

# **Notas Introdutórias Em Econometria Aplicada Usando R/RStudio**

**C.D. Shikida e Rodrigo N. Fernandez**

## Prefácio

A linguagem R tem sido amplamente utilizada na pesquisa estatística e, claro, na econometria. Uma de suas vantagens? Você não precisa pagar licença porque o programa é um *software* livre. Nesta categoria estão também o JMulti, Gretl e o EasyReg (cujo *mirror* brasileiro, agora, está no PPGOM/UFPeI).

Este texto é uma versão atualizada e bastante alterada de uma apostila anterior feita para um minicurso de 20 horas no PPGOM/UFPeI, salvo engano, em 2013 (repetido em 2014). Por que alterar a apostila? O principal motivo é que logo após finalizar a versão anterior, tomei contato com a plataforma *Rstudio* que funciona sobre o R, mas com uma visualização mais confortável (e com menos problemas para quem precisa, de vez em quando, salvar muitos gráficos...). Outros motivos? *Stargazer*, *RMarkdown*, etc<sup>1</sup>.

Os comentários recebidos por ex-alunos e amigos ao longo do tempo convenceram-me de que a melhor – embora mais trabalhosa – estratégia era a de optar por uma versão fortemente alterada. O leitor que conheceu a apostila anterior notará que não terminei a tarefa ainda, embora toda a apostila tenha sido revisada. Esta é a versão provisória que será usada no minicurso do R a ser ministrado em breve, no [PPGOM-UFPeI](#). Dos autores da versão anterior, o prof. Rodrigo Nobre Fernandez segue como co-autor (e com planos imperialistas, pelo que pude perceber). Também está em meus planos a participação de outros co-autores do PPGOM para uma versão mais organizada deste material.

Três avisos importantes: (i) aos que nunca tiveram contato comigo: o material a seguir pode parecer informal demais em muitas ocasiões. A intenção é que sua leitura seja menos árdua e um pouco mais divertida em alguns momentos. O problema, claro, é que economistas não são muito bem conhecidos pela qualidade de suas piadas; (ii) as planilhas para replicação de – quase – todos os exemplos desta apostila estarão disponíveis em minha página no [PPGOM-UFPeI](#) e; (iii) **este não é um curso de programação em R**.

Agradeço a André Carraro e César Tejada pela oportunidade de experimentar este material com os alunos de Mestrado de Ciências Econômicas na UFPEL nas edições anteriores deste minicurso. Agradeço também Elisangela Luzia Araújo pelo convite para ministrar o minicurso na XXXI Semana do Economista na UEM (Universidade Estadual de Maringá) em 2016.

Comentários podem ser enviados para o endereço [cdshikida at gmail dot com](mailto:cdshikida@gmail.com).

Pelotas, 16 de Outubro de 2016.

---

<sup>1</sup> Nesta versão da apostila ainda não há um tópico sobre *RMarkdown*. As referências importantes estão todas aqui: <http://rmarkdown.rstudio.com/>. Outro ponto importante é que apenas os tópicos feitos pelo professor Rodrigo Nobre Fernandez foram feitos apenas no R, não no *RStudio*. O leitor não terá, contudo, grandes dificuldades em acompanhar o conteúdo. Em breve, caso tudo corra bem, disponibilizo também os *scripts*.

## Índice

0. O que Homer Simpson disse para Peter Griffin? .....	3
1. Cheguei ao laboratório, mas não sei o que se passa. Alguém pode me ajudar? .....	9
2. Impressiona seus amigos e aquela garota bonita! .....	20
3. Como é que eu importo os meus dados para o R? Que loucura é esta? .....	32
4. Vamos começar a trabalhar? Tá tudo em ordem? Eu sei tudo? Tenho dúvidas? Opa, eu tenho dúvidas!.....	35
5. A Estatística dos Casos Extra-Conjugais de Maridos e Esposas.....	37
6. Regressão Linear Sistemas de Equações Simultâneas e Testes de Diagnósticos .....	41
6.1. O eterno debate da função consumo (O Primeiro Ato).....	41
Anexo 1 – A questão das observações influentes.....	53
6.2. O PIB, a Base Monetária e o Teste de Hipóteses.....	55
6.3. Morar longe da faculdade diminui seu salário? .....	56
6.4. Mais armas, menos crimes...com <i>dummies</i> .....	61
6.5. A Lei de Wagner, a Guerra do Paraguai e o teste “t” .....	64
Anexo 2 – Tabelas com várias regressões no R: salvando o dia do bolsista .....	66
Anexo 3 - Sorvetes, Matrizes e MQO (prof. Rodrigo N. Fernandez) .....	67
7. Modelos Microeconômicos.....	73
7.1. Coca ou Pepsi? .....	73
7.2. Esporte Espetacular .....	77
7.3. Os motivos da traição: o retorno dos casos extra-conjugais! .....	81
7.4. As decisões do Banco Central e nossa vida .....	83
7.5. Vamos dividir o Estado do Pará? .....	87
8. Econometria de Séries de Tempo em R (Parte I) .....	90
8.1. É fácil assim fazer a previsão da produção industrial brasileira? .....	92
Anexo 4 – Modelos ARIMA degenerados .....	100
Anexo 5 – Tendências, <i>dummies</i> sazonais e algumas dicas básicas .....	100
8.2. A função consumo keynesiana é estável para o Brasil? (O Segundo Ato).....	101
8.3. Rui Barbosa importa? .....	105
9. Econometria de Séries de Tempo em R (Parte II) .....	108
9.1. O leite nosso de cada dia.....	108
9.2. Quem veio primeiro: o ovo ou a galinha?.....	120
Anexo 6 – Ovos e Galinhas e a causalidade de Granger no pacote vars .....	125
10. Salário, Etnia e Educação: qual é a relação? – Uma Introdução à Econometria dos Dados em Pannel no R (Prof. Rodrigo N. Fernandez) .....	128
Bibliografia Parcial .....	136

## 0. O que Homer Simpson disse para Peter Griffin? <sup>2</sup>

Em 2014, o mundo (ou pelo menos parte dele) assistiu o *crossover* mais esperado da história da TV (na minha opinião): os dois episódios em que Homer Simpson encontrava Peter Griffin. Talvez você não saiba do que estou falando, mas os fãs de ambos os shows (ou de algum deles), sejam eles mestres, doutores ou calouros em Ciências Econômicas, sabem.

Um evento tão memorável não poderia passar batido a um usuário de R. Foi assim que tive a ideia de iniciar esta nova versão da apostila com um exemplo que, espero, mostre toda a potencialidade do R (notadamente, sob a camada que chamaremos de RStudio daqui em diante).

Considere, então, os verbetes de “Family Guy” e “The Simpsons” que existem na Wikipedia. Para minha sorte, Peter Meissner criou o pacote *wikipediatrend*<sup>3</sup>, que nos permite obter, diretamente da rede mundial de computadores, as visualizações diárias dos verbetes da enciclopédia gratuita.

Optei por me limitar aos verbetes da Wikipedia inglesa, embora o programa me permita explorar outras línguas e, portanto, os mesmos verbetes. Como todo usuário de R provavelmente sabe, a melhor maneira de começar a aprender uma técnica nova é por meio da replicação de algum exemplo. Você, caro leitor, provavelmente não sabe nada do R, mas siga este exemplo. Após instalar o pacote *wikipediatrend* (veremos como instalar pacotes logo mais), executei o comando seguinte.

```
|
page_views <-
  wp_trend(
    page = c( "Family_Guy", "The_Simpsons" ) ,
    from = "2008-01-01",
    to   = "2015-01-01",
    lang=c("en"),
    file = "C:/Users/cdshi_000/Documents/Meus Documentos/Meus Documentos/guysimpsons2.csv"
  )
```

O resultado? Algo que começava em 2008 e terminava em 2015. Um conjunto de variáveis.

---

<sup>2</sup> [Nota de Claudio D. Shikida] Agradeço a Leonardo M. Monasterio por me apresentar ao R. Levei muito tempo para entender que seria uma boa ideia seguir pelo menos este conselho dele (um pouco de Econometria mostrava que o melhor seria não seguir seus conselhos mas, neste caso, parece que tivemos um *outlier*). Meu primeiro contato mais formal com R foi facilitado pela profa. Edimeire A. Pinto. Ari Francisco de Araujo Jr me fez tantas perguntas que fui obrigado a melhorar a versão inicial desta apostila (e ainda falta um bocado...). Alunos nem sempre entendem, ao primeiro contato, a utilidade do R. O contato com o RStudio me veio por meio dos meus ex-alunos – e atualmente economistas – Raphael Molina e Lucas Farias. Boa parte desta apostila deve muito aos comentários, dicas e sugestões feitas por alunos do curso ministrado no PPGOM/UFPel, na semana de 22 a 26 de Abril de 2013. Agradeço também ao Carlos Cinelli e ao Vitor Wilher por terem sempre sido entusiastas do uso do R. O entusiasmo de muitos é sempre um estímulo adicional.

<sup>3</sup> Veja este endereço: <https://github.com/petermeissner/wikipediatrend>.

	date	count	lang	page	rank	month	title
1	2014-12-30	4599	en	Family_Guy	1033	201412	Family_Guy
2	2014-12-30	4888	en	The_Simpsons	847	201412	The_Simpsons
3	2014-12-31	4047	en	Family_Guy	1033	201412	Family_Guy
4	2014-12-31	4511	en	The_Simpsons	847	201412	The_Simpsons
5	2015-01-01	3863	en	Family_Guy	1033	201501	Family_Guy
6	2015-01-01	4247	en	The_Simpsons	847	201501	The_Simpsons

Sem sair de dentro do ambiente do R, reordenei a base de forma a obter as primeiras linhas para um show apenas e o restante para o outro, sempre sob a ordenação das datas.

```
shows<-data.frame(page_views)

head(shows)

library(plyr)
newshows<-arrange(shows,desc(page),date)
head(newshows)
tail(newshows)
```

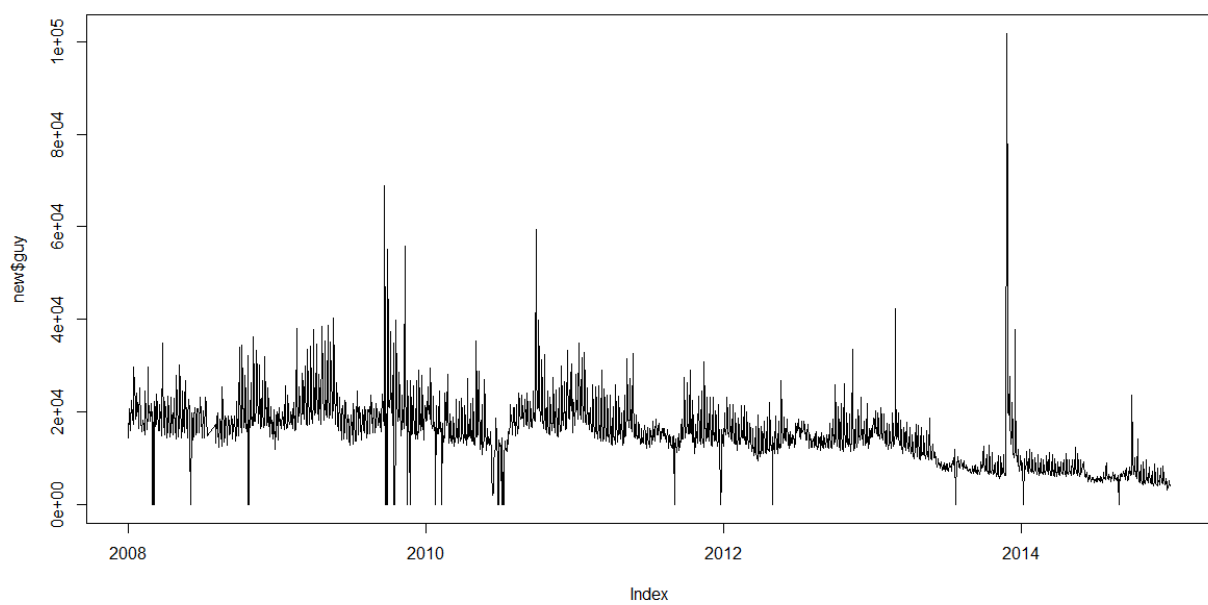
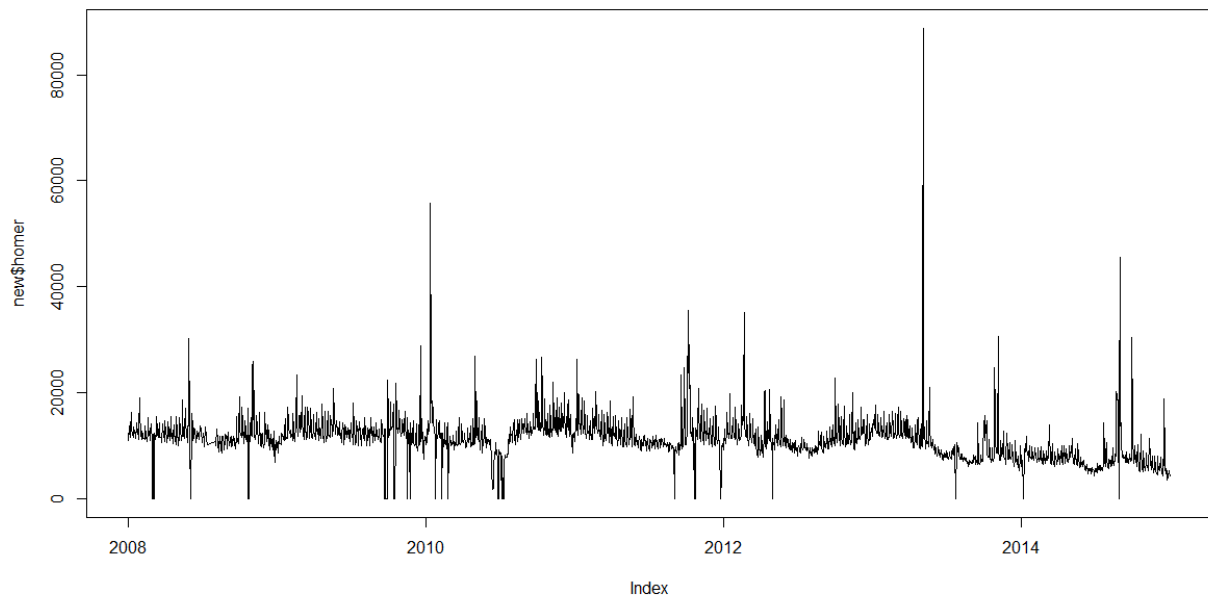
Visualizei meu resultado e descobri onde estava o ponto de corte. Tratava-se da linha de número 2537. Assim, dividi a base e criei uma nova base de dados apenas com a data e as visualizações diárias de cada show.

```
family<-newshows[2538:5074,1:7]
simpsons <-newshows[1:2537, 1:7]
head(family)

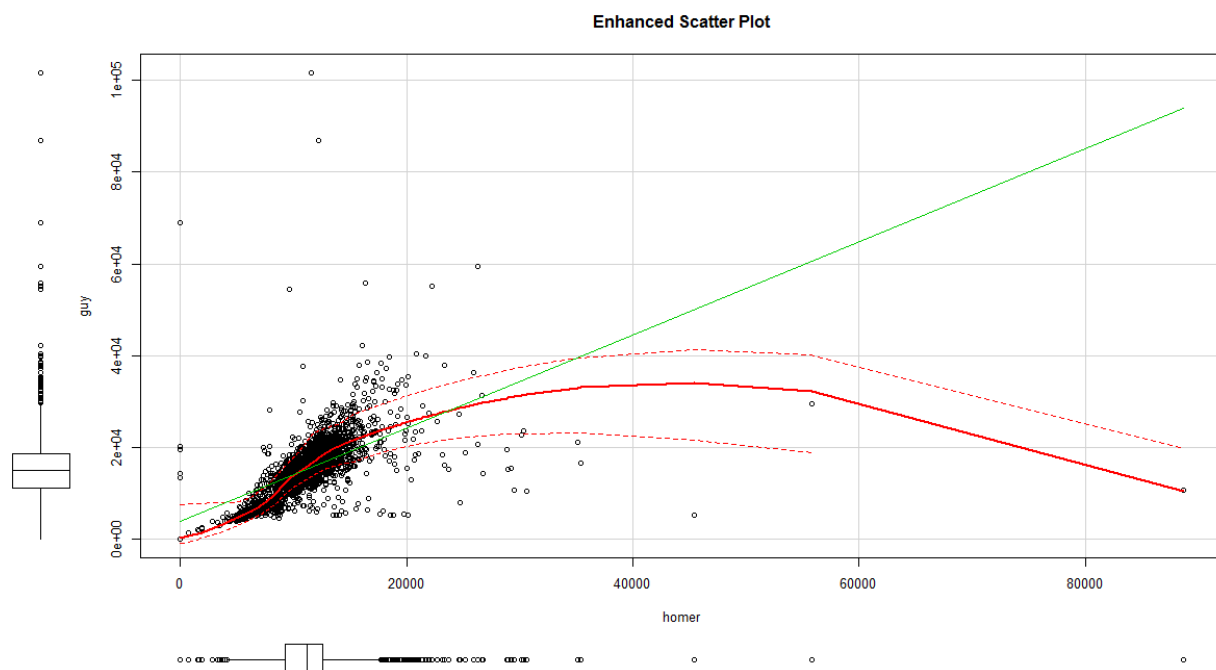
guy<-family$count
homer<-simpsons$count
time<-family$date
both<-data.frame(time, guy, homer)
head(both)

new<-read.zoo(both, format = "%Y-%m-%d")
head(new)
tail(new)
```

Confuso? Ao final deste minicurso você deverá estar pronto para começar uma análise como esta. Ah sim, vejamos como são as visualizações.



Parecem correlacionadas? Tudo bem, podemos tentar dar uma olhada nisto.



O cálculo da correlação indicou algo em torno de 0.60 mas, como sabemos “correlação não faz verão”, ou melhor, correlação não necessariamente implica causalidade<sup>4</sup>. Então, como um bom aluno, eu resolvi usar irresponsavelmente – como tantos – o teste de causalidade de Granger. Minha pergunta era mais ou menos esta: “será que as visualizações na página dos *Simpsons* aumentam/diminuem as visualizações na página do *Family Guy*?”

Ciente de que as visualizações na *Wikipedia* seriam uma *proxy* imperfeita da popularidade de cada show, mas ansioso para aprender a usar o *wikipediatrend*, prossegui. Após invocar um dos vários pacotes do R que fazem o teste de causalidade de Granger, experimentei três especificações, aleatoriamente, com defasagens de ordem 4, 8 e 12. O que eu encontrei?

<sup>4</sup> <https://xkcd.com/552/>.

```

> grangertest(diff(guy) ~diff(homer), order=4, data=new)
Granger causality test

Model 1: diff(guy) ~ Lags(diff(guy), 1:4) + Lags(diff(homer), 1:4)
Model 2: diff(guy) ~ Lags(diff(guy), 1:4)
      Res.Df Df      F Pr(>F)
1      2523
2      2527 -4  0.9049 0.4601
> grangertest(diff(homer) ~ diff(guy) , order=4, data=new)
Granger causality test

Model 1: diff(homer) ~ Lags(diff(homer), 1:4) + Lags(diff(guy), 1:4)
Model 2: diff(homer) ~ Lags(diff(homer), 1:4)
      Res.Df Df      F Pr(>F)
1      2523
2      2527 -4  4.4765 0.00132 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> grangertest(diff(guy) ~diff(homer), order=8, data=new)
Granger causality test

Model 1: diff(guy) ~ Lags(diff(guy), 1:8) + Lags(diff(homer), 1:8)
Model 2: diff(guy) ~ Lags(diff(guy), 1:8)
      Res.Df Df      F Pr(>F)
1      2511
2      2519 -8  1.4167 0.1841
> grangertest(diff(homer) ~ diff(guy) , order=8, data=new)
Granger causality test

Model 1: diff(homer) ~ Lags(diff(homer), 1:8) + Lags(diff(guy), 1:8)
Model 2: diff(homer) ~ Lags(diff(homer), 1:8)
      Res.Df Df      F Pr(>F)
1      2511
2      2519 -8 18.898 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> grangertest(diff(guy) ~diff(homer), order=12, data=new)
Granger causality test

Model 1: diff(guy) ~ Lags(diff(guy), 1:12) + Lags(diff(homer), 1:12)
Model 2: diff(guy) ~ Lags(diff(guy), 1:12)
      Res.Df Df      F Pr(>F)
1      2499
2      2511 -12 1.0768 0.3753
> grangertest(diff(homer) ~ diff(guy) , order=12, data=new)
Granger causality test

Model 1: diff(homer) ~ Lags(diff(homer), 1:12) + Lags(diff(guy), 1:12)
Model 2: diff(homer) ~ Lags(diff(homer), 1:12)
      Res.Df Df      F Pr(>F)
1      2499
2      2511 -12 13.139 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```



Em todas as especificações, verifiquei que o meu show favorito, *Family Guy*, (Granger-)causava o *The Simpsons*. Fiquei intrigado com o resultado mas, claro, não fiz meu dever de casa porque...não havia nenhum. Não era um exercício de Econometria!<sup>5</sup>

Mesmo sem ter a obrigação de apresentar um relatório ou uma lista de exercícios resolvida, eu passei um bom tempo fazendo esta “brincadeira”. Note também que não saí do ambiente do R em momento algum. Sim, eu poderia ter salvo os resultados coletados em uma planilha para, depois, fazer a leitura da mesma pelo R. Entretanto, nem isso foi necessário. Também é importante perceber outra característica do R: muitas das dúvidas que tive foram resolvidas *online* (sim, não dá para trabalhar com o R sem internet, a não ser que você seja muito bom na linguagem e/ou tenha uma boa quantidade de livros sobre os assuntos que deseja estudar por perto<sup>6</sup>).

Divertido, não? Ah sim, o que foi que Homer disse a Peter? Várias coisas, inclusive esta<sup>7</sup>:

*[The spaceship jumps over Springfield Gorge]*

**Peter:** We're gonna make it!

**Homer:** [Trust me, we're not.](#)

Você, que começa a usar o R agora, tal como você, que agora começa o mestrado em Economia, ou se prepara para sua primeira prova, ou, claro, você que está prestes a estimar os primeiros modelos de sua dissertação, reflita sobre esta frase de uma forma positiva.

Vamos ao R!

---

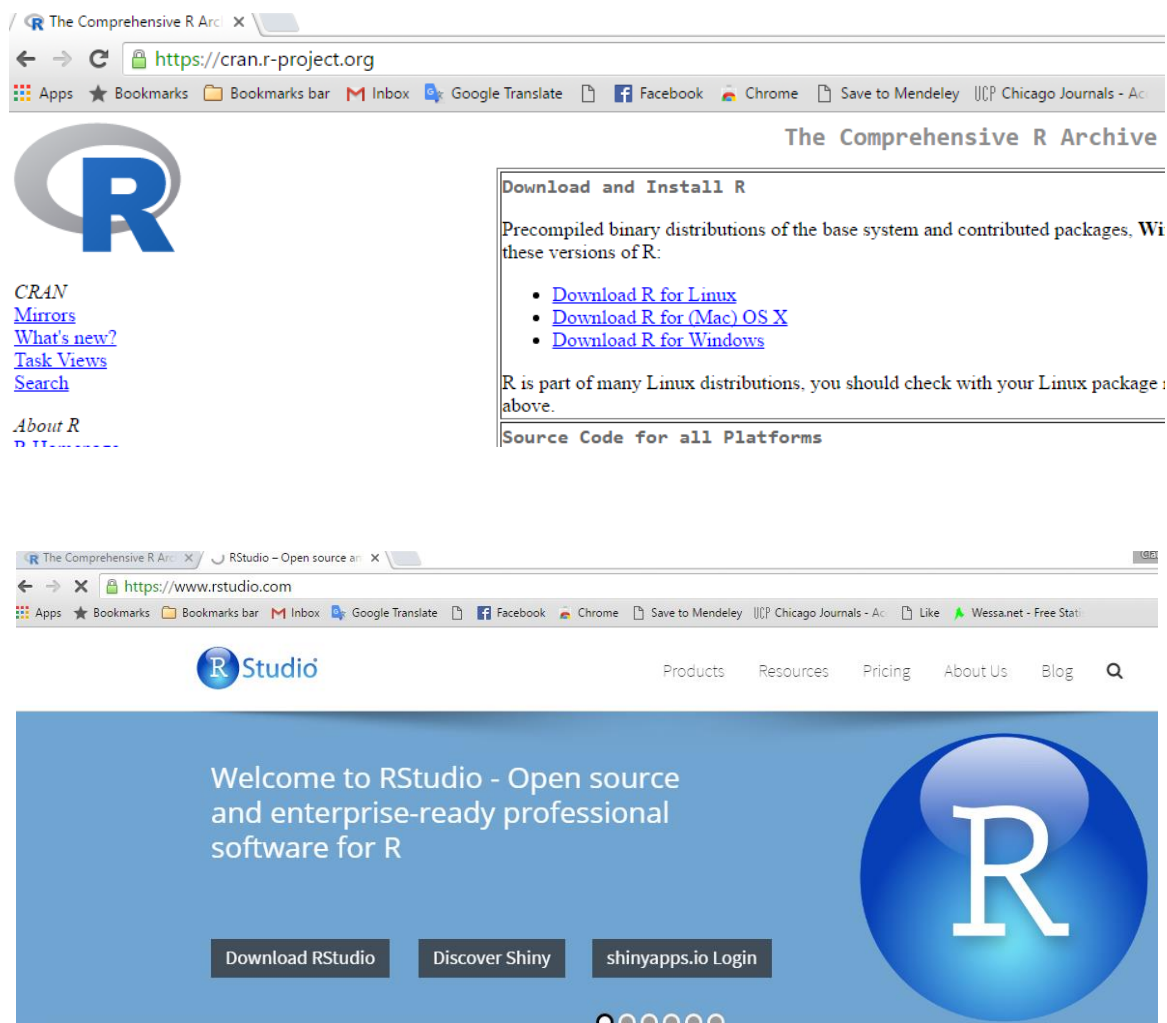
<sup>5</sup> Caso fosse, alguém já estaria levantando a bandeira dos testes de raiz unitária, não sem alguma razão.

<sup>6</sup> Morar em uma biblioteca poderia ser a solução, não é?

<sup>7</sup> Originalmente em: [http://familyguy.wikia.com/wiki/The\\_Simpsons\\_Guy/Quotes](http://familyguy.wikia.com/wiki/The_Simpsons_Guy/Quotes).

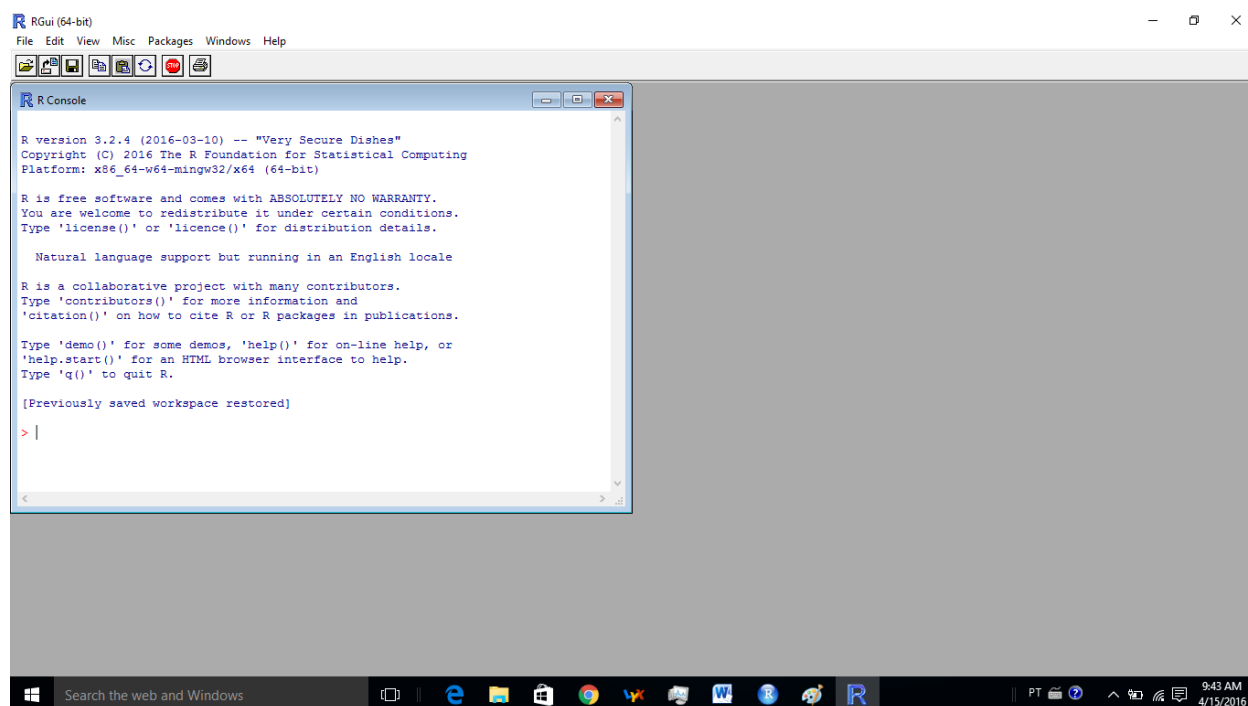
## 1. Cheguei ao laboratório, mas não sei o que se passa. Alguém pode me ajudar?

Para iniciar sua aventura pelo mundo da Econometria em R, vamos nos lembrar que utilizaremos o RStudio. Logo, precisamos instalar o R e o RStudio. A primeira coisa a se fazer é consultar os *sites* seguintes: <http://cran.r-project.org> e <http://www.rstudio.com>. Do primeiro se instala a versão do R mais adequada ao seu computador. Do segundo, a mesma coisa para o RStudio. Você deve ter encontrado as seguintes telas.



Digamos que você já está em um laboratório com ambos os programas instalados. Primeiramente, agradeça do fundo do seu coração aos bravos estagiários, bolsistas ou monitores dos professores que investiram horas de seu tempo para fazer as instalações. Eles merecem.

Mas, existe um *catch* aqui. Será que as versões dos programas são as mais recentes? É aqui que iniciaremos nossa primeira lição de R. Clique no ícone do programa e abra a tela do R. Você deve ter encontrado algo assim.



Vamos atualizar a versão do R que você acaba de encontrar em seu computador e vamos fazer isto de forma simples, usando o programa *installr*. Vá a este endereço: <https://github.com/talgali/installr/>. Vamos copiar e colar as instruções na tela do R. Na página do *installr*, copie o trecho seguinte.

## Installation

To install the stable version on CRAN:

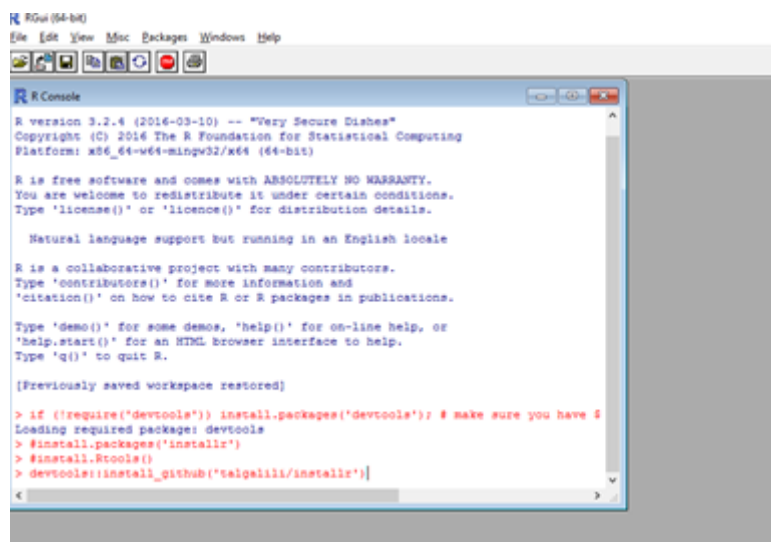
```
install.packages('installr')
```

To install the latest installr version from GitHub use:

```
if (!require('devtools')) install.packages('devtools'); # make sure you have Rtools installed first! if not, then run  
#install.packages('installr')  
#install.Rtools()  
devtools::install_github('talgalili/installr')
```

## Usage

Ao colar na janela do R que vimos acima (trata-se da janela *console*), você notará que ele já iniciará a execução dos comandos. Repare a posição do cursor na figura abaixo. Na última linha, o cursor encontra-se pronto para a execução do comando de atualização. Vamos apertar a tecla *Enter* e a mágica começará.



```
R GUI (64-bit)
File Edit View Misc Packages Windows Help

R Console
R version 3.2.4 (2016-03-10) -- "Very Secure Dishes"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

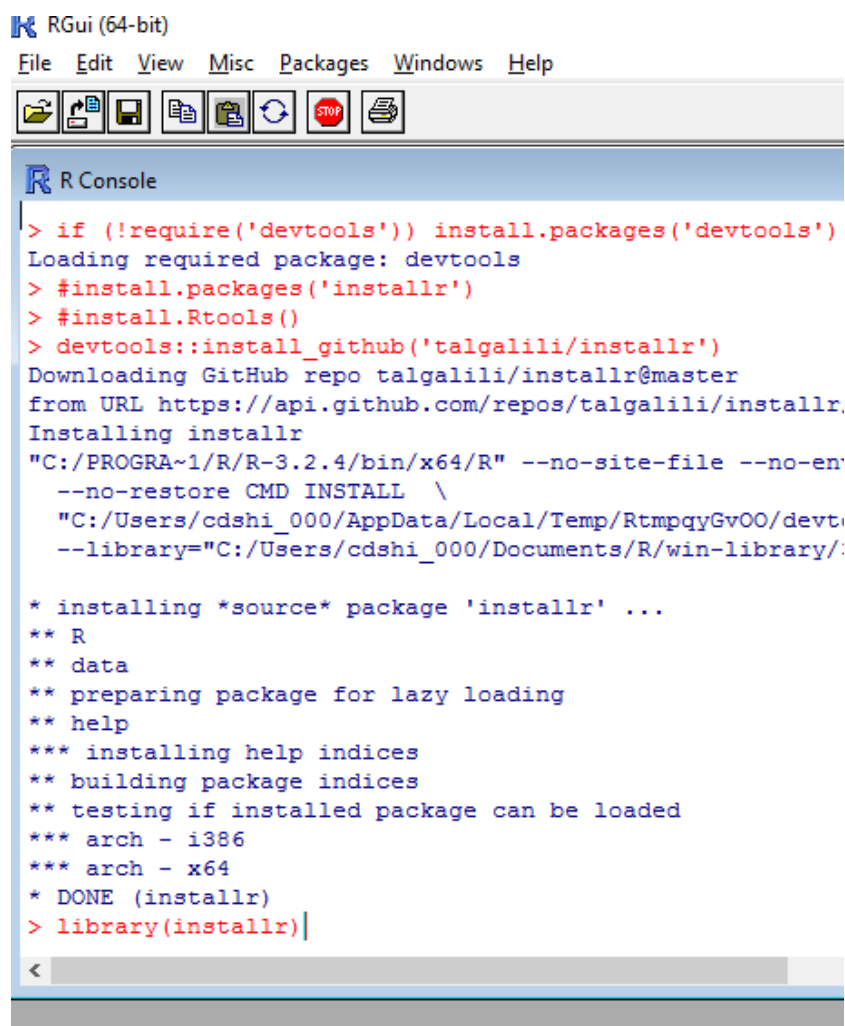
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> if (!require("devtools")) install.packages("devtools"); # make sure you have $
Loading required package: devtools
> #install.packages("installr")
> #install.Rtools()
> devtools::install_github("rsgalili/installr")
```

A primeira lição sobre o R é que os programas que executamos nele são chamados de *libraries* (bibliotecas). Em geral, eles estão disponíveis em um repositórios do R pelo mundo mas, muitos autores disponibilizam versões mais recentes em outros locais – a esmagadora maioria no *github* – como é o caso do *installr*. O que o *installr* faz é facilitar nossa vida com um dos mais penosos trabalhos que se executa em R: sua atualização.

Uma vez que o programa está em R, ele deve ser chamado (ou invocado). Faz-se isto com o comando *library(nome do fulano)*. Em nosso caso:



```

RGui (64-bit)
File Edit View Misc Packages Windows Help

> if (!require('devtools')) install.packages('devtools')
Loading required package: devtools
> #install.packages('installr')
> #install.Rtools()
> devtools::install_github('talgalili/installr')
Downloading GitHub repo talgalili/installr@master
from URL https://api.github.com/repos/talgalili/installr
Installing installr
"C:/PROGRA~1/R/R-3.2.4/bin/x64/R" --no-site-file --no-en
--no-restore CMD INSTALL \
"C:/Users/cdshi_000/AppData/Local/Temp/RtmpqyGvOO/devt
--library="C:/Users/cdshi_000/Documents/R/win-library/

* installing *source* package 'installr' ...
** R
** data
** preparing package for lazy loading
** help
*** installing help indices
** building package indices
** testing if installed package can be loaded
*** arch - i386
*** arch - x64
* DONE (installr)
> library(installr)|

```

As rotinas criadas por Ted Galili, autor do *installr* guiarão você pelo processo de instalação/atualização do R. Como o minicurso é feito em um laboratório, precisamos dar atenção à atualização do R. Voltemos à página do *github* em que estávamos e vamos copiar e colar outras linhas de comando.

## Usage

If you are using the Rgui, you will see a new menu added on your top right (just by "help"), giving you the option to update R, or install new software.

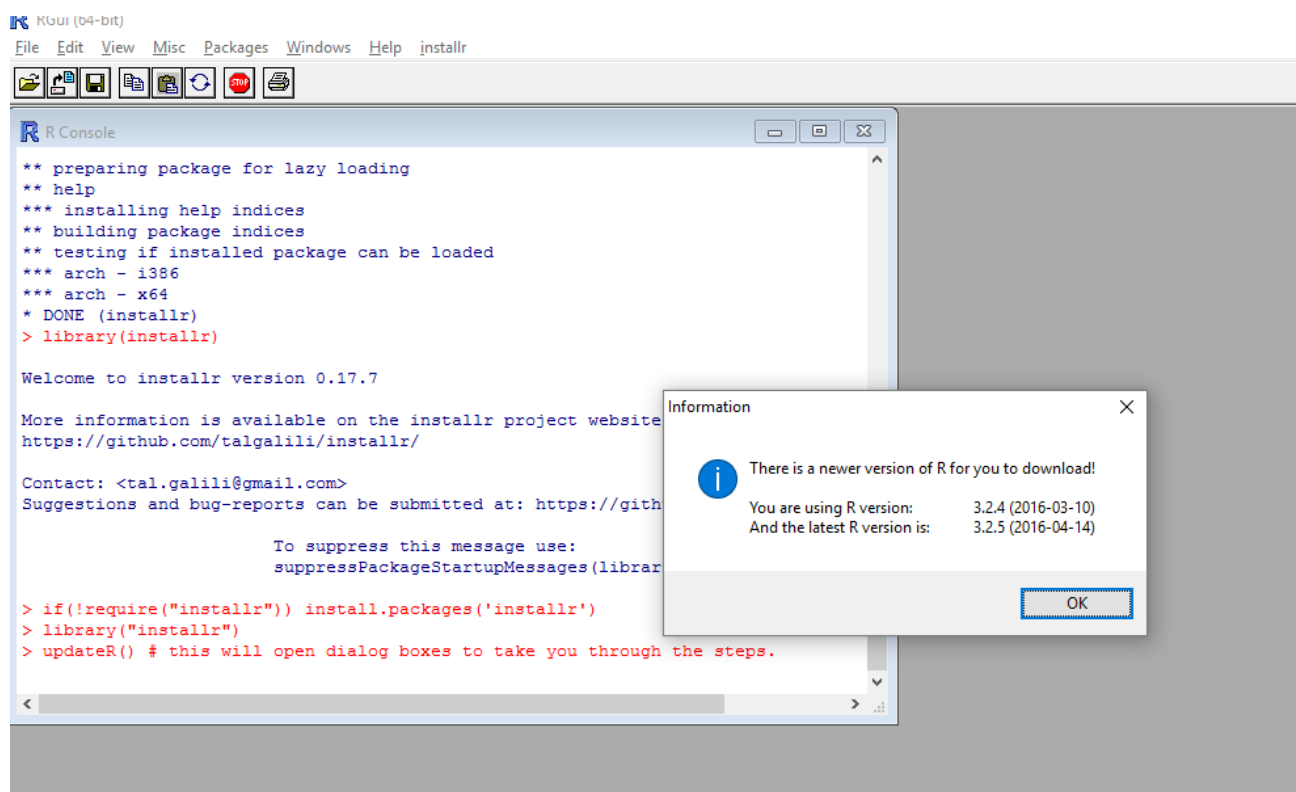
For command line use you can **update R** by running:

