Manipulação de Dados, Regressões e Métodos de Painel - Econometria

Universidade de Brasília - UnB

Monitoria 2/2021

Sumário

1 Breve Introdução ao R Markdown **2**

2 Instalando e Chamando Bibliotecas **2**

3 Regressão e Testes de Significância **3**
3.1 Regressão Múltipla Tradicional 3
3.2 Regressão de Corte Transversal 5
3.3 Testes de Significância 7
3.4 Interações e Heteroscedasticidade 8

4 Criando Variáveis nas Bases de Dados com o tidyverse **10**
4.1 Criando novas colunas 10
4.2 Agrupando 11
4.3 Filtragem e Seleção 12

5 Modelos de Painel: pacote `plm` **12**
5.1 Dados em Painel 12
5.2 Primeiras Diferenças (PD) 13
5.3 Efeitos Fixos 14
5.4 Efeitos Aleatórios e Teste de Hausman 16

A intenção desse arquivo é, por meio de exemplos dos capítulos 13 e 14 do livro *Introdução à Econometria* de Jeffrey Wooldridge, passar as noções básicas de **regressões**, **manipulações de dados**, **cortes transversais** e **modelos de painel** usando o R. Além disso, também falaremos um pouco do básico de como criar um caderno no R usando a sintaxe *markdown*.

1 Breve Introdução ao R Markdown

Um **notebook** do R (extensão `.Rmd`) permite juntar escrita e codificação em um só arquivo, muito semelhante a um arquivo `.ipynb` no Python.

Para começar, basta clicar no símbolo de + no canto superior esquerdo do RStudio e criar um novo *R Notebook*; quando ele abre, há inclusive um breve um tutorial pra você!

Para rodar os códigos dentro de cada célula no arquivo `.Rmd`, clique no botão"Run" ou aperte *Ctrl + Shift + Enter* (ou *Cmd* no Mac). Alternativamente, selecione toda a célula e pressione *Ctrl + Enter*. Os resultados aparecem abaixo da célula.

Para adicionar uma nova célula, clique em "Insert Chunk" ou aperte *Ctrl + Alt + I* (ou *Cmd + Opt* no Mac).

Importante: toda e qualquer coisa que tiver fora de um ambiente de célula será tratada como texto normal, o que permite escrever documentos inteiros (como este) no RStudio e depois exportá-los para Word ou PDF. Se você preferir, você pode fazer todas as listas (e até o trabalho final) só por aqui!

Para saber mais sobre a sintaxe do Markdown e como criar títulos, tabelas, negrito, etc, clique aqui!

Para saber mais sobre a exportação de um `.Rmd` para outros formatos (PDF, Word...), clique aqui! De início, já adiantamos que você vai precisar instalar o Latex no seu computador. Além disso, você vai precisar de uma função chamada `tinytex` (basta rodar `tinytex:install_tinytex()` em algum script do R).

2 Instalando e Chamando Bibliotecas

Para começar, precisamos chamar algumas bibliotecas. Se você nunca usou nenhuma, é preciso instalá-las com o comando `install.packages()` ou clicando no botão *Install* da seção *Packages* do RStudio.

Note o uso do #: ele é o símbolo de **comentário** e é muito útil para que a gente explique o que está acontecendo nos códigos. **POR FAVOR, USEM NA VIDA DE VOCÊS!!!**. Um código comentado é muito mais fácil de ler (e mais valorizado tanto aqui na matéria como no mercado de trabalho ;)).

```
## Para instalar os programas, retire o comentário (#) do código abaixo:
# install.packages("tidyverse", "wooldridge", ...)

## Usando as bibliotecas
library(tidyverse)  # manipulação de dados
library(magrittr)
library(wooldridge)  # baixar os dados do livro
library(robustbase)  # erros padrões robustos e clusterizados
library(sandwich)
library(clubSandwich)
library(plm)  # painel
library(car)  # testes
library(lmtest)
library(kableExtra)  # tabelas bonitinhas
library(broom)
library(stargazer)
library(modelsummary)
library(jtools)  # summ
```

Outras coisas importantes a serem feitas no começo de todo arquivo do R:

```
## Caso necessário, veja o diretório atual e mude para o que você está trabalhando

# getwd()
# setwd("caminho_da_pasta")
```

```r
## Para apagar todas as variáveis
# rm(list = ls())
## Limpar gráficos
# dev.off()
## Limpar console
# cat("\014")
```

3 Regressão e Testes de Significância

Vamos usar o exemplo 13.1 do livro do Wooldridge, que usa a base de dados `fertil1`.

3.1 Regressão Múltipla Tradicional

3.1.1 Dados

Os dados já estão prontos para uso na biblioteca `wooldridge`; não precisamos baixar nenhum arquivo.

```r
# Para manipularmos os dados, vamos atribuir a base a um objeto (df, sigla de DataFrame)
# Para atribuir algo, podemos usar = ou <-
# Assim, estou falando pro R que meu objeto df conterá os dados de fertil1
df <- fertil1

## Vendo a base de dados
view(df)
```

Alternativamente, você pode olhar o dataframe no RStudio clicando em "Environment" e no ícone de tabela no canto direito ao lado de `df`.

Para começar, vamos primeiro só fazer uma regressão básica com os dados de 1972.