```

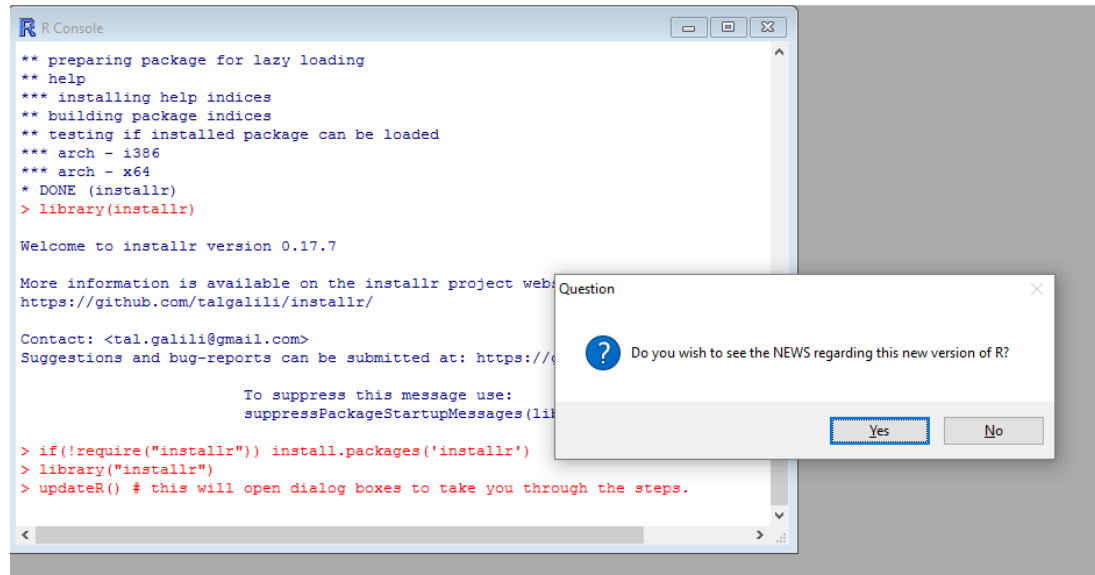
if(!require("installr")) install.packages('installr')
library("installr")
updateR() # this will open dialog boxes to take you through the steps.
# OR use:
# updateR(TRUE) # this will use common defaults and will be the safest/fastest option

```

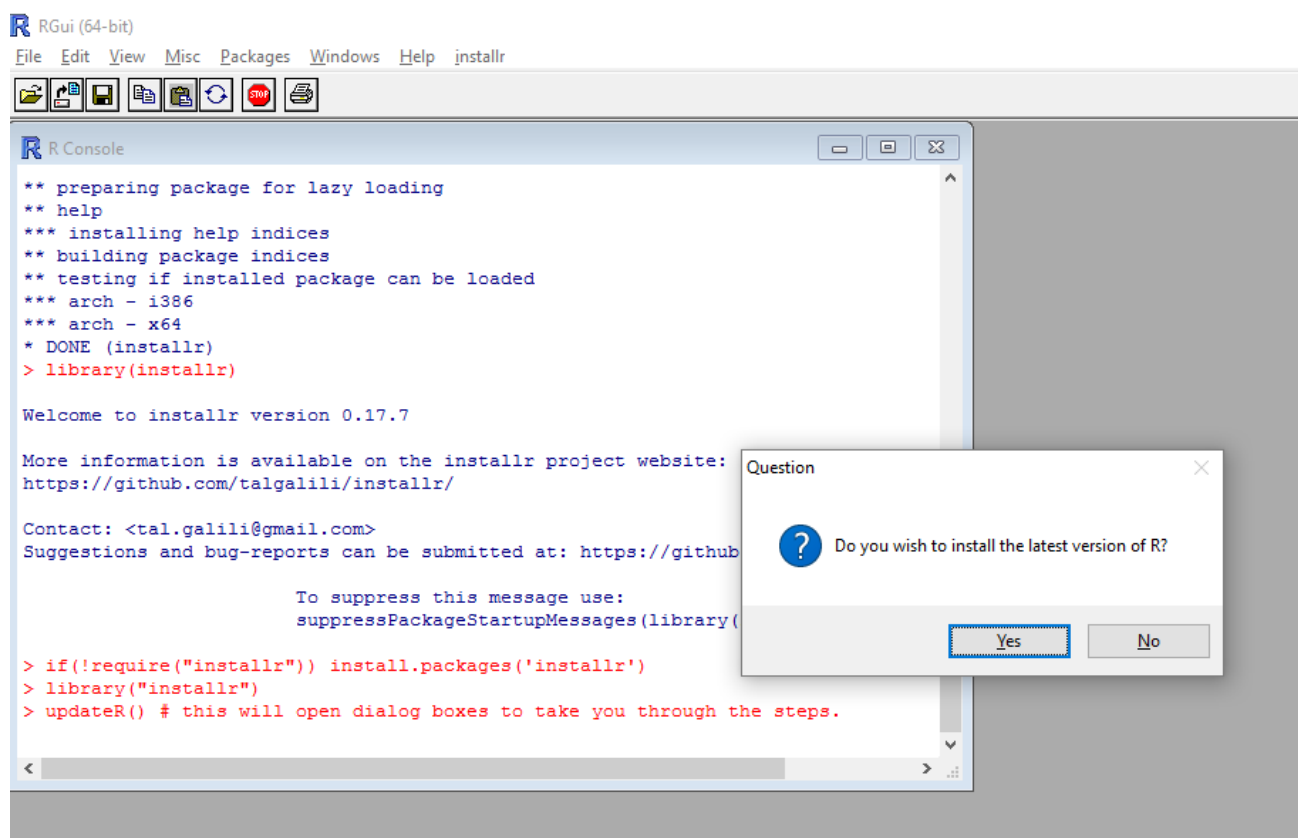
Exato. Copiamos as linhas acima e colamos na mesma janela do R. A execução será imediata. No meu caso, eis a janela que surge (ela poderá ser diferente para você).



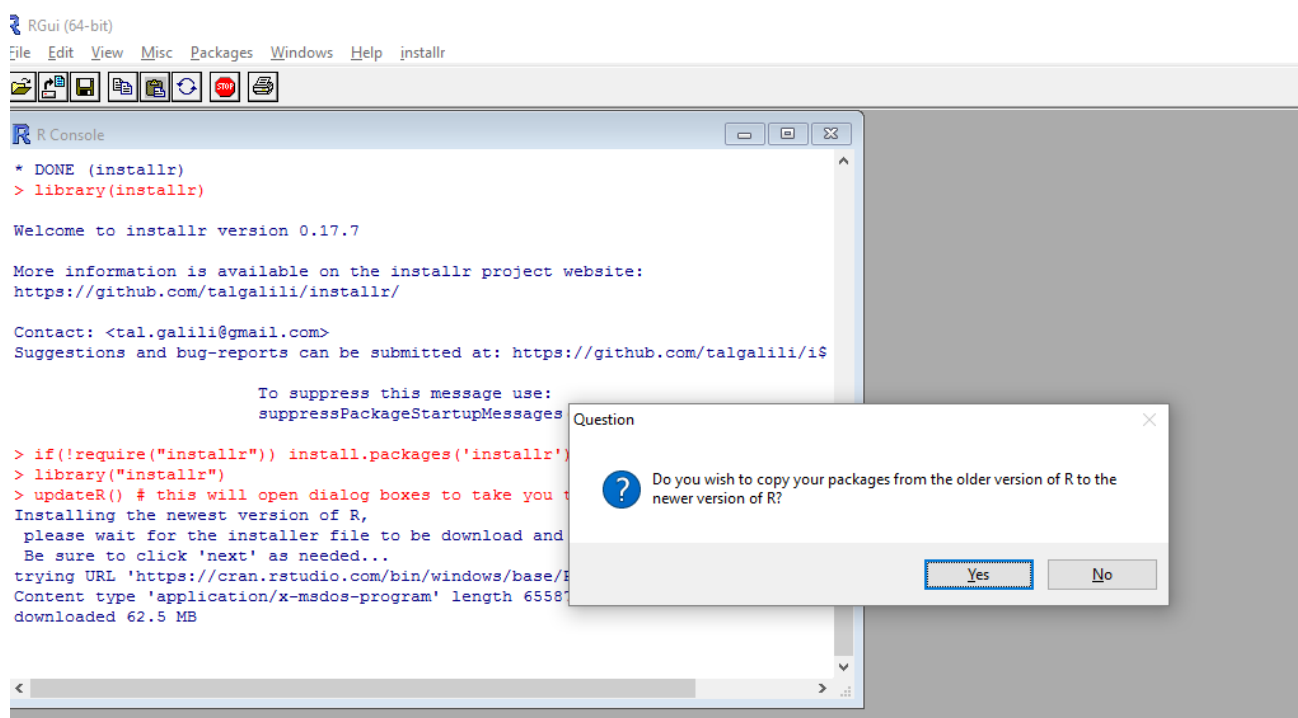
Em seguida, ele perguntará se desejamos saber algo sobre a nova versão do R (caso alguém esteja afim de uma leitura técnica, siga em frente. Não é o meu caso).



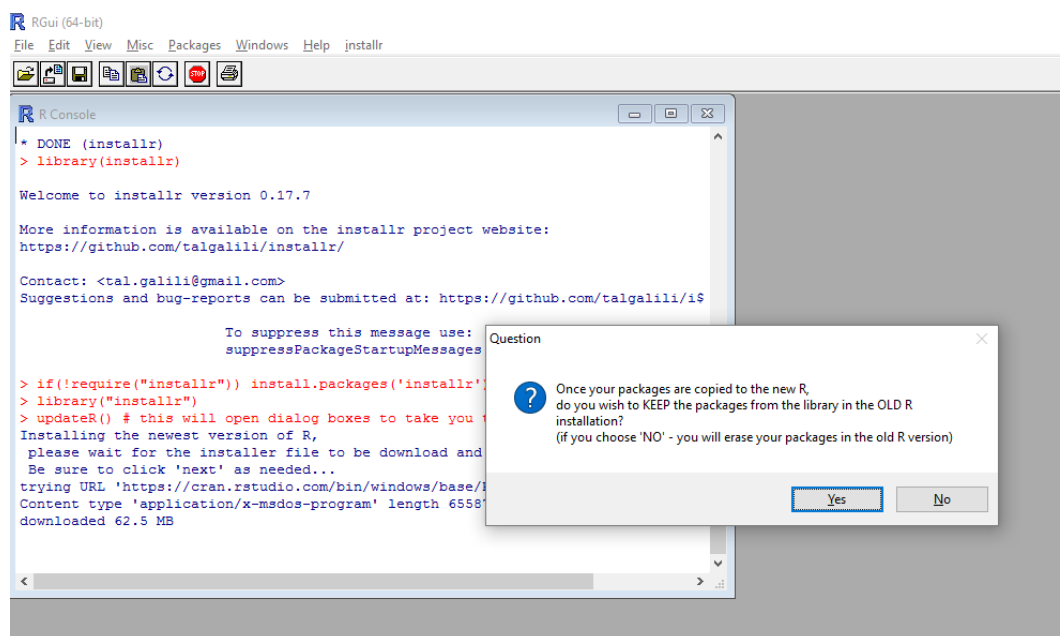
A próxima janela merece um "Yes".



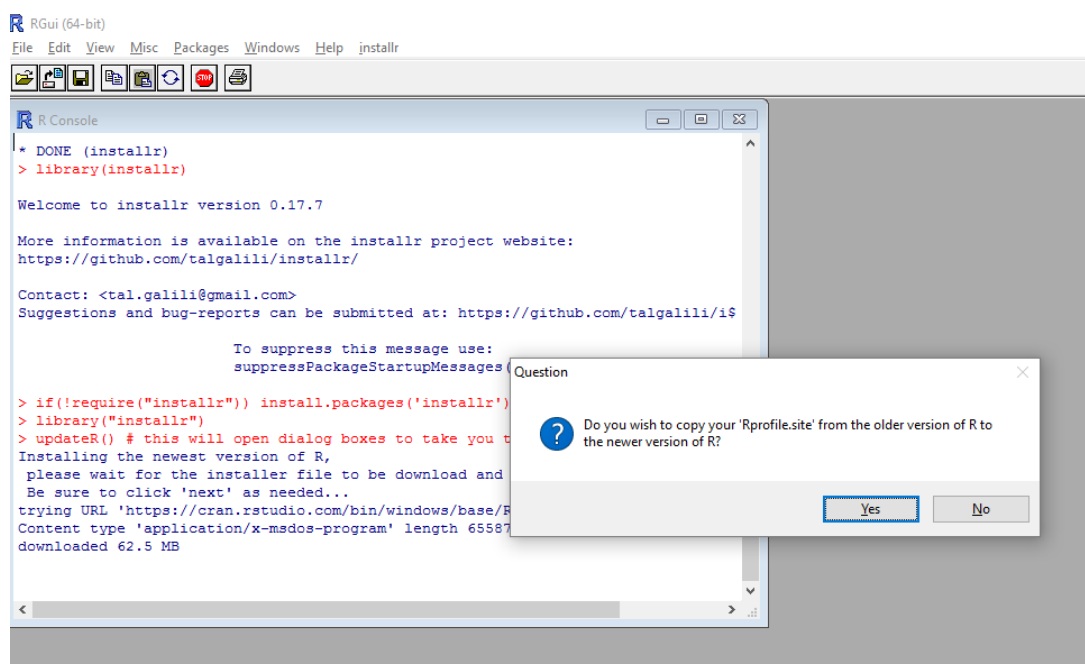
Um ponto importante, enquanto a rotina atua, é notar algumas coisas. Primeiro, nos comandos copiados, alguns deles continham o símbolo “#” no início. O “#” não é um comando. Ele indica que o que vem à sua direita não é executável. Usamos este símbolo para fazermos anotações. A próxima pergunta é muito importante e você dirá “Yes” mesmo que eu não explique o porquê.



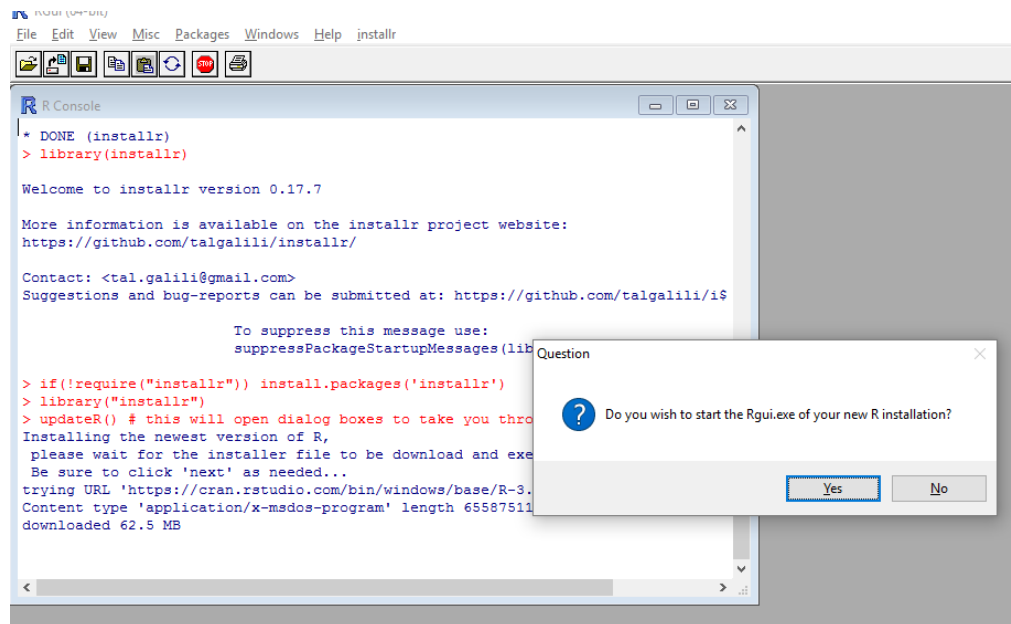
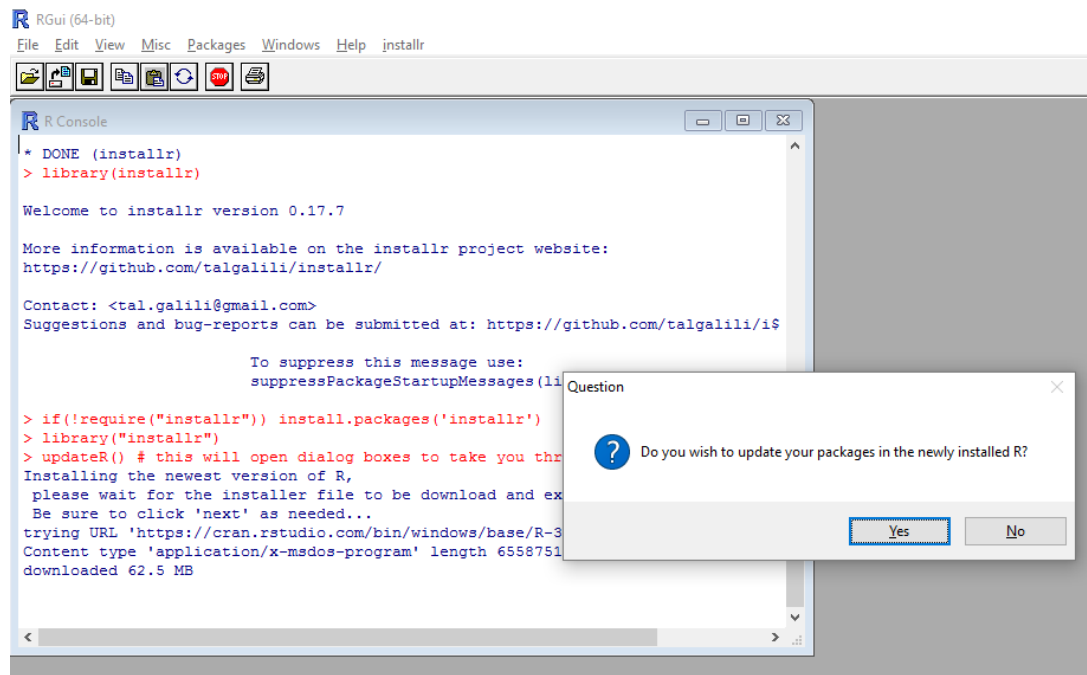
Como o *installr*, tudo o que usamos no R são os pacotes (*libraries*, bibliotecas, doravante chamados de pacotes). Estes são constantemente atualizados por seus autores e, sempre que há uma nova versão do R, há uma atualização para cada pacote. Claro que vamos aceitar. Mais ainda, ele nos perguntará se queremos manter os diretórios de suas antigas versões. Isto depende de cada um. Geralmente eu não faço isto (mas pode ser interessante e, se for o caso, explicarei em aula, ok?).

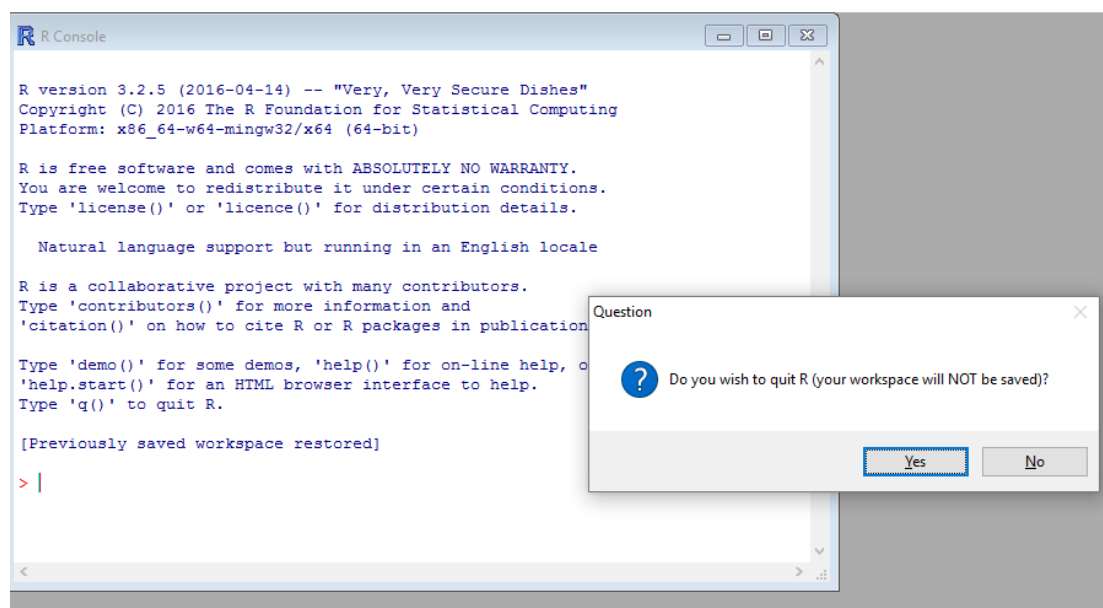


Para as quatro próximas perguntas, a resposta é, claro, “Yes”.

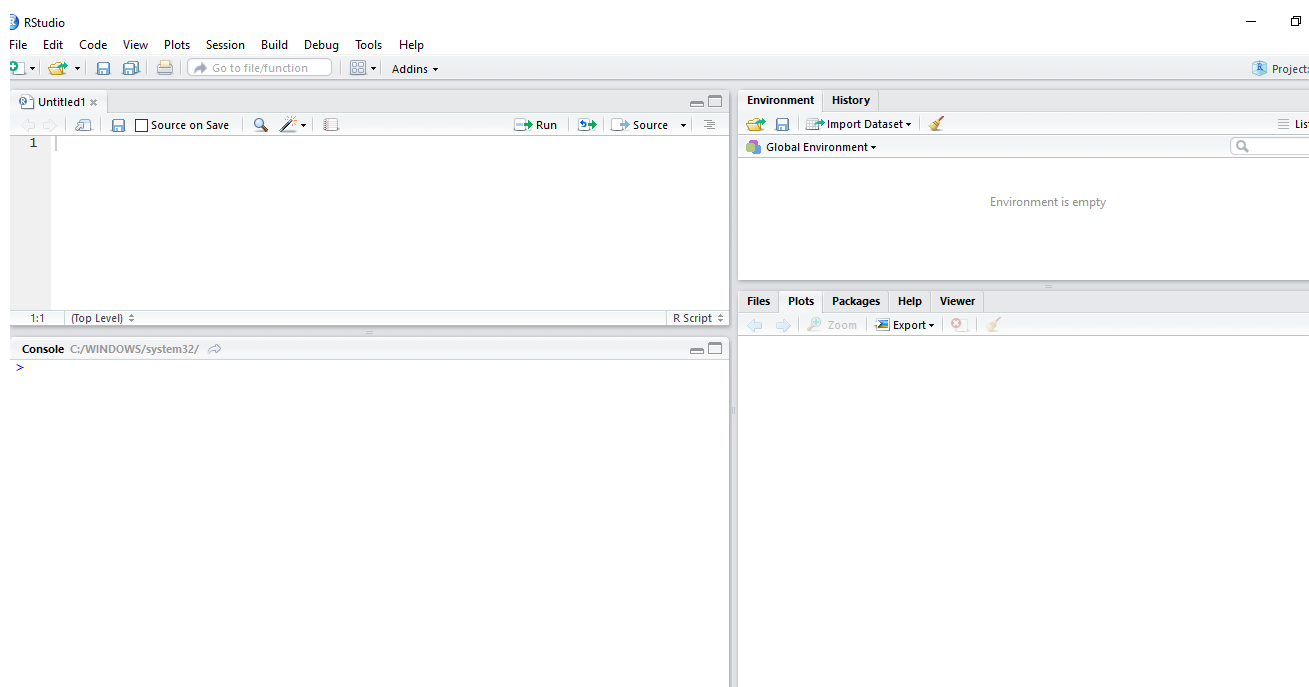








Podemos fechar o R agora. Vamos ao RStudio. Após sua instalação, podemos abri-lo e a visualização será algo como a figura seguinte.



Repare nas subdivisões. A janela do canto superior esquerdo é nosso principal ambiente. Nela criaremos nossos *scripts* (o corresponde ao *do file* do Stata ou ao *prg file* do Eviews ou ao *inp file* do Gretl). Escrever um *script* significa que temos que saber o que desejamos que o R execute e que temos um planejamento prévio, mínimo, sobre o que vamos fazer. Os comandos que copiamos e colamos direto na janela de execução (*console*) do R poderiam ter sido salvos em um *script* para posterior execução. O equivalente ao que foi feito seria colar os comandos na janela inferior à esquerda que, não por coincidência, é a janela de *console*.

No lado direito do painel temos as janelas superior e a inferior. Na primeira, há duas abas, ambas destinadas a nos informar sobre o que obtivemos por meio dos comandos digitados no *script* cuja execução aparecerá no *console*. A janela inferior tem várias abas e veremos mais sobre elas adiante.

Antes de terminarmos, uma breve palavrinha sobre instalações. Em breve será necessário instalar alguns pacotes como o *installR*. Como fazer isso? Você pode fazer isto manualmente. Por exemplo, se quiser se adiantar e instalar vários pacotes ao mesmo tempo, pode fazer:

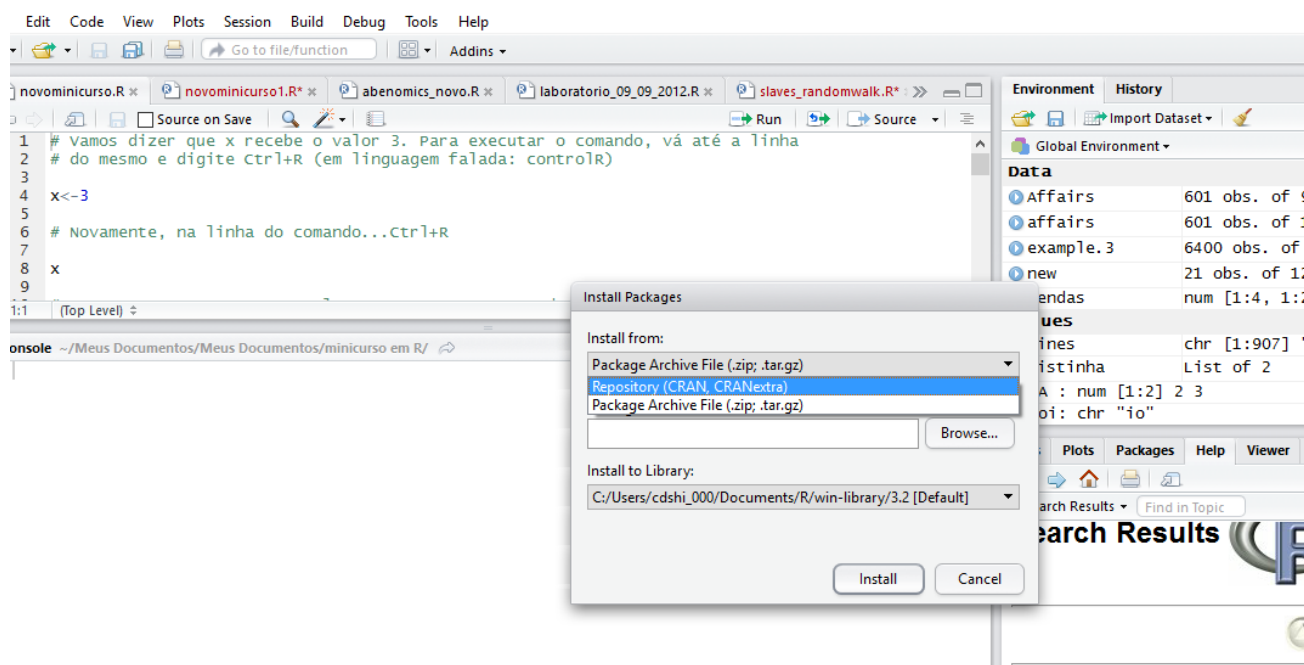
```
install.packages(c("AER", "lmtest", "sem", "car", "ROCR", "MASS", "pscl", "forecast", "astsa", "dynlm", "plm", "sandwich", "ggplot2", "pastecs", "vars", "urca", "strucchange", "foreign", "aod"))
```

Ou pode fazer separadamente. Por exemplo, para os dois primeiros pacotes da lista acima, teríamos:

```
install.package("AER")
```

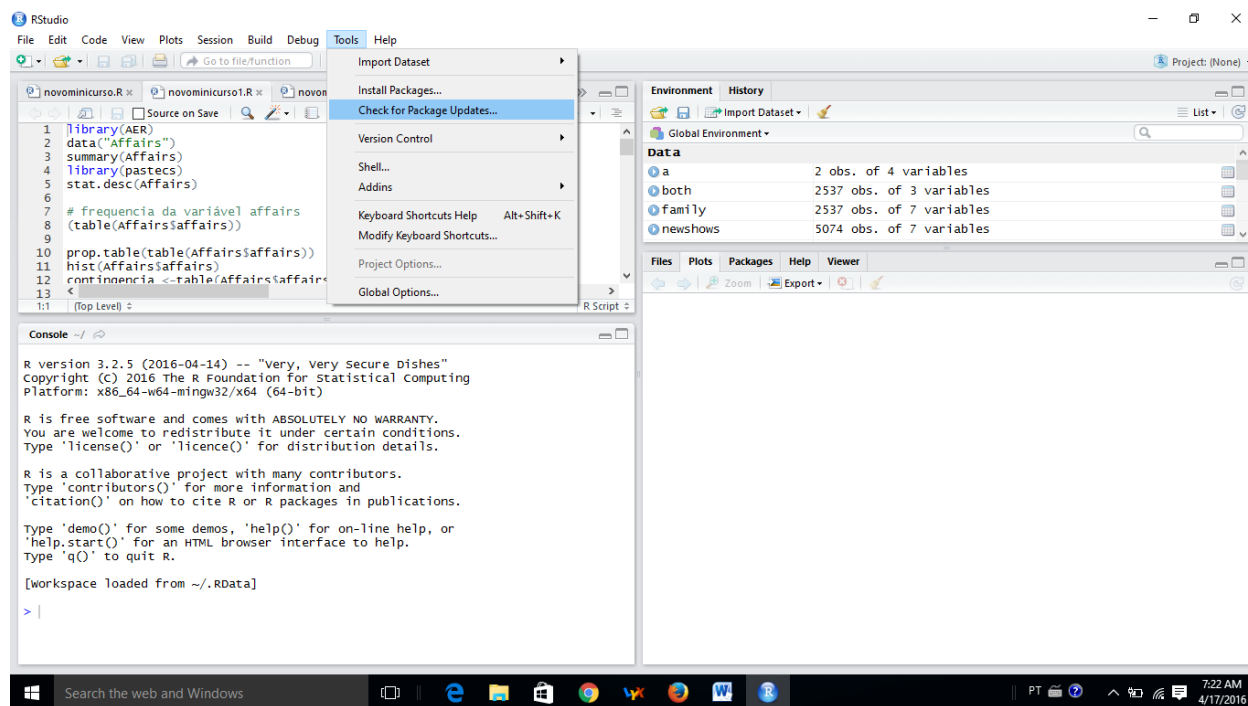
```
install.package("lmtest")
```

Claro, alguém deve lhe dizer que pacotes instalar. Quantos existem? Não faço ideia (...mas a dica é esta: <https://cran.r-project.org/web/views/Econometrics.html>). Ah sim, você pode também usar o *menu*.



Acho que podemos assumir que daqui em diante não é tão difícil, não? Basta escolher um repositório *CRAN* e buscar o nome do pacote. Repare que você também pode obter o arquivo compactado, salvá-lo no computador e, então, usar a segunda opção acima, *Package Archive File*.

É bem provável que você já esteja no laboratório e que seu R já tenha pacotes instalados. Tudo o que você precisa fazer é pedir para verificar se há atualizações dos mesmos. O resto do processo é simples.



### Box Necessário (mas não Suficiente)

Um aspecto interessante do uso do R é quando descobrimos a quantidade de recursos *online* para nos ajudar. Devo mencionar a comunidade do [StackOverflow](#) (ou [StackExchange](#), *nunca sei direito*) e o leitor poderá encontrar, facilmente, dicas *online* em forma de vídeo-aulas, cursos gratuitos ou pagos (como o do Vitor Wilher), ou mesmo em blogs (Carlos Cinelli tem disponibilizado [versões iniciais dos capítulos de seu livro em seu blog, por exemplo](#)). O prof. Rob Hyndman disponibilizou um excelente tutorial básico de séries de tempo [aqui](#). Os tutoriais da UCLA são fontes obrigatórias para quem usa [Stata](#) ou [R](#). Veja também [R for dummies](#), [Inside-R<sup>8</sup>](#), [Quick-R](#), por exemplo.

Certamente eu me esqueci de alguns *sites*, mas sugestões são sempre bem-vindas.

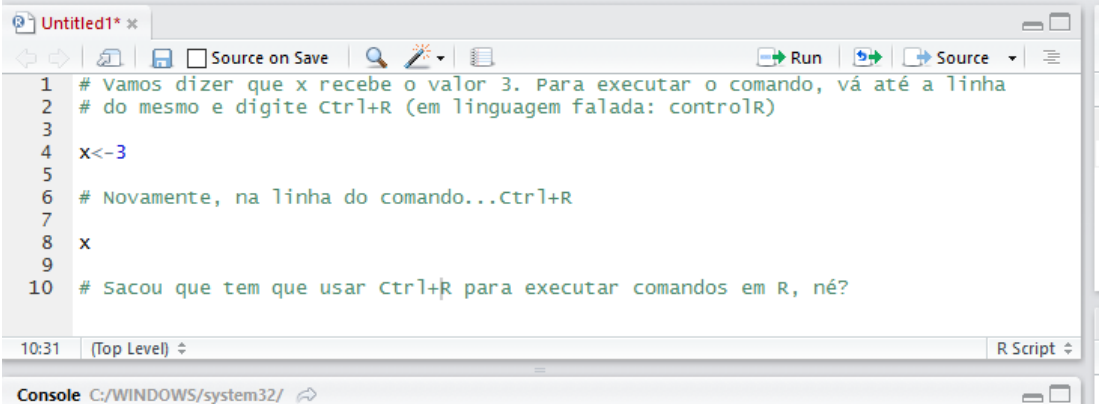
Você está de parabéns. Encerrou a primeira sessão inicial deste mini-curso. Pegue seu vale para um café.

<sup>8</sup> A partir de algum momento em 2016, este *link* foi substituído pelo da versão aberta do R oferecida pela Microsoft: <https://mran.microsoft.com/>.

## 2. Impressione seus amigos e aquela garota bonita!

Ok, você está ansioso(a) para mostrar ao mundo que já é um iniciado em R. Vamos do início então. Começemos com os *objetos* que podem ser manipulados no ambiente R. Temos números, vetores, matrizes, textos, gráficos, etc. Vejamos alguns exemplos que podem não parecer muito úteis agora, mas que eventualmente vão te ajudar no futuro.

Digamos que eu queira gerar um objeto “x” que é igual a “3”. Depois quero mostrá-lo a você. Não satisfeito, quero também mostrar que sou sofisticado e que sei fazer contas. Vamos ver se você consegue impressionar seus amigos (e aquela garota bonita) com seus conhecimentos em R. Então digite os comandos na janela de *scripts* (não precisa digitar os textos). Para executar os comandos, há duas opções: um por vez ou todos de uma vez só. No segundo caso, marque toda a janela (Ctrl+a) e execute (Ctrl+R). No primeiro, basta ir até a linha do comando e executar. Veja o exemplo.



```
1 # Vamos dizer que x recebe o valor 3. Para executar o comando, vá até a linha
2 # do mesmo e digite Ctrl+R (em linguagem falada: controlR)
3
4 x<-3
5
6 # Novamente, na linha do comando...Ctrl+R
7
8 x
9
10 # Sacou que tem que usar Ctrl+R para executar comandos em R, né?
```

10:31 (Top Level) R Script

Console C:/WINDOWS/system32/

Os resultados dos comandos aparecem na janela inferior.

```

RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function Addins
Untitled1*
1 # Vamos dizer que x recebe o valor 3. Para executar o comando, vá até a linha
2 # do mesmo e digite Ctrl+R (em linguagem falada: controlR)
3
4 x<-3
5
6 # Novamente, na linha do comando...Ctrl+R
7
8 x
9
10 # Sacou que tem que usar Ctrl+R para executar comandos em R, né?
8:2 (Top Level) R Script
Console C:/WINDOWS/system32/
> # vamos dizer que x recebe o valor 3. Para executar o comando, vá até a linha
> # do mesmo e digite Ctrl+R (em linguagem falada: controlR)
>
> x<-3
>
> # Novamente, na linha do comando...Ctrl+R
>
> x
[1] 3
>
> # Sacou que tem que usar Ctrl+R para executar comandos em R, né?
>

```

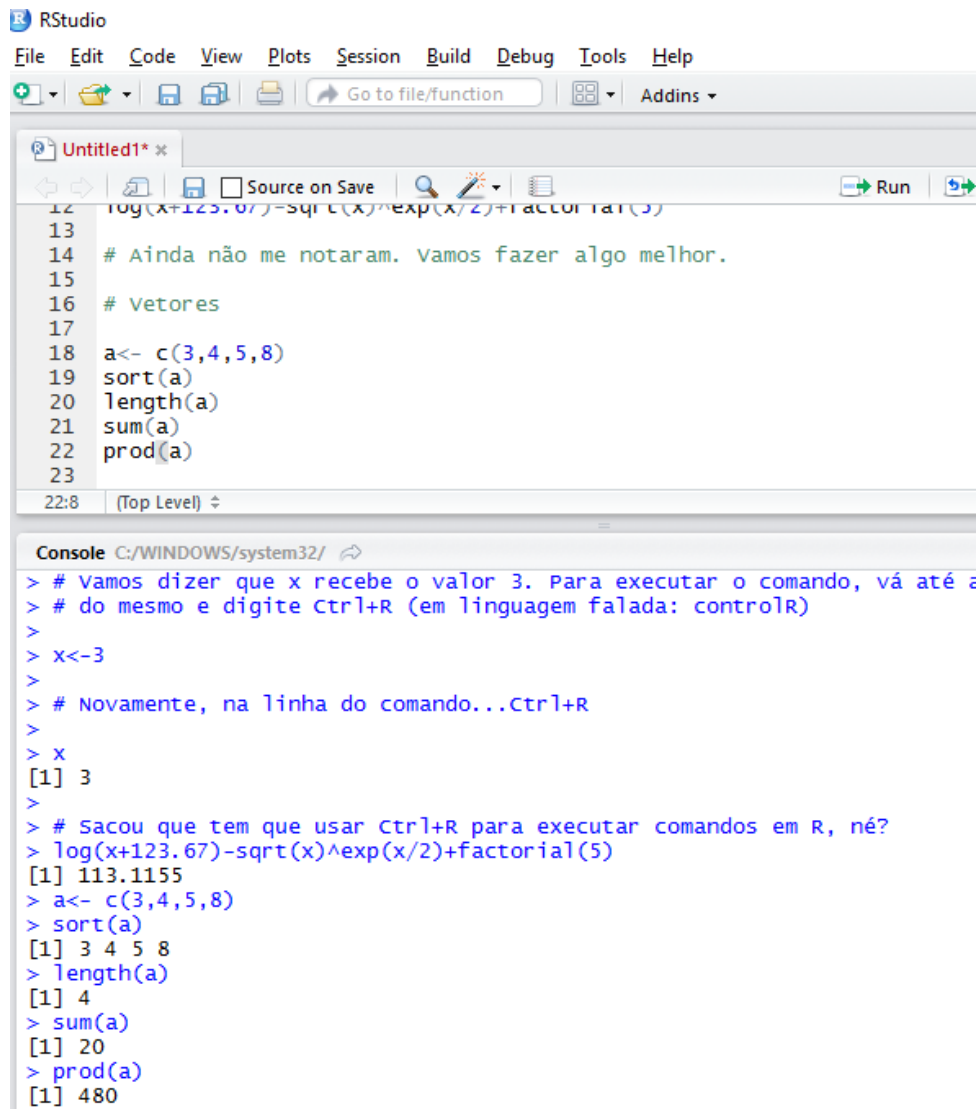
Ainda não impressionou ninguém, né? Ok, mais alguns comandos.

```

RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function Addins
Untitled1*
2 # do mesmo e digite Ctrl+R (em linguagem falada: controlR)
3
4 x<-3
5
6 # Novamente, na linha do comando...Ctrl+R
7
8 x
9
10 # Sacou que tem que usar Ctrl+R para executar comandos em R, né?
11
12 log(x+123.67)-sqrt(x)^exp(x/2)+factorial(5)
13
13:1 (Top Level) R Script
Console C:/WINDOWS/system32/
> # vamos dizer que x recebe o valor 3. Para executar o comando, vá até a linha
> # do mesmo e digite Ctrl+R (em linguagem falada: controlR)
>
> x<-3
>
> # Novamente, na linha do comando...Ctrl+R
>
> x
[1] 3
>
> # Sacou que tem que usar Ctrl+R para executar comandos em R, né?
> log(x+123.67)-sqrt(x)^exp(x/2)+factorial(5)
[1] 113.1155
>

```

Acho que agora você já começou a ser notado. Entretanto, tal como na vida, nenhuma garota quer saber de um cara que só sabe aritmética ou recitar números com quatro casas decimais. Vejamos, então, algo mais interessante.



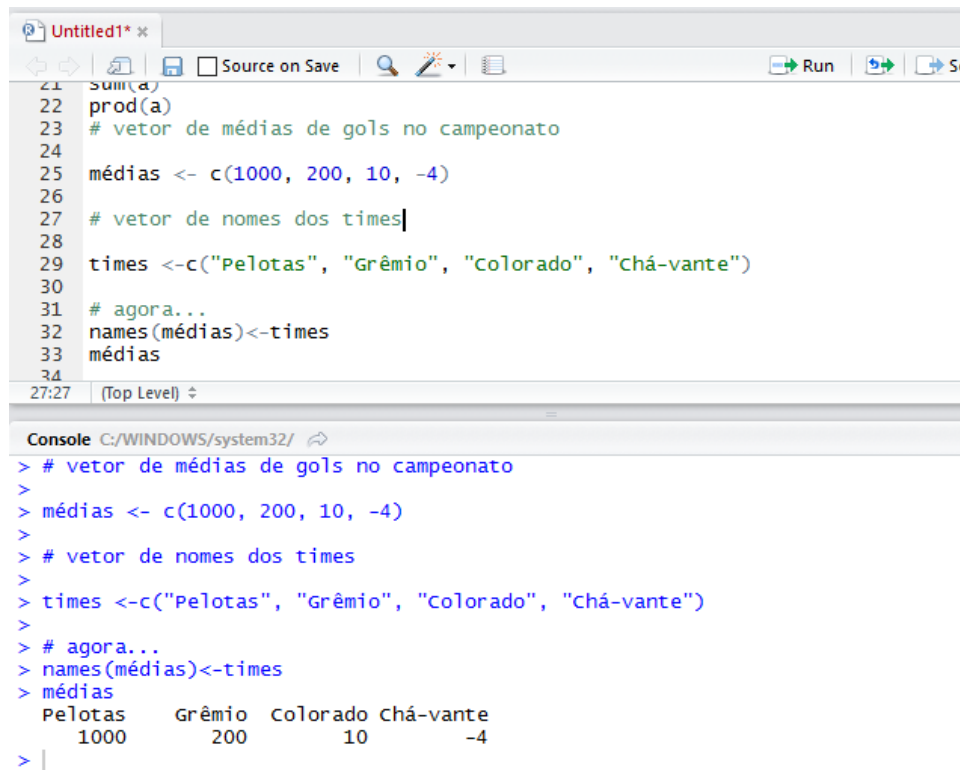
The screenshot shows the RStudio environment. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Tools, and Help. Below the menu is a toolbar with icons for file operations and a 'Go to file/function' search bar. The main editor window, titled 'Untitled1\*', contains R code. The code starts with a mathematical expression on line 12, followed by a comment on line 14, and then a series of vector operations on lines 16 through 23. The console window at the bottom shows the execution of these commands, with the prompt '>' and the output of each command. The console path is 'C:/WINDOWS/system32/'.