Para selecionar apenas um *subset* dos nossos dados, há duas maneiras: usando o R base (que é o que faremos aqui) ou a função `filter` da biblioteca `tidyverse` (mais a frente a usaremos)!

No R base, todo dataframe pode ter seus dados selecionados com um comando do tipo
`df[linhas, colunas]`
Se deixarmos um desses campos em branco, selecionados todas as linhas/colunas. Usando o exemplo:

```r
# Comando abaixo: df72 é um dataframe cujas linhas são só aquelas cujo ano em df
# é 72, mas mantendo todas as colunas (nada após a vírgula)
df72 <- df[df$year == 72, ]
```

Cuidado! Usamos == quando queremos fazer testes lógicos de igualdade (e não =).

Além desse método, também podemos usar o `select` do **tidyverse**; mais sobre isso na seção de **Filtragem e Seleção**.

3.1.2 Regressão

Vamos agora fazer nossa regressão, atribuindo seus resultados ao objeto `reg1`.

Para isso, precisamos de uma *formula* e de *data* (dados; no nosso caso, `df`). A sintaxe de uma fórmula no R é da seguinte forma, em que o ~ separa as variáveis dependentes e as independentes:
`var_dependente ~ var_independente1 + var_independente2 + ...`

Usando o nosso exemplo, podemos ver os resultados na célula abaixo usando ou a função `summary` ou `summ` do pacote `jtools`. summ() (do pacote jtools) produz um resultado mais formatado, próximo àqueles gerados no Stata e no Python. Ela também facilita a inclusão de erros robustos (`robust = hc1`), que usaremos muito durante o curso e que não é possível de ser feito de forma rápida com o `summary`.

Contudo, as tabelas podem não aparecer diretamente embaixo da célula no PDF (podem ficar na próxima página, por exemplo). Para modelos com muito regressores (como esse), é melhor usar o `summary()` e talvez combiná-lo com o `stargazer`, apesar dele ser um pouco mais feio.

```r
# Criando a fórmula
formula1 <- kids ~ educ + age + agesq + black + east + northcen +
                   west + farm + othrural + town + smcity

# Regredindo
reg1 <- lm(formula = formula1, data = df72)

# Caso queiramos ajudamos sobre a função lm:
# ?lm

## Vendo os resultados (pode-se usar summary ou summ)
summ(reg1, robust = "HC1")
```

Observations	156
Dependent variable	kids
Type	OLS linear regression

F(11,144)	1.47
R^2	0.10
Adj. R^2	0.03

	Est.	S.E.	t val.	p
(Intercept)	-7.34	9.73	-0.75	0.45
educ	-0.07	0.06	-1.14	0.26
age	0.53	0.44	1.21	0.23
agesq	-0.01	0.00	-1.32	0.19
black	0.86	0.58	1.49	0.14
east	0.68	0.41	1.67	0.10
northcen	0.61	0.45	1.35	0.18
west	0.57	0.50	1.14	0.26
farm	0.26	0.49	0.54	0.59
othrural	0.17	0.57	0.30	0.76
town	0.30	0.39	0.79	0.43
smcity	1.00	0.39	2.59	0.01

Standard errors: Robust, type = HC1

Por fim, vamos ver como são exibidos os resultados usando a função `summary()`. Caso você deseje tornar os resultados mais bonitinhos, você pode usar a biblioteca e a função `stargazer`, exemplificada ao longo deste documento :)

```r
# Vendo resultados com summary()
summary(reg1)
```

```
##
## Call:
## lm(formula = formula1, data = df72)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -3.7067 -1.1763 -0.1576  1.0429  4.1648
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.342532  10.351665  -0.709   0.4793
## educ        -0.070776   0.064525  -1.097   0.2745
## age          0.530593   0.465099   1.141   0.2558
## agesq       -0.006549   0.005220  -1.255   0.2117
## black        0.861039   0.573610   1.501   0.1355
## east         0.677352   0.411884   1.645   0.1023
## northcen     0.608414   0.417896   1.456   0.1476
## west         0.570929   0.509231   1.121   0.2641
## farm         0.264472   0.469443   0.563   0.5741
## othrural     0.172848   0.572658   0.302   0.7632
## town         0.304561   0.370206   0.823   0.4121
## smcity       1.001307   0.496056   2.019   0.0454 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.798 on 144 degrees of freedom
## Multiple R-squared:  0.1008, Adjusted R-squared:  0.03209
## F-statistic: 1.467 on 11 and 144 DF,  p-value: 0.15
```

3.2 Regressão de Corte Transversal

3.2.1 MQO Agrupado

Corte transversal: observações ao longo do tempo de *diferentes indivíduos*. O que muda para uma regressão tradicional é, basicamente, a inclusão de *dummies* temporais, criando, assim, um modelo de Mínimos Quadrados Ordinários Agrupados (cada ano, nesse caso, representa um grupo).

Usando o nosso exemplo:

```
## Criando a fórmula com as dummies de cada ano (já estão no dataset)
formula2 <- kids ~ educ + age + agesq + black + east + northcen +
                   west + farm + othrural + town + smcity +
                   y74 + y76 + y78 + y80 + y82 + y84
reg2 <- lm(formula = formula2, data = df)

summary(reg2)
```

```
##
## Call:
## lm(formula = formula2, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9878 -1.0086 -0.0767  0.9331  4.6548
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.742457   3.051767  -2.537 0.011315 *
## educ        -0.128427   0.018349  -6.999 4.44e-12 ***
## age          0.532135   0.138386   3.845 0.000127 ***
## agesq       -0.005804   0.001564  -3.710 0.000217 ***
```

```
## black        1.075658   0.173536   6.198 8.02e-10 ***
## east         0.217324   0.132788   1.637 0.101992
## northcen     0.363114   0.120897   3.004 0.002729 **
## west         0.197603   0.166913   1.184 0.236719
## farm        -0.052557   0.147190  -0.357 0.721105
## othrural    -0.162854   0.175442  -0.928 0.353481
## town         0.084353   0.124531   0.677 0.498314
## smcity       0.211879   0.160296   1.322 0.186507
## y74          0.268183   0.172716   1.553 0.120771
## y76         -0.097379   0.179046  -0.544 0.586633
## y78         -0.068666   0.181684  -0.378 0.705544
## y80         -0.071305   0.182771  -0.390 0.696511
## y82         -0.522484   0.172436  -3.030 0.002502 **
## y84         -0.545166   0.174516  -3.124 0.001831 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.555 on 1111 degrees of freedom
## Multiple R-squared:  0.1295, Adjusted R-squared:  0.1162
## F-statistic: 9.723 on 17 and 1111 DF,  p-value: < 2.2e-16
```

E caso as *dummies* de anos não existissem já prontas nos nossos dados? Teríamos que criá-las uma por uma? Felizmente, a resposta é **não**!

O R possui um tipo de dado chamado **fator**, que basicamente diz pro R que a coluna/variável deve ser tratada como uma variável categórica. Se só colocássemos `year` na nossa fórmula, não colocaríamos *dummies*, uma vez que o R a interpretaria como uma variável contínua.

```
## Transformando a coluna de uma variável contínua para um fator
df$year <- factor(df$year)

## Fazendo o mesmo modelo acima e já mostrando os resultados direto
summary(lm(
  formula = kids ~ educ + age + agesq + black + east + northcen +
                   west + farm + othrural + town + smcity + year,
  data = df
))
```

```
##
## Call:
## lm(formula = kids ~ educ + age + agesq + black + east + northcen +
##     west + farm + othrural + town + smcity + year, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9878 -1.0086 -0.0767  0.9331  4.6548
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.742457   3.051767  -2.537 0.011315 *
## educ        -0.128427   0.018349  -6.999 4.44e-12 ***
## age          0.532135   0.138386   3.845 0.000127 ***
## agesq       -0.005804   0.001564  -3.710 0.000217 ***
## black        1.075658   0.173536   6.198 8.02e-10 ***
## east         0.217324   0.132788   1.637 0.101992
```

```
## northcen     0.363114   0.120897   3.004 0.002729 **
## west         0.197603   0.166913   1.184 0.236719
## farm        -0.052557   0.147190  -0.357 0.721105
## othrural    -0.162854   0.175442  -0.928 0.353481
## town         0.084353   0.124531   0.677 0.498314
## smcity       0.211879   0.160296   1.322 0.186507
## year74       0.268183   0.172716   1.553 0.120771
## year76      -0.097379   0.179046  -0.544 0.586633
## year78      -0.068666   0.181684  -0.378 0.705544
## year80      -0.071305   0.182771  -0.390 0.696511
## year82      -0.522484   0.172436  -3.030 0.002502 **
## year84      -0.545166   0.174516  -3.124 0.001831 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.555 on 1111 degrees of freedom
## Multiple R-squared:  0.1295, Adjusted R-squared:  0.1162
## F-statistic: 9.723 on 17 and 1111 DF,  p-value: < 2.2e-16
```

3.3 Testes de Significância

Tanto o `summ()` quanto o `summary()` fornecem os p-valores para cada a significância de cada variável individual; mas e se quisermos fazer um Teste F de significância conjunta?

```
## Queremos testar se o ambiente de vida tem algum impacto na fertilidade
# c() é um vetor/lista de coisas separadas por ,
# white_adjust permite fazer testes robustos ("hc1" em geral)
linearHypothesis(reg2, c("farm=0", "othrural=0", "town=0", "smcity=0"),
                 white.adjust = "hc1")
```

```
## Linear hypothesis test
##
## Hypothesis:
## farm = 0
## othrural = 0
## town = 0
## smcity = 0
##
## Model 1: restricted model
## Model 2: kids ~ educ + age + agesq + black + east + northcen + west +
##     farm + othrural + town + smcity + y74 + y76 + y78 + y80 +
##     y82 + y84
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F Pr(>F)
## 1   1115
## 2   1111  4 1.1866 0.3149
```

Como o p-valor é maior que 0.05 (0.3149), não podemos rejeitar a hipótese nula de que todas as variáveis são conjuntamente nulas, ou seja, o ambiente de vida não parece afetar a fertilidade uma vez que outras características foram controladas. Sem os erros robustos, o p-valor é de 0.3275.