```
12 log(x+123.67)-sqrt(x)^exp(x/2)+factorial(5)
13
14 # Ainda não me notaram. Vamos fazer algo melhor.
15
16 # Vetores
17
18 a<- c(3,4,5,8)
19 sort(a)
20 length(a)
21 sum(a)
22 prod(a)
23
```

Console C:/WINDOWS/system32/

```
> # Vamos dizer que x recebe o valor 3. Para executar o comando, vá até a
> # do mesmo e digite Ctrl+R (em linguagem falada: controlR)
>
> x<-3
>
> # Novamente, na linha do comando...Ctrl+R
>
> x
[1] 3
>
> # Sacou que tem que usar Ctrl+R para executar comandos em R, né?
> log(x+123.67)-sqrt(x)^exp(x/2)+factorial(5)
[1] 113.1155
> a<- c(3,4,5,8)
> sort(a)
[1] 3 4 5 8
> length(a)
[1] 4
> sum(a)
[1] 20
> prod(a)
[1] 480
```

Ok, ainda não é muito. Vamos adaptar um exemplo de Heiss (2016) para uma base de dados hipotética.



The screenshot shows the RStudio interface. The top pane displays R code for creating a vector of goals and assigning team names to it. The bottom pane shows the console output, which includes the execution of the code and a printed table of the resulting vector.

```
21 sum(a)
22 prod(a)
23 # vetor de médias de gols no campeonato
24
25 médias <- c(1000, 200, 10, -4)
26
27 # vetor de nomes dos times|
28
29 times <-c("Pelotas", "Grêmio", "Colorado", "Chá-vante")
30
31 # agora...
32 names(médias)<-times
33 médias
34
```

27:27 (Top Level) ▾

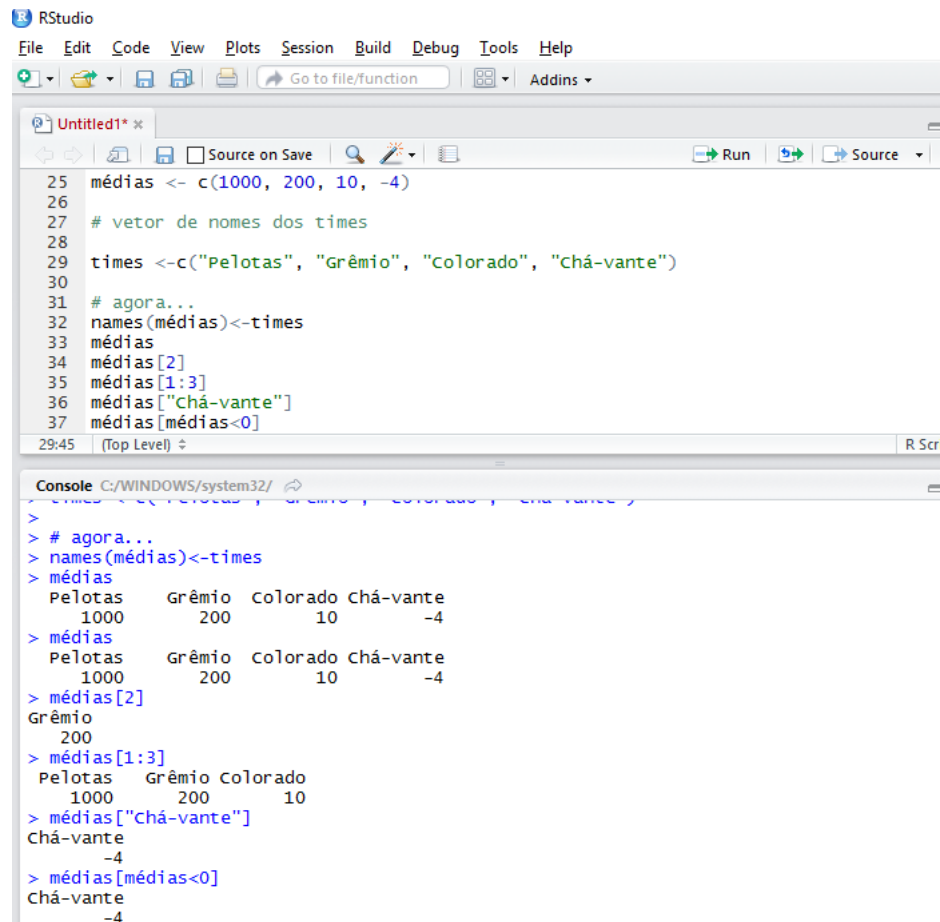
Console C:/WINDOWS/system32/ ↗

```
> # vetor de médias de gols no campeonato
>
> médias <- c(1000, 200, 10, -4)
>
> # vetor de nomes dos times
>
> times <-c("Pelotas", "Grêmio", "Colorado", "Chá-vante")
>
> # agora...
> names(médias)<-times
> médias
  Pelotas   Grêmio Colorado Chá-vante
    1000     200      10      -4
```

> |

Ok, neste exemplo você perdeu pontos com algumas garotas e ganhou com outras. Repare que o R aceita vetores numéricos e também de nomes. O uso das “aspas” é importante aqui. O RStudio é útil neste aspecto porque, bem, experimente escrever os nomes dos times sem as aspas e executar o comando. Ah sim, mais alguns comandos com este mesmo exemplo.





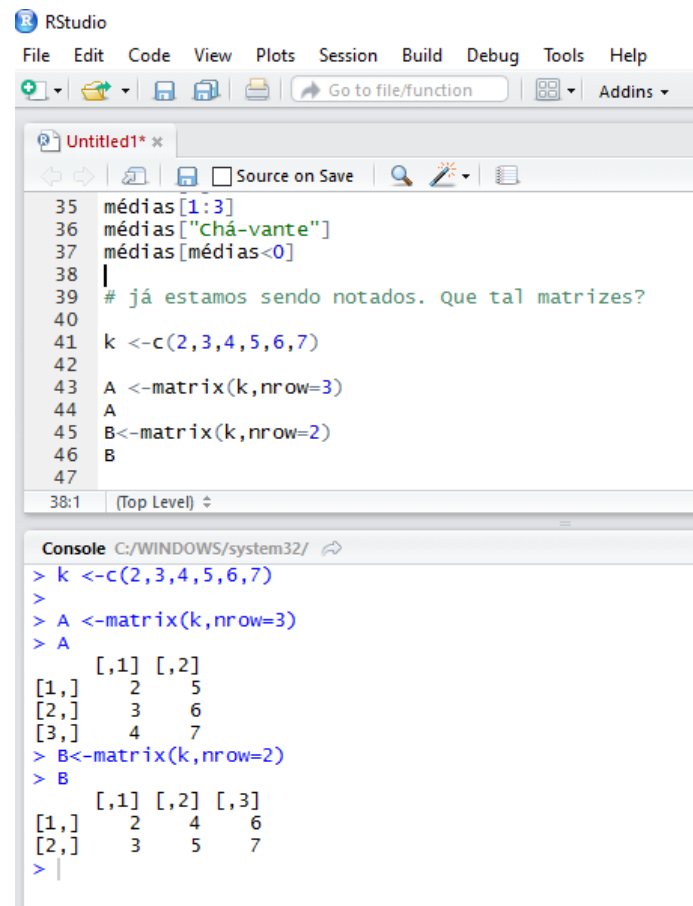
```
25 médias <- c(1000, 200, 10, -4)
26
27 # vetor de nomes dos times
28
29 times <-c("Pelotas", "Grêmio", "Colorado", "Chá-vante")
30
31 # agora...
32 names(médias)<-times
33 médias
34 médias[2]
35 médias[1:3]
36 médias["Chá-vante"]
37 médias[médias<0]
```

29:45 (Top Level) R Scr

Console C:/WINDOWS/system32/

```
> times <- c("Pelotas", "Grêmio", "Colorado", "Chá-vante")
>
> # agora...
> names(médias)<-times
> médias
Pelotas      Grêmio Colorado Chá-vante
1000         200        10       -4
> médias
Pelotas      Grêmio Colorado Chá-vante
1000         200        10       -4
> médias[2]
Grêmio
200
> médias[1:3]
Pelotas      Grêmio Colorado
1000         200        10
> médias["Chá-vante"]
Chá-vante
-4
> médias[médias<0]
Chá-vante
-4
```

Ok. Você já está experiente em vetores. Vejamos um pouco sobre matrizes. Note o comando *nrow* que diz respeito ao número de linhas da matriz que estamos criando.



```

35 médias[1:3]
36 médias["Chá-vante"]
37 médias[médias<0]
38 |
39 # já estamos sendo notados. Que tal matrizes?
40
41 k <-c(2,3,4,5,6,7)
42
43 A <-matrix(k,nrow=3)
44 A
45 B<-matrix(k,nrow=2)
46 B
47

```

38:1 (Top Level) ↕

```

Console C:/WINDOWS/system32/ ↗
> k <-c(2,3,4,5,6,7)
>
> A <-matrix(k,nrow=3)
> A
      [,1] [,2]
[1,]    2    5
[2,]    3    6
[3,]    4    7
> B<-matrix(k,nrow=2)
> B
      [,1] [,2] [,3]
[1,]    2    4    6
[2,]    3    5    7
> |

```

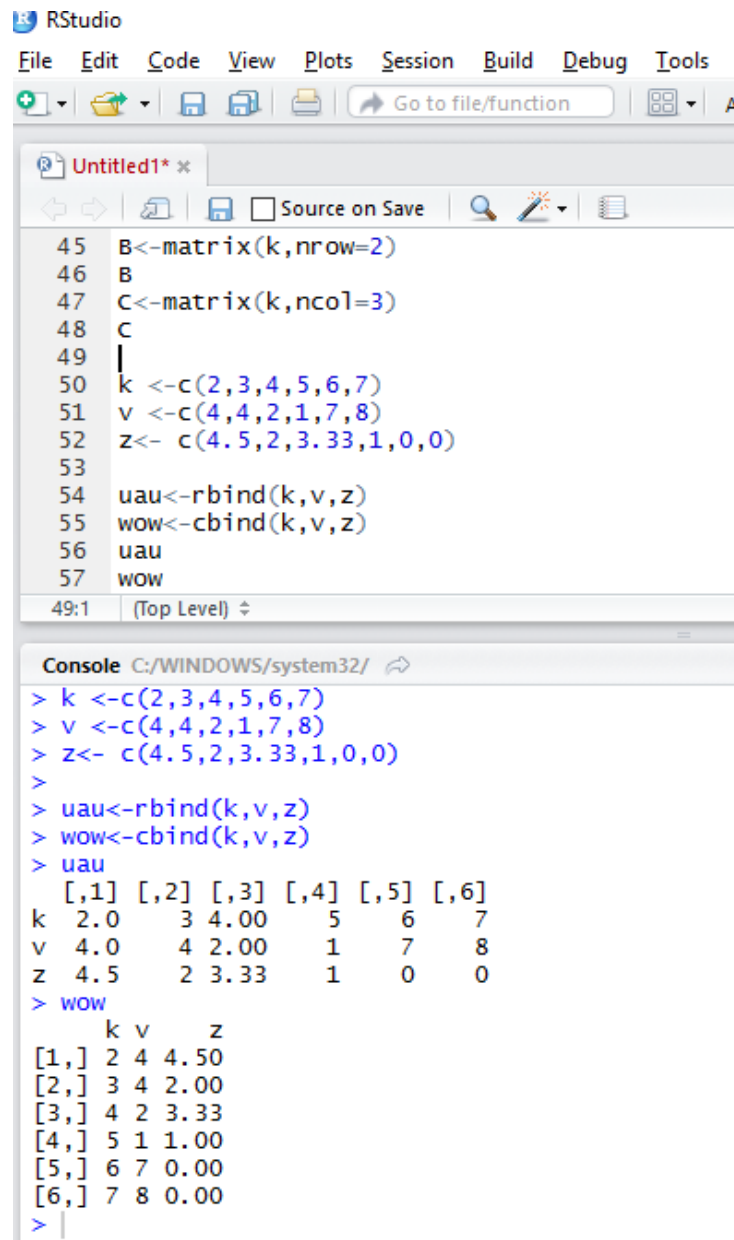
Que tal usar *ncol* (número de colunas)?

```

> C<-matrix(k,ncol=3)
> C
      [,1] [,2] [,3]
[1,]    2    4    6
[2,]    3    5    7
> |

```

Ok. Mas poderíamos também criar diferentes vetores e unifica-los em um objeto matriz por meio dos comandos *cbind* e *rbind*. No primeiro caso, os vetores são transformados em colunas de uma matriz e, no segundo caso, as linhas é que o são. Vejamos um exemplo.

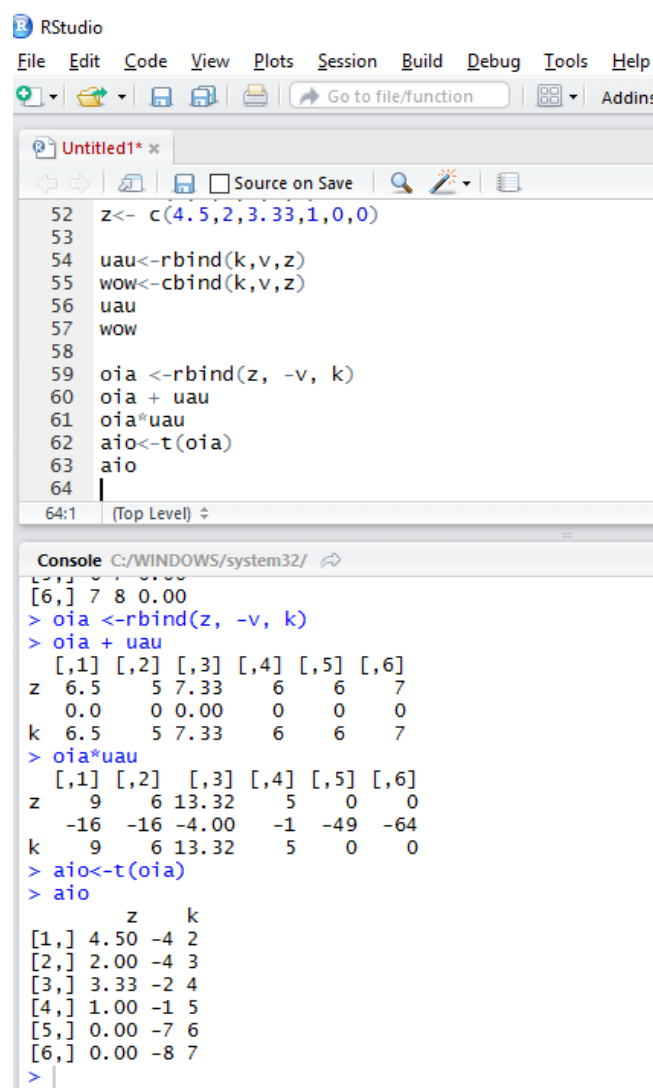


```

RStudio
File Edit Code View Plots Session Build Debug Tools
+ [Go to file/function]
Untitled1* x
Source on Save
45 B<-matrix(k,nrow=2)
46 B
47 C<-matrix(k,ncol=3)
48 C
49 |
50 k <-c(2,3,4,5,6,7)
51 v <-c(4,4,2,1,7,8)
52 z<- c(4.5,2,3.33,1,0,0)
53
54 uau<-rbind(k,v,z)
55 wow<-cbind(k,v,z)
56 uau
57 wow
49:1 (Top Level)
Console C:/WINDOWS/system32/
> k <-c(2,3,4,5,6,7)
> v <-c(4,4,2,1,7,8)
> z<- c(4.5,2,3.33,1,0,0)
>
> uau<-rbind(k,v,z)
> wow<-cbind(k,v,z)
> uau
  [,1] [,2] [,3] [,4] [,5] [,6]
k  2.0   3 4.00   5   6   7
v  4.0   4 2.00   1   7   8
z  4.5   2 3.33   1   0   0
> wow
      k v  z
[1,] 2 4 4.50
[2,] 3 4 2.00
[3,] 4 2 3.33
[4,] 5 1 1.00
[5,] 6 7 0.00
[6,] 7 8 0.00
>

```

Algumas operações básicas como somar, multiplicar (elemento por elemento) ou transpor são ilustradas abaixo.

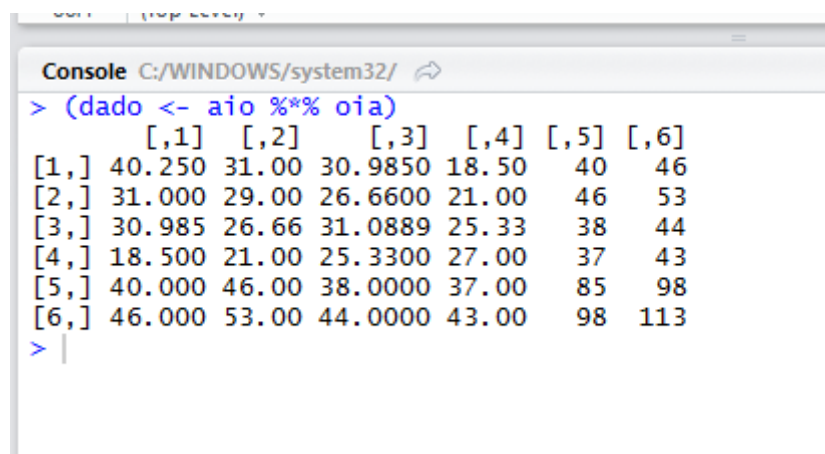


```

RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function Addins
Untitled1* x
Source on Save
52 z<- c(4.5,2,3.33,1,0,0)
53
54 uau<-rbind(k,v,z)
55 wow<-cbind(k,v,z)
56 uau
57 wow
58
59 oia <-rbind(z, -v, k)
60 oia + uau
61 oia*uau
62 aio<-t(oia)
63 aio
64
64:1 (Top Level)
Console C:/WINDOWS/system32/
[5,] 0 0 0.00
[6,] 7 8 0.00
> oia <-rbind(z, -v, k)
> oia + uau
      [,1] [,2] [,3] [,4] [,5] [,6]
z  6.5    5 7.33    6    6    7
   0.0    0 0.00    0    0    0
k  6.5    5 7.33    6    6    7
> oia*uau
      [,1] [,2] [,3] [,4] [,5] [,6]
z    9    6 13.32    5    0    0
   -16  -16 -4.00   -1  -49  -64
k    9    6 13.32    5    0    0
> aio<-t(oia)
> aio
      z    k
[1,] 4.50 -4 2
[2,] 2.00 -4 3
[3,] 3.33 -2 4
[4,] 1.00 -1 5
[5,] 0.00 -7 6
[6,] 0.00 -8 7
>

```

Quanto à multiplicação de matrizes, veja a figura seguinte. Note que ao colocarmos o comando entre parênteses, o resultado é automaticamente mostrado no *console*.



```

Console C:/WINDOWS/system32/
> (dado <- aio %%% oia)
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,] 40.250 31.00 30.9850 18.50 40 46
[2,] 31.000 29.00 26.6600 21.00 46 53
[3,] 30.985 26.66 31.0889 25.33 38 44
[4,] 18.500 21.00 25.3300 27.00 37 43
[5,] 40.000 46.00 38.0000 37.00 85 98
[6,] 46.000 53.00 44.0000 43.00 98 113
>

```

Para inverter matrizes precisamos, inicialmente de uma matriz quadrada que não seja singular (como é o caso da matriz acima, por motivos óbvios). Fica por sua conta construir uma matriz de dimensão  $2 \times 2$  e aplicar-lhe o comando *solve*. Digamos que sua

matriz se chame *banana*. Então, o comando para invertê-la será: *solve(banana)*. Ok, ajuda rápida.

```
[6,] 46.000 53.00 44.0000 43.00 98 113
> banana<-matrix(c(2,3,4,5),nrow=2)
> solve(banana)
      [,1] [,2]
[1,] -2.5  2
[2,]  1.5 -1
> |
```

Uma das vantagens do R é podermos criar uma lista de elementos de diferentes tipos. Neste caso, criamos uma lista composta de dois objetos. O primeiro deles é uma sequência de números que se inicia em dois e termina em três, gerados de forma discreta, com um passo de distância. O segundo é a variável “oi” que recebe a expressão “io”. Em seguida, você percebe que pode visualizar os nomes dos componentes e também que o símbolo “\$” serve para visualizar os itens do componente “A” da “listinha”.

```
> listinha<-list(A=seq(2,3,1), oi="io")
> listinha
$A
[1] 2 3

$oi
[1] "io"

> names(listinha)
[1] "A" "oi"
> listinha$A
[1] 2 3
|
```

Este tipo de objeto naturalmente nos leva a um dos mais usados em R que é o *data frame*. Embora lembre uma matriz, ele não é uma matriz e pode acomodar objetos bem diferentes ao mesmo tempo. Na tentativa de ganhar a atenção da garota (ou dos seus amigos), você tenta emplacar uma conversa sobre vendas semanais dos produtos “v1” e “v2”.

```

> v1 <- c(10, 2, 11, -4)
> v2<-c(1,2,3,0)
> semana<-c(1,2,3,4)
>
> vendas<-cbind(v1,v2)
> rownames(vendas) <-semana
>
> vendas
      v1 v2
1 10  1
2  2  2
3 11  3
4 -4  0
> vendasnovo<-as.data.frame(vendas)
> vendasnovo
      v1 v2
1 10  1
2  2  2
3 11  3
4 -4  0
>

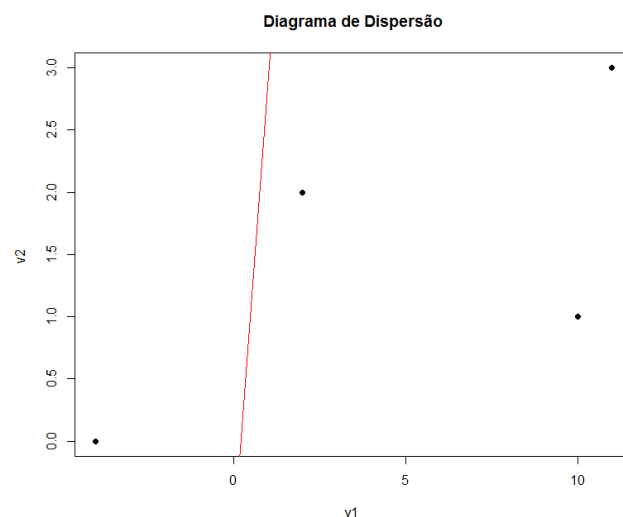
```

O que acontece se você digitar (e executar) `vendasnovo$v1`? Experimente. Voltaremos aos arquivos *data frame* posteriormente, para ilustrar outros comandos. Por enquanto, acho que você já conseguiu impressionar seus amigos (e aquela garota). Ou não? Ok, vamos tentar mais uma última cartada: fazer um diagrama de dispersão e incluir a reta de regressão.

```

# um grafico simples
plot(vendasnovo$v1, vendasnovo$v2, main="Diagrama de Dispersão",
      xlab="v1 ", ylab="v2 ", pch=19)
abline(lm(v1~v2,data=vendasnovo), col="red")

```



Poucos dados, não? Podemos tentar algo mais interessante. Que tal um pouco de ação? Primeiro, vamos abrir uma planilha (*japao.xlsx*), copiar e colar os dados no R.

Clipboard		Font		Alignment					
A1		fx		year					
	A	B	C	D	E	F	G	H	
1	year	total	gangsterir	pistol	gdp_grow	total_crim	gangster_	pistol_growth	
2	1980	199	142	127	2.81757	0.151641	0.143643	0.171358	
3	1981	198	148	131	4.17684	-0.00504	0.041385	0.03101	
4	1982	194	150	129	3.37664	-0.02041	0.013423	-0.01538	
5	1983	291	232	220	3.06072	0.405465	0.436102	0.533815	
6	1984	191	158	163	4.46389	-0.42105	-0.38414	-0.29988	
7	1985	254	221	209	6.33337	0.285061	0.335568	0.248584	
8	1986	256	232	219	2.83107	0.007843	0.048575	0.046737	
9	1987	265	230	219	4.10745	0.034552	-0.00866	0	
0	1988	197	154	155	7.14668	-0.29653	-0.40113	-0.34565	
1	1989	202	170	157	5.37017	0.025064	0.098846	0.012821	
2	1990	200	173	171	5.57239	-0.00995	0.017493	0.085418	
3	1991	155	125	132	3.32433	-0.25489	-0.32498	-0.25886	
4	1992	176	140	151	0.81905	0.127059	0.113329	0.134478	
5	1993	130	99	97	0.17106	-0.30295	-0.34652	-0.44257	
6	1994	159	117	131	0.86356	0.20137	0.167054	0.300486	
7	1995	156	111	127	1.94234	-0.01905	-0.05264	-0.03101	

Aí está. O crescimento do PIB (*gdp\_growth*) foi obtido a partir de uma série descontinuada do banco de dados FRED (<https://research.stlouisfed.org/fred2/>). O restante são de fontes oficiais (governo do Japão). Temos, então, mais observações e a possibilidade de você ganhar a garota com um bom papo sobre máfias e economia (ou, eventualmente, sobre filmes japoneses de máfia).

Para esta empreitada, usaremos um pacote um pouco mais avançado – mas muito mais útil – para fazer gráficos. O comando *plot()* usado acima é básico do R. Vejamos um pouco do *ggplot2*. Os primeiros comandos abaixo nos mostram como é fácil carregar dados no R (fácil, mas o problema é que há tantas possibilidades que você geralmente tem que pesquisar para saber qual a melhor forma de importar seus dados para o R). Repare que, em seguida, já uso o comando do *ggplot*. Vale a pena ampliarmos a parte final da figura para vermos a janela com o *script*<sup>9</sup>.

<sup>9</sup> Para detalhes, ver: <http://stackoverflow.com/questions/10599708/difference-between-read-table-and-read-delim-functions>.

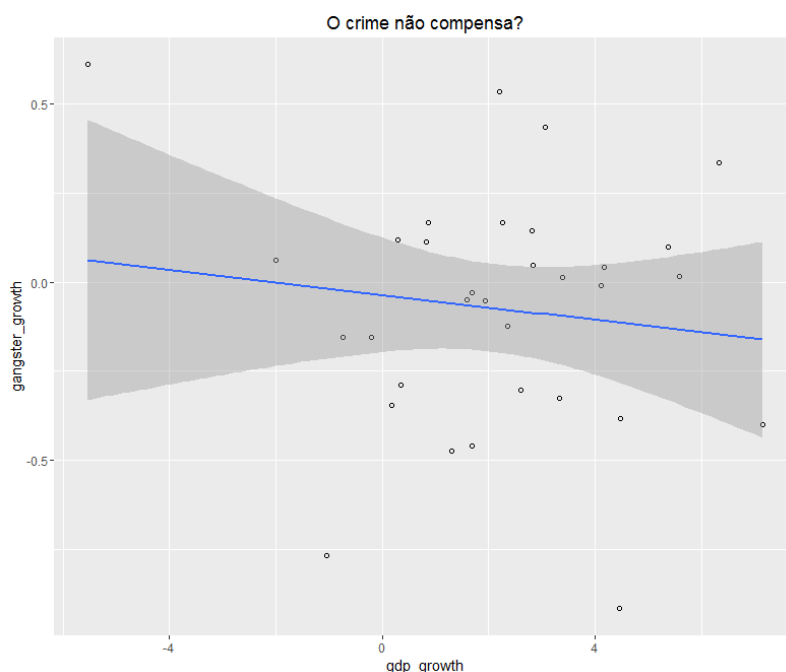
```

> jap<-read.delim("clipboard", header=TRUE)
> # para planilhas o melhor é: read.table(file = "clipboard", sep = "\t", header=TRUE)
> head(jap)
  year total gangsterinvolved pistol gdp_growth total_crime_growth gangster_growth pistol_growth
1 1980  199             142    127   2.81757      0.151641268      0.14364270    0.17135825
2 1981  198             148    131   4.17684     -0.005037794      0.04138522    0.03101024
3 1982  194             150    129   3.37664     -0.020408872      0.01342302   -0.01538492
4 1983  291             232    220   3.06072      0.405465108      0.43610208    0.53381514
5 1984  191             158    163   4.46389     -0.421049839     -0.38414234   -0.29987735
6 1985  254             221    209   6.33337      0.285060839      0.33556767    0.24858405
> library(ggplot2)
> ggplot(jap, aes(x=gdp_growth, y=gangster_growth)) +
+   geom_point(shape=1) +      # bolinhas
+   geom_smooth(method=lm) +    # reta de regressão (default 95%)
+   ggtitle("O crime não compensa?")

head(jap)
library(ggplot2)
ggplot(jap, aes(x=gdp_growth, y=gangster_growth)) +
  geom_point(shape=1) +      # bolinhas
  geom_smooth(method=lm) +    # reta de regressão (default 95%)
  ggtitle("O crime não compensa?")

```

A partir da janela acima você percebe que o “+” que aparece no *script* faz parte da sintaxe do comando. Não se trata do tradicional “+” que aparece na janela de resultados (*console*) para indicar que a linha anterior continua. Cuidado com isso, heim? E o gráfico para impressionar a garota?



Não ficou feio, ficou? A partir daí, a conversa prossegue com uma discussão sobre o crime, a natureza humana, os limites da Estatística, as maravilhas da Econometria e, antes que você perceba, já estará sozinho, no bar, bêbado, sem a garota por perto. Ou não.



### 3. Como é que eu importo os meus dados para o R? Que loucura é esta?

Não deve haver um tópico mais polêmico do que este. Há várias formas de se importar dados de planilhas, *sites* da internet, etc, para o R. Minha sugestão é que você se especialize em um tipo de arquivo (minha escolha pessoal é pelos arquivos no formato *csv*) e pesquise sobre outros formatos conforme sua necessidade. De qualquer forma, vamos a alguns exemplos.

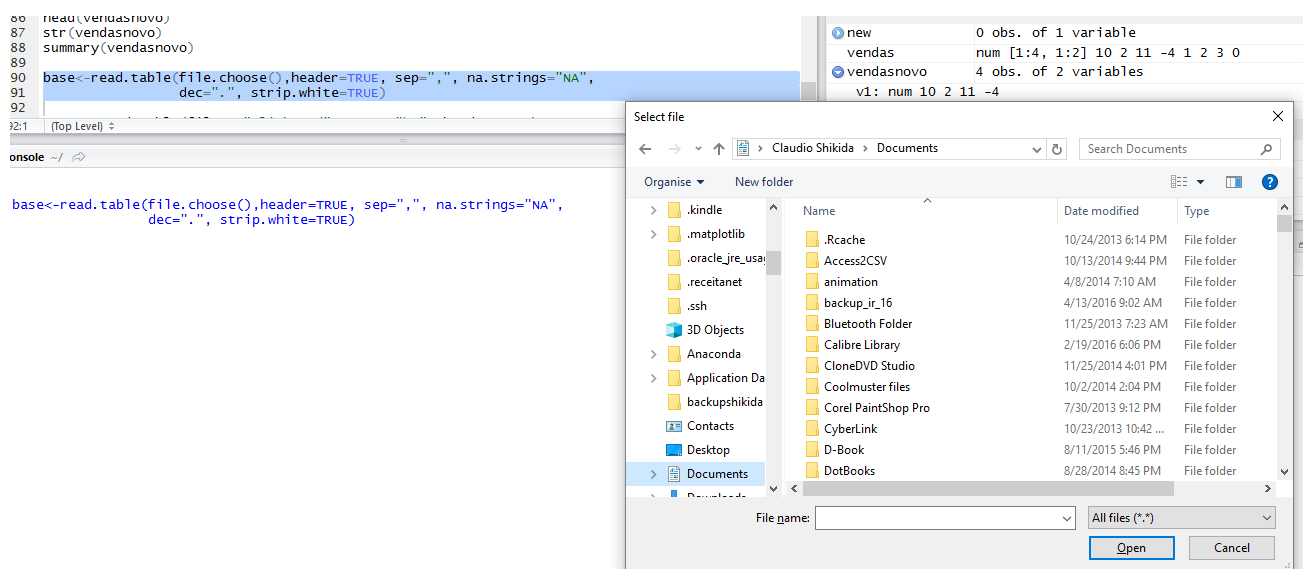
```
> new<-read.table("C:/Users/cdshi_000/Documents/Meus Documentos/Meus Documentos/textos/
+ textos em andamento (e planilhas)/escravos de ganho rent seeking/rationalslaves.csv",
+ header=TRUE, sep=",", na.strings="NA", dec=".", strip.white=TRUE)
```

Outra dica do R: estamos na janela de *console*, certo? O que você visualiza aí em cima é o comando que digitei, exceto por um detalhe: na terceira linha, na extrema esquerda, há um sinal de "+". Ele não foi digitado. É que o R, ao encontrar a limitação da janela à direita, passa para a linha de baixo com o sinal "+" indicando que o que se segue é continuação da linha anterior e não um novo comando. Veja o mesmíssimo comando na janela do *script*. Ele é tão comprido que tive que passar para a linha de baixo em algum momento. A regra é não dividir a linha pelo menos até terminar o nome do arquivo, no caso, *rationalslaves.csv*. Após a vírgula, **que deve permanecer na mesma linha**, tudo bem.

```
1 new<-read.table("C:/Users/cdshi_000/Documents/Meus Documentos/Meus Documentos/textos/textos em
2 header=TRUE, sep=",", na.strings="NA", dec=".", strip.white=TRUE)
```

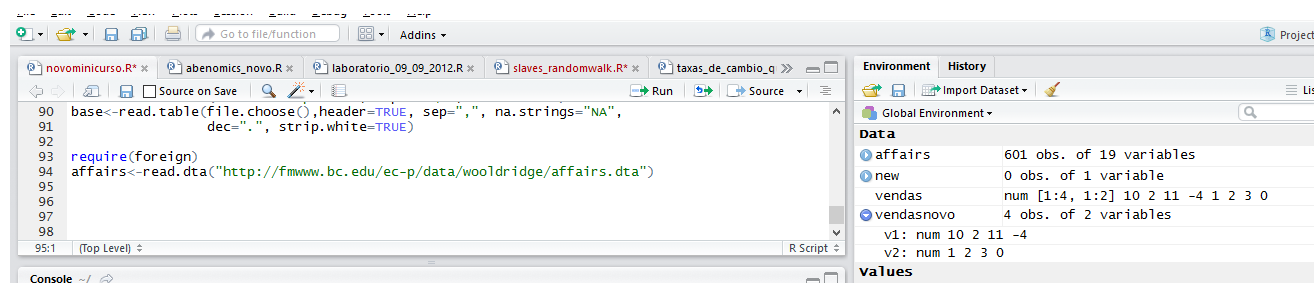
Algumas observações. O *header=TRUE* diz ao R que ele deve considerar a primeira linha da planilha como o cabeçalho. O *sep=","* informa que o separador entre as colunas do arquivo *csv* é a vírgula (não é coincidência que *csv* seja o acrônimo de *comma separated values*). O *na.strings="NA"* informa que as células em branco devem ser preenchidos com a expressão *NA*. O *dec="."* Informa que o separador decimal é o ponto. Finalmente, *strip.white=TRUE* é usado, em *read.table*, quando a planilha original tem delimitadores. Basicamente, o que ele faz é considerar espaços em branco à esquerda e à direita da célula (entre os delimitadores) como parte da mesma. Por exemplo, digamos que a célula (o separador utilizado são as aspas) é: " Altamira do Carmo ". Assim, os espaços em branco à esquerda e à direita do nome serão considerados como parte da célula.

Achou desagradável? Você pode simplesmente pedir para abrir uma janela para que você mesmo selecione seu arquivo (mas repare que os comandos auxiliares são os mesmos: cabeçalho, separadores, etc).



Basicamente, o `file.choose()` permite que escolhamos o arquivo a ser lido por meio de uma janela. A vantagem é que não se precisa indicar o caminho do mesmo. A desvantagem é que se gasta mais tempo para se obter acesso ao arquivo<sup>10</sup>.

Você poderia importar dados diretamente da *internet* sem necessidade de transferência para o computador primeiro. Por exemplo, os dados utilizados no famoso estudo de Fair sobre casos extra-conjugais é usado em vários livros e está disponibilizado em vários endereços *online* como o do exemplo seguinte. Repare que o formato de dados importado, no exemplo, é o do *Stata* (extensão *.dta*).



Uma terceira opção é copiar os dados da sua planilha e colar no R salvando-os, em seguida, em um *data frame*. Lembra do exemplo do Japão? Veja aqui o que foi feito.

<sup>10</sup> O comando `strip.white=TRUE` é usado, em `read.table`, quando a planilha original tem delimitadores. Basicamente, o que ele faz é considerar espaços em branco à esquerda e à direita da célula (entre os delimitadores) como parte da mesma. Digamos que a célula (o separador utilizado são as aspas) é: "Altamira do Carmo". Assim, os espaços em branco à esquerda e à direita do nome serão considerados como parte da célula.

```

> jap<-read.delim("clipboard", header=TRUE)
> # para planilhas o melhor é: read.table(file = "clipboard", sep = "\t", header=TRUE)
> head(jap)
  year total gangsterinvolved pistol gdp_growth total_crime_growth gangster_growth pistol_growth
1 1980  199                142   127    2.81757    0.151641268    0.14364270    0.17135825
2 1981  198                148   131    4.17684   -0.005037794    0.04138522    0.03101024
3 1982  194                150   129    3.37664   -0.020408872    0.01342302   -0.01538492
4 1983  291                232   220    3.06072    0.405465108    0.43610208    0.53381514
5 1984  191                158   163    4.46389   -0.421049839   -0.38414234   -0.29987735
6 1985  254                221   209    6.33337    0.285060839    0.33556767    0.24858405
> library(ggplot2)
> ggplot(jap, aes(x=gdp_growth, y=gangster_growth)) +
+ geom_point(shape=1) + # bolinhas
+ geom_smooth(method=lm) + # reta de regressão (default 95%)
+ ggtitle("O crime não compensa?")

```

Para salvar como *data.frame* basta aplicar o comando homônimo e eis o resultado.