Por fim, caso quiséssemos checar se `farm` e `town` são iguais, basta escrever `c("farm"="town")`. Como o p-valor do teste é maior que 0.05, não podemos rejeitar essa hipótese de igualdade.

```r
linearHypothesis(reg2, c("farm=town"), white.adjust = "hc1")
```

```
## Linear hypothesis test
##
## Hypothesis:
## farm - town = 0
##
## Model 1: restricted model
## Model 2: kids ~ educ + age + agesq + black + east + northcen + west +
##     farm + othrural + town + smcity + y74 + y76 + y78 + y80 +
##     y82 + y84
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F Pr(>F)
## 1   1112
## 2   1111  1 1.0267 0.3112
```

3.4 Interações e Heteroscedasticidade

Às vezes, precisamos interagir variáveis, adicionando a multiplicação delas nos modelos. O R tem um jeito bem fácil que faz com que não seja necessário criar todas essas variáveis de multiplicações na nossa base de dados.

Digamos que queiramos fazer um modelo simples, levando em contas apenas educação e os anos na amostra. Além disso, queremos ver se o retorno da educação muda ao longo do tempo; para isso, precisamos interagir essas variáveis:

```r
## Dois jeitos equivalentes:
# : coloca apenas a interação
# * coloca a interação e cada variável individualmente
formula_interacao <- kids ~ educ + year + educ:year
formula_interacao <- kids ~ educ*year

reg_interacao <- lm(formula = formula_interacao, data = df)
summary(reg_interacao)
```

```
##
## Call:
## lm(formula = formula_interacao, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6039 -1.0319 -0.0793  1.0088  5.0622
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.52041    0.67520   5.214  2.2e-07 ***
## educ        -0.04071    0.05456  -0.746   0.4557
## year74       0.33395    0.91888   0.363   0.7163
## year76       0.57799    0.89653   0.645   0.5193
## year78       1.23811    0.96970   1.277   0.2019
## year80       0.61995    0.91263   0.679   0.4971
## year82       1.10174    0.89024   1.238   0.2161
## year84       1.57082    0.91416   1.718   0.0860 .
```

```
## educ:year74 -0.01183    0.07380  -0.160   0.8727
## educ:year76 -0.06524    0.07205  -0.905   0.3654
## educ:year78 -0.11386    0.07679  -1.483   0.1384
## educ:year80 -0.06204    0.07170  -0.865   0.3871
## educ:year82 -0.12706    0.06945  -1.830   0.0676 .
## educ:year84 -0.17443    0.07109  -2.454   0.0143 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.591 on 1115 degrees of freedom
## Multiple R-squared:  0.08542,    Adjusted R-squared:  0.07476
## F-statistic: 8.011 on 13 and 1115 DF,  p-value: 1.624e-15
```

Mas será que os erros-padrão são válidos? Em outras palavras, será que há **homo** ou **heteroscedasticidade**?

Para averiguar isso, precisamos testar para **heteroscedasticidade**, o que é feito por meio do teste de *Breusch-Pagan*. No R, isso é feito usando a função `bptest` do pacote `lmtest`:

```
bptest(reg_interacao)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  reg_interacao
## BP = 39.151, df = 13, p-value = 0.0001891
```

Como o p-valor é menor que 0.05, podemos rejeitar a hipótese nula de homoscedasticidade e os erros-padrão normais não são válidos. Assim, precisaremos de erros robustos.

Para pegarmos os erros robustos sem usar o `summ()`, precisamos usar o pacote *sandwich*:

```
## Pegando a matriz de covariância robusta
cov <- vcovHC(reg_interacao, type = "HC1")

## Pegando o erro-padrão (raiz quadrada da diagonal da matriz)
robust.se.reg_interacao <- sqrt(diag(cov))
```

Para criarmos uma tabela com esses erros sem usar o `summ()`, podemos usar a biblioteca `stargazer`, feita justamente para construir tabelas no R:

```
stargazer(reg_interacao,
          title = "Regressão de Interações com Erros Robustos",
          type = "text",
          se = list(robust.se.reg_interacao),
          decimal.mark = ",", digit.separator = ".",
          align = T, no.space = T, single.row = T)
```

```
##
## Regressão de Interações com Erros Robustos
## ================================================
##                          Dependent variable:
##                      ---------------------------
##                                  kids
## ------------------------------------------------
## educ                        -0,041 (0,067)
```

```
## year74                      0,334 (1,056)
## year76                      0,578 (1,133)
## year78                      1,238 (1,337)
## year80                      0,620 (1,098)
## year82                      1,102 (1,035)
## year84                      1,571 (1,019)
## educ:year74                -0,012 (0,083)
## educ:year76                -0,065 (0,090)
## educ:year78                -0,114 (0,104)
## educ:year80                -0,062 (0,087)
## educ:year82                -0,127 (0,081)
## educ:year84               -0,174** (0,079)
## Constant                  3,520*** (0,834)
## ------------------------------------------------
## Observations                     1.129
## R2                               0,085
## Adjusted R2                      0,075
## Residual Std. Error       1,591 (df = 1115)
## F Statistic           8,011*** (df = 13; 1115)
## ================================================
## Note:                *p<0,1; **p<0,05; ***p<0,01
```

4 Criando Variáveis nas Bases de Dados com o tidyverse

4.1 Criando novas colunas

De modo geral, só usamos as variáveis que já estavam dadas e criadas na base de dados. Contudo, muitas problemas exigem a criação de novas variáveis no banco de dados; o intuito aqui é mostrar o básico de como fazer isso usando o **tidyverse** [1], que é um conjunto de bibliotecas como o `dplyr`, o `ggplot` e outras afins.