```

121:1 | (Top Level) | R Script
Console ~/
1 2009  64                35   33   -5.52696    0.28768207    0.6109091    0.2776317
2 2010  46                14   20    4.44669   -0.33024169   -0.9162907   -0.5007753
3 2011  31                12   10   -0.73255   -0.39465419   -0.1541507   -0.6931472
new<-data.frame(new)

head(new)
  year total gangsterinvolved pistol gdp_growth total_crime_growth gangster_growth pistol_growth
1979  171                123   107    5.48396                NA                NA                NA
1980  199                142   127    2.81757    0.151641268    0.14364270    0.17135825
1981  198                148   131    4.17684   -0.005037794    0.04138522    0.03101024
1982  194                150   129    3.37664   -0.020408872    0.01342302   -0.01538492
1983  291                232   220    3.06072    0.405465108    0.43610208    0.53381514
1984  191                158   163    4.46389   -0.421049839   -0.38414234   -0.29987735
tail(new)
  year total gangsterinvolved pistol gdp_growth total_crime_growth gangster_growth pistol_growth
8 2006  73                24   33    1.69291    0.04196420   -0.4595323   -0.4155154
9 2007  74                41   43    2.19219    0.01360565    0.5355182    0.2646926
0 2008  48                19   25   -1.04164   -0.43286408   -0.7691331   -0.5423243
1 2009  64                35   33   -5.52696    0.28768207    0.6109091    0.2776317
2 2010  46                14   20    4.44669   -0.33024169   -0.9162907   -0.5007753

```

Para conferir, uso dois comandos simples para ver o início (*head*) e o final (*tail*) da minha nova base de dados importada.

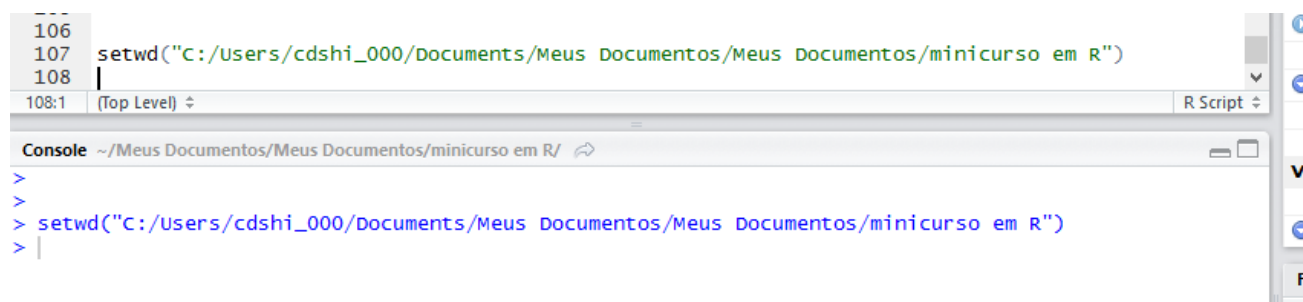
Existem muitos outros jeitos de se importar dados para o R. Você pode, por exemplo, programar e capturar dados da internet. Veja este [exemplo](#) para um conjunto de dados em formato ASCII e extensão “.txt” que está na página de um professor e é importado diretamente para o R.

Para mais dicas sobre como importar arquivos, pesquise um pouco, ok?<sup>11</sup>

<sup>11</sup> Veja, por exemplo, este [material](#).

#### 4. Vamos começar a trabalhar? Tá tudo em ordem? Eu sei tudo? Tenho dúvidas? Opa, eu tenho dúvidas!

Agora que já sabemos alguns detalhes do R, vamos começar a pensar em termos do trabalho que vamos desenvolver. Primeiramente, um pouco de organização. Vamos combinar aqui que é necessário ter um diretório de trabalho. Há quem goste disto. Bem, isto pode ser feito com o comando `setwd()`.

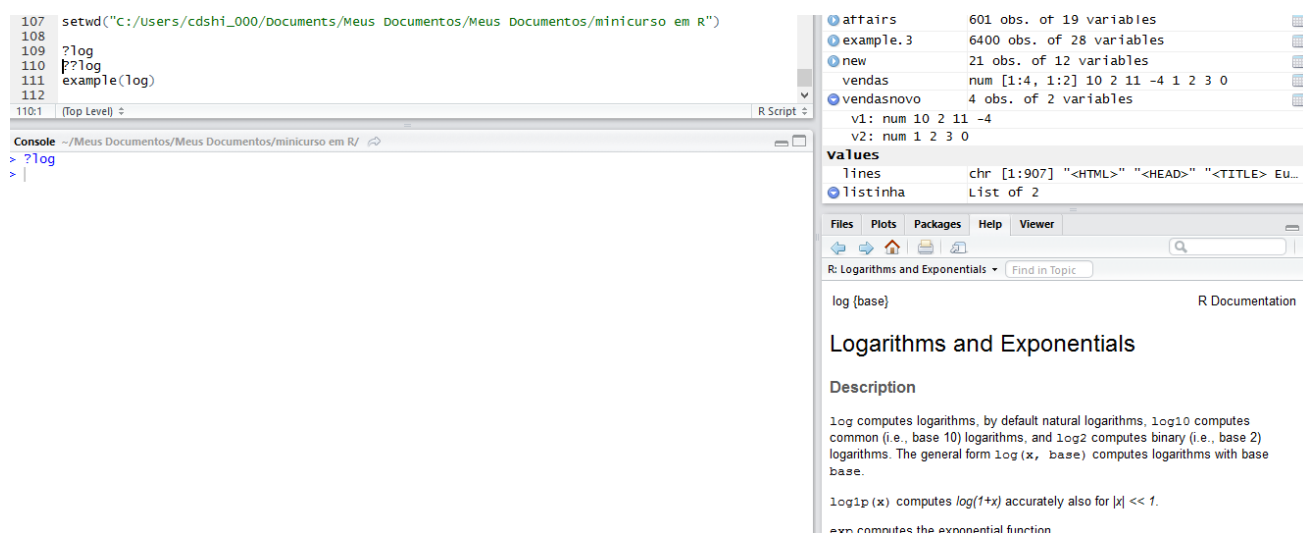


```

106
107 setwd("C:/Users/cdshi_000/Documents/Meus Documentos/Meus Documentos/minicurso em R")
108
108:1 (Top Level)
R Script
Console ~/Meus Documentos/Meus Documentos/minicurso em R/
>
>
> setwd("C:/Users/cdshi_000/Documents/Meus Documentos/Meus Documentos/minicurso em R")
>

```

Pronto! A partir de agora, já podemos nos concentrar no trabalho. Bem, não exatamente. Seria bom poupar o professor de tantas perguntas. Digamos, por exemplo, que você tem dúvida sobre o comando `log`. Como saber mais? Um pouco de tentativa-e-erro vem a calhar aqui. A maneira mais simples é digitar `?log`.



```

107 setwd("C:/Users/cdshi_000/Documents/Meus Documentos/Meus Documentos/minicurso em R")
108
109 ?log
110 ?log
111 example(log)
112
110:1 (Top Level)
R Script
Console ~/Meus Documentos/Meus Documentos/minicurso em R/
> ?log
>

```

**Values**

lines	chr [1:907] "<HTML>" "<HEAD>" "<TITLE> EU..."
listinha	List of 2

**R: Logarithms and Exponentials**

**log (base)**

**Logarithms and Exponentials**

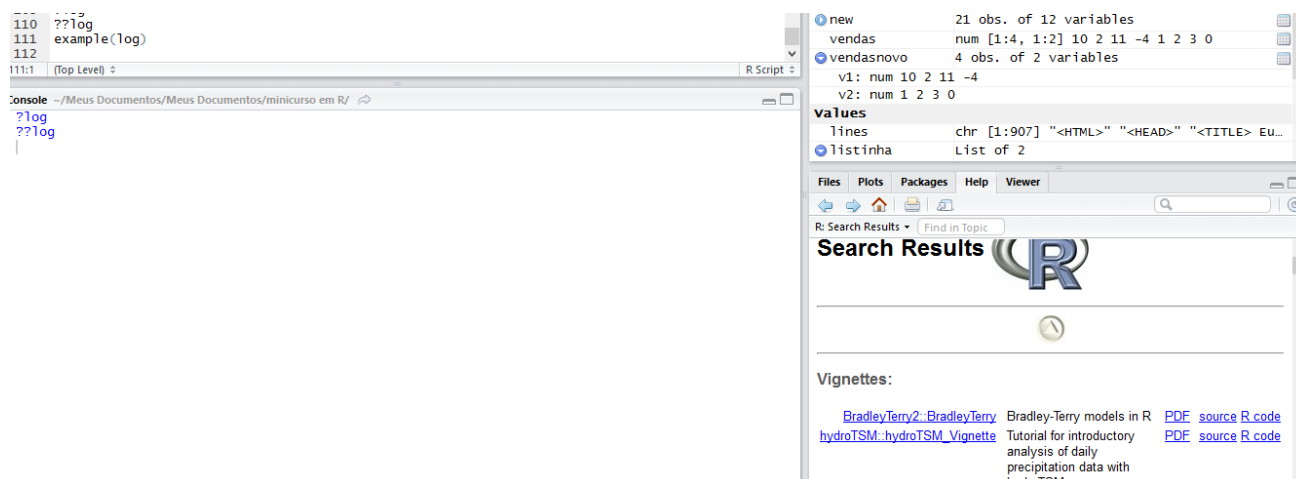
**Description**

log computes logarithms, by default natural logarithms, log10 computes common (i.e., base 10) logarithms, and log2 computes binary (i.e., base 2) logarithms. The general form log(x, base) computes logarithms with base base.

log1p(x) computes log(1+x) accurately also for |x| << 1.

exp computes the exponential function

Repare que no canto inferior direito obtivemos alguma ajuda. Digamos que você não conseguiu ajuda assim. Então tente `??log` (note que ele já apareceu na figura acima). Neste caso, o R tenta uma busca mais ampla. No caso, encontrou referências em alguns pacotes.



Finalmente, o famoso *example(log)* que serve para funções simples (algumas).

```

110  ??log
111  example(log)
112  |
112:1 (Top Level) ↕

Console ~/Meus Documentos/Meus Documentos/minicurso em R/ ↗
> ??log
> ??log
> example(log)

log> log(exp(3))
[1] 3

log> log10(1e7) # = 7
[1] 7

log> x <- 10^-(1+2*1:9)

log> cbind(x, log(1+x), log1p(x), exp(x)-1, expm1(x))
      x
[1,] 1e-03 9.995003e-04 9.995003e-04 1.000500e-03 1.000500e-03
[2,] 1e-05 9.999950e-06 9.999950e-06 1.000005e-05 1.000005e-05
[3,] 1e-07 1.000000e-07 1.000000e-07 1.000000e-07 1.000000e-07
[4,] 1e-09 1.000000e-09 1.000000e-09 1.000000e-09 1.000000e-09
[5,] 1e-11 1.000000e-11 1.000000e-11 1.000000e-11 1.000000e-11
[6,] 1e-13 9.992007e-14 1.000000e-13 9.992007e-14 1.000000e-13
[7,] 1e-15 1.110223e-15 1.000000e-15 1.110223e-15 1.000000e-15
[8,] 1e-17 0.000000e+00 1.000000e-17 0.000000e+00 1.000000e-17
[9,] 1e-19 0.000000e+00 1.000000e-19 0.000000e+00 1.000000e-19
> |

```

Outra forma de se pedir ajuda é por meio do “*help.start()*”. Este comando abre seu navegador padrão diretamente na página do R (obviamente você deve estar conectado...).

Já que vamos trabalhar, pode ser interessante ganhar um pouco de memória removendo alguns objetos. Digamos que eu queira remover o objeto *vendasnovo* criado há algumas páginas. Posso escrever *rem(vendasnovo)* ou *remove(vendasnovo)* e obterei meu resultado (não será apresentada a imagem aqui: verifique você mesmo que o objeto desaparecerá da listagem no canto superior direito).

## 5. A Estatística dos Casos Extra-Conjugais de Maridos e Esposas

Para os próximos exemplos importarei uma base de dados famosa, do não menos famoso economista Ray C. Fair<sup>12</sup>. A base, *Affairs*, está em um dos pacotes mais úteis do R, o pacote *AER*. Não é preciso importar os dados de planilhas mas apenas carregar o pacote *AER*. Vejamos, inicialmente, os dados.

```

1 library(AER)
2 data("Affairs")
3 summary(Affairs)
4 library(pastecs)
5 stat.desc(Affairs)
6

```

4:1 (Top Level) ↕

Console ~/Meus Documentos/Meus Documentos/minicurso em R/ ↗

```

· library(AER)
· data("Affairs")
· summary(Affairs)
  affairs      gender      age      yearsmarried      children      religi
Min.   : 0.000  female:315  Min.   :17.50  Min.   : 0.125  no :171  Min.
1st Qu.: 0.000  male :286   1st Qu.:27.00  1st Qu.: 4.000  yes:430  1st Qu
Median : 0.000                Median :32.00  Median : 7.000                Median
Mean   : 1.456                Mean   :32.49  Mean   : 8.178                Mean
3rd Qu.: 0.000                3rd Qu.:37.00  3rd Qu.:15.000               3rd Qu
Max.   :12.000                Max.   :57.00  Max.   :15.000                Max.

  education      occupation      rating
Min.   : 9.00  Min.   :1.000  Min.   :1.000
1st Qu.:14.00  1st Qu.:3.000  1st Qu.:3.000
Median :16.00  Median :5.000  Median :4.000
Mean   :16.17  Mean   :4.195  Mean   :3.932
3rd Qu.:18.00  3rd Qu.:6.000  3rd Qu.:5.000
Max.   :20.00  Max.   :7.000  Max.   :5.000
· tail(Affairs$affairs)
[1] 7 1 7 2 2 1
· library(pastecs)
loading required package: boot
attaching package: 'boot'
the following object is masked from 'package:survival':
  am1

```

Repare que, inicialmente usei o *summary*, mas você pode obter mais estatísticas usando o pacote *pastecs*. Ok, você pode estar achando que estou inventando. Afinal, como assim uma base de dados sobre casos extra-conjugais? Primeiramente, veja referência original do artigo no rodapé<sup>13</sup>. Trata-se de um exemplo clássico de aplicação do modelo de regressão Tobit<sup>14</sup>.

<sup>12</sup> Quem já trabalhou com modelos macroeconômicos em larga escala deve se lembrar do modelo de Fair usado como exemplo no *Evievs*. Parte do que se segue é adaptado de material que produzi para o blog do falecido Nepom. Eis o endereço: <https://nepom.wordpress.com/2015/05/23/a-economia-dos-casos-extraconjugais-ou-monica-lewinsky-visita-o-nepom/>.

<sup>13</sup> <http://people.stern.nyu.edu/wgreene/Lugano2013/Fair-ExtramartialAffairs.pdf>.

<sup>14</sup> Caso você tenha tido bons cursos de Econometria na graduação, saberá do que falo. Caso contrário, veremos pelo menos como estimar um tobit mais adiante. Obviamente, não é da minha conta se você tem casos extra-conjugais. Nenhum marido, esposa ou amante foi ferido ou morto (sério) durante a confecção desta apostila. Bem, não que eu saiba.

Vejamos quantos o número de observações da variável *affair* (que se refere ao número de casos extraconjugais do indivíduo no ano anterior). Entretanto, os valores assumidos pela variável devem ser interpretados com cautela (ver rodapé 5). Por exemplo, quando *affairs* assume valores 0, 1, 2 ou 3, estes são exatamente os números de casos reportados no ano anterior. Caso o respondente tenha tido entre 4 e 10 casos, ele é computado como “7”. Finalmente, alguns reportaram ter tido casos em frequência mensal, semanal ou diária (!). Para todos estes foi atribuído o valor “12”. Dito isto, vejamos os dados.

```

Console ~/Meus Documentos/Meus Documentos/minicurso em R/
> # frequencia da variável affairs
> (table(Affairs$affairs))

 0    1    2    3    7   12 
451   34   17   19   42   38 
> prop.table(table(Affairs$affairs))

      0      1      2      3      7      12 
0.75041597 0.05657238 0.02828619 0.03161398 0.06988353 0.06322795 
> |

```

Repare que temos 451 indivíduos que não “pularam a cerca” no ano anterior, o que corresponde a 75% das observações desta variável. Quer um histograma?

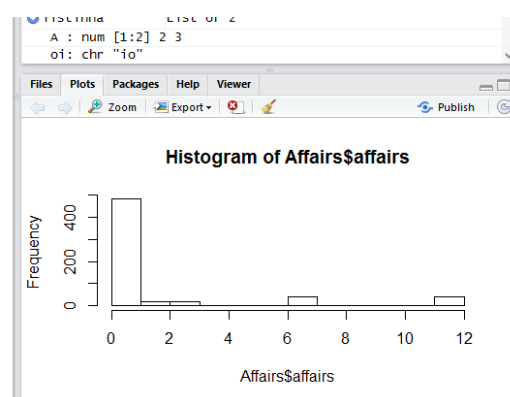
```

# frequencia da variável affairs
(table(Affairs$affairs))

 0    1    2    3    7   12 
51   34   17   19   42   38 
prop.table(table(Affairs$affairs))

      0      1      2      3      7      12 
.75041597 0.05657238 0.02828619 0.03161398 0.06988353 0.06322795 
hist(Affairs$affairs)

```



Uma das vantagens do RStudio em relação ao R é que os gráficos gerados não desaparecem quando da geração de novos gráficos. Eles vão se acumulando na aba *Plots*, na janela inferior direita.

Digamos que você queira a tabela de contingência entre *affairs* e *children*. Eis como fazer.



```

      0      1      2      3      7
0.75041597 0.05657238 0.02828619 0.03161398 0.06988353 0.0632
> hist(Affairs$affairs)
> contingencia <-table(Affairs$affairs, Affairs$children)
> contingencia

      no yes
0  144 307
1    7  27
2    2  15
3    4  15
7    7  35
12   7  31
> |

```

As distribuições marginais na linha e na coluna são obtidas conforme exemplo abaixo.

```
> prop.table(contingencia,margin=1)
```

```

      no      yes
0 0.3192905 0.6807095
1 0.2058824 0.7941176
2 0.1176471 0.8823529
3 0.2105263 0.7894737
7 0.1666667 0.8333333
12 0.1842105 0.8157895

```

```
> prop.table(contingencia,margin=2)
```

```

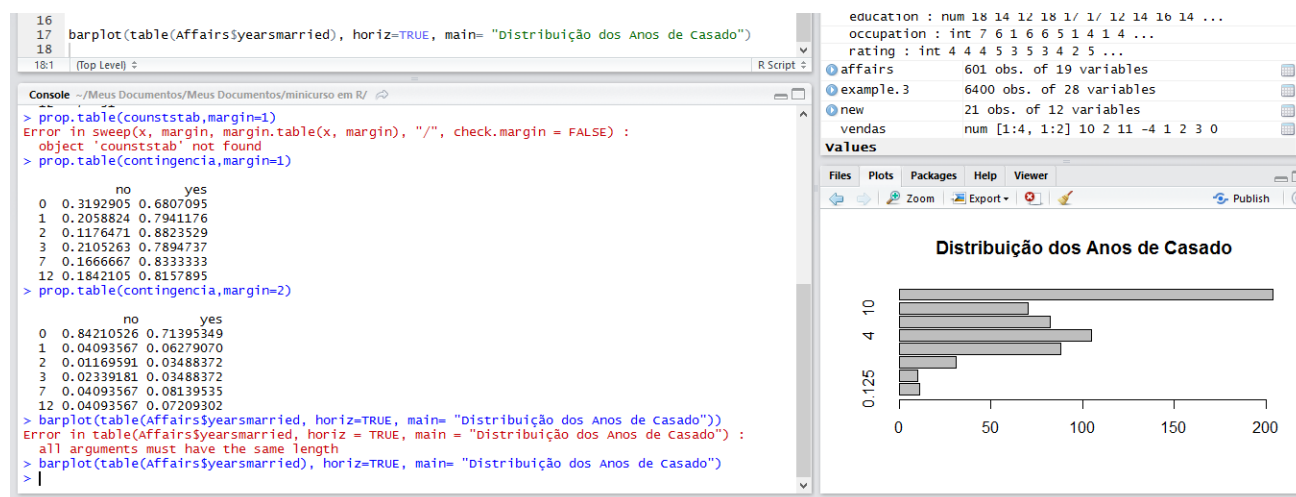
      no      yes
0 0.84210526 0.71395349
1 0.04093567 0.06279070
2 0.01169591 0.03488372
3 0.02339181 0.03488372
7 0.04093567 0.08139535
12 0.04093567 0.07209302

```

Na primeira saída, para a linha “0”, ou seja, para os indivíduos que não tiveram casos extra-conjugais no ano anterior, 31.93% eram indivíduos sem filhos. Na segunda saída, a soma em 100% se dá em cada coluna e vemos que 84% dos indivíduos que não tinham filhos não tiveram casos conjugais no ano anterior.

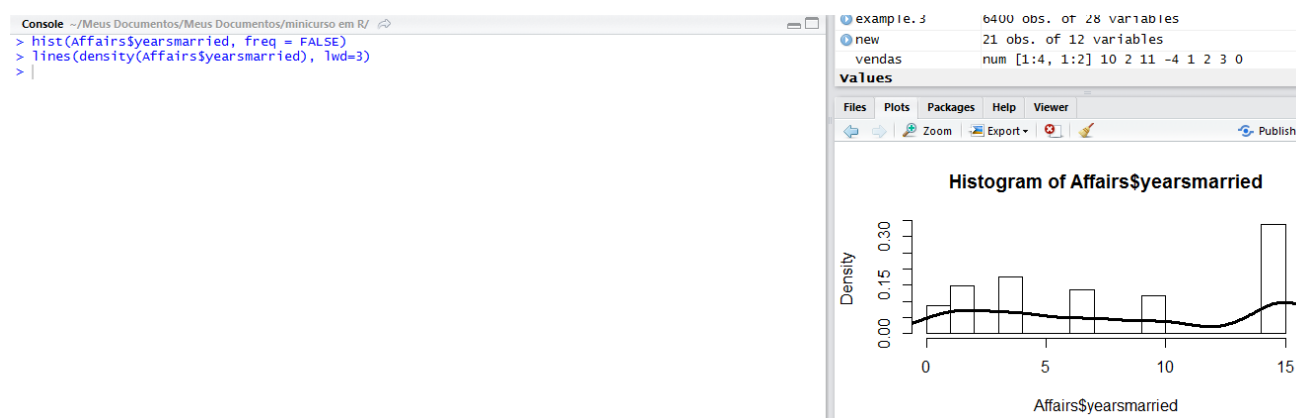
Aproveitando para mostrar como você identifica um erro no R, gero o gráfico de barras para a variável *yearsmarried*.





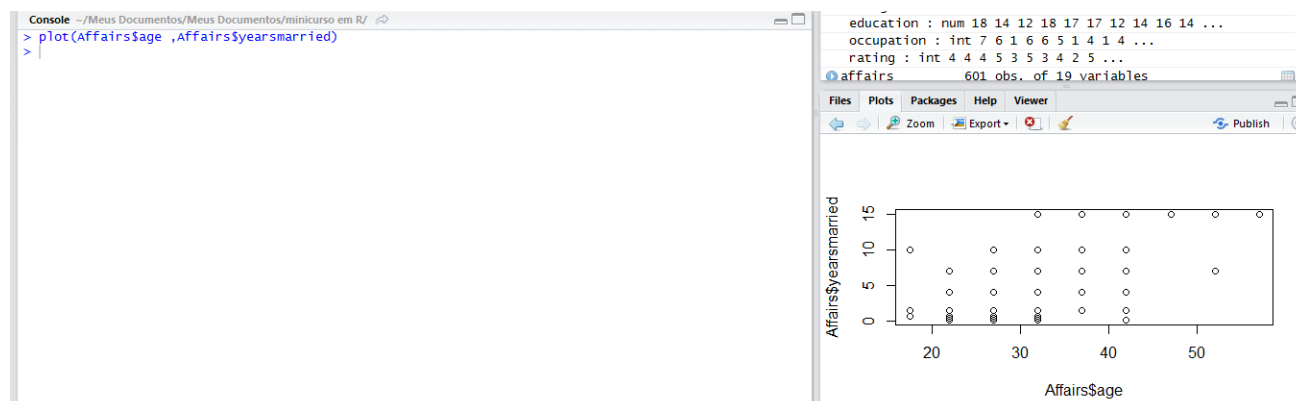
Repare que, na primeira tentativa, eu me esqueci do parênteses logo após *Affairs\$yearsmarried* e o R apontou o erro em vermelho com uma mensagem.

Poderíamos sobrepor também a densidade ao histograma desta variável. Eis como fazer isto de forma simples.



No comando acima, primeiro faz-se o histograma. Em seguida, o comando *lines* faz o desenho da densidade (*density*) com uma linha mais espessa (*lwd=3*).

Digamos que você esteja interessado na relação gráfica entre a idade do indivíduo (*age*) e a variável *yearsmarried*. Em outras palavras, você quer um diagrama de dispersão.

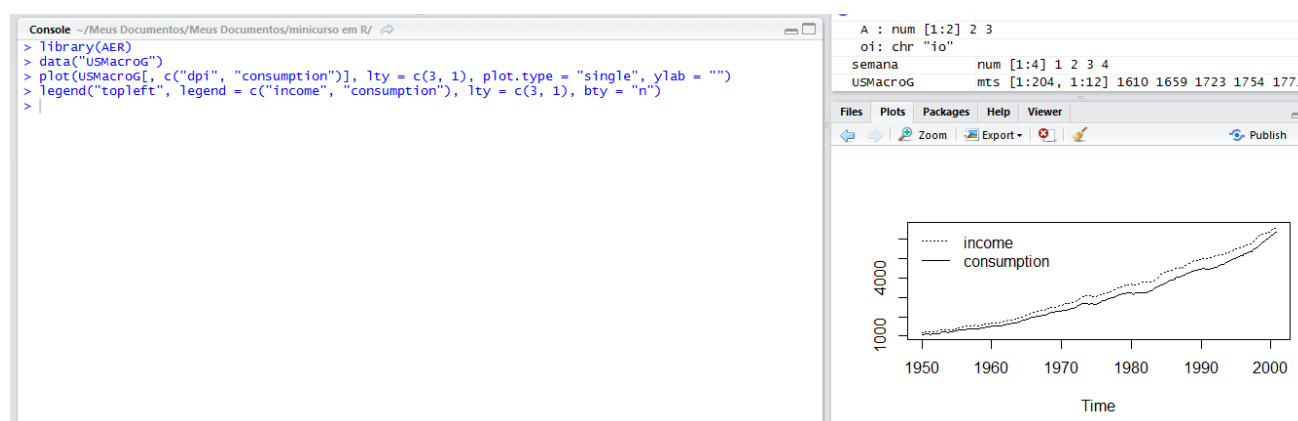


Eu não disse ainda, mas se você pode explorar as opções da aba *Plots*. Para não tomar mais nosso tempo (e espaço), deixo este exercício por sua conta e, com isto, encerramos a parte básica do R. A partir da próxima seção, passaremos às aplicações com análise de regressão e tudo aquilo que você esperava com ou sem casos extra-conjugais.

## 6. Regressão Linear Sistemas de Equações Simultâneas e Testes de Diagnósticos

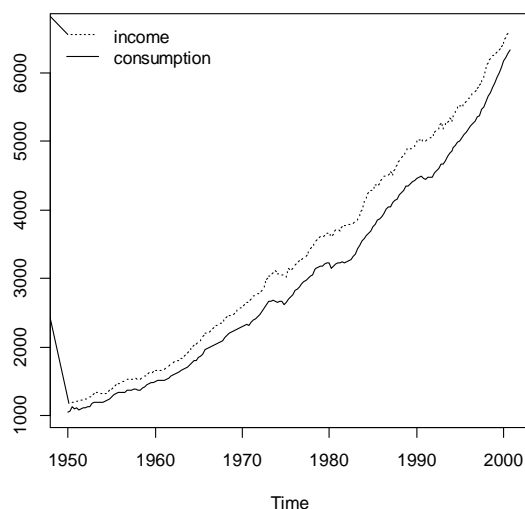
### 6.1. O eterno debate da função consumo (O Primeiro Ato)

Talvez o modelo mais intuitivo para um economista seja o da função consumo. Não apenas mais intuitivo como também o mais controverso. Nesta aula, estimamos algumas funções consumo tradicionais e algumas outras fornecidas como exemplos em Kleiber & Zeileis (2008). Os dados estão no pacote *AER*, e dizem a dados trimestrais dos EUA, em bilhões de dólares já analisados no livro-texto de Greene. Após carregar *AER*, incorporamos os dados: `data("USMacroG")`. Podemos pedir uma visão inicial dos dados com o comando `plot`<sup>15</sup>.



Podemos, como eu disse antes, trazer o gráfico separadamente como imagem para este documento. Ei-lo a seguir.

<sup>15</sup> Quantas observações temos para o *consumption*? Para saber esta informação, basta pedir o `length(consumption)`.



Usaremos os comandos “lm” e “dynlm” – e, eventualmente, “dyn” – para estimar as funções consumo. O comando “lm” já está no pacote básico do R, *stats*. Isto significa que você não precisa instalar nada para utilizar o comando. Já o “dynlm” (ou o “dyn”) deve ser instalado(s) previamente.

Dito isto, vejamos as regressões estimadas para três funções consumo. Primeiramente, para a hipótese tradicional keynesiana, temos (note que utilizamos o comando *lm*). Digitando a primeira linha abaixo, obtemos a estimação da função consumo keynesiana tradicional. A última linha nos retorna o resumo (*summary*, lembra?) dos resultados.

```
> legend("topleft", legend = c("income", "consumption"), lty = c
> cons_key <- lm(consumption ~dpi, data=USMacroG)
> summary(cons_key)
```

Call:  
lm(formula = consumption ~ dpi, data = USMacroG)

Residuals:

Min	1Q	Median	3Q	Max
-191.42	-56.08	1.38	49.53	324.14

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-80.354749	14.305852	-5.617	6.38e-08 ***
dpi	0.921686	0.003872	238.054	< 2e-16 ***

---  
signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 87.21 on 202 degrees of freedom  
Multiple R-squared: 0.9964, Adjusted R-squared: 0.9964  
F-statistic: 5.667e+04 on 1 and 202 DF, p-value: < 2.2e-16

Repare que a propensão marginal a consumir é próxima da unidade e o intercepto tem o impalatável valor negativo. Ambos são significativos ao nível de confiança de 1%.

Qual o número de observações utilizadas na regressão? Usando o comando `nobs()`, vemos que a regressão `cons_key` usou...204 observações.

```
> nobs(cons_key)
[1] 204
```

Veja o que acontece quando usamos o comando `names()`.

```
Residual standard error: 87.21 on 202 degrees of freedom
Multiple R-squared: 0.9964, Adjusted R-squared: 0.9964
F-statistic: 5.667e+04 on 1 and 202 DF, p-value: < 2.2e-16

> names(cons_key)
[1] "coefficients" "residuals" "effects" "rank"
[5] "fitted.values" "assign" "qr" "df.residual"
[9] "xlevels" "call" "terms" "model"
> |
```

A utilidade deste comando é que ele nos informa os nomes dos componentes do objeto em questão - no caso, `cons_key` - facilitando manipulações posteriores (por exemplo, construindo testes de hipótese manualmente). Experimente, por exemplo, digitar: `cons_key$coefficients` ou `cons_key$residuals`.

Seria interessante observar os resíduos e proceder a outros testes de diagnósticos. Faremos isso adiante. Antes, contudo, vamos estimar a função consumo estimada sob a hipótese de renda permanente (na qual a renda permanente esperada segue um ajuste de Koyck).

```
Console ~/meus documentos/meus documentos/minicurso em R / ~
> library("dynlm")
> cons_rp <- dynlm(consumption ~ dpi + L(consumption)-1, data=USMacroG )
> summary(cons_rp)

Time series regression with "ts" data:
Start = 1950(2), End = 2000(4)

Call:
dynlm(formula = consumption ~ dpi + L(consumption) - 1, data = USMacroG)

Residuals:
    Min       1Q   Median       3Q      Max
-101.247   -9.541    1.238   12.797   45.706

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
dpi          -0.003071   0.014982  -0.205   0.838
L(consumption) 1.012147   0.016748  60.433 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 21.47 on 201 degrees of freedom
Multiple R-squared: 1, Adjusted R-squared: 1
F-statistic: 2.459e+06 on 2 and 201 DF, p-value: < 2.2e-16

> |
```

Para esta estimação fizemos uso do comando `dynlm`. Isto porque é sabido que trabalhar com *lags* e *leads* no R é uma benção e uma maldição. Benção, porque você possui

formas diretas de criar *lags* e *leads* de variáveis<sup>16</sup>. Alguém poderia utilizar o comando *lm* da seguinte forma<sup>17</sup>:

```
cons_rp1 <-lm(consumption~dpi + lag(consumption, -1)-1, na.action=NULL,
data=USMacroG)
```

O resultado será, sim, distinto, para os comandos *dynlm* e *lm*, **exceto no caso em que a regressão não envolva defasagens**. Para fins de comparação, veja a tabela abaixo, com duas estimações da função consumo sob a hipótese da renda permanente.

Dependent variable:		
	consumption	
	dynamic linear (1)	OLS (2)
dpi	-0.003 (0.015)	0.000 (0.000)
L(consumption)	1.012*** (0.017)	
lag(consumption, -1)		1.000*** (0.000)
Observations	203	204
R2	1.000	1.000
Adjusted R2	1.000	1.000
Residual Std. Error	21.470 (df = 201)	0.000 (df = 202)
Note: *p<0.1; **p<0.05; ***p<0.01		

Perceba que, no modelo estimado pelo comando *dynlm*, há, como se esperaria, a perda de uma observação na estimação.

### Uma breve digressão que pode lhe confundir agora

O problema – a maldição – está na forma como o comando *lag* desloca as variáveis no tempo, o que exige algum tipo de tratamento prévio dos dados no sentido de alinhar a série com ela mesma defasada. Inicialmente, dizemos ao R que as variáveis são séries de tempo por meio do comando *as.ts*.

```
cons<-as.ts(USMacroG[, "consumption"])
```

```
dpi<- as.ts(USMacroG[, "dpi"])
```

Em seguida, geramos as defasagens das variáveis.

```
conslag=lag(cons,-1)
```

```
dpilag=lag(dpi,-1)
```

Finalmente, alinhamos conjuntamente (*tie*) as variáveis previamente.

```
consbi=ts.intersect(cons, conslag)
```

<sup>16</sup> Uma interessante aplicação de *leads* é o teste de causalidade de Granger-Sims.

<sup>17</sup> Para saber quantas observações há na regressão *cons\_rp1*, basta digitar *cons\_rp1\$N*. Como saber que atributos de uma saída de regressão podemos obter? Digite *names(cons\_rp1)* neste caso.

```
dpibi = ts.intersect(dpi, dpilag)
consbi
```

A vantagem de se adotar esta abordagem está na comparação de regressões por meio de testes de Wald. A desvantagem é que, neste caso, você tem que ter muito cuidado com os nomes das variáveis. Só para matar sua curiosidade, eis como seria a sintaxe para cada uma das regressões (keynesiana, renda permanente e consumo com hábito).

```
cons_keyn <- lm(consbi[,1] ~ dpibi[,1])
cons_rpn <- lm(consbi[,1] ~ dpibi[,1] + consbi[,2] - 1)
cons_habn <- lm(consbi[,1] ~ dpibi[,1] + consbi[,2])

summary(cons_keyn)
summary(cons_rpn)
summary(cons_habn)
```

No anexo desta aula entramos em mais detalhes sobre os testes de Wald para modelos dinâmicos no R. Por ora, voltemos à análise da função consumo com renda permanente estimada acima.

Neste caso, a renda corrente torna-se estatisticamente não-significativa e a autoregressividade do consumo parece ser a variável mais importante, indicando que a escolha intertemporal deve ser importante.

Outra função consumo poderia ser aquela na qual o consumidor tem *hábito*. Neste caso, o consumo é que segue uma movimentação no tempo ao estilo das defasagens de Koyck<sup>18</sup>. A forma reduzida estimada é similar à da renda permanente, exceto que, agora, há um intercepto (o que nos ajuda a racionalizar, economicamente, o famoso “consumo autônomo”).

---

<sup>18</sup> Uma outra forma de obter um intercepto na função consumo é calcular a demanda para uma função de utilidade Cobb-Douglas que apresente um nível mínimo de subsistência. Para um mesmo período de tempo, hábito ou subsistência seriam racionalizações similares para a função consumo do tipo  $C = a + bY$ .

```
> cons_hab <- dynlm(consumption ~ dpi + L(consumption), na.action=NULL, data=USmacroG )
> summary(cons_hab)
```

Time series regression with "ts" data:  
Start = 1950(2), End = 2000(4)

Call:  
dynlm(formula = consumption ~ dpi + L(consumption), data = USmacroG,  
na.action = NULL)

Residuals:

Min	1Q	Median	3Q	Max
-101.303	-9.674	1.141	12.691	45.322

Coefficients:

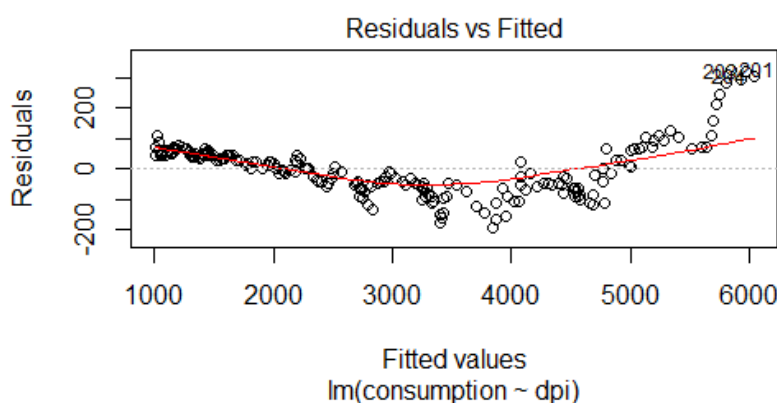
	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	0.535216	3.845170	0.139	0.889
dpi	-0.004064	0.016626	-0.244	0.807
L(consumption)	1.013111	0.018161	55.785	<2e-16 ***

---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

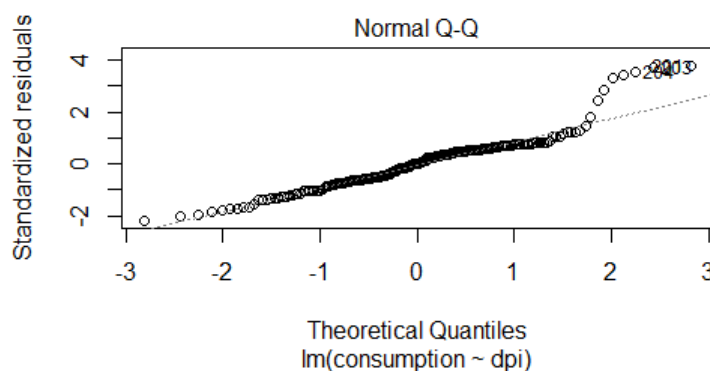
Residual standard error: 21.52 on 200 degrees of freedom  
Multiple R-squared: 0.9998, Adjusted R-squared: 0.9998  
F-statistic: 4.627e+05 on 2 and 200 DF, p-value: < 2.2e-16

Os resultados não parecem se diferenciar muito daqueles encontrados no modelo da renda permanente. Observe que a última equação estimada equivale a uma forma *irrestrita* que contém as outras duas funções consumo como casos particulares. Antes de prosseguir, vejamos alguns problemas típicos das regressões. A primeira coisa que deveria ser feita seria a análise dos resíduos. Vejamos, assim, alguns diagnósticos para cada uma das funções .

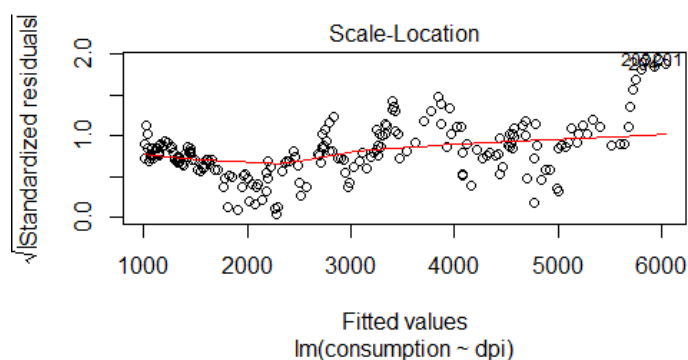
### Diagnósticos – Função Consumo Keynesiana



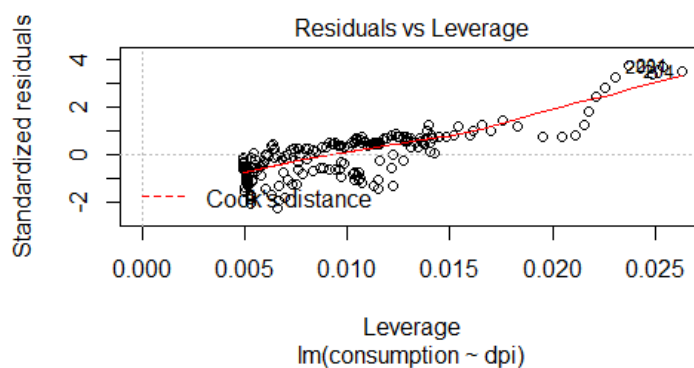
Vejamos cada um dos diagnósticos. O primeiro gráfico faz a correspondência entre valores estimados e resíduos. Conforme destacado por Kleiber & Zeileis (2008), a utilidade deste gráfico é nos dar uma ideia visual da hipótese de que os elementos da matriz X são não correlacionados com o erro (em nosso gráfico, as variações parecem sistemáticas ou aleatórias?).



Este é o famoso QQ. Por ele percebemos que os erros certamente não se encaixam em uma distribuição normal (repare nos quantis mais altos).



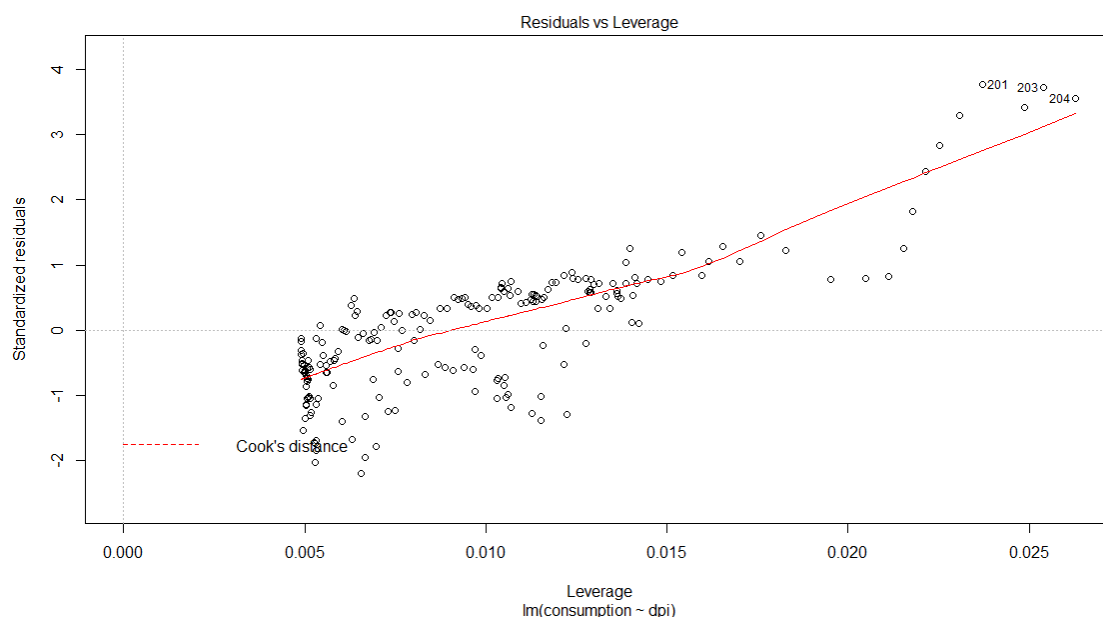
O gráfico acima nos dá a raiz quadrada dos erros padronizados e os valores ajustados. A ideia do gráfico é verificar evidências de heterocedasticidade.



O grau de *leverage* de uma observação indica o quanto ela se afasta da variância da matriz de variância-covariância. Para mais detalhes, veja Kleiber & Zeileis (2008), cap.4. Mas o que o diagnóstico nos diz é que as observações 201, 203 e 204 merecem nossa atenção. Repare que o gráfico acima está um pouco confuso e você deve estar pensando que tenho



poderes de *jedi* para distinguir entre 201, 203 e 204. O truque é pedir um *zoom* do gráfico. Veja como ele fica.



Outra forma de se pesquisar observações “não-usuais” é dada pelo conjunto de medidas de Belsley, DFFIT, DFBETA, COVRATIO e D<sup>2</sup>. As mesmas podem ser obtidas pelo comando *influence.measures()*. De acordo com o resumo destas medidas, as observações influentes seriam a 139 e 198 a 204 (ou seja, 1984.III e 1992.II – 2000.IV).

Vejamos o mesmo conjunto de gráficos para as outras duas funções consumo bem como os diagnósticos das medidas de influência<sup>19</sup>. Os comandos, algumas saídas e um resumo estão a seguir:

```
summary(influence.measures(cons_key))
```

```
summary(influence.measures(cons_rp))
```

```
summary(influence.measures(cons_hab))
```

<sup>19</sup> Faremos a discussão destas medidas caso haja tempo.

```
> summary(influence.measures(cons_key))
Potentially influential observations of
lm(formula = consumption ~ dpi, data = USmacroG) :

      dfb.1_ dfb.dpi dffit   cov.r   cook.d hat
139  0.02  -0.09  -0.18  0.97_*  0.02  0.01
198 -0.22  0.33   0.37_*  0.97   0.07  0.02
199 -0.26  0.39   0.44_*  0.95_*  0.09  0.02
200 -0.32  0.46   0.52_*  0.93_*  0.13  0.02
201 -0.37  0.54   0.61_*  0.89_*  0.17  0.02
202 -0.35  0.50   0.56_*  0.92_*  0.15  0.02
203 -0.39  0.56   0.62_*  0.90_*  0.18  0.03
204 -0.38  0.54   0.60_*  0.91_*  0.17  0.03
```

### Medidas de Influência

Função Consumo	Pontos influentes detectados
Keynesiana	1984.III e 1999.II – 2000.IV
Renda Permanente	1950.III, 1974.I, 1974.IV, 1980.II, 1981.IV, 1984.III, 1984.IV, 1985.II, 1990.IV, 1991.I, 1991.IV, 1996.IV, 1997.IV, 1998.III, 1999.I – 2000.IV
Hábito	1974.I, 1974.IV, 1980.II, 1981.IV, 1984.III, 1990.IV, 1991.I, 1991.IV, 1999.II – 2000.IV.

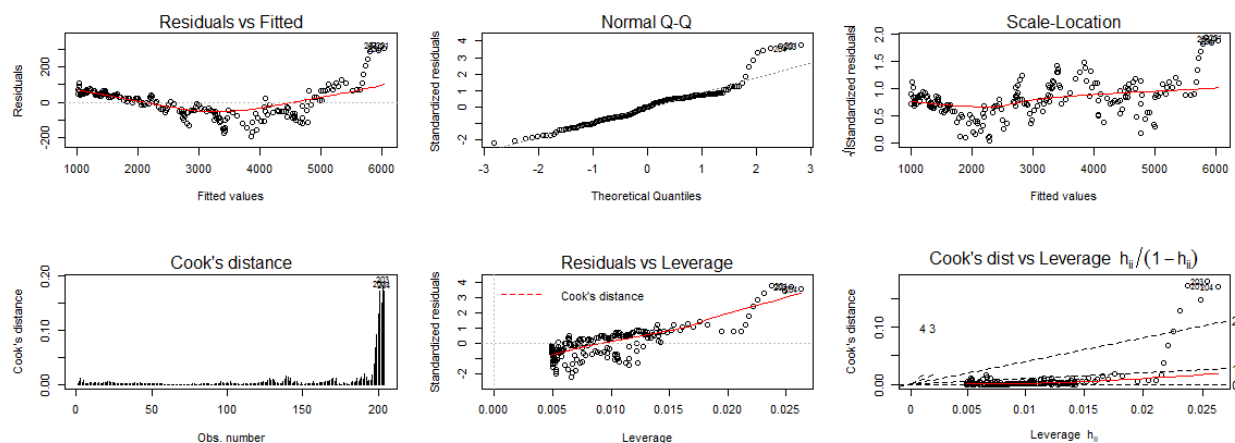
A estimação de funções sob a existência de pontos influentes pode ser feita considerando-se erros-padrão robustos. Isso será feito adiante. Para gerar o conjunto de gráfico a seguir, eis os comandos<sup>20</sup>:

```
par(mfrow=c(3,3))
```

```
plot(cons_key, which=1:6)
```

### Diagnósticos – Função Consumo Keynesiana

<sup>20</sup> Cuidado ao usar o RStudio. Para obter mais gráficos na mesma janela, você tem que ampliar o tamanho da mesma em detrimento das demais. Ao fazer isto, em nosso exemplo, descobre-se que dois gráficos adicionais de diagnóstico de resíduos estavam ocultos.



Fica como exercício para você gerar os conjuntos correspondentes às outras funções estimadas. Vejamos alguns outros testes de diagnósticos que não precisam de pacotes (já estão no pacote básico que é carregado por *default* no R, o *stats*)<sup>21</sup> (Breusch-Godfrey, Ljung-Box), o teste Breusch-Pagan e o RESET (carregados a partir do pacote *lmtest*).

```
> bgtest(cons_key)

Breusch-Godfrey test for serial correlation of order up to 1

data: cons_key
LM test = 185.27, df = 1, p-value < 2.2e-16

> Box.test(residuals(cons_key),type="Ljung-Box")

Box-Ljung test

data: residuals(cons_key)
X-squared = 176.28, df = 1, p-value < 2.2e-16

>
```

<sup>21</sup> Para o teste de autocorrelação, reporto Breusch-Godfrey e Ljung-Box. É possível também obter o Durbin-Watson. Entretanto, para modelos ADL, o teste de Breusch-Godfrey parece ser recomendado (vide Gujarati & Porter (2011)). O teste “h” de Durbin também poderia ser construído manualmente a partir dos resultados reportados.

```

> bptest(cons_key)

        studentized Breusch-Pagan test

data:  cons_key
BP = 42.52, df = 1, p-value = 6.998e-11

> reset(cons_key)

        RESET test

data:  cons_key
RESET = 542.66, df1 = 2, df2 = 200, p-value < 2.2e-16

> |

```

Você pode checar o restante das estatísticas com a tabela que construí abaixo.

#### Testes de Diagnóstico - Vários

Função Consumo	Breusch – Pagan (p-valor)	RESET	Breusch- Godfrey	Ljung- Box
Keynesiana	42.5195 (6.998e-11)	542.6619 (2.2e-16)	185.2714 (2.2e-16)	176.2777 (2.2e-16)
Renda Permanente	0.6182 (0.4317)	1.9413 (0.1651)	7.6098 (0.005805)	7.4906 (0.006202)
Hábito	3.7984 (0.1497)	2.1295 (0.1461)	7.5542 (0.005987)	7.4044 (0.006507)

Os resultados acima mostram que a função keynesiana parece ter problemas de heterocedasticidade. Seria prudente reestimar as equações utilizando algum estimador HC para obtermos erros padrão robustos. Vejamos os resultados para os três modelos com HC4, já que há observações influentes.

```

Console ~/Meus Documentos/Meus Documentos/minicurso em R/
> library(lmtest)
> coeftest(cons_key, vcov=vcovHC, type="HC4")

t test of coefficients:

              Estimate Std. Error t value Pr(>|t|)
(Intercept) -80.3547488  15.3362349  -5.2395 4.034e-07 ***
dpi          0.9216857   0.0056234  163.9022 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> coeftest(cons_rp, vcov=vcovHC, type="HC4")

t test of coefficients:

              Estimate Std. Error t value Pr(>|t|)
dpi          -0.0030715  0.0150578  -0.204  0.8386
L(consumption) 1.0121469  0.0168058  60.226 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> coeftest(cons_hab, vcov=vcovHC, type="HC4")

t test of coefficients:

              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.5352156  3.1591518  0.1694  0.8656
dpi          -0.0040641  0.0165039  -0.2463  0.8057
L(consumption) 1.0131107  0.0180623  56.0898 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Pode-se observar que os erros-padrões não se alteram muito, exceto, talvez, no caso da função consumo keynesiana. Finalmente, vale notar que um problema importante destas estimações diz respeito ao fato de os dados estarem no domínio do tempo. Por que?

Bem, note que a função consumo keynesiana apresenta evidências de não-estacionaridade, medida pela aplicação da regra de bolso (" $R^2 > DW$ "). O comando `dwtest(cons_key)` nos mostra que esta desigualdade seria, para este modelo, " $0.9964 > 0.092$ "<sup>22</sup>. Assim, antes de chegarmos a alguma conclusão sobre a função consumo, deveríamos considerar a modelagem da função consumo conforme as características temporais das séries, algo que será visto nas últimas aulas.

Como sabemos, a função consumo não é apenas a única explicação de toda história do ciclo econômico. O investimento agregado, a balança comercial, etc também são extremamente importantes. O problema econométrico, portanto, é o de saber como estimar relações levando em conta o aspecto sistêmico de alguns fenômenos econômicos. Em poucas palavras, é o que você já estudou na econometria de sistemas de equações (recursivas ou simultâneas). Para sistemas de equações simultâneas, um pacote geralmente utilizado é a *systemfit*. Entretanto, os resultados acima praticamente pedem por uma comparação entre os três modelos. Afinal, o modelo do consumo com hábito pode ser tido como um caso irrestrito, tendo como subconjuntos os outros dois modelos.

<sup>22</sup> O teste de Durbin Watson é o que você quer, não? O pacote *lmtest* possui o comando `dwtest()`. Assim, o que fiz foi: `dwtest(cons_key)`, obtendo o valor reportado.

Entretanto, esta comparação só é possível se o número de observações utilizadas for o mesmo. O uso do pacote *dynlm*, embora facilite nossa análise no curto prazo, não é útil neste caso. A maneira correta de se proceder, assim, é alinhar conjuntamente as variáveis (eliminando células vazias) e depois fazer as regressões. Vimos como juntar as variáveis e como estima-las no *box* desta seção. Quanto aos testes de Wald, temos:

```
> waldtest(cons_rpn, cons_habn, cons_keyn)
wald test

Model 1: consbi[, 1] ~ dpibi[, 1] + consbi[, 2] - 1
Model 2: consbi[, 1] ~ dpibi[, 1] + consbi[, 2]
Model 3: consbi[, 1] ~ dpibi[, 1]
      Res.Df Df      F Pr(>F)
1         201
2         200  1    0.0194 0.8894
3         201 -1 3111.9666 <2e-16 ***
---
signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A primeira comparação é entre o modelo de renda permanente e o de hábito e a segunda, entre o de hábito e a função keynesiana. Embora a função keynesiana se saia pior do que o modelo de hábito, esta parece ser superada pela função com renda permanente. Obviamente, a análise não é conclusiva ainda pois, como vimos, os dados apresentam características próprias típicas de séries de tempo.

Outra forma de se analisar cada modelo, separadamente, é fazer um teste *Type I* ANOVA sobre o mesmo. O teste faz, simplesmente, uma análise sequencial, com sucessivas inclusões das variáveis independentes. Exemplificando para o modelo de consumo com hábito, temos:

```
> anova(cons_habn)
Analysis of Variance Table

Response: consbi[, 1]
      Df Sum Sq Mean Sq F value    Pr(>F)
dpibi[, 1]  1 427222555 427222555  922287 < 2.2e-16 ***
consbi[, 2]  1  1441528  1441528    3112 < 2.2e-16 ***
Residuals    200    92644     463
---
signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Legal, né?

## Anexo 1 – A questão das observações influentes

Considere a seguinte função consumo estimada para dados brasileiros (a saída apresentada está em R, não em RStudio):

```
cons_hab3<-dynlm(cons~dpi+L(cons,1), data=basebr)

summary(cons_hab3)
```

```

> cons_hab3<-dynlm(cons~dpi+L(cons,1), data=base)
> summary(cons_hab3)

Time series regression with "ts" data:
Start = 1954(2), End = 2004(4)

Call:
dynlm(formula = cons ~ dpi + L(cons, 1), data = base)

Residuals:
    Min       1Q   Median       3Q      Max
-101.303   -9.674    1.141   12.691   45.322

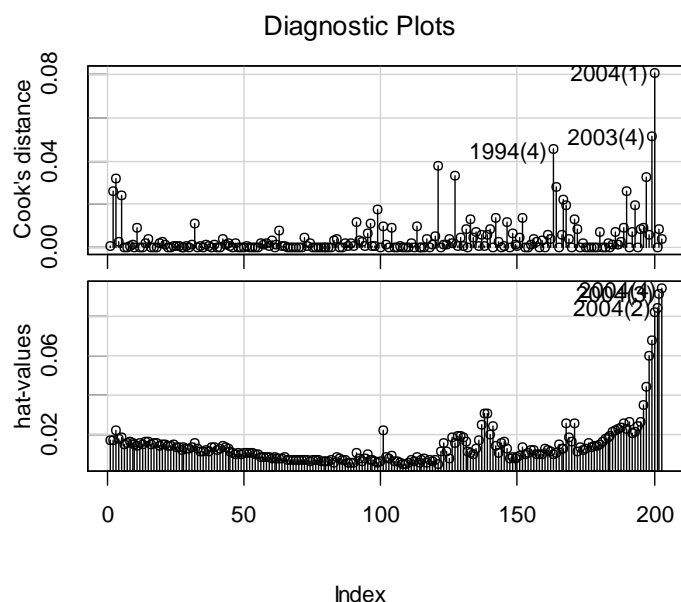
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.535216   3.845170   0.139   0.889
dpi          -0.004064   0.016626  -0.244   0.807
L(cons, 1)    1.013111   0.018161  55.785 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 21.52 on 200 degrees of freedom
Multiple R-squared:  0.9998,    Adjusted R-squared:  0.9998
F-statistic: 4.627e+05 on 2 and 200 DF,  p-value: < 2.2e-16

```

Uma forma de se identificar pontos não-usuais da amostra é usar o pacote *car* e o comando *InfluenceIndexPlot*. No exemplo a seguir, pede-se que o R nos mostre, no gráfico, os três pontos mais distantes da média da variável do eixo horizontal do gráfico nos critérios especificados.

```
influenceIndexPlot(cons_hab3,vars=c("Cook", "hat"), id.n=3)
```



Seria interessante, assim, discutir a origem destes deslocamentos na discussão da função consumo keynesiana<sup>23</sup>.

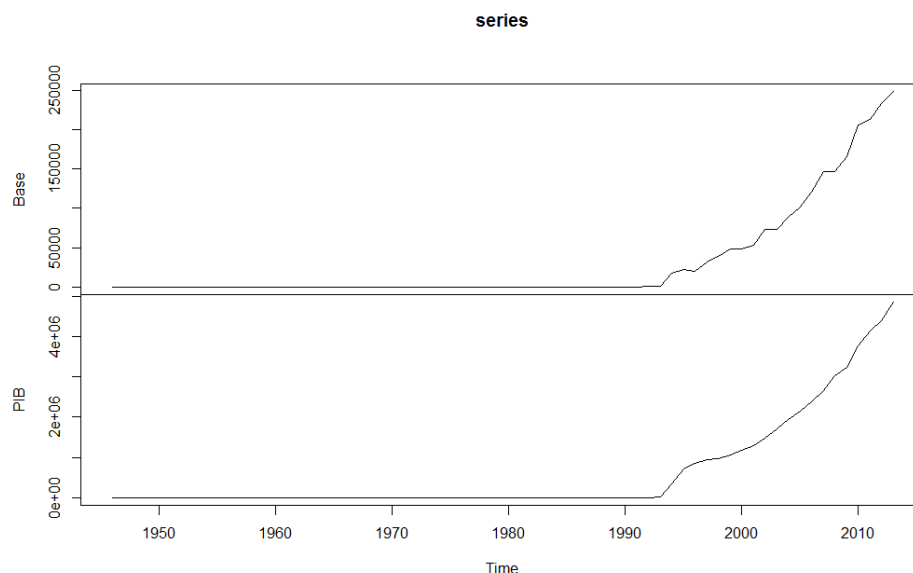
## 6.2. O PIB, a Base Monetária e o Teste de Hipóteses

Um exercício que anima muita gente em Macroeconomia é tentar testar hipóteses polêmicas. Quem nunca brigou com o amigo, a namorada ou o vizinho por conta da proposição da neutralidade da moeda? Duvido que haja alguém que tenha cursado Ciências Econômicas e passado incólume por esta questão.

Gujarati & Porter (2010) apresentam um exemplo de teste interessante para este problema, por meio dos modelos ADL (*autoregressive distributed lags*). A ideia é a seguinte: sabemos que a teoria é suficientemente abstrata para não nos dizer se a neutralidade da moeda se dá em amostras de dados *cross-section*, de séries de tempo, etc. Assim, a proposta de teste dos autores é simples: em um modelo ADL, supondo que a causalidade seja da moeda para o PIB, a moeda seria neutra se os coeficientes defasados da moeda não forem estatisticamente significativos.

Obviamente, quem já passou pela graduação, sabe que este tipo de protocolo de teste não é imune a críticas (a própria exogeneidade suposta *ad hoc* poderia ser questionada) mas não estamos em um debate profundo neste momento (graças a Deus!).

Considere, então, a replicação do exercício com dados brasileiros. Temos o PIB do Brasil e a Base Monetária, anuais, ambos no período 1946-2013. Vejamos os dados.



Veja, agora, a estimação do modelo com três defasagens e um termo autoregressivo.

<sup>23</sup> Segundo Bollen & Jackman (1990), pontos com distâncias de Cook acima de  $4/n$  ( $n$  = tamanho da amostra) devem ser analisados com cautela. Agradeço a Ari Francisco de Araujo Jr por esta observação.



```

> reg<-dynlm(PIB~Base+L(Base)+L(Base,2)+L(Base,3)+L(PIB), data=series)
> coeftest(reg)

t test of coefficients:

              Estimate Std. Error t value Pr(>|t|)
(Intercept) 8480.706694 7707.072241  1.1004 0.2756361
Base          7.946277    1.453447  5.4672 9.709e-07 ***
L(Base)       1.092258    1.654417  0.6602 0.5116903
L(Base, 2)    -6.450371    1.660887 -3.8837 0.0002626 ***
L(Base, 3)    -1.370143    1.542555 -0.8882 0.3780244
L(PIB)        0.946044    0.072117 13.1181 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> linearHypothesis(reg,matchCoefs(reg,"Base"))
Linear hypothesis test

Hypothesis:
Base = 0
L(Base) = 0
L(Base, 2) = 0
L(Base, 3) = 0

Model 1: restricted model
Model 2: PIB ~ Base + L(Base) + L(Base, 2) + L(Base, 3) + L(PIB)

      Res.Df    RSS Df Sum of Sq    F    Pr(>F)
1         63 3.131e+11
2         59 1.623e+11  4  1.508e+11 13.705 5.805e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>

```

Ao invés de pedir o sumário, desta vez usei o comando *coeftest*. Vemos que a base monetária tem algum efeito, mas ele não existe para a primeira e nem para a terceira defasagem. Assim, usamos o comando *linearHypothesis* do pacote *car* para testar a hipótese de que os coeficientes da Base são igualmente iguais a zero. O teste nos diz que, em seu conjunto, os coeficientes da Base não são nulos (com p-valor bem baixo).

Será que a moeda não é mesmo neutra? Bem, esta é uma questão que exige mais do que este exemplo consegue responder<sup>24</sup>.

### 6.3. Morar longe da faculdade diminui seu salário?

Antes que você fique inquieto,<sup>25</sup> trata-se apenas de uma aplicação de variáveis instrumentais e do método dos MQ2E. Talvez a distância da faculdade (ou do colégio, como no exemplo) possa dificultar a obtenção de seu diploma, mas será que afetará seu salário? A distância, assim, está sendo utilizada como instrumento para “educação”.

Vejamos o exemplo em dois estágios e depois, com o comando *ivreg* (pacote *AER*) e, finalmente, com o comando *tsls* (pacote *sem*). As outras variáveis são *urbanyes* (sim, se o

<sup>24</sup> Fosse eu um aluno de graduação, neste momento estaria pedindo *dummies*, reclamando da exogeneidade ad hoc, da falta de análise dos resíduos, da ausência dos erros-padrão robustos, etc. Fosse eu um simpatizante (ou não) da tese da neutralidade da moeda no longo prazo pediria testes com diferentes especificações de defasagens. Fosse eu um aluno de pós-graduação, estaria recitando o mantra: “raiz unitária-cointegração” e reclamando de outros detalhes. Enfim, se alguém quiser testar isto, em algum nível do conhecimento (desde que seja para obter avanços marginais em seu capital humano), sinta-se à vontade para seguir em frente com este tema. Não tem muito a ver com o R em si, mas esta discussão sempre me provoca.

<sup>25</sup> Originalmente em: <http://r.789695.n4.nabble.com/instrumental-variables-regression-using-ivreg-AER-or-tsls-sem-td4651366.html>.

colégio se localiza na área urbana), *genderfemale*, *ethnicityyafam* (afro-americano), *ethnicityhispanic*, *unemp* (taxa de desemprego do condado em 1980), *education* (número de anos de educação), *distance* (distância de um *college* de 4 anos, em 10 milhas) e *wage* (salário-hora estadual na indústria, 1980).

Primeiro, vejamos o primeiro estágio.

```

93 library(AER)
94 library(sem)
95 library(lmtest)
96 data("CollegeDistance")
97 cd.d<-CollegeDistance
98 reg.simples<-lm(education~urban+gender+ethnicity+unemp+distance, data=cd.d)
99 summary(reg.simples)
100 |

```

100:1 (Top Level) R Script

Console ~/Meus Documentos/Meus Documentos/minicurso em R/

```

lm(formula = education ~ urban + gender + ethnicity + unemp +
    distance, data = cd.d)

Residuals:
    Min       1Q   Median       3Q      Max
-2.1742 -1.7035 -0.4755  1.8914  4.5255

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  14.060680   0.083075  169.253 < 2e-16 ***
urbanyes     -0.092308   0.065039  -1.419   0.156
genderfemale -0.024645   0.051731  -0.476   0.634
ethnicityafam -0.524317   0.072444  -7.238 5.30e-13 ***
ethnicityhispanic -0.274761  0.067879  -4.048 5.25e-05 ***
unemp         0.010267   0.009768   1.051   0.293
distance     -0.086846   0.012244  -7.093 1.51e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.77 on 4732 degrees of freedom
Multiple R-squared:  0.02298, Adjusted R-squared:  0.02174
F-statistic: 18.55 on 6 and 4732 DF, p-value: < 2.2e-16

```

Desta regressão obtemos a previsão de *education* para usá-la na equação do salário. O comando é: `cd.d$ed.pred<- predict(reg.simples)`. O segundo estágio, assim:

```

100 cd.d$ed.pred<- predict(reg.simples)
101 segundo.estagio<-lm(wage~urban+gender+ethnicity+unemp+ed.pred, data=cd.d)
102 summary(segundo.estagio)
103 |

```

103:1 (Top Level) ↕ R Script

Console ~/Meus Documentos/Meus Documentos/minicurso em R/ ↕

```

lm(formula = wage ~ urban + gender + ethnicity + unemp + ed.pred,
    data = cd.d)

Residuals:
    Min       1Q   Median       3Q      Max
-3.1692 -0.8294  0.1502  0.8482  3.9537

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.359032    1.412087  -0.254  0.79931
urbanyes      0.046144    0.044691   1.033  0.30188
genderfemale -0.070753    0.036978  -1.913  0.05576 .
ethnicityafam -0.227240    0.072984  -3.114  0.00186 **
ethnicityhispanic -0.351291    0.057021  -6.161 7.84e-10 ***
unemp         0.139163    0.006748  20.622 < 2e-16 ***
ed.pred       0.647099    0.100592   6.433 1.38e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.263 on 4732 degrees of freedom
Multiple R-squared:  0.1175,    Adjusted R-squared:  0.1163
F-statistic: 105 on 6 and 4732 DF,  p-value: < 2.2e-16

```

Agora vejamos o resultado por meio do método de variáveis instrumentais.

```

104 m <- ivreg(wage ~ urban + gender + ethnicity + unemp + education |
105             urban + gender + ethnicity + unemp + distance, data = cd.d)
106 <- summary(m)
107 |

```

107:1 (Top Level) ↕ R Script

Console ~/Meus Documentos/Meus Documentos/minicurso em R/ ↕

```

Call:
ivreg(formula = wage ~ urban + gender + ethnicity + unemp + education |
      urban + gender + ethnicity + unemp + distance, data = cd.d)

Residuals:
    Min       1Q   Median       3Q      Max
-5.20896 -1.14578 -0.02361  1.33303  4.77571

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.35903    1.90830  -0.188  0.8508
urbanyes      0.04614    0.06039   0.764  0.4449
genderfemale -0.07075    0.04997  -1.416  0.1569
ethnicityafam -0.22724    0.09863  -2.304  0.0213 *
ethnicityhispanic -0.35129    0.07706  -4.559 5.28e-06 ***
unemp         0.13916    0.00912  15.259 < 2e-16 ***
education     0.64710    0.13594   4.760 1.99e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.706 on 4732 degrees of freedom
Multiple R-Squared: -0.6118,    Adjusted R-squared: -0.6138
Wald test: 57.48 on 6 and 4732 DF,  p-value: < 2.2e-16

```

Vejamos o mesmo resultado pelo método MQ2E.

```

m2 <- tsls(wage ~ urban + gender + ethnicity + unemp + education,
           ~ urban + gender + ethnicity + unemp + distance, data = cd.d)

```

```

108 m2 <- tsls(wage ~ urban + gender + ethnicity + unemp + education,
109             ~ urban + gender + ethnicity + unemp + distance, data = cd.d)
110 summary(m2)
111 |

```

111:1 (Top Level) ↕ R Scri

Console ~/Meus Documentos/Meus Documentos/minicurso em R/ ↗

```

> summary(m2)

2SLS Estimates

Model Formula: wage ~ urban + gender + ethnicity + unemp + education

Instruments: ~urban + gender + ethnicity + unemp + distance

Residuals:
    Min.    1st Qu.    Median      Mean    3rd Qu.     Max.
-5.20900 -1.14600 -0.02361  0.00000  1.33300  4.77600

      Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.359031993  1.908299652  -0.18814  0.850773
urbanyes     0.046144402  0.060395344   0.76404  0.444882
genderfemale -0.070752726  0.049971930  -1.41585  0.156885
ethnicityafam -0.227239937  0.098630957  -2.30394  0.021269 *
ethnicityhispanic -0.351290603  0.077058167  -4.55877  5.2751e-06 ***
unemp        0.139162515  0.009119739  15.25948 < 2.22e-16 ***
education    0.647098596  0.135940586   4.76016  1.9921e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.7061773 on 4732 degrees of freedom

```

Repare que, em ambos os casos, os instrumentos são as variáveis exógenas (que são instrumentos delas mesmas) *urban*, *gender*, *ethnicity*, *unemp* e *distance* (que é instrumento da *education*). Os resultados são interessantes? Talvez não para os afro-americanos ou os hispânicos. A educação tem o efeito esperado sobre os salários e o desemprego parece ter um efeito invertido ao usual (positivo) o que pode ser sinal de problemas de especificação (ou, alternativamente, temos que pensar em uma explicação melhor para o resultado...).

Será que *education* é realmente endógena? Vamos testar isto! No entanto, utilizaremos mais de uma variável como instrumento. Adicionalmente à distância da faculdade usaremos *score* que é uma medida de desempenho médio destes alunos no ensino médio. Primeiramente verificaremos se *education* é correlacionada com a *distância* e com o *score*:

```

> corr_variaveis = data.frame(cbind(cd.d$education, cd.d$distance, cd.d$score))
> colnames(corr_variaveis) = cbind("education", "distance", "score")
> cor(corr_variaveis)
      education distance score
education 1.00000000 -0.09318309 0.46518719
distance -0.09318309 1.00000000 -0.06797927
score      0.46518719 -0.06797927 1.00000000
> |

```

Agora devemos regressar educação contra todas as variáveis exógenas *urban*, *gender*, *ethnicity*, *unemp* adicionando seus possíveis instrumentos *distance* e *score*.

```

116 reg2 <- lm(education~urban + gender + ethnicity + unemp +
117             distance+score,data=cd.d)
118 summary(reg2)
119

```

119:1 (Top Level) ↕

Console ~/Meus Documentos/Meus Documentos/minicurso em R/ ↗

Call:  
lm(formula = education ~ urban + gender + ethnicity + unemp +  
distance + score, data = cd.d)

Residuals:

Min	1Q	Median	3Q	Max
-3.9101	-1.1694	-0.2191	1.2107	5.2813

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	8.605052	0.172206	49.970	< 2e-16	***
urbanyes	-0.018997	0.057980	-0.328	0.743190	
genderfemale	0.100044	0.046223	2.164	0.030486	*
ethnicityafam	0.255808	0.068262	3.747	0.000181	***
ethnicityhispanic	0.226996	0.062140	3.653	0.000262	***
unemp	0.008922	0.008702	1.025	0.305276	
distance	-0.049178	0.010961	-4.487	7.41e-06	***
score	0.099971	0.002849	35.087	< 2e-16	***

---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.576 on 4731 degrees of freedom  
Multiple R-squared: 0.2247, Adjusted R-squared: 0.2236  
F-statistic: 195.9 on 7 and 4731 DF, p-value: < 2.2e-16

Precisaremos dos resíduos desta regressão, obtemo-los da seguinte forma:

$$v2.hat = reg2\$residuals$$

O próximo passo é utilizar estes resíduos na nossa equação original e verificarmos se eles são estatisticamente diferentes de zero, como segue:

$$reg3 = lm(wage \sim urban + gender + ethnicity + unemp + v2.hat, data=cd.d)$$

$$summary(reg3)$$

```

121 reg3 <- lm(wage ~ urban + gender + ethnicity + unemp + v2.hat,data=cd.d)
122 summary(reg3)
123

```

123:1 (Top Level) ↕

Console ~/Meus Documentos/Meus Documentos/minicurso em R/ ↗

Call:  
lm(formula = wage ~ urban + gender + ethnicity + unemp + v2.hat,  
data = cd.d)

Residuals:

Min	1Q	Median	3Q	Max
-3.3940	-0.8600	0.1723	0.8073	3.9865

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	8.716798	0.059399	146.750	<2e-16	***
urbanyes	0.070318	0.044708	1.573	0.1158	
genderfemale	-0.085363	0.037054	-2.304	0.0213	*
ethnicityafam	-0.558807	0.051877	-10.772	<2e-16	***
ethnicityhispanic	-0.545620	0.048552	-11.238	<2e-16	***
unemp	0.133050	0.006707	19.836	<2e-16	***
v2.hat	-0.023114	0.011689	-1.977	0.0481	*

---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.268 on 4732 degrees of freedom  
Multiple R-squared: 0.1105, Adjusted R-squared: 0.1094  
F-statistic: 97.96 on 6 and 4732 DF, p-value: < 2.2e-16

Observe que realmente estávamos corretos “a 10% de significância” em nossas suspeitas, *education* realmente é endógeno<sup>26</sup>.

#### 6.4. Mais armas, menos crimes...com *dummies*

O que você prefere? Andar armado ou deixar que apenas os que têm porte de armas o façam? Qual seria o impacto do porte de armas mais “liberal” sobre a violência? Talvez não exista um exemplo mais polêmico sobre políticas públicas do que este. Embora haja uma discussão muito importante sobre o tema, vou usar a base de dados para ilustrar o uso de *dummies* no R. Basicamente, uma *dummy*, no R, é uma variável do tipo “fator” (*factor*). Vejamos, por exemplo, como é a base original.

```
> library(AER)
> data(Guns)
> head(Guns)
```

	year	violent	murder	robbery	prisoners	afam	cauc	male	population
1	1977	414.4	14.2	96.8	83	8.384873	55.12291	18.17441	3.780403
2	1978	419.1	13.3	99.1	94	8.352101	55.14367	17.99408	3.831838
3	1979	413.3	13.2	109.5	144	8.329575	55.13586	17.83934	3.866248
4	1980	448.5	13.2	132.1	141	8.408386	54.91259	17.73420	3.900368
5	1981	470.5	11.9	126.5	149	8.483435	54.92513	17.67372	3.918531
6	1982	447.7	10.6	112.0	183	8.514000	54.89621	17.51052	3.925229

	income	density	state	law
1	9563.148	0.0745524	Alabama	no
2	9932.000	0.0755667	Alabama	no
3	9877.028	0.0762453	Alabama	no
4	9541.428	0.0768288	Alabama	no
5	9548.351	0.0771866	Alabama	no
6	9478.919	0.0773185	Alabama	no

Perceba que as duas últimas variáveis, *state* e *law* são não-numéricas e já estão, automaticamente, classificadas como fatores.

Variable	Class	Levels
year	num	
violent	num	
murder	num	
robbery	num	
prisoners	num	
afam	num	
cauc	num	
male	num	
population	num	
income	num	
density	num	
state	Factor	w/ 51 levels "Alabama", "Alaska", ...
law	Factor	w/ 2 levels "no", "yes"

O que seria preciso para fazer a regressão? Praticamente nada. Eis os comandos que utilizarei.

```
exemplo <- lm(log(violent) ~ law + prisoners + density + income +
               population + afam + cauc + male + state, data = Guns)
summary(exemplo)
```

<sup>26</sup> Acerca dos testes de exogeneidade neste contexto, ver, por exemplo, Stock & Watson (2010).



Vejamos uma parcial da tabela de resultados.

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	4.037e+00	3.894e-01	10.368	< 2e-16	***
lawyes	-4.614e-02	1.887e-02	-2.446	0.014613	*
prisoners	-7.101e-05	9.360e-05	-0.759	0.448240	
density	-1.723e-01	8.504e-02	-2.026	0.042994	*
income	-9.204e-06	5.908e-06	-1.558	0.119572	
population	1.152e-02	8.724e-03	1.321	0.186756	
afam	1.043e-01	1.776e-02	5.873	5.65e-09	***
cauc	4.086e-02	5.074e-03	8.052	2.07e-15	***
male	-5.027e-02	6.404e-03	-7.851	9.68e-15	***
stateAlaska	5.596e-02	7.272e-02	0.770	0.441712	
stateArizona	2.404e-01	8.877e-02	2.708	0.006867	**
stateArkansas	-1.273e-01	6.648e-02	-1.915	0.055811	.
stateCalifornia	2.442e-01	2.304e-01	1.060	0.289393	
stateColorado	-1.051e-01	1.122e-01	-0.937	0.349061	
stateConnecticut	-9.556e-02	1.348e-01	-0.709	0.478622	
stateDelaware	9.760e-02	7.674e-02	1.272	0.203682	
stateDistrict of Columbia	2.759e+00	7.797e-01	3.539	0.000418	***
stateFlorida	6.771e-01	1.135e-01	5.965	3.28e-09	***
stateGeorgia	2.253e-02	5.385e-02	0.418	0.675707	
stateHawaii	-1.128e+00	2.590e-01	-4.354	1.46e-05	***

Repare que a variável *law* aparece como *lawyes* e algo similar ocorre com os estados dos EUA. Claro que o valor *no* da variável *law* é a categoria de referência (e algum dos estados cumpre esta função de forma similar, claro). Podemos redefinir a referência, por exemplo, para a variável *law*, simplesmente digitando o seguinte.

```
exemplo <- lm(log(violent) ~ law + prisoners + density + income +
               population + afam + cauc + male+state, data = Guns)
summary(exemplo)

Guns$law<-relevel(Guns$law,"yes")
```

Uma visão parcial da saída da nova regressão nos mostra que tivemos sucesso aqui.

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	3.991e+00	3.960e-01	10.078	< 2e-16	***
lawno	4.614e-02	1.887e-02	2.446	0.014613	*
prisoners	-7.101e-05	9.360e-05	-0.759	0.448240	
density	-1.723e-01	8.504e-02	-2.026	0.042994	*
income	-9.204e-06	5.908e-06	-1.558	0.119572	
population	1.152e-02	8.724e-03	1.321	0.186756	
afam	1.043e-01	1.776e-02	5.873	5.65e-09	***
cauc	4.086e-02	5.074e-03	8.052	2.07e-15	***
male	-5.027e-02	6.404e-03	-7.851	9.68e-15	***
stateAlaska	5.596e-02	7.272e-02	0.770	0.441712	

Podemos aproveitar para ilustrar outro ponto interessante do R: os termos não-lineares. Digamos que haja motivos teóricos para imaginar que a violência seja impactada negativamente pela renda mas que também existam efeitos de segunda ordem. Basta usar a função *I()*.

```
exemplo2 <- lm(log(violent) ~ law + prisoners + density + income + I(income^2)+
               population + afam + cauc + male+state, data = Guns)
summary(exemplo2)
```

Eis a visão parcial dos resultados.

```

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.176e+00  3.963e-01  8.014 2.80e-15 ***
lawno         7.761e-02  1.869e-02  4.153 3.53e-05 ***
prisoners     1.540e-05  9.140e-05  0.168 0.866255
density      -3.700e-01  8.581e-02 -4.312 1.76e-05 ***
income        2.035e-04  2.599e-05  7.828 1.15e-14 ***
I(income^2)   -6.810e-09  8.118e-10 -8.388 < 2e-16 ***
population    4.498e-03  8.506e-03  0.529 0.597021
afam          9.560e-02  1.726e-02  5.539 3.80e-08 ***
cauc          2.327e-02  5.351e-03  4.349 1.49e-05 ***
male         -3.040e-02  6.649e-03 -4.572 5.36e-06 ***
stateAlaska   -5.926e-03  7.094e-02 -0.084 0.933444

```

Será que este último modelo está especificado corretamente? O teste RESET é o adequado para se fazer o teste. Vamos comparar este modelo - “*exemplo2*”- com o original (“*exemplo*”). Para fazer isto, precisaremos do comando *linearHypothesis* do pacote *car*.

```

> linearHypothesis(exemplo2, matchCoefs(exemplo2, "income"))
Linear hypothesis test

Hypothesis:
income = 0
I(income^2) = 0

Model 1: restricted model
Model 2: log(violent) ~ law + prisoners + density + income + I(income^2) +
         population + afam + cauc + male + state

   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
1   1115 28.839
2   1113 27.066  2     1.7738 36.471 4.554e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> |

```

Outro uso da *dummy* é na forma de termo de interação. Assim, suponha, apenas para fins de ilustração, que tenhamos motivos para imaginar que o coeficiente da variável *density* possa ser alterado conforme a lei assuma um de seus valores (vale lembrar que, no caso, “1” diz respeito ao fato de que a lei que permite o porte de armas está ativo no estado em questão). Como fazer esta estimação?

```

exemplo3 <- lm(log(violent) ~ law + law*(density)+ prisoners + density + income + I(income^2)+
               population + afam + cauc + male+state, data = Guns)
summary(exemplo3)
```

O resultado? Vejamos (a tabela é extensa, note que resumi um pouco...).



Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	2.890e+00	3.979e-01	7.263	7.12e-13	***
lawno	1.486e-01	2.424e-02	6.130	1.22e-09	***
density	5.140e-01	2.125e-01	2.419	0.015708	*
prisoners	2.816e-05	9.065e-05	0.311	0.756135	
income	1.964e-04	2.581e-05	7.610	5.81e-14	***
I(income^2)	-6.734e-09	8.049e-10	-8.366	< 2e-16	***
population	-4.084e-03	8.641e-03	-0.473	0.636595	
afam	1.111e-01	1.745e-02	6.370	2.76e-10	***
statevermont	-9.798e-01	1.333e-01	-7.352	3.78e-13	***
statevirginia	-5.848e-01	5.923e-02	-9.873	< 2e-16	***
statewashington	4.602e-02	1.016e-01	0.453	0.650569	
statewest virginia	-6.603e-01	1.223e-01	-5.401	8.11e-08	***
statewisconsin	-6.041e-01	1.147e-01	-5.267	1.66e-07	***
statewyoming	-3.425e-01	1.183e-01	-2.896	0.003850	**
lawno: density	-9.059e-01	1.995e-01	-4.541	6.22e-06	***

---  
 signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1546 on 1112 degrees of freedom  
 Multiple R-squared: 0.9456, Adjusted R-squared: 0.9427

No caso, vemos que o efeito de interação é significativo o que significa que quando a *law* varia, o coeficiente associado à variável *density* também varia (estados nos quais a lei que permite o porte de armas não é adotada, ou seja, *law* assume valor “0”, são tais que o impacto da densidade sobre a violência são menores (aproximadamente:  $-0.5 - 0.9 = -1.4$ ). Repare que este efeito de interação poderia não existir mas é justamente isto que está sendo testado.

Um exercício interessante, contudo, é considerar os efeitos de painel desta base de dados. Sim, eu não disse nada, mas o que foi feito aí em cima foi simplesmente aplicar um “mínimos quadrados ordinários” sobre um painel empilhado, mas você não vai me dar um tiro na testa por conta disto, certo?