Digamos que queiramos logaritimizar `kids` (para termos os coeficientes das variáveis independentes*100 em termos percentuais, ou seja, a semi-elasticidade) e definir o grau de ensino a depender dos anos de estudo da pessoa.

Usaremos o `mutate` para falar para o R que queremos modificar nosso dataframe; no caso, estaremos criando uma nova coluna chamada `log_kids`, que contérá o logaritmo do número de filhos. Contudo, a função `log()` é indefinida no 0, de modo que precisaremos levar em consideração as pessoas com 0 filhos usando uma condição `ifelse()`. A sintaxe da função é igual à `SE()` no Excel.

Além disso, vamos criar uma variável categórica se diz se a pessoa é muito ou pouca educada (sem nenhum juízo de valor, só para ilustrar o comando do R). Se ela tiver menor de 9 anos de educação, diremos que ela tem pouca; caso contrário, muita.

```
## Vamos modificar nosso dataframe passando-o para um pipe
df <- df %>%
  mutate(log_kids = ifelse(kids == 0, 0, log(kids)))  %>%
  mutate(grau_educ = ifelse(educ < 9, "Pouca", "Muita"))
```

Vamos colocar a variável `grau_educ` como fator para dizer ao R que os dados representam diferentes categorias e, assim, conseguimos colocá-la numa regressão.

```
## Colocando a variável categórica grau_educ como um dado do tipo fator
df$grau_educ <- factor(df$grau_educ)
```

[1] Para usarmos o tidyverse, vamos precisar usar um operador chamado *pipe* (`%>%`), que pega o objeto da linha anterior e o passa para uma função que estará na próxima linha. Vale a pena pesquisar mais sobre isso!

```r
## Sumário da regressão simples
summ(lm(formula = log_kids ~ grau_educ, data = df), robust = "HC1")
```

Observations	1129
Dependent variable	log_kids
Type	OLS linear regression

F(1,1127)	0.98
R^2	0.00
Adj. R^2	-0.00

	Est.	S.E.	t val.	p
(Intercept)	0.88	0.02	50.67	0.00
grau_educPouca	0.07	0.09	0.85	0.39

Standard errors: Robust, type = HC1

Sem outros controles, uma pessoa com pouca educação tem 7% mais filhos que uma com muita educação, apesar desse resultado ser insignificante ($p > 0.05$).

4.2 Agrupando

Uma outra coisa importante de ser feita é encontrar estatísticas (média, soma, desvio-padrão, maior, menor) por grupo. Na análise descritiva do trabalho final, por exemplo, vocês precisarão calcular a média de rendimentos por cor/etnia, gênero e até uma combinação dos dois (homem negro, mulher branca)... Para isso, usaremos a função `group_by` do **tidyverse**.

Digamos que queiramos encontrar a média, o máximo, o mínimo e o desvio-padrão de `educ` por cor – além do número de observações de cada grupo –, o que faremos a partir da identificação dada pela variável `black` (== 1 se a pessoa se identifica como negra).

Além disso, vamos brincar com os tipos de célula do R Markdown: vamos colocar `results ='hide'` na célula abaixo para que o resultado um pouco mais feio do `group_by` não seja mostrado. Além disso, colocaremos `echo=FALSE` na célula seguinte para que o código do `stargazer` não seja exibido no PDF, mostrando apenas a tabela gerada (veja o `.Rmd` para entender melhor).

```r
## Estatísticas por cor
estatisticas.grupo <- df %>%
  group_by(black) %>%
  summarise(Média = mean(educ),
            Máximo = max(educ),
            Mínimo = min(educ),
            Desvio_Padrão = sd(educ),
            Observações = n()) %>%
  round(2) %>%  # arredondando para 2 dígitos
  as.data.frame()  # colocando como DataFrame (round transforma em lista)
```

```
##
## Estatísticas por Cor
## ======================================================
##   black Média  Máximo Mínimo Desvio_Padrão Observações
## ------------------------------------------------------
```

```
## 1   0   12,700   20     0        2,620        1.033
## 2   1   12,610   20     7        2,870         96
## ---------------------------------------------------
```

4.3 Filtragem e Seleção

Caso queiramos filtrar a nossa base de dados para apenas um *subset* de observações, podemos usar a função `filter` do **tidyverse**.

Se quisermos apenas as pessoas com pouca educação, executamos o código abaixo:

```
## Criando uma nova base de dados apenas com as pessoas com pouca educação
# Selecionamos apenas algumas colunas; isso não modifica os dados, mas
# pode tornar manipulações em bases maiores mais rápidas
df_poucaeduc <- df %>%
  select(kids, educ, age, grau_educ) %>%
  filter(grau_educ == "Pouca")
```

5 Modelos de Painel: pacote `plm`

Uma base de dados em painel contém informações de um **mesmo indivíduo** ao longo do tempo, o que nos permite, através de manipulações, estimar modelos que expurgam características individuais não-observáveis relegadas ao termo de erro u que estejam correlacionadas com a nossa variável dependente (matematicamente, $cov(x_j, u) \neq 0$) e que, consequentemente, violariam a condição de **exogeneidade estrita** e tornariam nossos estimadores inconsistentes.

Há três modelos principais que vamos usar: **Primeiras Diferenças, Efeitos Fixos e Efeitos Aleatórios**. Antes disso, vamos repassar o que são dados em painel.