## 6.5. A Lei de Wagner, a Guerra do Paraguai e o teste “t”

A Lei de Wagner é uma correlação clássica na História do Pensamento Econômico e na Economia do Setor Público. A bem da verdade, acho que não seria incorreto dizer que suas raízes remontam à famosa *Methodenstreit* (literalmente: “a disputa entre métodos”). Mas correlação entre que variáveis? Basicamente entre o nível de gastos do governo e o PIB. A ideia é que o aumento do PIB levaria a um aumento dos gastos do governo. A causalidade, neste caso, é *ad hoc*, apenas por suposição do autor, por isso chamo-a de correlação.

Vejamos o caso do Brasil. Graças a Goldsmith (1987), temos uma série de PIB e de Gastos do Governo que se inicia em 1850, o que nos permite ilustrar outro aspecto interessante do R por meio do uso de *dummies*, qual seja, os fatores de interação.

No exemplo, introduzimos uma variável *dummy* que capta os períodos de conflitos internacionais do Brasil: a Guerra do Paraguai e as duas guerras mundiais. A regressão pretende testar se a lei de Wagner sofre algum tipo de efeito com as guerras. Os dados foram importados e foram deflacionados. Em seguida, foram transformados em séries de tempo.

```
> wag<-read.table("c:/Users/cdshi_000/Documents/Meus Documentos/Meus Documentos/minicurso em R/leiwagner_m
inicurso.csv",
+ header=TRUE, sep=";", na.strings="NA", dec=".", strip.white=TRUE)
> wag<-data.frame(wag)
> head(wag)
  inflacao gov PIB war
1 1.000000 29.0 365 0
2 1.022740 33.2 405 0
3 1.065047 42.8 417 0
4 1.152634 31.7 429 0
5 1.235000 36.2 452 0
6 1.312285 38.7 492 0
> #
> wag$pib_real<-wag$PIB/wag$inflacao
> wag$gov_real<-wag$gov/wag$inflacao
> wag<-ts(wag, start=c(1850))
> head(wag)
  inflacao gov PIB war pib_real gov_real
[1,] 1.000000 29.0 365 0 365.0000 29.00000
[2,] 1.022740 33.2 405 0 395.9952 32.46183
[3,] 1.065047 42.8 417 0 391.5319 40.18601
[4,] 1.152634 31.7 429 0 372.1909 27.50222
[5,] 1.235000 36.2 452 0 365.9918 29.31174
[6,] 1.312285 38.7 492 0 374.9185 29.49054
>
```

O passo seguinte é estimar a regressão.

```
> wagner<-lm(gov_real~war*pib_real, data=wag)
> summary(wagner)
```

Call:

```
lm(formula = gov_real ~ war * pib_real, data = wag)
```

Residuals:

Min	1Q	Median	3Q	Max
-873.64	-125.75	-41.61	111.17	1651.09

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	406.71708	67.19613	6.053	3.05e-08 ***
war	637.19408	123.04097	5.179	1.31e-06 ***
pib_real	-0.33676	0.08963	-3.757	0.000301 ***
war:pib_real	-1.23433	0.29571	-4.174	6.78e-05 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 350.5 on 92 degrees of freedom

Multiple R-squared: 0.4453, Adjusted R-squared: 0.4273

F-statistic: 24.62 on 3 and 92 DF, p-value: 8.758e-12

Curiosamente, a lei de Wagner aparece com o sinal invertido. A *dummy* de guerra (*war*) é significativa. Repare no último termo, *war:pib\_real*, que diz respeito ao termo de interação. Todos os coeficientes estimados foram significativos. Vamos interpretar alguns deles. Sabemos que o gasto real do governo tem, *ceteris paribus*, uma média de 406.71. Em períodos de guerra, este valor é acrescido de 637.19. O coeficiente do *pib\_real* é de -0.33 mas,

em períodos de guerra, o coeficiente é menor ainda  $(-0.33 + (-1.23))$ . Perceba que o comando para criar o efeito de interação é simples<sup>27</sup>.

## Anexo 2 – Tabelas com várias regressões no R: salvando o dia do bolsista

Uma das tarefas mais inglórias do estagiário/monitor/bolsista é fazer tabelas com várias regressões. Existem algumas opções no R que minimizam este problema. Por exemplo, suponha que tenhamos estimado três funções consumo e queiramos um quadro resumo. Podemos usar o programa *stargazer* para nos ajudar.

```
> stargazer(cons_key, cons_rp, cons_hab, type = "text")
```

```
=====
Dependent variable:
-----
              OLS              consumption              dynamic
              (1)              (2)              linear              (3)
-----
dpi              0.922***              -0.003              -0.004
              (0.004)              (0.015)              (0.017)
L(consumption)              1.012***              1.013***
              (0.017)              (0.018)
Constant              -80.355***              0.535
              (14.306)              (3.845)
-----
Observations              204              203              203
R2              0.996              1.000              1.000
Adjusted R2              0.996              1.000              1.000
Residual Std. Error      87.210 (df = 202)      21.470 (df = 201)      21.523 (df = 200)
F Statistic      56,669.720*** (df = 1; 202)  2,458,600.000*** (df = 2; 201)  462,699.600*** (df = 2; 200)
=====
Note:                               *p<0.1; **p<0.05; ***p<0.01
> |
```

Outro exemplo.

```
> stargazer(cons_key, cons_rp, cons_hab, title="Exemplo", no.space = TRUE, type="text")
```

```
Exemplo
=====
Dependent variable:
-----
              OLS              consumption              dynamic
              (1)              (2)              linear              (3)
-----
dpi              0.922***              -0.003              -0.004
              (0.004)              (0.015)              (0.017)
L(consumption)              1.012***              1.013***
              (0.017)              (0.018)
Constant              -80.355***              0.535
              (14.306)              (3.845)
-----
Observations              204              203              203
R2              0.996              1.000              1.000
Adjusted R2              0.996              1.000              1.000
Residual Std. Error      87.210 (df = 202)      21.470 (df = 201)      21.523 (df = 200)
F Statistic      56,669.720*** (df = 1; 202)  2,458,600.000*** (df = 2; 201)  462,699.600*** (df = 2; 200)
=====
Note:                               *p<0.1; **p<0.05; ***p<0.01
> |
```

<sup>27</sup> Veja também este exemplo, para uma variável *dummy* categórica com três categorias.

<http://datascienceplus.com/assessing-significance-of-slopes-in-regression-models-with-interaction/>

Repare que o comando tem também a opção de gerar o código da tabela para LaTeX. A *vignette* do programa é bastante útil<sup>28</sup> e o lamentável é que nem todos os pacotes do R são compatíveis com *stargazer*. Mesmo assim, ajuda a alegrar a vida do pesquisador, não ajuda?

### Anexo 3 - Sorvetes, Matrizes e MQO (prof. Rodrigo N. Fernandez)

Agora faremos um exemplo relembrando nossas aulas de econometria. O R é um software bastante poderoso quando desejamos trabalhar com matrizes. Primeiramente vejamos o nosso  $\hat{\beta}$ :

$$\hat{\beta} = (X'X)^{-1}X'Y$$

Para calcularmos nosso parâmetro usaremos a base de dados Icecream do pacote Ecdat. Este pacote contém um diverso conjunto de dados para aplicações em econometria. Nossa primeira tarefa será instalar este pacote ou carregá-lo com os seguintes comandos:

```
install.packages("Ecdat")
```

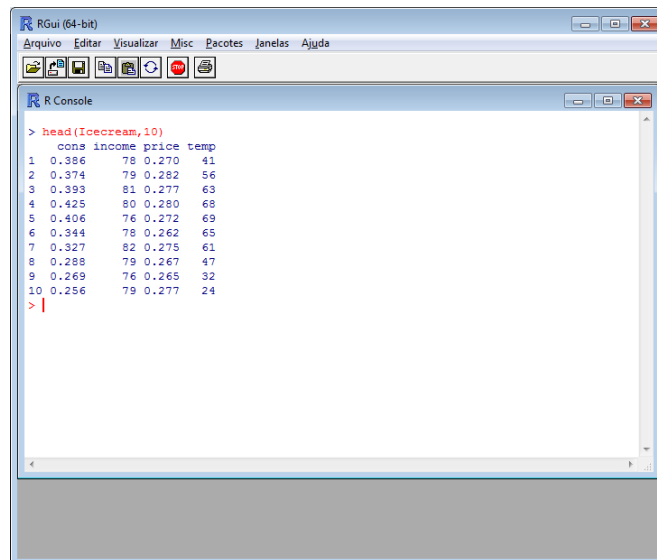
```
library("Ecdat")
```

```
data("Icecream")
```

Após carregarmos os dados usaremos o comando `attach("Icecream")` para podermos usar os nomes das variáveis que estão na base de dados. Se você for curioso como eu e deseja saber o que está na base de dados digite `head(Icecream,10)` :

---

<sup>28</sup> <https://cran.r-project.org/web/packages/stargazer/vignettes/stargazer.pdf>.



```
RGui (64-bit)
Arquivo  Editar  Visualizar  Misc  Pacotes  Janelas  Ajuda

R Console

> head(Icecream,10)
  cons income price temp
1  0.386    78 0.270   41
2  0.374    79 0.282   56
3  0.393    81 0.277   63
4  0.425    80 0.280   68
5  0.406    76 0.272   69
6  0.344    78 0.262   65
7  0.327    82 0.275   61
8  0.288    79 0.267   47
9  0.269    76 0.265   32
10 0.256    79 0.277   24
> |
```

Você verá as dez primeiras observações do nosso conjunto de dados. Nessa base de dados temos 30 observações para o consumo semanal de sorvete nos EUA entre março de 1951 a novembro de 1953 nos EUA. As variáveis são as seguintes:

**cons** = consumo de sorvete por pessoa (em litros)

**income** = renda semanal média da família por semana (em Dólares)

**price** = preço do sorvete (por litro);

**temp** = temperature média em graus Fahrenheites

O próximo passo é “traduzir” o nome das variáveis fazendo o seguinte:

*consumo=cons*

*renda=income*

*preco=price*

*temperatura=temp*

Tendo nossos dados é necessário criar nosso vetor de constantes para podemos utilizar na nossa regressão múltipla:

*n=nrow(Icecream)*

$$uns = rep(1,n)$$

O primeiro comando nos retorna quantas linhas a nossa base de dados possui, como mencionamos previamente o tamanho de  $n$  é igual a 30. O segundo comando cria o vetor de constantes. Em seguida construiremos as nossas matrizes  $X$  e  $Y$  da seguinte forma:

$$X = cbind(uns,renda,preco,temperatura)$$

$$Y = cons$$

O comando *cbind* monta uma matriz com todos os nossos vetores coluna, que são as variáveis dependentes na nossa regressão múltipla. O próximo passo é fazermos operações com essas matrizes. Para isso usaremos o seguinte:

$$solve(t(X) \% \% X)$$

Aqui estamos fazendo a primeira parte do cálculo do nosso  $\hat{\beta}$ . Na linguagem do R o comando *%%* indica a multiplicação de matrizes,  $t(X)$  é a matriz transposta e *solve* calcula a matriz inversa, deste modo temos  $(X'X)^{-1}$ . Para facilitar nossas contas adicionaremos ao comando acima o seguinte:

$$xlinhax = solve(t(X) \% \% X) \% \% t(X)$$

Por fim multiplicamos por  $Y$ :

$$beta.hat = xlinhax \% \% Y$$

```
> beta.hat
           [,1]
uns      0.19731507
renda    0.00330776
preco    -1.04441399
temperatura 0.00345843
> |
```

Para calcularmos nosso  $\hat{Y}$  estimado e os resíduos fazemos o seguinte:

$$y.hat = X \% \% beta.hat$$

$$e.hat = Y - y.hat$$

Podemos também calcular a matriz de variância e covariância fazendo o seguinte:

$$k = ncol(X)$$

$$sig.2 = sum(e.hat^2)/(n-k)$$

$$cov.b = sig.2 * solve(t(X) \% \% X)$$

Digite cov.b e veja o resultado no terminal:

```

      uns      renda      preco      temperatura
uns      7.301677e-02 -1.656223e-04 -2.069700e-01 -4.034752e-05
renda     -1.656223e-04  1.372221e-06  1.482448e-04  1.776084e-07
preco     -2.069700e-01  1.482448e-04  6.961521e-01  5.657431e-05
temperatura -4.034752e-05  1.776084e-07  5.657431e-05  1.985120e-07
> |

```

Para calcularmos o desvio padrão precisamos da raiz quadrada da diagonal principal da matriz de variância e covariância:

$$dp.b = \text{sqrt}(\text{diag}(cov.b))$$

Para obtermos nossa estatística t:

$$est.t = \text{beta.hat}/dp.b$$

E agora nossos p-valores:

$$p.valor.t = 2 * pt(\text{abs}(est.t), df = n - k, \text{lower.tail} = \text{FALSE})$$

Finalmente temos tudo que precisamos referente a nossa regressão. Vamos então construir uma matriz que resuma estes resultados:

$$\text{sumario} = \text{matrix}(1, nrow = 4, ncol = 4)$$

$$\text{sumario} = \text{cbind}(\text{beta.hat}, dp.b, est.t, p.valor.t)$$

$$\text{colnames}(\text{sumario}) = c(\text{"Beta"}, \text{"DP"}, \text{"Estat-T"}, \text{"P-Valor"})$$

```

> sumario=matrix(1,nrow=4,ncol=4)
> sumario = cbind(beta.hat,dp.b,est.t,p.valor.t)
> colnames(sumario)=c("Beta","DP","Estat-T","P-Valor")
>
> sumario
      Beta      DP      Estat-T      P-Valor
uns      0.19731507 0.2702161566 0.730212 4.717894e-01
renda      0.00330776 0.0011714185 2.823722 8.988730e-03
preco     -1.04441399 0.8343573214 -1.251759 2.218027e-01
temperatura 0.00345843 0.0004455469 7.762213 3.100024e-08
> |

```

Que tal compararmos nossos resultados com o da função *lm*? Para isto devemos usar o seguinte comando:

```
reg = lm(consumo~renda+temperatura+preco,data=Icecream)
```

Obtemos o resumo da regressão com o comando `summary(reg)`:

```
R Console
temperatura -1.212664e-05  2.240786e-07 -5.273816e-05  1.748438e-07
> reg = lm(consumo~renda+temperatura+preco,data=Icecream)
> summary(reg)

Call:
lm(formula = consumo ~ renda + temperatura + preco, data = Icecream)

Residuals:
    Min       1Q   Median       3Q      Max
-0.065302 -0.011873  0.002737  0.015953  0.078986

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.1973151   0.2702162    0.730   0.47179
renda        0.0033078   0.0011714    2.824   0.00899 **
temperatura  0.0034584   0.0004455    7.762   3.1e-08 ***
preco       -1.0444140   0.8343573   -1.252   0.22180
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03683 on 26 degrees of freedom
Multiple R-squared:  0.719,    Adjusted R-squared:  0.6866
F-statistic: 22.17 on 3 and 26 DF,  p-value: 2.451e-07

> |
```

Veja que acertamos na mosca! Desde aquela época o consumo de sorvete dependia positivamente da renda e da temperatura. Também observamos que o preço afeta negativamente o consumo da delícia gelada, mas esta variável não é estatisticamente significativa.

Adicionalmente, usaremos a matriz de White para termos erros padrões robustos e corrigirmos um possível problema de heterocedasticidade:

$$\hat{V} = (X'X)^{-1}(\hat{\sigma}^2 X'X)(X'X)^{-1}$$

Construiremos a nossa matriz `v.hat`:

```
v.hat = matrix(0,ncol=n,nrow=n);
```

Inseriremos na diagonal principal da nossa matriz de White os erros padrões estimados ao quadrado:

```
diag(v.hat) = e.hat^2;
```

Calculamos então segunda parte com os seguintes comandos:

```
xlinhax1=X%%solve(t(X)%%X)
```

```
v.white = xlinhax1%%v.hat%%xlinhax1
```



```
> v.white
              uns          renda          preco  temperatura
uns          7.167345e-02 -1.689010e-04 -2.034772e-01 -1.212664e-05
renda        -1.689010e-04  1.148268e-06  2.174210e-04  2.240786e-07
preco        -2.034772e-01  2.174210e-04  6.724605e-01 -5.273816e-05
temperatura  -1.212664e-05  2.240786e-07 -5.273816e-05  1.748438e-07
> |
```

Obtemos então os erros padrões robustos:

$$dp.b.rob = \sqrt{diag(v.white)}$$

Você pode estar se perguntando, será que o R não faz isso de um jeito mais fácil? É claro que sim, primeiro carregue o pacote sandwich (`library(sandwich)`). E digite o seguinte comando:

$$vcovHC(reg,type="HC0")$$

O leitor observará que os resultados são equivalentes aos apresentados acima. Como se vê, sorvetes também podem nos ajudar a aprender Econometria e, melhor ainda, Econometria em R.

## 7. Modelos Microeconômicos

### 7.1. Coca ou Pepsi? <sup>29</sup>

O exemplo clássico de bens substitutos perfeitos está resumido no subtítulo acima. Em Hill, Griffiths & Lim (2011), cap.7, temos uma aplicação econométrica desta pergunta por meio de modelos lineares generalizados (binomial, gaussiano ou Poisson). Os dados são relativos a uma amostra de 1140 indivíduos. A base contém o preço relativo entre uma lata de Coca-Cola e uma de Pepsi-Cola, a existência ou não (*dummies*) de anúncios no ponto de venda de um ou de outro e, finalmente, duas *dummies* indicando se o sujeito comprou um ou outro refrigerante.

Vejamos o arquivo.

```
> coke<-read.table("C:/Users/cdshi_000/Documents/Meus Documentos/Meus Documentos/minicurs
o em R/coke.csv",header=TRUE, sep=";", na.strings="NA", dec=".", strip.white=TRUE)
> summary(coke)
```

coke		pr_pepsi		pr_coke		disp_pepsi		disp_coke	
Min.	:0.0000	Min.	:0.680	Min.	:0.68	Min.	:0.000	Min.	:0.0000
1st Qu.	:0.0000	1st Qu.	:0.980	1st Qu.	:0.89	1st Qu.	:0.000	1st Qu.	:0.0000
Median	:0.0000	Median	:1.190	Median	:1.19	Median	:0.000	Median	:0.0000
Mean	:0.4474	Mean	:1.203	Mean	:1.19	Mean	:0.364	Mean	:0.3789
3rd Qu.	:1.0000	3rd Qu.	:1.390	3rd Qu.	:1.39	3rd Qu.	:1.000	3rd Qu.	:1.0000
Max.	:1.0000	Max.	:1.790	Max.	:1.79	Max.	:1.000	Max.	:1.0000

```
pratio
Min. :0.4972
1st Qu.:0.8571
Median :1.0000
Max. :1.0000
```

Os autores, inicialmente, os autores estimam um modelo linear por meio de MQO, para a Coca-Cola. A leitura é a de que estamos estimando os determinantes da probabilidade do indivíduo da amostra comprar uma lata de Coca-Cola. Eis os resultados.

```
> coke_linear<-lm(coke~pratio+disp_coke+disp_pepsi, data=coke)
> summary(coke_linear)
```

Call:  
lm(formula = coke ~ pratio + disp\_coke + disp\_pepsi, data = coke)

Residuals:

Min	1Q	Median	3Q	Max
-0.7681	-0.4186	-0.1560	0.4547	1.2073

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	0.89022	0.06548	13.594	< 2e-16 ***
pratio	-0.40086	0.06135	-6.534	9.64e-11 ***
disp_coke	0.07717	0.03439	2.244	0.025 *
disp_pepsi	-0.16566	0.03560	-4.654	3.65e-06 ***

---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4672 on 1136 degrees of freedom  
Multiple R-squared: 0.1201, Adjusted R-squared: 0.1177  
F-statistic: 51.67 on 3 and 1136 DF, p-value: < 2.2e-16

<sup>29</sup> Particularmente, eu preferiria escolher entre Coca Zero e Pepsi (comum).

A leitura é a seguinte: a existência de anúncios de Coca-Cola, *ceteris paribus*, aumentam em 0.07% a probabilidade de se comprar este refrigerante. A propaganda do concorrente, por sua vez, tem um efeito negativo, *ceteris paribus*, de queda de 0.16%. A razão de preços, medida como o preço relativo da Coca-Cola em relação à Pepsi, tem um impacto negativo sobre a probabilidade de compra do primeiro. Seguindo os autores, se a diferença de preços for de 10%, com a Coca-Cola mais cara do que a Pepsi (ou seja,  $\text{pratio} = 1.10$ ), então a probabilidade do indivíduo comprar a Coca diminui em 0.04% ( $= 0.10 \times 0.04$ ), sem considerar a existência de anúncios (não se esqueça: *ceteris paribus*).

Sabemos que este modelo pode não ser adequado dado seu caráter linear e a necessidade de que as probabilidades estejam entre zero e um. Assim, podemos usar o comando *glm* para estimar um probit ou um logit para a mesma base. Eis os resultados para ambos.

```
> coke_probit <-glm(coke~pratio+disp_coke+disp_pepsi, data=coke, family=binomial(link="probit"))
> summary(coke_probit)
```

call:  
glm(formula = coke ~ pratio + disp\_coke + disp\_pepsi, family = binomial(link = "probit"), data = coke)

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.7273	-1.0192	-0.6052	1.0958	2.7535

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )	
(Intercept)	1.10804	0.19250	5.756	8.61e-09	***
pratio	-1.14594	0.18392	-6.231	4.64e-10	***
disp_coke	0.21719	0.09622	2.257	0.024	*
disp_pepsi	-0.44730	0.10103	-4.427	9.54e-06	***

---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1567.7 on 1139 degrees of freedom  
Residual deviance: 1421.9 on 1136 degrees of freedom  
AIC: 1429.9

```
> coke_logit <-glm(coke~pratio+disp_coke+disp_pepsi, data=coke, family=binomial(link="logit"))
> summary(coke_logit)
```

call:  
glm(formula = coke ~ pratio + disp\_coke + disp\_pepsi, family = binomial(link = "logit"), data = coke)

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.7476	-1.0031	-0.5872	1.0913	2.6377

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )	
(Intercept)	1.9230	0.3258	5.902	3.59e-09	***
pratio	-1.9957	0.3146	-6.344	2.23e-10	***
disp_coke	0.3516	0.1585	2.218	0.0266	*
disp_pepsi	-0.7310	0.1678	-4.356	1.33e-05	***

---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1567.7 on 1139 degrees of freedom  
Residual deviance: 1418.9 on 1136 degrees of freedom  
AIC: 1426.9

Repare que, com distintas distribuições, os coeficientes mudam de magnitude, mas não de sinal. A significância estatística também é mantida. Em relação ao modelo linear, repare que o impacto do preço relativo é muito maior. Os coeficientes relativos à propaganda também são maiores, em magnitude, quando se os comparam com o modelo linear. Entretanto, salta aos olhos a importância relativa dos preços relativos em relação à propaganda. Em qualquer das três estimativas, a variação dos preços é muito mais importante, o que é uma evidência favorável, creio eu, à visão de que estes bens, teoricamente, são substitutos perfeitos para o consumidor<sup>30</sup>.

Os efeitos marginais podem ser calculados, no caso do probit, para fins de comparação com o modelo original. Eis os resultados:

```
> coke_marg<-mean(dnorm(predict(coke_probit, type="link")))
> coke_marg*coef(coke_probit)
(Intercept)      pratio disp_coke disp_pepsi
0.39613890 -0.40968928 0.07764731 -0.15991392
```

Repare que os efeitos marginais não são muito distintos daqueles obtidos pelo modelo linear. Os efeitos marginais foram obtidos, neste caso, por meio da média dos efeitos marginais amostrais. Como destacam Kleiber & Zeileis (2008), é possível calcular os efeitos sobre a média do regressor, mas a existência de fatores do lado direito da regressão (disp\_coke, disp\_pepsi) faz com que o procedimento reportado seja preferível.

Em termos de ajuste aos dados, podemos calcular o pseudo-R<sup>2</sup> de McFadden ou verificar a previsão do modelo (considerando um ponto de corte de 0.5).

```
> coke_marg<-mean(dnorm(predict(coke_probit, type="link")))
> coke_marg*coef(coke_probit)
(Intercept)      pratio disp_coke disp_pepsi
0.39613890 -0.40968928 0.07764731 -0.15991392
> coke_probit0 <- update(coke_probit, formula =, ~1)
> 1 - as.vector(logLik(coke_probit)/logLik(coke_probit0))
[1] 0.0930162
>
> table(true = coke$coke, pred = round(fitted(coke_probit)))
      pred
true   0   1
  0  507 123
  1  263 247
```

Da tabela, temos que 66%  $((507 + 247) / (507+123+263+247) = 0.66)$  das observações são previstas corretamente. Hill, Griffiths & Lim (2011), em sua análise dos dados, indicam que os três modelos – linear, probit e logit – têm desempenho semelhante no que diz respeito à previsão. Por meio do pacote *ROCR*, pode-se obter algumas outras medidas de previsão. Portanto, sem entrar em mais detalhes, eis os comandos e os resultados gerados.

<sup>30</sup> Ok, aqui eu creio que valeria a pena pensar em modelos de competição (duopólio) e gastos em propaganda. Creio que é uma discussão bem interessante para o usuário de R com vício em questões econômicas.

```

> library("ROCR")
Loading required package: gplots

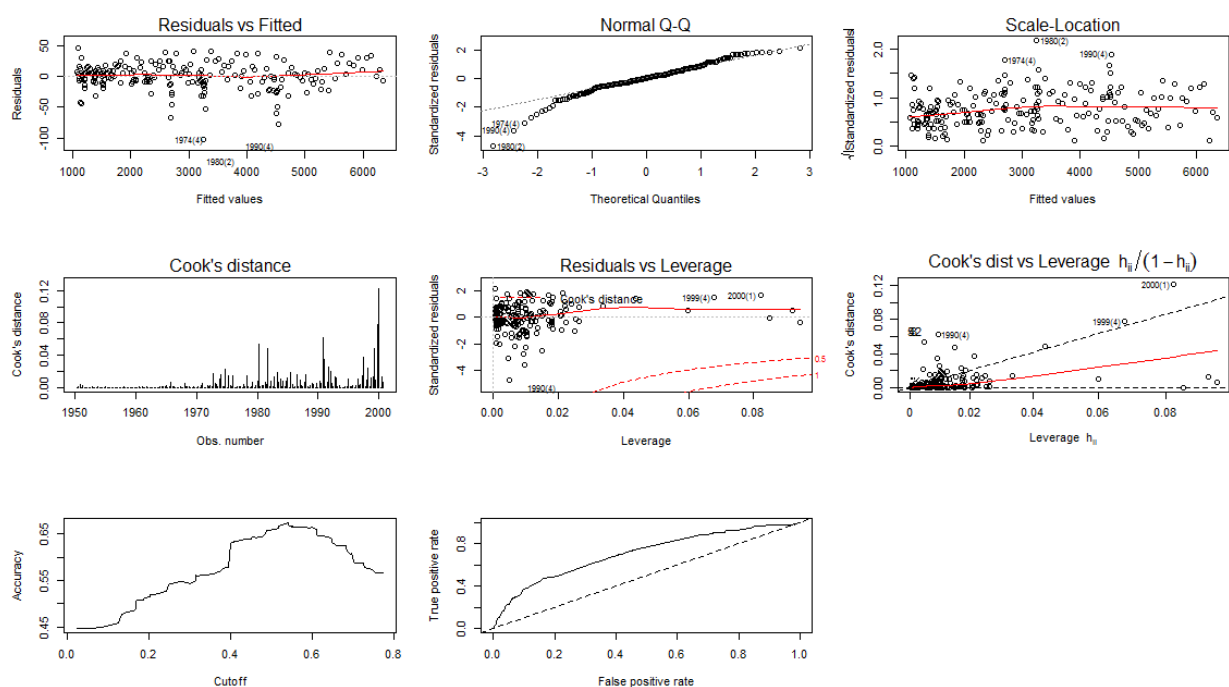
Attaching package: 'gplots'

The following object is masked from 'package:stats':

    lowess

> pred<-prediction(fitted(coke_probit), coke$coke)
> plot(performance(pred, "acc"))
> plot(performance(pred, "tpr", "fpr"))
> abline(0,1, lty=2)

```



Observe a última linha. Resumidamente, o primeiro gráfico nos diz que o ponto de corte é um pouco maior que 0.5, sugerindo que a tabela das previsões que apresentamos acima (baseada no corte em 0.5) não é uma medida adequada. O segundo gráfico, por sua vez, nos diz que o modelo tem bom desempenho. Segundo Kleiber & Zeileis (2008), o desempenho é melhor quanto mais a noroeste da reta de 45 graus está a curva.<sup>31</sup>

Seguindo Hill, Griffiths & Lim (2011), pode-se fazer um teste de hipóteses sobre o efeito das propagandas dos dois refrigerantes. Por meio do pacote *car*, podemos fazer tal teste. Eis os resultados.

<sup>31</sup> Para detalhes sobre estas medidas, ver Kleiber & Zeileis (2008), seção 5.2.

```
> linearHypothesis(coke_probit, "disp_coke=-disp_pepsi")
Linear hypothesis test

Hypothesis:
disp_coke + disp_pepsi = 0

Model 1: restricted model
Model 2: coke ~ pratio + disp_coke + disp_pepsi

    Res.Df Df    Chisq Pr(>Chisq)
1      1137
2      1136  1  5.4362    0.01972 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Assim, a 10%, a hipótese nula é rejeitada. Vale lembrar que, para testes de hipóteses em modelos GLM, Kleiber & Zeileis (2008) não recomendam o uso de erros-padrão robustos (*sandwich*), exceto em regressões do tipo Poisson. Para outros testes, o leitor é convidado a consultar Kleiber & Zeileis (2008) e a documentação do R.

## 7.2. Esporte Espetacular

Nesta seção reproduzimos o exemplo das medalhas em Olimpíadas de Hill, Griffiths & Lim (2011). A ideia é implementar o exemplo de uma regressão Poisson, já que os dados da variável dependente são de contagem (número de medalhas). Repare que a regressão foi feita apenas para o anos de 1988, por meio de `subset=olympics$year=="88"`. Note que a forma de delimitar a amostra é ligeiramente distinta da que vimos anteriormente. O arquivo é *olympics.csv*.

```
> olympics<-read.table("C:/Users/cdshi_000/Documents/Meus Documentos/Meus Documentos/minicurso
em R/olympics.csv",
+ header=TRUE, sep=";", na.strings="NA", dec=".", strip.white=TRUE)
> summary(olympics)
```

country		year	gdp	pop
Min. :	1.0	Min. :60	Min. :3.910e+07	Min. :1.496e+04
1st Qu.:	54.0	1st Qu.:68	1st Qu.:1.492e+09	1st Qu.:1.428e+06
Median :	105.0	Median :78	Median :5.790e+09	Median :5.353e+06
Mean :	105.1	Mean :78	Mean :1.173e+11	Mean :2.677e+07
3rd Qu.:	156.0	3rd Qu.:88	3rd Qu.:4.190e+10	3rd Qu.:1.675e+07
Max :	211.0	Max :96	Max :7.280e+12	Max :1.220e+09

Eis a regressão.

```
> medal_pois <- glm(medaltot~log(gdp)+log(pop), data=olympics, family=poisson, subset=olympics$
year=="88")
> summary(medal_pois)
```

Call:  
glm(formula = medaltot ~ log(gdp) + log(pop), family = poisson,  
data = olympics, subset = olympics\$year == "88")

Deviance Residuals:

Min	1Q	Median	3Q	Max
-8.1558	-1.8279	-1.0438	-0.4306	18.0856

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-15.88746	0.51180	-31.042	< 2e-16 ***
log(gdp)	0.57660	0.02472	23.324	< 2e-16 ***
log(pop)	0.18004	0.03228	5.577	2.44e-08 ***

---  
signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 2994.5 on 150 degrees of freedom  
Residual deviance: 1266.5 on 148 degrees of freedom  
(54 observations deleted due to missingness)  
AIC: 1450.7

Number of Fisher Scoring iterations: 7

É interessante pensar em alguma coisa similar a um ponto ótimo neste exemplo. Afinal, pode-se imaginar que exista um número de medalhas ótimo relativamente ao PIB ou à população de um país. Eis o que encontramos ao incluir alguns termos não-lineares nesta mesma regressão.

```
Console ~/Meus Documentos/Meus Documentos/minicurso em R/ ↗
> medal_pois3 <- glm(medaltot~log(gdp)+log(pop)+I((log(pop))^2)+I((log(gdp))^2)
+
+I((log(pop))^3)+I((log(gdp))^3), data=olympics, family=poisson, subset=ol
ympics$year=="88")
> summary(medal_pois3)
```

Call:  
glm(formula = medaltot ~ log(gdp) + log(pop) + I((log(pop))^2) +  
I((log(gdp))^2) + I((log(pop))^3) + I((log(gdp))^3), family = poisson,  
data = olympics, subset = olympics\$year == "88")

Deviance Residuals:

Min	1Q	Median	3Q	Max
-6.9403	-1.7146	-0.8773	-0.2052	17.0361

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-11.363333	64.473746	-0.176	0.8601
log(gdp)	-9.420762	6.962864	-1.353	0.1761
log(pop)	12.994670	6.058992	2.145	0.0320 *
I((log(pop))^2)	-0.746512	0.346498	-2.154	0.0312 *
I((log(gdp))^2)	0.423077	0.270091	1.566	0.1172
I((log(pop))^3)	0.014352	0.006565	2.186	0.0288 *
I((log(gdp))^3)	-0.005864	0.003479	-1.686	0.0919 .

---  
signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 2994.5 on 150 degrees of freedom  
Residual deviance: 1232.8 on 144 degrees of freedom  
(54 observations deleted due to missingness)  
AIC: 1425.1

Number of Fisher Scoring iterations: 7



Repare que, nesta especificação, apenas a população tem algum efeito diferente de zero, e não-linear (a 10%). Uma questão interessante seria pensar também no modelo em função do PIB per capita, ao invés dos termos separados. A racionalização, para não-linearidades, poderia ser similar a de uma curva de Engel. Eis os resultados.

```
> medal_pois4 <- glm(medaltot~log(gdp/pop)+I((log(gdp/pop))^2)+I((log(gdp/pop))^3), data=olympics, family=poisson, subset=olympics$year=="88")
> summary(medal_pois4)
```

Call:  
glm(formula = medaltot ~ log(gdp/pop) + I((log(gdp/pop))^2) + I((log(gdp/pop))^3), family = poisson, data = olympics, subset = olympics\$year == "88")

Deviance Residuals:

Min	1Q	Median	3Q	Max
-4.828	-3.053	-1.555	-1.058	22.888

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )	
(Intercept)	20.98745	7.58015	2.769	0.005627	**
log(gdp/pop)	-10.29864	2.87828	-3.578	0.000346	***
I((log(gdp/pop))^2)	1.55029	0.35824	4.328	1.51e-05	***
I((log(gdp/pop))^3)	-0.07069	0.01463	-4.831	1.36e-06	***

---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 2994.5 on 150 degrees of freedom  
Residual deviance: 2461.6 on 147 degrees of freedom  
(54 observations deleted due to missingness)  
AIC: 2647.9

Number of Fisher Scoring iterations: 8

Podemos analisar esta especificação, por exemplo, em termos de super-dispersão (*overdispersion*), um problema comum em regressões Poisson. Para tanto, usamos o pacote *AER*. Há dois testes de super-dispersão neste pacote, conforme a especificação que se queira supor para a variância. Apresentamos, sem maiores discussões, os resultados destes testes<sup>32</sup>.

<sup>32</sup> Para detalhes sobre sua especificação, ver Kleiber & Zeileis (2008), cap.5. Para detalhes sobre o problema, ver Fox & Weisberg (2011), seção 5.10.4. Basicamente, o problema diz respeito ao fato de que a variância amostral geralmente não é igual à média, como se supõe quando a distribuição assumida é a de Poisson.



```

> dispersiontest(medal_pois4)

overdispersion test

data: medal_pois4
z = 2.2358, p-value = 0.01268
alternative hypothesis: true dispersion is greater than 1
sample estimates:
dispersion
 39.88167

> dispersiontest(medal_pois4, trafo=2)

overdispersion test

data: medal_pois4
z = 2.3111, p-value = 0.01041
alternative hypothesis: true alpha is greater than 0
sample estimates:
alpha
 6.334819

```

Ambos sugerem problemas de super-dispersão e, segundo Kleiber & Zeileis (2008), o procedimento comum é usar uma distribuição binomial negativa em lugar da Poisson ou usar erros-padrão robustos (Huber-White). Primeiramente, vejamos a binomial negativa obtida por meio do pacote MASS.

```

> library("MASS")
> medal_nb <- glm.nb(medaltot~log(gdp/pop)+I((log(gdp/pop))^2)+I((log(gdp/pop))^3), data=olympi
cs,subset=olympics$year=="88")
> summary(medal_nb)

Call:
glm.nb(formula = medaltot ~ log(gdp/pop) + I((log(gdp/pop))^2) +
      I((log(gdp/pop))^3), data = olympics, subset = olympics$year ==
      "88", init.theta = 0.1291593362, link = log)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.0781  -0.9254  -0.7149  -0.2759   2.7780

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)    24.45577   30.69854   0.797   0.426
log(gdp/pop)  -11.43740   12.32573  -0.928   0.353
I((log(gdp/pop))^2)  1.66964   1.61278   1.035   0.301
I((log(gdp/pop))^3) -0.07470   0.06882  -1.085   0.278

(Dispersion parameter for Negative Binomial(0.1292) family taken to be 1)

Null deviance: 119.86  on 150  degrees of freedom
Residual deviance: 102.14  on 147  degrees of freedom
(54 observations deleted due to missingness)
AIC: 519.48

Number of Fisher Scoring iterations: 1

              Theta: 0.1292
            Std. Err.: 0.0224

2 x log-likelihood: -509.4790
> |

```

A segunda opção é reestimar nossa regressão Poisson com a matriz de variância-covariância ajustada por Huber-White.

```

> coeftest(medal_pois4, vcov=sandwich)

z test of coefficients:

              Estimate Std. Error z value Pr(>|z|)
(Intercept)    20.987455   30.528174   0.6875   0.4918
log(gdp/pop)  -10.298644   11.625568  -0.8859   0.3757
I((log(gdp/pop))^2)  1.550291   1.460205   1.0617   0.2884
I((log(gdp/pop))^3) -0.070690   0.060445  -1.1695   0.2422
> |

```

Percebe-se que os resultados da binomial negativa são mantidos e a relação entre PIB per capita e medalhas mantém-se não significativa aos níveis usuais<sup>33</sup>. Vale a pena ser sede de uma Olimpíada? Talvez não. Pelo menos é o que este exercício aponta. Como melhorá-lo? Eis a pergunta que vale uma medalha de ouro.

### 7.3. Os motivos da traição: o retorno dos casos extra-conjugais!

Nesta seção, basicamente repetiremos o exercício de Kleiber & Zeileis (2008) com os dados de casos extraconjugais, um exemplo de dados censurados (caso observados, assumem valores positivos, senão, zero). Lembre-se que já tratamos desta base de dados na seção 5 deste texto.

O objetivo é estudar os determinantes de casos extraconjugais. Então, quais os motivos da traição? Vejamos o resultado da estimação.

<sup>33</sup> Para maiores detalhes sobre outros modelos de contagem, ver Kleiber & Zeileis, cap.5, especialmente a discussão sobre *zero-inflated Poisson* e *Hurdle Models*. Ambos são estimáveis por meio do pacote *pscl*.

```

Console ~/Meus Documentos/Meus Documentos/minicurso em R/
> aff_tob<-tobit'affairs~age+yearsmarried + religiousness +occupation+rating, data=Affairs)
> summary(aff_tob)

Call:
tobit(formula = affairs ~ age + yearsmarried + religiousness +
      occupation + rating, data = Affairs)

Observations:
      Total    Left-censored    Uncensored Right-censored
      601         451          150           0

Coefficients:
      Estimate Std. Error z value Pr(>|z|)
(Intercept)   8.17420    2.74145   2.982  0.00287 **
age          -0.17933    0.07909  -2.267  0.02337 *
yearsmarried   0.55414    0.13452   4.119 3.80e-05 ***
religiousness -1.68622    0.40375  -4.176 2.96e-05 ***
occupation     0.32605    0.25442   1.282  0.20001
rating        -2.28497    0.40783  -5.603 2.11e-08 ***
Log(scale)     2.10986    0.06710  31.444 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Scale: 8.247

Gaussian distribution
Number of Newton-Raphson Iterations: 4
Log-likelihood: -705.6 on 7 Df
Wald-statistic: 67.71 on 5 Df, p-value: 3.0718e-13

```

Os resultados mostram evidências de que pessoas felizes no casamento (*rating*) tendem a trair menos. Também observamos que a religiosidade é um fator com influência negativa sobre a ocorrência de casos extraconjugais. Poderíamos seguir adiante com a discussão sobre a vida conjugal das pessoas (um bom tema para fofocas, eu sei), mas o objetivo, aqui, é apenas o de mostrar como se estima um modelo Tobit no R.

De qualquer forma, fica aqui uma observação importante: a variável dependente, lembre-se, é o número de casos extraconjugais do indivíduo no ano anterior. Em outras palavras, é uma variável de contagem e, portanto, o modelo mais adequado para a estimação seria uma regressão de Poisson. Assim, a regressão mais adequada talvez fosse esta.

```

Console ~/Meus Documentos/Meus Documentos/minicurso em R/
> summary(m1 <- glm(affairs~age+yearsmarried + religiousness +occupation+rating, family="poisson", data=Affairs))

Call:
glm(formula = affairs ~ age + yearsmarried + religiousness +
     occupation + rating, family = "poisson", data = Affairs)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.5968  -1.5728  -1.1627  -0.7067   8.3473

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.533905   0.196924  12.867 < 2e-16 ***
age          -0.032255   0.005851  -5.512 3.54e-08 ***
yearsmarried  0.115698   0.009908  11.677 < 2e-16 ***
religiousness -0.354037   0.030892 -11.460 < 2e-16 ***
occupation    0.079828   0.019449   4.105 4.05e-05 ***
rating       -0.409443   0.027381 -14.953 < 2e-16 ***
---
signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 2925.5  on 600  degrees of freedom
Residual deviance: 2360.1  on 595  degrees of freedom
AIC: 2866.1

Number of Fisher Scoring iterations: 7

```

Repare que, neste caso, todas as variáveis tornam-se estatisticamente significantes e, comparando-se com a estimação obtida pela regressão Tobit, os sinais dos coeficientes, embora as magnitudes mudem um pouco. O modelo da traição traiu o Tobit e foi estimado por dois diferentes métodos no ano passado? Eis aí uma pergunta que não desperta a menor vontade de rir, não é? Pelo menos você já sabe como estimar os modelos *tobit* e *Poisson* no R.

## 7.4. As decisões do Banco Central e nossa vida

Uma das perguntas mais interessantes em termos de análise de conjuntura econômica é, no contexto da política monetária, tentar prever o que o Banco Central fará com a taxa de juros SELIC<sup>34</sup>. Sabidamente, o volume de informações envolvido nesta análise é considerável. Entretanto, com um pouco de Econometria, podemos tentar uma resposta que envolva probabilidades acerca das ações do Banco Central.

<sup>34</sup> Nesta seção, ilustro o trabalho de alguns anos de experiência didático-pedagógica na faculdade com o grupo de alunos denominado “Nepom” (Núcleo de Estudos de Política Monetária). Juntamente com o então aluno Pedro H.C.G. Sant’Anna (campeão do *Econometric Game* em sua edição de 2013 e, atualmente, professor da Vanderbilt University), começamos a experiência do extinto “Nepom” no início de 2008. O modelo apresentado abaixo é, basicamente, a versão ampliada, ao longo dos anos, por vários outros alunos, do modelo criado por Pedro (vale lembrar alguns nomes aqui. Primeiro, o criador do modelo, Pedro H.C. G. Sant’Anna. Além dele, correndo o risco de estar me esquecendo de alguns alunos: Luiz André B. Miranda, Jéssica Dutra, Lucas Farias Lima, Raphael Molina, Leonardo Oliveira e Thomaz Lino Amorim. Este exemplo é, de certa forma, o produto de seu trabalho. Para maiores detalhes sobre o que foi o Nepom, ver <http://nepom.wordpress.com>. Tenho cá comigo um texto sobre esta experiência que nunca consigo terminar. Caso eu desapareça da face da Terra nos próximos minutos, pode ser interessante deixar isto registrado.

Neste exemplo, ilustramos um dos modelos utilizado pelo Nepom em suas apresentações em nossa faculdade. O modelo tem como variável dependente as categorias de ação do Banco Central. Especificamente, a variável categoriza as diversas decisões do Copom em termos da taxa Selic. A base de dados é uma série de tempo irregular, iniciando-se em 2001. Primeiramente carregamos os dados e verificando as primeiras observações.

```
> nepom<-read.table("C:/Users/cdshi_000/Documents/Meus Documentos/Meus Documentos/base_lagmini.csv",
+ header=TRUE, sep=",", na.strings="NA", dec=".", strip.white=TRUE)
> head(nepom)
  lag_deciso es_copom decisoes_copom inflacao_esperada meta_inflacao prod_ind_dessaz
1          0          0              4.84             3.5          94.74
2          0         -1              4.62             3.5          95.02
3         -1         -1              4.74             3.5          97.38
4         -1          0              4.83             3.5          99.13
5          0          0              4.42             3.5         100.32
6          0          0              4.54             3.5          99.13
  prod_ind_hp      hiato
1  96.10798 -1.367979
2  96.39618 -1.376180
3  96.68429  0.695713
4  96.97211  2.157892
5  97.25950  3.060499
6  97.54647  1.583526
> |
```

Renomeando os dados, obtemos nomes mais compactos.

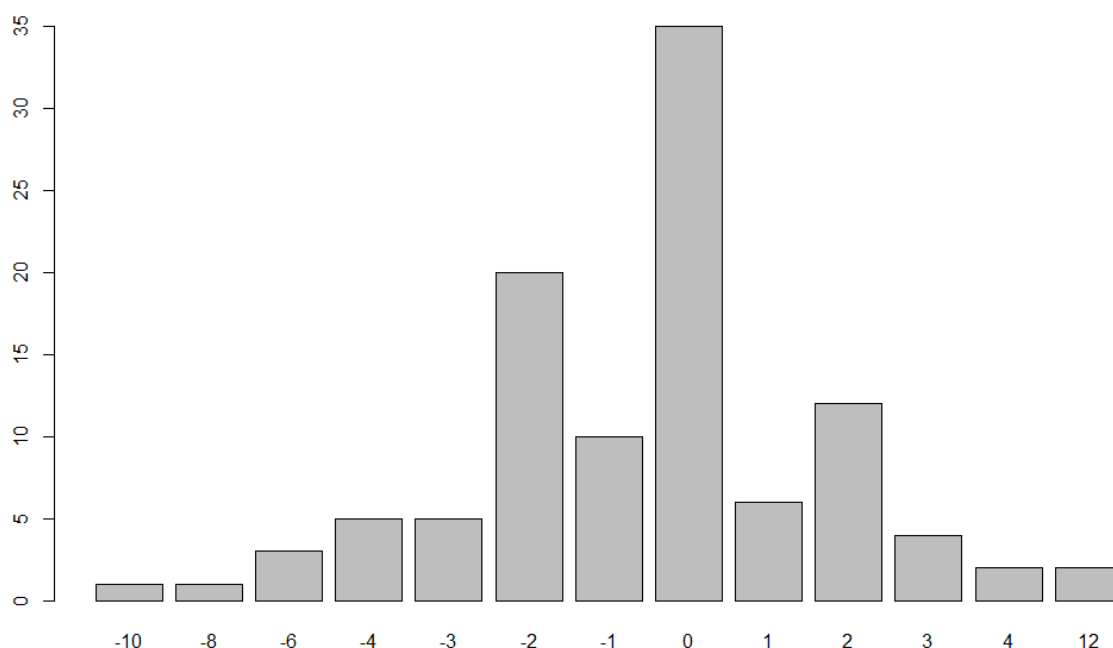
```
> dec<-nepom$deciso es_copom
> dec lag<-nepom$lag_deciso es_copom
> infesp<-nepom$inflacao_esperada
> meta<-nepom$meta_inflacao
> prod_dess<-nepom$prod_ind_dessaz
> prod_ind_hp<-nepom$prod_ind_hp
```

Repare que as decisões do Copom foram convertidas em uma variável ordenada. Desta forma, por exemplo, “0” significa que o Copom não alterou a taxa. Caso tenha ocorrido um aumento de 0.25 pontos percentuais (p.p.), então a variável assume valor “1”. Definimos aumentos de 0.25 em 0.25 pontos (e quedas de forma análoga). O período de análise se inicia em 2001.

Entretanto, alguém poderia querer criar as categorias dentro do R. Neste caso, o comando seria:

```
novacategoria<-factor(nepom$deciso es_copom)
```

Neste caso, as categorias serão criadas a partir dos dados originais e, diferentemente do caso anterior, no qual construímos os extremos das probabilidades agrupando alguns pontos, neste temos treze categorias, ilustradas no histograma a seguir.



As categorias são:

```
> summary(novacategoria)
-10 -8 -6 -4 -3 -2 -1  0  1  2  3  4 12
  1  1  3  5  5 20 10 35  6 12  4  2  2
```

Este é um exemplo de um modelo no qual a variável dependente representa respostas ordinais. Uma forma de se estimar um modelo como este é por meio de um probit ordenado. Considere as seguintes variáveis independentes – sob a ótica de um Banco Central que segue um sistema de metas de inflação – a decisão do Copom no período anterior, a diferença entre a inflação e a meta de inflação (o centro da meta, no caso do exemplo) e o hiato do produto.

Para se estimar um probit ordenado pode-se usar o pacote *MASS*.

```

> hiatoinf = infesp-meta
> hiatoprod = prod_dess/prod_ind_hp
> library("MASS")
> nepom_polr_ex2 <- polr(novacategoria~hiatoinf+log(hiatoprod), Hess=TRUE, method=c("probit"))
> summary(nepom_polr_ex2)
Call:
polr(formula = novacategoria ~ hiatoinf + log(hiatoprod), Hess = TRUE,
      method = c("probit"))

Coefficients:
              value Std. Error t value
hiatoinf      0.5904   0.09808   6.020
log(hiatoprod) 22.4373   3.54911   6.322

Intercepts:
      Value Std. Error t value
-10|-8 -3.9066  0.5702   -6.8516
-8|-6  -3.4887  0.4806   -7.2583
-6|-4  -2.7192  0.3711   -7.3270
-4|-3  -1.8468  0.2573   -7.1774
-3|-2  -1.3443  0.1997   -6.7329
-2|-1  -0.3691  0.1514   -2.4385
-1|0   -0.0062  0.1472   -0.0421
0|1     1.3009  0.1774    7.3341
1|2     1.5771  0.1897    8.3141
2|3     2.4031  0.2571    9.3478
3|4     3.0622  0.3741    8.1864
4|12     3.8041  0.5718    6.6532

Residual Deviance: 342.831
AIC: 370.831
~ |

```

As probabilidades podem ser estimadas por meio do comando *predict*.

```
Phat_nepom_polr_ex2 <- predict(nepom_polr_ex2, type="probs")
```

Neste exemplo, temos as seguintes probabilidades para a última observação da amostra (a reunião de 16/01/2013:

Ação sobre a Taxa Selic (em p.p.)	Probabilidade (%)
Cai 2.5	0.00
Cai 2	0.00
Cai -1.5	0.01
Cai 1	0.29
Cai 0.75	0.96
Cai 0.50	9.08
Cai 0.25	8.08
Não se altera	47.41
Sobe 0.25	9.47
Sobe 0.50	18.14
Sobe 0.75	5.05
Sobe 1	1.32
Sobe 3	0.18

Uma vez que tenhamos as previsões para os hiatos, é possível fazer a previsão um passo a frente e, assim, tentar verificar as probabilidades das ações do Copom nestes cenários. Obviamente, o modelo acima poderia ser modificado, por exemplo, em sua categorização, ou mesmo na especificação, já que, por exemplo, imagina-se que a decisão do Copom sobre a taxa SELIC incorpore alguma variável indicadora de desempenho futuro.

Este tipo de exercício é bem interessante, notadamente para você que curte estudar política monetária e discutir com colegas da faculdade, sejam eles economista ou não<sup>35</sup>.

## 7.5. Vamos dividir o Estado do Pará?

Ok, você já se deparou com diversas especificações para a variável dependente de uma regressão. Agora, o que fazer quando esta é expressa apenas em forma de percentual como, por exemplo, o percentual de votos em uma urna?

Pensemos em um exemplo que une motivos econômicos e políticos. No final do ano de 2011, a população do Pará foi às urnas decidir se o estado continuaria sendo o mesmo ou se seria dividido em três: Pará, Carajás e Tapajós. Este é um problema que se enquadra exatamente no escopo desta seção. O modelo mais adequado, neste caso, é a regressão beta que, no R, é implementada pelo pacote *betareg*<sup>36</sup>.

Como dito acima, quando a variável dependente é uma proporção, o modelo adequado é a regressão beta<sup>37</sup>. Neste exemplo, a variável dependente é justamente a proporção de votos (resultado da votação) em cada município do Pará. Quanto às variáveis independentes, elas são: o logaritmo da área de cada município em 2010, o logaritmo do PIB municipal de 2009, o logaritmo do PIB per capita por município de 2009, a distância do município para a capital, Índice FIRJAN de Desenvolvimento Municipal (IFDM) de 2009 e o Índice FIRJAN de Gestão Fiscal (IFGF) de 2009, uma *dummy* para os municípios que fariam parte de Carajás e outra para os municípios que fariam parte de Tapajós (sendo 1 para os que fariam parte da região e 0 para os que fariam parte de outra)<sup>38</sup>.

---

<sup>35</sup> Obviamente, o modelo acima é bem simples e sujeito a diversas críticas (como qualquer modelo econométrico ou de concurso de *Miss*). Destaco, por exemplo, dois problemas sempre apontados por vários colegas: (a) a ausência de uma variável de expectativa *forward looking* e (b) a ausência de algum termo capaz de captar mudanças radicais na política monetária (como a politização do Banco Central em detrimento do sistema de metas).

<sup>36</sup> Fernanda Faria, uma ex-aluna, decidiu explorar este tema e seu trabalho resultou em Shikida, Faria e Araujo Jr (2014) (<http://portalrevistas.ucb.br/index.php/EALR/article/view/5%20EALR%201>). Esta seção baseia-se neste artigo.

<sup>37</sup> Para detalhes sobre a regressão beta, ver, por exemplo, Cribari-Neto & Zeileis (2009).

<sup>38</sup> As *dummies* para Carajás e Tapajós foram adicionadas ao modelo para avaliar a influência estatística do pertencimento do municípios à região que pleiteia tornar-se novo estado. Espera-se que coeficientes estimados das *dummies* sejam positivos. Para Carajás a *dummy* foi denominada “dum\_car” e para Tapajós “dum\_tap”.



Primeiramente, carrega-se o pacote *betareg* e, em seguida, são estimadas algumas especificações. Vejamos os comandos para estimar uma delas<sup>39</sup>.

```
paracar31<-carajsim~ifdm+ifgf+lpib+lpib_pcap+larea+dist_cap+dum_car+dum_tap
paracar31_logit<-betareg(paracar31,data=fer,type="BR")
summary(paracar31_logit)
```

O comando na primeira linha especifica a regressão. A segunda linha estima a regressão beta com a família logística, utilizando a base de dados “fer”, na qual estão as variáveis citadas. A estimação é feita com redução de viés (*bias reduction*, BR). Finalmente, a visualização dos resultados é obtida com o comando *summary()*.

```
> paracar31<-carajsim~ifdm+ifgf+lpib+lpib_pcap+larea+dist_cap+dum_car+dum_tap
> paracar31_logit<-betareg(paracar31,data=zy,type="BR")
> summary(paracar31_logit)
```

Call:  
betareg(formula = paracar31, data = zy, type = "BR")

Standardized weighted residuals 2:

	Min	1Q	Median	3Q	Max
	-2.3118	-0.6260	0.0945	0.6838	2.7436

Coefficients (mean model with logit link):

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-1.7772746	1.0824636	-1.642	0.100615
ifdm	-2.6137810	1.6184587	-1.615	0.106315
ifgf	-0.3819611	0.4309806	-0.886	0.375477
lpib	0.2217971	0.0914566	2.425	0.015302 *
lpib_pcap	-0.1430727	0.1916773	-0.746	0.455411
larea	-0.2215384	0.0600050	-3.692	0.000223 ***
dist_cap	0.0018556	0.0002774	6.689	2.24e-11 ***
dum_car	5.5864174	0.2030224	27.516	< 2e-16 ***
dum_tap	4.6041371	0.3065156	15.021	< 2e-16 ***

Phi coefficients (precision model with identity link):

	Estimate	Std. Error	z value	Pr(> z )
(phi)	60.92	8.64	7.051	1.77e-12 ***

---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Type of estimator: BR (bias-reduced)  
Log-likelihood: 250 on 10 Df  
Pseudo R-squared: 0.9656  
Number of iterations: 22 (BFGS) + 15 (Fisher scoring)

Repare que um resultado adicional é a estimação do coeficiente  $\Phi$  (phi). Valores elevados do mesmo indicam baixa dispersão da variável dependente. A ocorrência de *observações influentes* e *outliers* pode afetar a estimação. Assim, deve-se analisar os resíduos da regressão estimada e, se necessário, reestimar o modelo sem estas observações. O comando *plot()* retorna vários testes de diagnóstico de resíduos como a distância de Cook, por exemplo. Desta forma, o passo seguinte seria reestimar a especificação, retirando tais observações.

<sup>39</sup> Nesta parte do *script* você vai se deparar com alguns ajustes prévios na base de dados. Para nos poupar tempo, apresento apenas o resultado final.

Mas vale comentar alguns dos resultados. Vejamos as três últimas: *dist\_cap*, *dum\_car*, *dum\_tap*. A primeira é a distância do município à capital do Pará. O sinal positivo, embora baixo, é fortemente significativo, estatisticamente falando, o que indica um impacto positivo na probabilidade de se votar pela emancipação do, então, “estado de Carajás”.

As duas outras variáveis são *dummies* para os municípios que estariam, respectivamente, nos potenciais estados de Carajás e Tapajós. O efeito positivo verificado em ambos os coeficientes, supondo que seja realmente digno de análise (e que eventuais ajustes do modelo não alterem este resultado) parece indicar uma espécie de efeito “solidário” entre as populações envolvidas no processo de criação dos estados<sup>40</sup>.

---

<sup>40</sup> Lembra, de certa forma, o famoso *logrolling* (ou “comércio de votos”), exemplo clássico em livros-texto de *Public Choice*.

## 8. Econometria de Séries de Tempo em R (Parte I)

O estudo de dados no domínio do tempo é um dos mais interessantes e difíceis da Econometria moderna. Nestas duas próximas seções veremos algumas abordagens tradicionalmente usadas em dados que se situam no domínio do tempo.

Antes de mais nada, contudo, vamos tratar de um problema muito comum para o usuário do R: a questão de como trabalhar com séries de tempo. Primeiramente, sabemos que séries de tempo podem ocorrer em intervalos regulares (e.g., diários, semanais, mensais, trimestrais, semestrais, anuais, etc) e irregulares. O formato padrão do R para dados no tempo regulares é o formato *ts*. Para dados irregulares ou de elevada frequência (e.g. diários, semanais, etc), existe o pacote *zoo*<sup>41</sup>.

Veja o que acontece, por exemplo, com esta base de dados de taxas de câmbio diárias.

```
> base <- read.table("C:/Users/cdshi_000/Documents/Meus Documentos/Meus Documentos/cambio_Brasil.csv",
+ header=TRUE, sep=",", na.strings="NA", dec=".", strip.white=TRUE)
> head(base)
```

	Data	BRL_usd
1	1/2/1995	0.844
2	1/3/1995	0.845
3	1/4/1995	0.845
4	1/5/1995	0.843
5	1/6/1995	0.840
6	1/9/1995	0.844

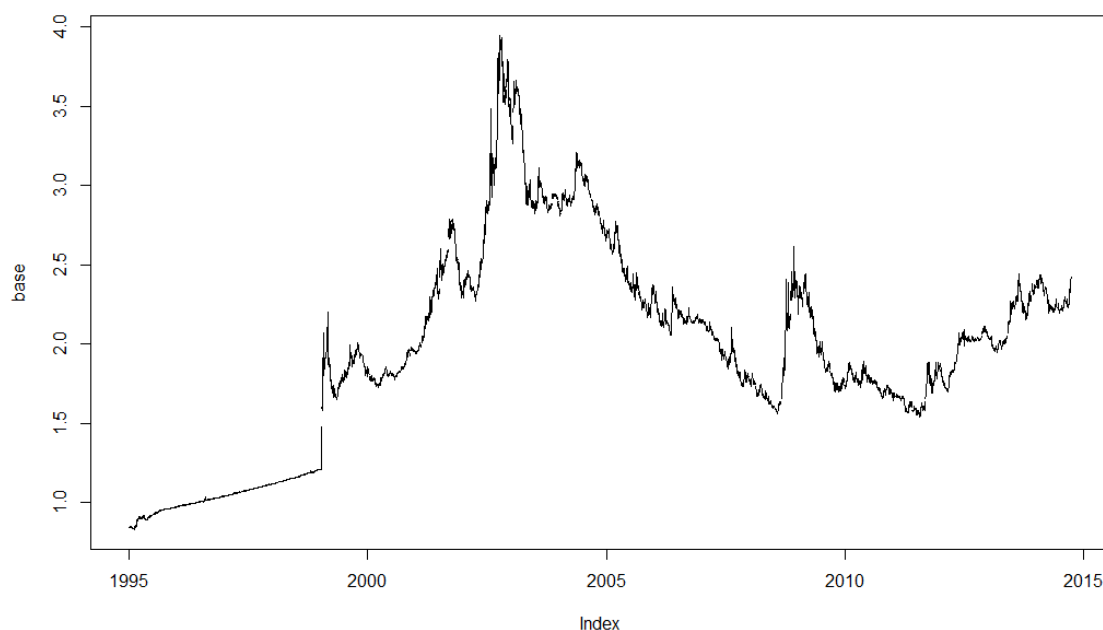
Note que os dados são diários e, claro, não existem valores para sábados e domingos. Alguém poderia querer usar o comando *ts*, mas o mesmo só funciona, como dito, para intervalos regulares. A saída, neste caso, é usar o formato *zoo*. Veja como a base pode ser lida neste caso.

```
> base <- read.zoo("C:/Users/cdshi_000/Documents/Meus Documentos/Meus Documentos/cambio_Brasil3.csv",
+ header=TRUE, sep=",", format = "%Y-%m-%d")
> head(base)
```

1995-01-02	1995-01-03	1995-01-04	1995-01-05	1995-01-06	1995-01-09
0.844	0.845	0.845	0.843	0.840	0.844

A aplicação simples do comando *plot* nos dá o gráfico da série.

<sup>41</sup> Vale a pena consultar as documentações do *zoo*: <https://cran.r-project.org/web/packages/zoo/index.html>.



Outro formato que é utilizado na comunidade R é o das séries de tempo em formato *xts*, do pacote homônimo<sup>42</sup>. Aparentemente, a pretensão é que este formato seja tal que se possa trabalhar com todas as séries de tempo de formato *zoo* ou *ts*.

Finalmente, o formato mais popular para dados de frequência mensal, trimestral ou anual, é *ts*, cuja sintaxe é razoavelmente simples. Vejamos o caso do índice da taxa de câmbio real efetivo calculado pelo BIS. Os dados são mensais e seguem o formato ilustrado abaixo. Note que, para a China, por exemplo, o código da variável é RBCN.

1	BIS effective exchange rate													
2	Real (CPI-based), Broad Indices													
3	Monthly averages; 2010=100													
4	EER for:	Algeria	Argentina	Australia	Austria	Belgium	Brazil	Bulgaria	Canada	Chile	China	Chinese	Colombia	Croatia
5		RBDZ	RBAR	RBAU	RBAT	RBBE	RBBR	RBBG	RBCA	RBCL	RBCN	RBTW	RBCO	RBHF
6	01-1994	175.35	251.07	74.28	105.79	100	83.13	46.35	88.28	93.5	65.58	144.53	71.15	86.1
7	02-1994	182.45	250.26	75.36	105.69	100.71	81.12	46.87	85.32	93.36	65.87	144.32	72.75	87.1
8	03-1994	183.77	247.43	74.3	106.8	101.53	81.46	38.89	83.43	93	66.08	142.69	83.63	88.1

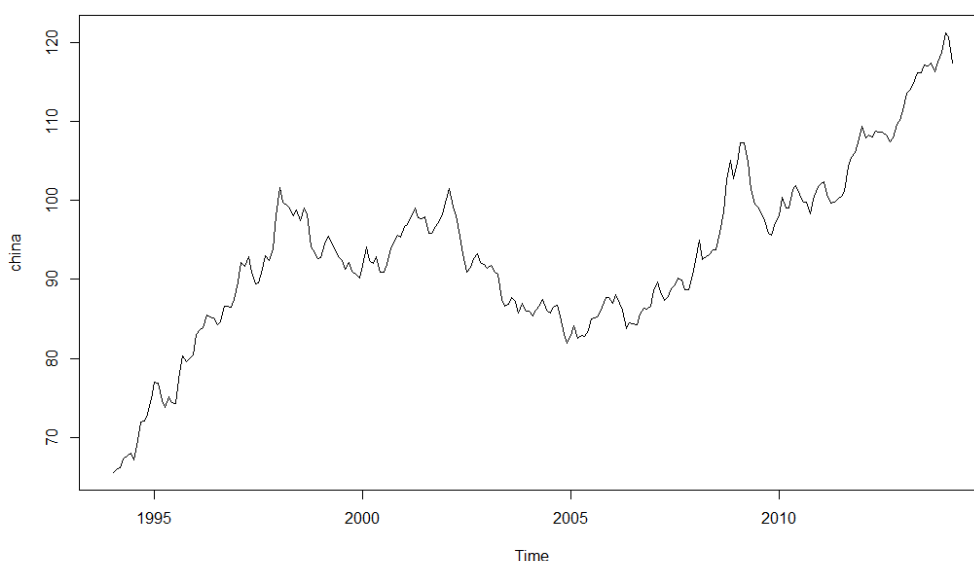
Pode-se marcar a planilha a partir da célula B5, copiar seu conteúdo e...

```
> exemp<-read.table(file="clipboard",sep="\t",header=TRUE)
> head(exemp)
  RBDZ  RBAR  RBAU  RBAT  RBBE  RBBR  RBBG  RBCA  RBCL  RBCN  RBTW
1 175.35 251.07 74.28 105.79 100.00 83.13 46.35 88.28 93.50 65.58 144.53
2 182.45 250.26 75.36 105.69 100.71 81.12 46.87 85.32 93.36 65.87 144.32
3 183.77 247.43 74.30 106.80 101.53 81.46 38.89 83.43 93.00 66.08 142.69
4 137.33 247.53 74.50 106.04 101.23 80.92 40.70 82.08 94.43 67.20 143.08
5 128.55 246.55 74.90 106.67 102.04 80.10 42.76 81.72 94.99 67.60 141.20
6 124.91 244.40 75.37 107.05 102.61 81.65 44.72 81.34 95.58 67.93 138.49
  RBDK  RBEE  RBXM  RBF1  RBFR  RBDE  RBGR  RBHK  RBHU  RBIS  RBIN
```

<sup>42</sup> Para detalhes, ver: <https://cran.r-project.org/web/packages/xts/vignettes/xts.pdf>.

Repare que o primeiro comando “cola” a área copiada da planilha a partir da área de transferência (*clipboard*) para o R. O segundo comando serve apenas para uma conferência superficial. Note que o R não atribuiu às colunas qualquer propriedade associadas ao tempo. Sabemos, contudo, da figura anterior, que as séries começam em 1994. Vamos usar apenas a série da China.

```
> china<-ts(exemp$RBCN,start=c(1994,1),freq=12)
> plot(china)
```



O comando informa o início da série e sua frequência (mensal = 12, trimestral = 4, anual = 1, etc). Objetos *ts* como este podem ser trabalhados sob diferentes janelas de tempo. Por exemplo, suponha que se quisesse trabalhar com esta série apenas no período de jan/2009 até dez/2012. O comando *window()* pode ser aplicado aqui. Digamos que eu queira salvar a nova série. Então, a sintaxe seria: *novo\_nome <- window(china, start = c(2009, 1), end = c(2012,12))*. Não é difícil perceber que boa parte dos dados macroeconômicos pode ser tratada como objeto *ts* e que dados financeiros geralmente são tratados como *zoo* ou *xts*<sup>43</sup>.

### 8.1. É fácil assim fazer a previsão da produção industrial brasileira?

Eis aí uma pergunta com uma resposta direta: não, não é fácil. Para entender o porquê, lembre-se do problema das tendências na seção que tratou da função consumo. O leitor já familiarizado com a literatura sabe que um dos problemas mais estudados é o da não-estacionariedade das séries. No R, este problema pode ser trabalhado com vários pacotes e veremos um pouco sobre isto adiante. Após importar os dados, transformarmos a série em um objeto de série de tempo.

<sup>43</sup> Veja esta comparação entre objetos *zoo* e *xts*: <http://blog.revolutionanalytics.com/2014/01/quantitative-finance-applications-in-r-plotting-xts-time-series.html>.

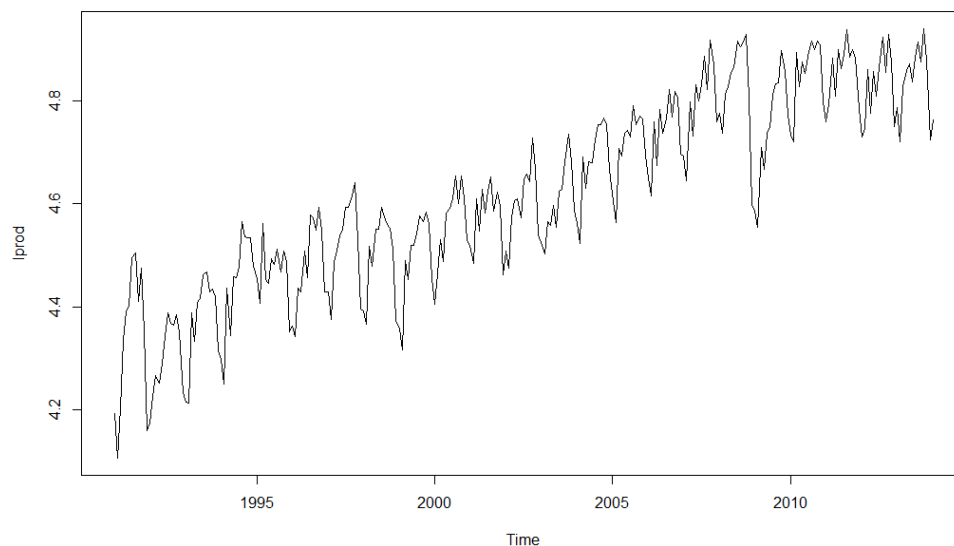
```

> base<-read.table("C:/Users/cdshi_000/Documents/Meus Documentos/Meus Documentos/prod_ind_teacher.csv",header=TRUE, sep=","
+ na.strings="NA", dec=".", strip.white=TRUE)
> summary(base)
      prodind
  Min.   : 60.75
  1st Qu.: 88.50
  Median : 99.54
  Mean   :102.99
  3rd Qu.:117.94
  Max.   :139.76
> base$prodind <- ts(base$prodind,start=c(1991,1),freq=12)
> prod<-base$prodind
> lprod<-log(prod)

```

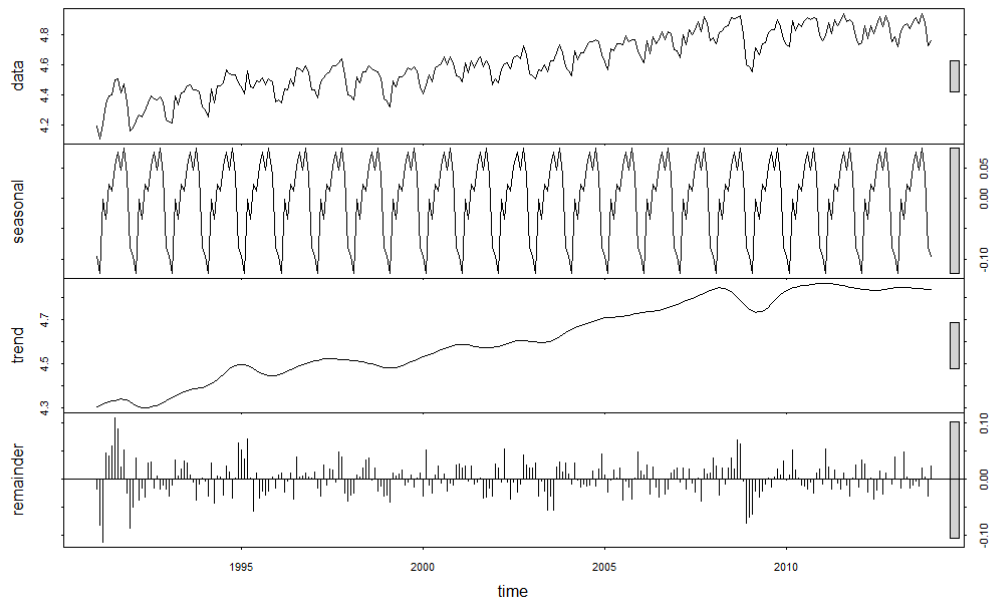
Como sempre, é bom visualizar o gráfico de uma série de dados. Em nosso caso, temos o índice da produção industrial brasileira (IBGE) (sem a aplicação prévia de qualquer método de dessazonalização). Iniciamos com o gráfico da mesma, em escala logarítmica.

### Produção Industrial

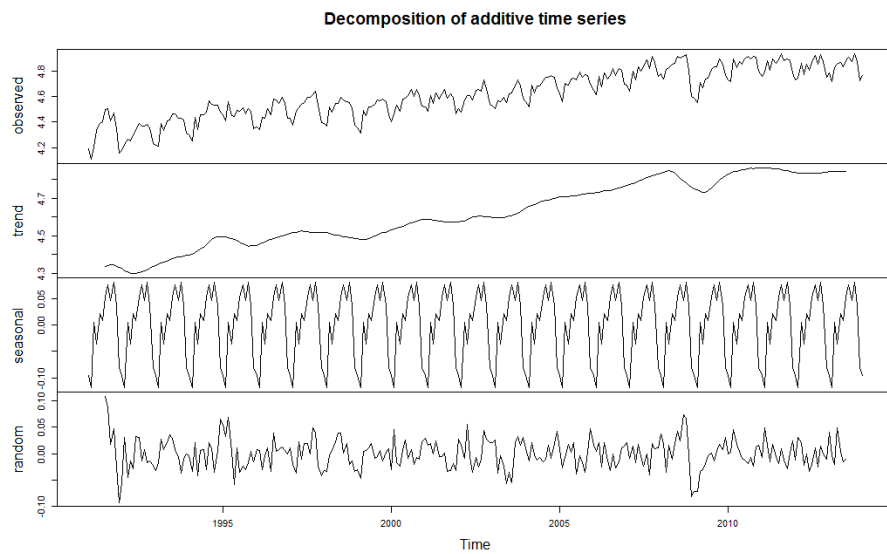


No pacote básico do R, *stats*, temos o comando *stl* que serve para decompor a série de tempo em componentes sazonais, tendência e componente irregular. Vejamos como nossa série se apresenta em ambas as decomposições.

### Produção Industrial – Decomposição via *Loess smoothing*



### Produção Industrial – Decomposição Clássica



Como se vê, não há grande diferença nos resultados da decomposição, seja por um método, seja pelo outro.

Um outro resumo da série pode ser obtido pelo comando *tsdisplay* do pacote *forecast*<sup>44</sup>.

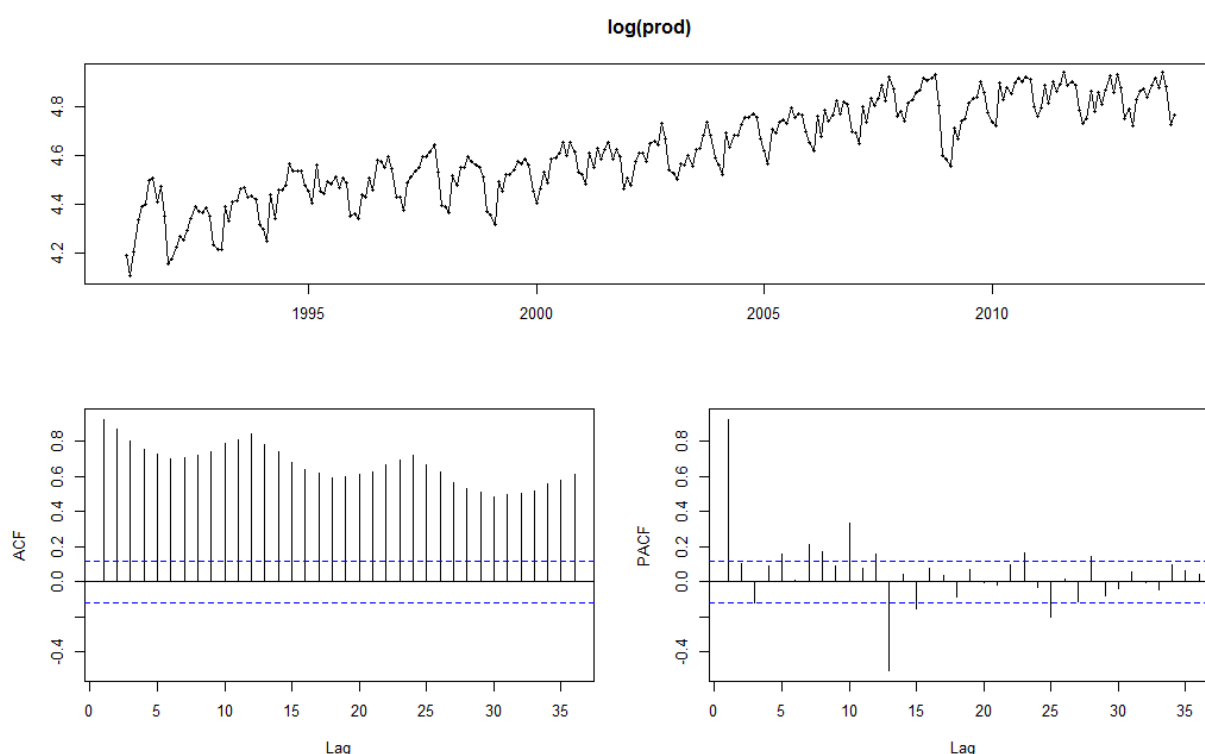
```
> library(forecast)
Loading required package: timeDate
This is forecast 7.1

Attaching package: 'forecast'

The following object is masked from 'package:astsa':

    gas

> tsdisplay(log(prod))
> |
```



A metodologia Box-Jenkins exige uma análise detalhada da série, partindo dos gráficos e do padrão dos correlogramas. Para se estimar modelos SARIMA em R é necessário observar, inicialmente, os correlogramas das séries. Como apontado por alguns autores, os correlogramas do R apresentam a desconfortável propriedade de apresentar uma correlação (unitária) na defasagem zero. Sabemos que isto é verdade, mas usualmente, não é assim que correlogramas são apresentados. Os pacotes mais úteis, neste caso, são a *forecast* e *astsa*.

<sup>44</sup> Pode-se utilizar os comandos *Acf()* e *Pacf()* da biblioteca *astsa*. Até a versão atual (3.2.5), os comandos *acf()* e *pacf()* do pacote básico *stats* apresentam o desconfortável problema de visualização de apresentar uma correlação no ponto de defasagem zero...



Faremos uso do pacote *forecast* para estimar um SARIMA para a produção industrial. O leitor poderia querer usar o comando *arima*, do pacote básico do R, *stats*. Ocorre que este comando gera problemas de interpretação na estimação do intercepto de alguns modelos. Para maiores detalhes, ver Shumway & Stoffer (2011)<sup>45</sup>.

Desta forma, o comando que utilizaremos é o *sarima* (lembre-se: do pacote *forecast*)<sup>46</sup>. Eis o primeiro modelo, um SARIMA (2,1,1) x (2,0,2).

```
Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
REPORT = 1, reltol = tol))

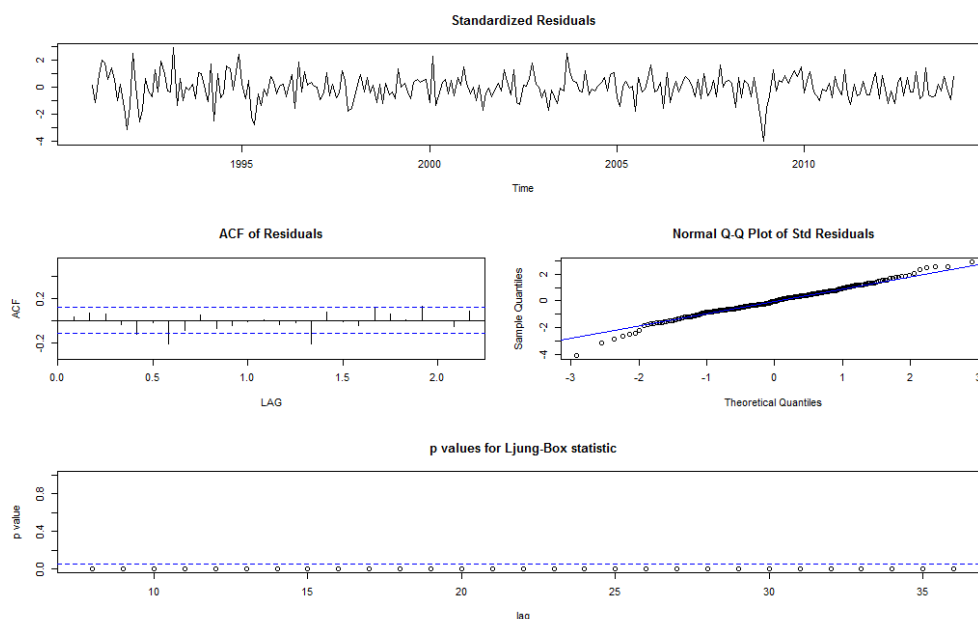
Coefficients:
      ar1      ar2      ma1      sar1      sar2      sma1      sma2
-0.8215 -0.3693  0.4839  0.2952  0.7047 -0.1826 -0.7753
s.e.    0.1233  0.0620  0.1213  0.3326  0.3326  0.2929  0.2827

sigma^2 estimated as 0.001041:  log likelihood = 532.15,  aic = -1048.31

$AIC
[1] -5.816919

$AICC
[1] -5.807759

$BIC
[1] -6.725337
```



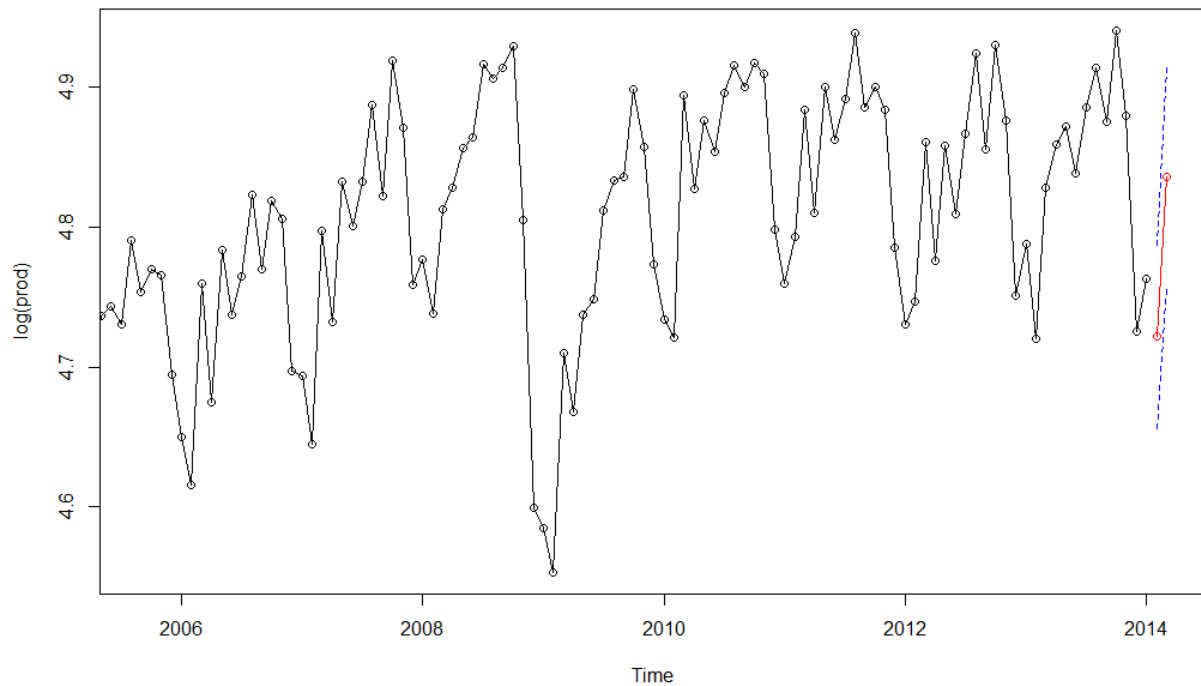
Vejamos como fazer a previsão dois passos à frente.

<sup>45</sup> O problema não é exclusivo do R, mas aparece em saídas de outros programas como o Eviews e o Gretl e diz respeito ao problema da interpretação do intercepto em modelos ARIMA. Embora não haja maiores consequências para os outros parâmetros estimados, caso o seu objetivo seja estimar modelos deste tipo para previsão, então este é um problema relevante.

<sup>46</sup> O autor do pacote *forecast*, Rob Hyndman, tem várias dicas boas para o trabalho com séries de tempo em R. Confira mais a partir daqui: <http://robjhyndman.com/>.