5.1 Dados em Painel

Como dito anteriormente, dados em painel são aqueles em que cada indivíduo é observado em diferentes períodos de tempo.

Para mexermos com dados em painel, precisamos primeiro entender como é a estrutura desses dados, o que faremos através do Exemplo 13.8 do livro do Wooldridge. Leia o exemplo pra ter uma base teórica melhor!

```
## Atribuindo a base de dados que analisa os efeitos de zonas industriais sobre
## os pedidos de seguro-desemprego
df <- ezunem

## Coletando informações sobre a base
?ezunem

## Vendo a base
view(df)
```

Note que temos observações para 22 cidades ao longo de 9 anos. Além disso, veja a organização dos dados: primeiro, a cidade 1 aparece 9 vezes; depois, vem todas as observações da cidade 2 (e assim por diante). Essa organização é **muito importante** para fazer os modelos e até ajuda a entender o que está acontecendo.

5.1.1 Agrupando com `mutate()`

Uma outra coisa importante que vocês precisarão fazer é criar uma nova coluna com a média de determinada variável agrupada por indivíduo. Isso será útil quando vocês forem fazer o modelo de *Efeitos Aleatórios Correlacionados*.

Criando uma nova coluna que contém à média de `uclms`, ou seja, o número médio de seguros-desemprego **de cada cidade**; note que, como criaremos uma nova coluna, **precisamos** usar o `mutate()`! Se alguém usa Python, isso seria semelhante a usar um `media_uclms = df.groupby('city').transform('mean')`.

```
## Criando coluna com a média temporal de cada cidade
### Vamos usar um pipe modificado do maggritr, que substitui o df <- df %>%
df %<>%
  group_by(city) %>%
  mutate(media_uclms = mean(uclms))
```

5.2 Primeiras Diferenças (PD)

Sem entrar muito na teoria, o modelo de PD basicamente regride as diferenças temporais das variáveis. Assim, se temos observações para 9 anos, o 1º ano seria descartado e faríamos a regressão com os dados do 2º ao 9º ano, subtraídos daqueles do ano anterior.

Como assumimos que as características não-observáveis são constantes no tempo, podemos eliminar os vieses! Note que, consequentemente, também eliminaremos qualquer variável que for **constante no tempo**.

Vamos usar a função `plm()` para executar todos os modelos de painel. A sintaxe é semelhante a da função `lm()` que usamos anteriormente, mas temos que passar também o argumento `index = c(var_individuo, var_tempo)`. No nosso caso, os nossos indivíduos são as cidades (`city`) e a variável de tempo é `year`.

Além disso, devemos indicar qual o modelo de painel que queremos que o R calcule, o que é feito através do parâmetro `model`. Aqui, usaremos `"fd"`, abreviação de *First Differences*.

```
# Tornando year um fator
df$year <- factor(df$year)

# Fazendo a formula e expressamente incluindo um intercepto (1 +)
# ez é uma dummy = 1 se a cidade tinha uma zona industrial naquele ano
formula_pd <- luclms ~ 1 + year + ez

# Fazendo o modelo (note que os nomes dentro do vetor de index estão entre aspas)
modelo_pd <- plm(formula = formula_pd, data = df,
                 index = c("city", "year"),
                 model = "fd")

# Vendo o sumário (obs: summ não suporta modelos de painel)
summary(modelo_pd)
```

```
## Oneway (individual) effect First-Difference Model
##
## Call:
## plm(formula = formula_pd, data = df, model = "fd", index = c("city",
##     "year"))
##
## Balanced Panel: n = 22, T = 9, N = 198
## Observations used in estimation: 176
##
## Residuals:
##       Min.    1st Qu.     Median    3rd Qu.       Max.
## -0.4925475 -0.1426735 -0.0091986  0.1494602  0.6062249
##
## Coefficients: (1 dropped because of singularities)
##              Estimate Std. Error t-value  Pr(>|t|)
## (Intercept) -0.149053   0.016881 -8.8296 1.382e-15 ***
```

```
## year1981    -0.172579   0.043317 -3.9841 0.0001010 ***
## year1982     0.433601   0.057112  7.5921 2.117e-12 ***
## year1983     0.227903   0.064468  3.5351 0.0005275 ***
## year1984     0.038186   0.065241  0.5853 0.5591348
## year1985     0.188688   0.064468  2.9268 0.0039018 **
## year1986     0.308263   0.057112  5.3975 2.287e-07 ***
## year1987     0.189632   0.043317  4.3777 2.109e-05 ***
## ez          -0.181878   0.078186 -2.3262 0.0212087 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    20.678
## Residual Sum of Squares: 7.7958
## R-Squared:      0.623
## Adj. R-Squared: 0.60494
## F-statistic: 34.496 on 8 and 167 DF, p-value: < 2.22e-16
```

Vamos fazer o teste de Breusch-Pagan para modelos de painel usando a função `bptest`. Ela não funciona com o pacote `plm`, então criaremos um novo modelo de primeiras diferenças na mão usando as variáveis de diferença já presente no dataset.

```
## Para tirar a diferença, usamos o comando diff()
# Como diff() elimina a primeira observação, colocaremos um NA no início
## e juntamos o NA com as diferenças em um vetor c()
df$diff_luclms <- c(NA, diff(df$luclms, differences = 1))
df$diff_ez <- c(NA, diff(df$ez, differences = 1))

## Contudo, temos que ter cuidado quando os dados transacionam de cidade,
## ou seja, quando os dados passam da cidade 1 para a cidade 2
df$diff_luclms <- ifelse(df$year == 1980, NA, df$diff_luclms)
df$diff_ez <- ifelse(df$year == 1980, NA, df$diff_ez)

# Formula e modelo
formula_pd_bruta <- diff_luclms ~ 1 + year + diff_ez
modelo_pd_bruto <- lm(formula = formula_pd_bruta, data = df)
bptest(modelo_pd_bruto)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  modelo_pd_bruto
## BP = 6.914, df = 8, p-value = 0.5459
```

Não há indícios de heteroscedasticidade :) (p-valor > 0.05).

5.3 Efeitos Fixos

O modelo de Efeitos Fixos (EF ou `within`) atua subtraindo a média das variáveis do valor de cada tempo, o que também elimina as características não-observáveis constantes no tempo.

Novamente, não conseguimos estimar variáveis de interesse que sejam constantes no tempo. Contudo, podemos interagi-las com *dummies* temporais para ver se esses efeitos mudam ao longo do tempo. Por exemplo, nunca encontraremos o valor real do retorno da educação para o salário, mas poderíamos ver como ele mudou ao longo do tempo interagindo `educ` com *dummies* temporais.

Para exemplificar, vamos usar o Exemplo 14.2 do livro do Wooldridge.

```r
## Pegando os dados e atribuindo ao nosso objeto favorito, df
df <- wagepan

## Descrição da base de dados
?wagepan

## Vendo a base
view(df)
```

Vamos estimar o modelo de EF usando `model = within` no `plm`. Nossos indivíduos são as pessoas (identificadas por `nr`) e o tempo é o ano (`year`).

```r
## Colocando ano como fator
df$year <- factor(df$year)

## Formula (lembrando da sintaxe das interações)
formula_ef <- lwage ~ married + union + year*educ

## Modelo e sumário
modelo_ef <- plm(formula = formula_ef, data = df,
                 model = "within",
                 index = c("nr", "year"))
summary(modelo_ef)
```

```
## Oneway (individual) effect Within Model
##
## Call:
## plm(formula = formula_ef, data = df, model = "within", index = c("nr",
##     "year"))
##
## Balanced Panel: n = 545, T = 8, N = 4360
##
## Residuals:
##      Min.   1st Qu.    Median   3rd Qu.      Max.
## -4.152111 -0.125630  0.010897  0.160800  1.483401
##
## Coefficients:
##                 Estimate Std. Error t-value  Pr(>|t|)
## married        0.0548205  0.0184126  2.9773  0.002926 **
## union          0.0829785  0.0194461  4.2671 2.029e-05 ***
## year1981      -0.0224158  0.1458885 -0.1537  0.877893
## year1982      -0.0057611  0.1458558 -0.0395  0.968495
## year1983       0.0104297  0.1458579  0.0715  0.942999
## year1984       0.0843743  0.1458518  0.5785  0.562965
## year1985       0.0497253  0.1458602  0.3409  0.733190
## year1986       0.0656064  0.1458917  0.4497  0.652958
## year1987       0.0904448  0.1458505  0.6201  0.535216
## year1981:educ  0.0115854  0.0122625  0.9448  0.344827
## year1982:educ  0.0147905  0.0122635  1.2061  0.227872
## year1983:educ  0.0171182  0.0122633  1.3959  0.162830
## year1984:educ  0.0165839  0.0122657  1.3521  0.176437
## year1985:educ  0.0237085  0.0122738  1.9316  0.053479 .
## year1986:educ  0.0274123  0.0122740  2.2334  0.025583 *
## year1987:educ  0.0304332  0.0122723  2.4798  0.013188 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    572.05
## Residual Sum of Squares: 474.35
## R-Squared:      0.1708
## Adj. R-Squared: 0.048567
## F-statistic: 48.9069 on 16 and 3799 DF, p-value: < 2.22e-16
```

Note que, como `educ` é constante no tempo (os homens da amostra são mais velhos e não estudam mais), não podemos estimar seus retornos. Contudo, vemos que, frente ao retorno de 1980, a educação parece estar sendo mais valorizada ao longo do tempo. Em 1983, por exemplo, ela dava um retorno 1,7% maior do que fazia em 1980; em 1987, 3%! Note também que a diferença só passa a ser significante a partir de 1986.

Vamos fazer um teste F de significância conjunta das interações (ao contrário do `bptest()`, `linearHypothesis()` funciona com objetos do `plm`):

```
linearHypothesis(modelo_ef,
                 c("year1981:educ=0", "year1982:educ=0", "year1983:educ=0",
                   "year1984:educ=0", "year1985:educ=0", "year1986:educ=0",
                   "year1987:educ=0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## year1981:educ = 0
## year1982:educ = 0
## year1983:educ = 0
## year1984:educ = 0
## year1985:educ = 0
## year1986:educ = 0
## year1987:educ = 0
##
## Model 1: restricted model
## Model 2: lwage ~ married + union + year * educ
##
##   Res.Df Df  Chisq Pr(>Chisq)
## 1   3806
## 2   3799  7 8.6554     0.2784
```

O p-valor do teste é de 0.27, ou seja, não podemos rejeitar a hipótese nula de que elas são conjuntamente iguais a 0 (apesar de algumas delas serem individualmente significantes).

5.4 Efeitos Aleatórios e Teste de Hausman

O modelo de Efeitos Aleatórios (EA ou `random`) atua fazendo a subtração de uma quasi-média, o que permite estimar variáveis constantes ao longo do tempo. Teoricamente, os resultados de EA sempre estão entre os de EF e MQO Agrupado.

Contudo, EA pode ser viesado se houver correlação entre o vetor de variáveis explicativas X e as características não-observáveis relegadas ao termo de erro (a_i). Matematicamente, EA só é consistente se $cov(X, a_i) = 0$, o que precisa ser testado via Teste de Hausman (`phtest()` no pacote `plm`).

Se rejeitarmos a hipótese nula de que $cov(X, a_i) = 0$, temos que usar Efeitos Fixos e EA é inconsistente. Caso não consigamos rejeitar, prefere-se EA (apesar de EF não ser viesado, EA é mais eficiente, ou seja, produz menores erros-padrão).

Para exemplificar, vamos usar o Exemplo 14.4 do livro do Wooldridge, usando os mesmos dados que usamos na exemplificação dos Efeitos Fixos. Além disso, vamos estimar MQO Agrupado, Efeitos Fixos e Efeitos Aleatórios e compará-los em uma tabela bonitinha com o `stargazer`.

5.4.1 MQO Agrupado

```
formula_ea <- lwage ~ educ + black + hisp + exper + expersq + married +
                      union + year
modelo_ols_comp <- lm(formula <- formula_ea, data = df)
```

5.4.2 Efeitos Fixos

```
modelo_ef_comp <- plm(formula = formula_ea, data = df,
                      model = "within",
                      index = c("nr", "year"))
```

5.4.3 Efeitos Aleatórios

```
modelo_ea_comp <- plm(formula = formula_ea, data = df,
                      model = "random",
                      index = c("nr", "year"))
```

5.4.4 Tabela Comparativa

```
# Tabela comparativa
stargazer(modelo_ols_comp, modelo_ea_comp, modelo_ef_comp,
          title = "Comparação de Modelos de Painel",
          decimal.mark = ",", digit.separator = ".",
          align = T, no.space = T, column.sep.width = "2pt",
          type = "text", dep.var.labels = c("lwage"), model.names = F,
          column.labels = c("OLS", "EA", "EF"))
```

```
##
## Comparação de Modelos de Painel
## ================================================================================
##                                        Dependent variable:
##                      ------------------------------------------------------------
##                                lwage                          lwage
##                                 OLS              EA               EF
##                                 (1)              (2)              (3)
## --------------------------------------------------------------------------------
## educ                         0,091***         0,092***
##                               (0,005)          (0,011)
## black                        -0,139***        -0,139***
##                               (0,024)          (0,048)
## hisp                           0,016            0,022
##                               (0,021)          (0,043)
## exper                        0,067***         0,106***         0,132***
##                               (0,014)          (0,015)          (0,010)
## expersq                      -0,002***        -0,005***        -0,005***
##                               (0,001)          (0,001)          (0,001)
## married                      0,108***         0,064***          0,047**
##                               (0,016)          (0,017)          (0,018)
## union                        0,182***         0,106***         0,080***
```

```
##                              (0,017)          (0,018)          (0,019)
## year1981                     0,058*           0,040            0,019
##                              (0,030)          (0,025)          (0,020)
## year1982                     0,063*           0,031            -0,011
##                              (0,033)          (0,032)          (0,020)
## year1983                     0,062*           0,020            -0,042**
##                              (0,037)          (0,042)          (0,020)
## year1984                     0,090**          0,043            -0,038*
##                              (0,040)          (0,051)          (0,020)
## year1985                     0,109**          0,058            -0,043**
##                              (0,043)          (0,061)          (0,020)
## year1986                    0,142***          0,092            -0,027
##                              (0,046)          (0,071)          (0,020)
## year1987                    0,174***          0,135*
##                              (0,049)          (0,081)
## Constant                      0,092            0,024
##                              (0,078)          (0,151)
## ------------------------------------------------------------------------------------
## Observations                  4.360            4.360            4.360
## R2                            0,189            0,181            0,181
## Adjusted R2                   0,187            0,178            0,061
## Residual Std. Error     0,480 (df = 4345)
## F Statistic         72,459*** (df = 14; 4345) 957,774*** 83,851*** (df = 10; 3805)
## ====================================================================================
## Note:                                                *p<0,1; **p<0,05; ***p<0,01
```

Lembrando: EF não consegue estimar variáveis constantes no tempo; por isso, há menos resultados reportados na última coluna.

5.4.5 Teste de Hausman

Para ver qual modelo é melhor entre EA e EF, fazemos um teste de Hausman usando a função `phtest()` da biblioteca `plm`, onde $H_0 : cov(X, a_i) = 0$:

```
## Teste de Hausman
phtest(x = formula_ea, data = df)
```

```
##
##  Hausman Test
##
## data:  formula_ea
## chisq = 31.707, df = 10, p-value = 0.000448
## alternative hypothesis: one model is inconsistent
```

Como o p-valor é menor que 0.05, EA é inconsistente e devemos usar o modelo de Efeitos Fixos!