```
> sarima.for(log(prod),2,2,1,1,2,0,2,12,no.constant=TRUE)
$pred
      Feb      Mar
2014 4.721579 4.835933

$se
      Feb      Mar
2014 0.03263715 0.03914554
```



Eis uma forma alternativa de se obter o mesmo modelo. Usamos o pacote *astsa* e o comando *Arima* (não confundir com o comando *arima*).

```

> library(astsa)
> Arima(log(prod),order=c(2,1,1), seasonal=list(order=c(2,0,2), period=12))
Series: log(prod)
ARIMA(2,1,1)(2,0,2)[12]

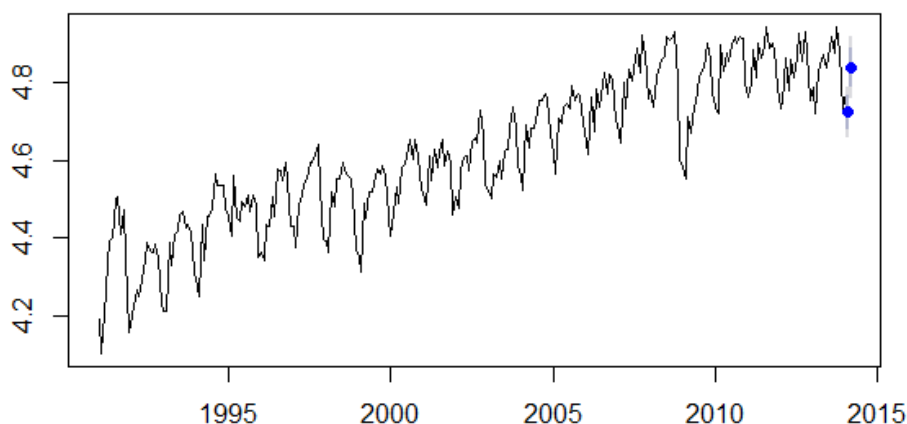
Coefficients:
      ar1      ar2      ma1      sar1      sar2      sma1      sma2
-0.8215 -0.3693  0.4839  0.2952  0.7047 -0.1826 -0.7753
s.e.    0.1233  0.0620  0.1213  0.3326  0.3326  0.2929  0.2827

sigma^2 estimated as 0.001068:  log likelihood=532.15
AIC=-1048.31  AICC=-1047.77  BIC=-1019.34
> fit1<-Arima(log(prod),order=c(2,1,1), seasonal=list(order=c(2,0,2), period=12))
> plot(forecast(fit1, h=2))
> accuracy(fit1)

```

	ME	RMSE	MAE	MPE	MAPE	MASE
Training set	-0.0006340156	0.03220905	0.02508822	-0.01686626	0.5482307	0.4561027
ACF1						
Training set	0.02835718					

**Forecasts from ARIMA(2,1,1)(2,0,2)[12]**



Repare que aproveitei para obter também as medidas de acurácia para a previsão, úteis para fins de comparação entre modelos. Um detalhe: o comando *accuracy* é do pacote *forecast*, mas eu o apliquei ao SARIMA estimado pelo comando *Arima* do pacote *astsa*. Esta flexibilidade é bem útil em alguns casos, não acha?

Voltando às medidas de acurácia, você provavelmente já as viu mas caso não se lembre, independentemente de sua escala de medida, a leitura é trivial: modelos com menores erros de previsão, sob qualquer escala de medida escolhida, possuem maior acurácia. Logo, em ME, RMSE, MAE, MPE, MAPE ou MASE, queremos sempre o menor valor possível, comparado à correspondente medida do(s) modelo(s) concorrente(s).

Vamos estimar um outro modelo, que chamaremos de *fit2*, que será um SARIMA (3,1,0)×(1,0,1). Vamos usá-lo para ilustrar o teste de Diebold-Mariano (e você mesmo pode checar as medidas de acurácia). No pacote *forecast*, este teste é executado facilmente.

## Diebold-Mariano Test

```
data: residuals(fit1)residuals(fit2)
DM = 0.30003, Forecast horizon = 1, Loss function power = 2, p-value =
0.7644
alternative hypothesis: two.sided
```

A hipótese nula é a de que ambos os modelos têm a mesma acurácia preditiva. O p-valor indica uma forte evidência de que esta hipótese não é rejeitada. Em outras palavras, o desempate dos modelos não será obtido por este teste.

É bem provável que você esteja se perguntando sobre um aspecto central do estudo de séries de tempo na abordagem ARIMA: o número de diferenciações de uma série. Em termos modernos, o procedimento envolve, geralmente, o recurso a algum teste de raiz unitária.

Voltemos ao início desta análise. Tínhamos, basicamente, que apresentava um forte caráter sazonal e também uma tendência. Isto significa que é possível que a série possa ser diferenciada em seu nível e também em seu nível sazonal. O pacote *forecast* nos fornece dois comandos úteis para uma análise rápida acerca disto: *ndiffs* e *nsdiffs*.

O que os comandos nos dizem sobre a série?

```
> ndiffs(lprod)
[1] 1
> nsdiffs(lprod)
[1] 0
```

Obviamente, aquele que já teve contato com a Econometria deve estar se perguntando sobre qual teste foram utilizados. Use “*?ndiffs*” e “*?nsdiffs*” para descobrir que o *default* de cada teste é, respectivamente, o KPSS e o Osborn-Chui-Smith-Birchenhall<sup>47</sup>.

Finalmente, na abordagem ARIMA, um comando útil do pacote *forecast* é o *auto.arima*. Não se deve confiar cegamente em algoritmos automáticos, mas é divertido comparar o que fazemos com um modelo e o que um algoritmo automático faz (no caso, o melhor modelo estimado pelo comando foi um SARIMA (3,1,3)×(1,0,0)).

A produção industrial poderia ser modelada de outras formas. O leitor é convidado a verificar, por exemplo, as possibilidades no pacote *forecast*. Também é recomendável analisar outros tipos de modelos como os apresentados, por exemplo, em Hyndman et al (2008). Finalmente, a famosa “marolinha”, um aspecto tão comentado na imprensa, poderia

<sup>47</sup> Outra opção é usar os testes de raiz unitária da biblioteca *urca*. Veremos isso adiante.

ser testada por meio de testes de quebra estrutural. Não faremos isto aqui, mas a próxima seção dá pistas sobre fazê-lo<sup>48</sup>.

#### Anexo 4 – Modelos ARIMA degenerados

Nesta seção vimos como usar o R para a estimação de modelos (S)ARIMA. Entretanto, é bom lembrar que, em alguns casos, pode ser importante estimar modelos desta classe *degenerados*. Suponha que você obteve uma outra série, *ppi*, e estimou o modelo SARIMA (2,1,1) × (2,0,2) e notou que os termos sazonais autoregressivo e de médias móveis de primeira ordem não são estatisticamente significativos a 5%. Então, poderíamos querer estimar um SARIMA (2,1,1) × (2\*, 0, 2\*) no qual o “\*” ilustra a ausência **apenas dos termos de primeira ordem autoregressivos e de média móvel**.

O comando *sarima* não faz isso e, assim, temos que usar o *arima* ou *Arima*, vistos previamente. O comando seria assim:

```
Arima(ppi, order = c(2,1,0),seasonal=list(order=c(2,0,2)),include.constant=TRUE,
      fixed = c(NA,NA, NA, 0,NA, 0, NA), transform.pars = FALSE)
```

Repare em *fixed = c(NA,NA, NA, 0,NA, 0, NA)*. Ele nos diz que o programa não deve (os dois primeiros “NA”) colocar restrições na ordem *p* da parte não-sazonal do modelo. Em seguida, ele restringe *P* = 1 a zero, mas permite o termo autoregressivo sazonal de segunda ordem (os “0” e “NA” seguintes). A mesma coisa para *q* = 1 e *q* = 2 (os seguintes “0” e “NA”). Finalmente, o último “0” indica que não se deve restringir o *drift*<sup>49</sup>.

#### Anexo 5 – Tendências, *dummies* sazonais e algumas dicas básicas

Um outro jeito de se tratar a sazonalidade é por meio de *dummies* sazonais. Desta forma, cria-las, no R, pode ser um bocado trabalhoso. Entretanto, existem atalhos. Por exemplo, no pacote *forecast*, podemos usar a frequência da variável “CA” para criar *dummies* sazonais.

```
library(forecast)
```

```
SEASON <- ts(seasonaldummy(CA), start = start(CA),frequency = frequency(CA))
```

Os dados são mensais e, assim, são onze *dummies* (excluindo Dezembro). Esta é uma forma rápida e simples de se criar *dummies* no R, quando o estudo envolve amostras de dados de séries de tempo.

Outras dicas úteis dizem respeito a como criar uma tendência determinista e como transformá-la em uma série de tempo. Já vimos anteriormente como criar a série de

<sup>48</sup> Uma nota rápida, mas necessária. Suponha que se quisesse trabalhar com a produção industrial apenas no período de jan/2009 até dez/2012. Poderíamos usar o comando *window()*, adequado para trabalhar com sub-amostras em séries de tempo. Eis a sintaxe: *novo\_nome <- window(log(prod), start = c(2009, 1), end = c(2012,12))*.

<sup>49</sup> A convergência deste tipo de algoritmo, contudo, nem sempre ocorre.

tendência. Entretanto, para transformá-la em série de tempo basta fazer como na leitura de dados como séries de tempo. Assim, por exemplo, uma tendência como a série *trend* pode ser facilmente criada como se vê a seguir.

```
trend<-seq(1:349)
```

Para transformá-la em série de tempo:

```
tendencia<-ts(trend, start=c(1985), freq=12)
```

Implementar uma regressão com estas variáveis é bem simples. Por exemplo, fazendo uso do pacote *dynlm*, obtemos:

```
dynlm(Imp_US~Yen_Dolar+tendencia+SEASON))
```

O restante do trabalho é trivial.

## 8.2. A função consumo keynesiana é estável para o Brasil? (O Segundo Ato)

A pergunta é muito pretenciosa, mas serve para introduzirmos alguns testes de quebra estrutural. Respectivamente, o teste CUSUM e o sup\_F. O procedimento é estimar a regressão e fazer o teste. Nesta seção aproveitamos para alterar nossa base de dados: usaremos dados da famosa *Penn World Tables*. Existem dois pacotes em R para estas bases: *pwt* e *pwt8* sendo que o último inclui as atualizações mais recentes da base: 8.0 e 8.1. Deve-se construir a série do consumo, isolar os dados do Brasil e, finalmente informar ao R que o consumo e o PIB são séries de tempo. Isto é feito nos comandos abaixo.

```
> library(pwt8)
> pwt8.1$cons<-(pwt8.1$cs_h_c)*pwt8.1$cgdpe
> pwt8.1$pmec<-pwt8.1$cons / pwt8.1$cgdpe
> data <- subset(pwt8.0, isocode %in% c("BRA"))
> data$cons<-ts(data$cons, start=c(1950),freq=1)
> data$cgdpe<-ts(data$cgdpe, start=c(1950), freq=1)
> |
```

---

Em seguida, estima-se a função consumo keynesiana.

```

> consbr_keyn<-lm(data$cons~data$cgdpe)
> summary(consbr_keyn)

Call:
lm(formula = data$cons ~ data$cgdpe)

Residuals:
    Min       1Q   Median       3Q      Max
-155548  -18722   -4489   29488  104369

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.434e+04  8.929e+03   2.726  0.00838 **
data$cgdpe    6.006e-01  1.044e-02  57.519 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 42290 on 60 degrees of freedom
Multiple R-squared:  0.9822,    Adjusted R-squared:  0.9819
F-statistic: 3308 on 1 and 60 DF,  p-value: < 2.2e-16

```

Percebemos que a propensão marginal a consumir, nesta especificação, é de aproximadamente 0.6. A questão é se esta função é “estável” no sentido de que os parâmetros estimados – dentre os quais a propensão marginal a consumir é claramente a que mais nos interessa – é constante ao longo do período. Como sempre, vamos aos testes. O pacote utilizado é *strucchange*. A hipótese nula de qualquer um dos testes é a de que não há quebra estruturais. Vejamos o primeiro teste.

```

> library(strucchange)
> consbr_keyn_ocus<-efp(data$cons~data$cgdpe, type="OLS-CUSUM")
> sctest(consbr_keyn_ocus)

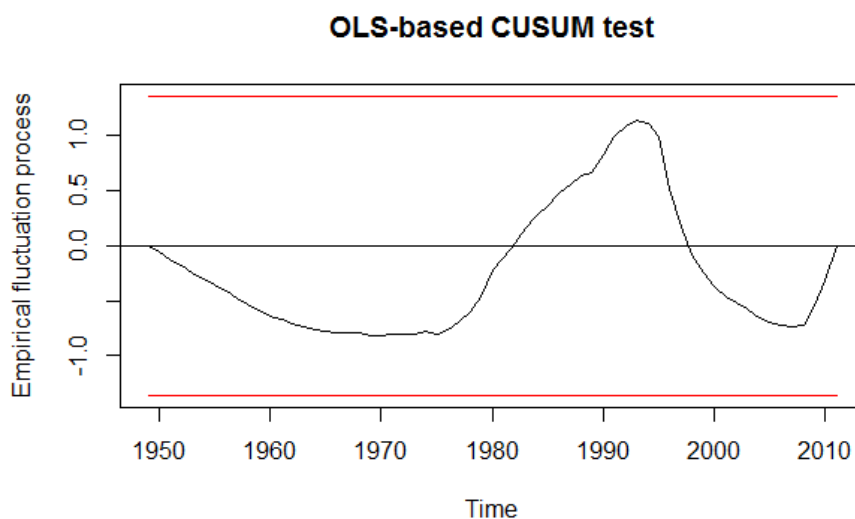
      OLS-based CUSUM test

data:  consbr_keyn_ocus
S0 = 1.137, p-value = 0.1507

> plot(consbr_keyn_ocus)
<

```

O resultado não é tão animador e, assim, vale a pena ter uma ideia visual do teste. Para tanto, basta usar `plot(consbr_keyn_ocus)`. O gráfico mostra que não existem quebras (a(s) mesma(s) ocorreria(m) se o processo da flutuação ultrapassasse(m) alguma das fronteiras (inferior ou superior).



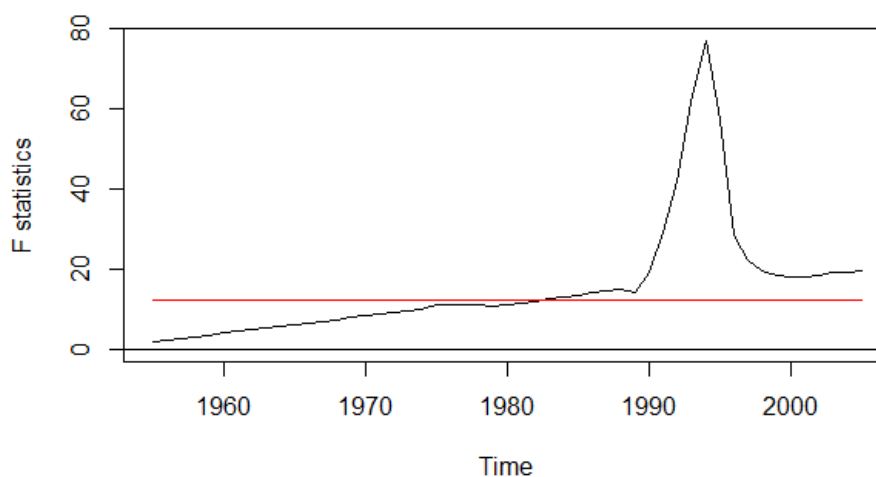
O segundo teste, o teste do sup F, abaixo ilustrado, mostra que deve existir uma quebra estrutural na série.

```
> plot(consbr_keyn_ocus)
> consbr_keyn_fs<-Fstats(data$cons~data$cgdpe,from=0.1)
> sctest(consbr_keyn_fs)
```

supF test

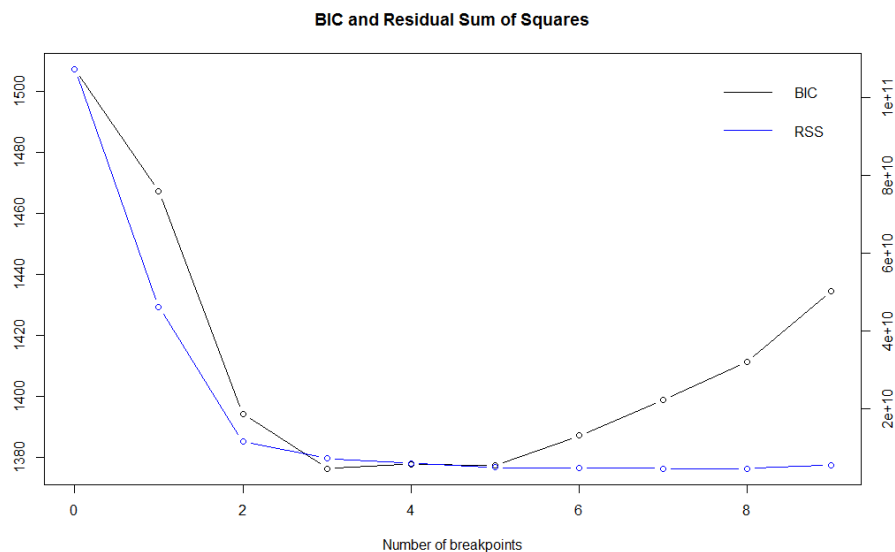
```
data: consbr_keyn_fs
sup.F = 76.967, p-value = 7.037e-16
```

```
> plot(consbr_keyn_fs)
```



Como destacam Kleiber & Zeileis (2008), não existe um teste de quebra estrutural superior a outro. Assim, o importante é estudar o contexto histórico da série. Como último exemplo, vamos aplicar o teste de quebras endógenas de Bai- Perron nesta regressão.





Vejamos o que diz a saída do teste.

```
> consbr_bp <- breakpoints(data$cons ~ data$cgdpe, h=0.1)
> consbr_bp

      optimal 4-segment partition:

call:
breakpoints.formula(formula = data$cons ~ data$cgdpe, h = 0.1)

Breakpoints at observation number:
29 40 49

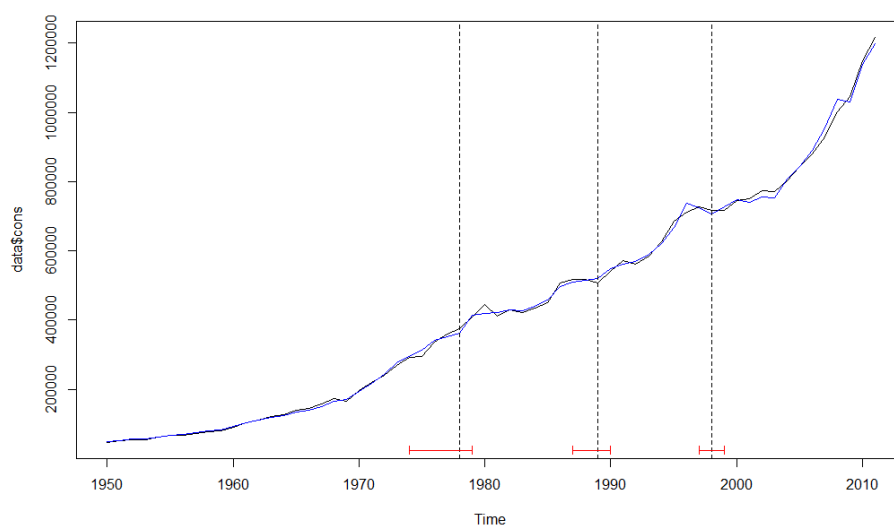
Corresponding to breakdates:
1978 1989 1998
> plot(consbr_bp)
>
```

Quais seriam os segmentos e quais seriam as propensões marginais a consumir estimadas em cada um deles?

```
> coef(consbr_bp, breaks=3)
              (Intercept) data$cgdpe
1950 - 1978    -1957.113    0.6843976
1979 - 1989   152850.608    0.4638467
1990 - 1998   320425.498    0.2965864
1999 - 2011  -304792.264    0.8299015
```

Perceba que a propensão marginal a consumir oscila um bocado e, claro, é importante lembrar que alguns segmentos são muito pequenos como os dois intermediários. Vejamos como fica o gráfico com as quebras para uma visualização melhor.

```
> plot(data$cons)
> lines(fitted(consbr_bp, breaks=3), col=4)
> lines(confint(consbr_bp, breaks=3))
>
```



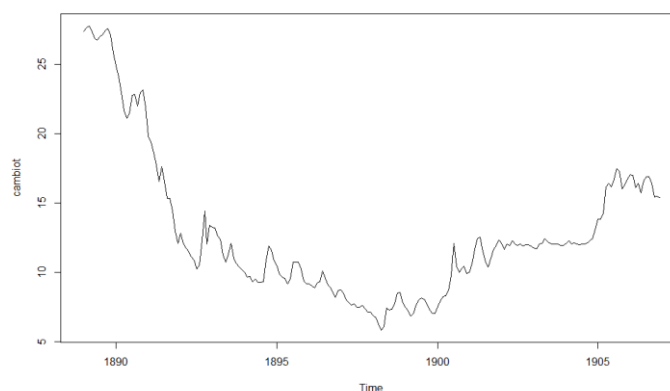
Claro, a discussão sobre a estabilidade da função consumo não termina aqui e você poderia continuar o exercício escolhendo outros países ou formas funcionais para a função consumo.

No exemplo seguinte, mostramos como realizar um teste de quebra estrutural em uma única série, em homenagem à História Econômica do Brasil.

### 8.3. Rui Barbosa importa?

Uma famosa discussão em história econômica brasileira diz respeito ao *Encilhamento* e ao seu “criador”, o famoso Rui Barbosa. No item anterior, vimos como fazer alguns testes de quebra estrutural em uma regressão. Neste item, vemos como fazer o teste em uma série. Basicamente, o processo é o mesmo só que, ao invés de se fazer uma regressão múltipla, a série que se deseja estudar é regredida contra uma constante.

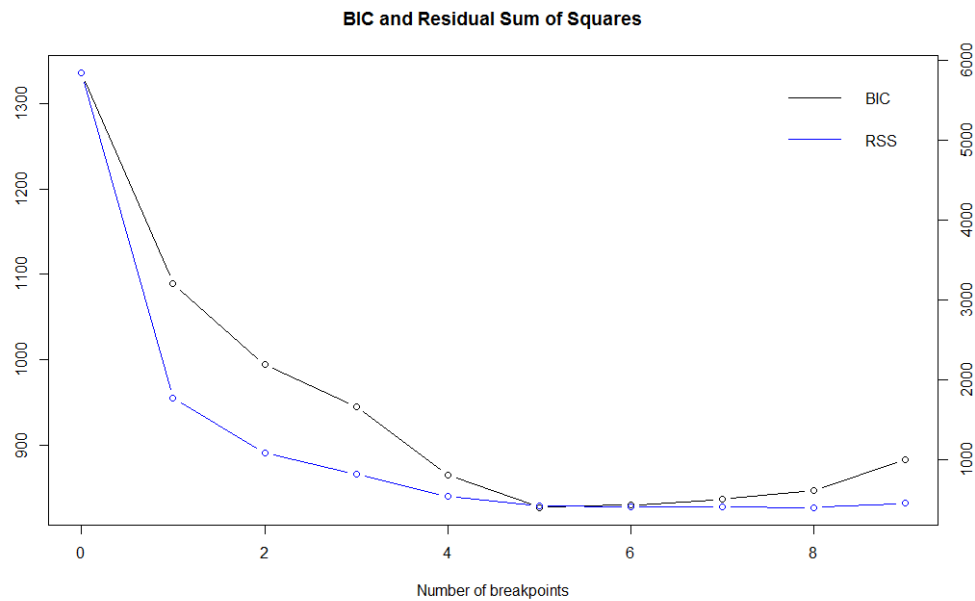
Considere, assim, os dados de Calógeras (1960)<sup>50</sup> para a taxa de câmbio (libra por mil-réis) abaixo reproduzido.



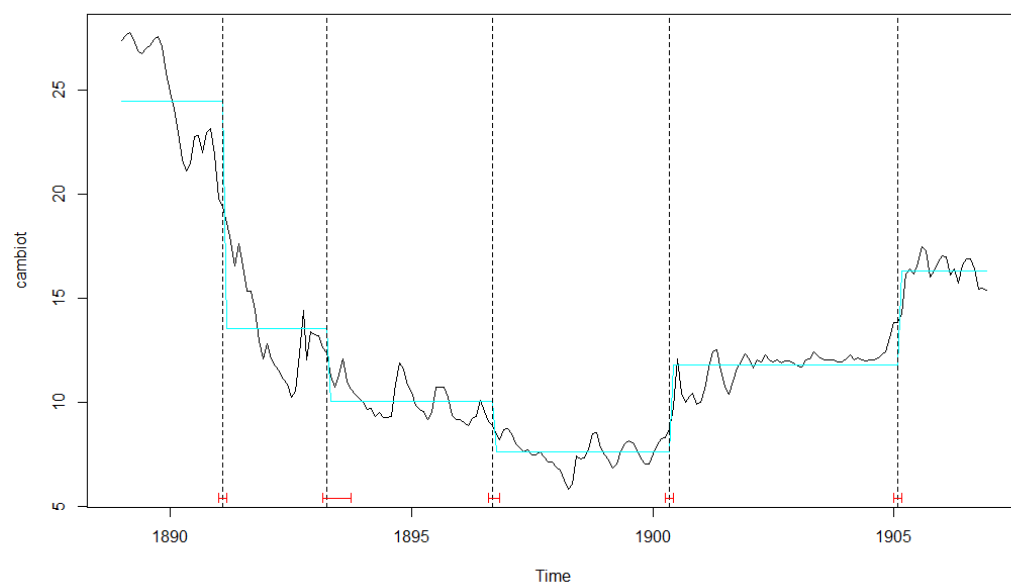
<sup>50</sup> Edição original de 1910.

Façamos apenas o teste de Bai-Perron. Usamos o pacote *strucchange* aqui. Os comandos e o gráfico pertinente são:

```
> cambiot <- ts(tanuri$pence_mil_reis,start=c(1889,1),freq=12)
> plot(cambiot)
> cambiot_dd<-breakpoints(cambiot~1, data=tanuri,h =0.1)
> plot(cambiot_dd)
```



Com cinco quebras temos:



Com os seguintes períodos:

```
> coef(cambiot_dd, breaks = 5)
(Intercept)
1889(1) - 1891(2)    24.482981
1891(3) - 1893(4)    13.518479
1893(5) - 1896(9)    10.016006
1896(10) - 1900(5)    7.589015
1900(6) - 1905(2)    11.774671
1905(3) - 1906(12)   16.299006
```

Shikida & Cortes (2013) buscaram explicar as quebras acima e você pode consultar o texto para mais detalhes sobre o Encilhamento e quebras estruturais. Quanto aos aspectos econométricos, lembre-se que o *trade-off*, neste caso, é sobre o critério de quebra relativamente o consenso dos historiadores. Não há motivo para se escolher um sobre o outro, mas uma combinação convexa de ambos é uma boa forma de se começar uma análise econométrica (no caso, cliométrica)<sup>51</sup>.

---

<sup>51</sup> A Cliometria é a aplicação da econometria a problemas de história econômica. Vários artigos cliométricos têm sido publicados no *Journal of Economic History* e, claro, no *Cliometrica*.

## 9. Econometria de Séries de Tempo em R (Parte II)

Nesta seção desta apostila – já estamos todos cansados deste minicurso, não? - tratamos dos problemas sistêmicos de variáveis econômicas em um contexto dinâmico. Na verdade, queremos fazer uma análise multivariada. Por exemplo, quem não se lembra de ter estudado os métodos de equação simultânea? Naquele contexto, fica claro que a estimação de uma função consumo não é isenta de viés (dado o aspecto simultâneo das variáveis envolvidas em um ambiente macroeconômico)<sup>52</sup>. Embora a simultaneidade já tenha sido brevemente tratada em tópico anterior, no contexto de séries de tempo pode ser mais interessante pensar em sistemas de equações autoregressivas, ou seja, em vetores autoregressivos (VAR).

Começamos então com um exemplo quase a-teórico,

### 9.1. O leite nosso de cada dia

Do Centro de Estudos Avançados em Economia Aplicada (ESALQ/USP) obtivemos algumas séries de preços para alguns derivados do leite, quais sejam, leite cru integral, leite pasteurizado, leite UHT, queijo prato, leite em pó, manteiga e queijo muçarela<sup>53</sup>. Como as mesmas possuem alguns problemas de compatibilidade (ausência irregular de dados), optamos por trabalhar com o leite pasteurizado, cujo preço médio é encontrado para Goiás, Minas Gerais, Paraná, Rio Grande do Sul e São Paulo.

De posse destes dados, faremos uma análise de cointegração. A hipótese básica é que estas séries devem ter uma relação de longo prazo. O leitor interessado pode consultar os livros de Econometria de Séries de Tempo para maiores detalhes. Inicialmente, carregamos os dados e transformamos as variáveis em séries de tempo.

```
> leite<-read.table("C:/Users/cdshi_000/Documents/Meus Documentos/Meus Documentos/minic
urso em R/leitepasteurizado.csv",
+ header=TRUE, sep=";", na.strings="NA", dec=".", strip.white=TRUE)
> summary(leite)
```

LP_GO		LP_MG		LP_PR		LP_RS		LP_SP	
Min.	:0.840	Min.	:0.890	Min.	:0.840	Min.	:0.800	Min.	:0.880
1st Qu.	:0.930	1st Qu.	:0.955	1st Qu.	:0.950	1st Qu.	:0.970	1st Qu.	:0.960
Median	:1.170	Median	:1.220	Median	:1.120	Median	:1.110	Median	:1.200
Mean	:1.139	Mean	:1.167	Mean	:1.135	Mean	:1.114	Mean	:1.173
3rd Qu.	:1.300	3rd Qu.	:1.300	3rd Qu.	:1.305	3rd Qu.	:1.290	3rd Qu.	:1.320
Max.	:1.540	Max.	:1.420	Max.	:1.480	Max.	:1.400	Max.	:1.500

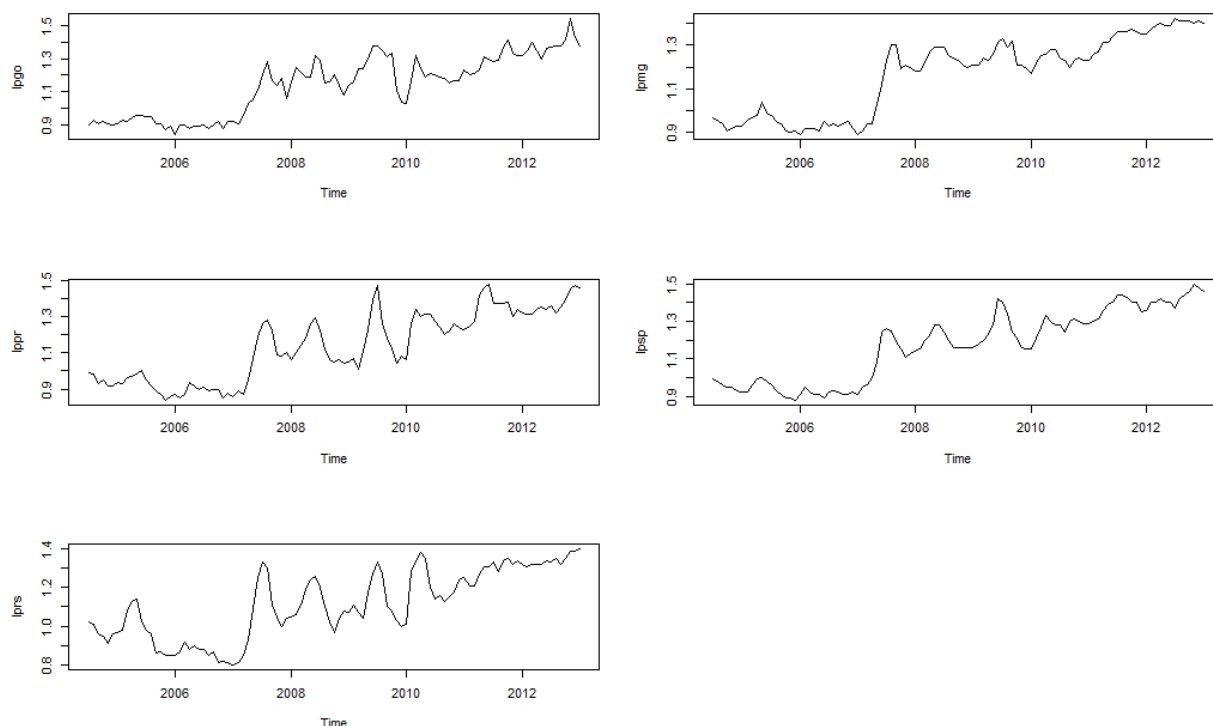
```
> leite$LP_GO<-ts(leite$LP_GO, start=c(2004,7), freq=12)
> leite$LP_MG<-ts(leite$LP_MG, start=c(2004,7), freq=12)
> leite$LP_PR<-ts(leite$LP_PR, start=c(2004,7), freq=12)
> leite$LP_RS<-ts(leite$LP_RS, start=c(2004,7), freq=12)
> leite$LP_SP<-ts(leite$LP_SP, start=c(2004, 7), freq=12)
> lpgo<-leite$LP_GO
> lpmg<-leite$LP_MG
> lppr<-leite$LP_PR
> lprs<-leite$LP_RS
> lpsp<-leite$LP_SP
```

<sup>52</sup> Johnston (1991), cap.11 é uma boa exposição sobre o tema.

<sup>53</sup> Originalmente em: <http://cepea.esalq.usp.br/leite/?page=160>. Tratam-se de preços nominais, isto é, não apliquei qualquer deflator nestes dados. O exercício interessante envolveria a deflação por índices de preços regionais, mas isto fica para algum leitor mais interessado no tema (eu mesmo não aprecio leite...).

Vale a pena visualizar as séries.

```
> par(mfrow=c(3,2))
> plot(lpgo)
> plot(lpmg)
> plot(lprr)
> plot(lpsp)
> plot(lprs)
```



Podemos notar que as séries parecem apresentar comportamento cíclico (sazonal) com tendência. Além disso, observamos uma mudança perceptível no nível médio dos preços de todas elas por volta do ano de 2007. A decomposição das séries (não mostrada) mostra um claro padrão sazonal (ou cíclico) e uma tendência que parece mudar de inclinação mais de uma vez. Uma hipótese plausível é a de que cada uma das médias de preços dos cinco estados sejam tendências estocásticas.

Sempre que nos deparamos com visualizações aparentes de tendências, a pergunta que salta aos olhos é: que tipo de tendência estamos observando na série?<sup>54</sup> Trata-se de uma tendência determinista ou estocástica? Para responder esta pergunta, utilizamos os testes de raiz unitária. No R, é possível fazer vários deles. O pacote utilizado aqui é o *urca*<sup>55</sup>. Assim, vejamos como fazer alguns testes de raiz unitária nas séries. Como primeiro exemplo, vejamos a média do preço em Minas Gerais e o tradicional teste ADF.

```
summary(ur.df(lpmg, type = c("trend"), lags = 10, selectlags = c("BIC")))
```

<sup>54</sup> Tentei explicar a diferença entre os tipos de tendências com uma interpretação macroeconômica [aqui](#).

<sup>55</sup> Mas veja também o pacote *fUnitRoots*.

No commando, o “trend” é a opção para que o teste inclua tendência e *drift*. O procedimento se inicia com dez defasagens e seleciona o número ideal conforme o critério BIC.

```
Test regression trend

Call:
lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)

Residuals:
    Min       1Q   Median       3Q      Max
-0.097067 -0.014725 -0.002792  0.015532  0.073352

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.1266067   0.0381695   3.317  0.00133 **
z.lag.1      -0.1451613   0.0429794  -3.377  0.00110 **
tt           0.0008425   0.0002661   3.166  0.00213 **
z.diff.lag1  0.2368649   0.0994480   2.382  0.01941 *
z.diff.lag2  0.2330529   0.1020690   2.283  0.02485 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03039 on 87 degrees of freedom
Multiple R-squared:  0.1767,    Adjusted R-squared:  0.1389
F-statistic: 4.669 on 4 and 87 DF,  p-value: 0.00184

value of test-statistic is: -3.3775 4.0215 5.7573

Critical values for test statistics:
      1pct  5pct 10pct
tau3  -3.99 -3.43 -3.13
phi2   6.22  4.75  4.07
phi3   8.43  6.49  5.47
```

A saída é bem completa. Repare nas informações abaixo das estatísticas da regressão. Temos três estatísticas, tau3 (-3.3775), phi2 (4.0215) e phi3 (5.7573) que podem ser comparados com os valores críticos da tabela para 1, 5 ou 10%. Cada estatística diz respeito a uma modalidade do teste ADF no que diz respeito à inclusão de termos deterministas. Em resumo, o que os resultados acima nos dizem são equivalentes à tabela abaixo com a observação de que tau3 equivale a  $\tau_\tau$ .

Percebe-se que, em qualquer uma das três especificações possíveis da equação estimada (correspondente ao primeiro modelo da tabela abaixo), não rejeitamos a existência de raiz unitária a 1 ou a 5%. É bom lembrar que, conforme Gujarati & Porter (2011), a equação de teste pode ser desconsiderada se o coeficiente da variável em nível defasada for positivo, pois isto implicaria em uma tendência explosiva. Em nosso caso, o coeficiente é negativo (-0.1451), o que não nos causa problemas.

Pode-se proceder ao restante dos testes. O comando seria, para cada linha da tabela anterior: `ur.df(lpmg, type = c("drift"), lags = 10, selectlags = c("BIC"))` geraria o modelo com *drift* e, finalmente, `ur.df(lpmg, type = c("none"), lags = 10, selectlags = c("BIC"))` geraria a última equação de teste.

#### Sumário dos testes de Dickey-Fuller

Modelo	Hipótese Nula	Estatística de Teste
$\Delta y_t = \alpha + \beta T_t + (\rho - 1)y_{t-1} + \sum \delta_k \Delta y_{t-k} + \varepsilon_t$	$(\rho - 1) = 0$	$\tau_\tau$
	$(\rho - 1) = \beta = 0$	$\phi_3$
	$(\rho - 1) = \beta = \alpha = 0$	$\phi_2$
$\Delta y_t = \alpha + (\rho - 1)y_{t-1} + \sum \delta_k \Delta y_{t-k} + \varepsilon_t$	$(\rho - 1) = 0$	$\tau_\mu$
	$(\rho - 1) = \alpha = 0$	$\phi_1$
$\Delta y_t = (\rho - 1)y_{t-1} + \sum \delta_k \Delta y_{t-k} + \varepsilon_t$	$(\rho - 1) = 0$	$\tau$

Fonte: Enders (2004). Obs: A distribuição dos testes  $\tau$  é unicaudal.

Podemos fazer o exercício para as outras séries. Em homenagem aos patrocinadores deste minicurso, trabalhemos com o preço médio no RS. Podemos fazer outros testes de raiz unitária para esta série, além, claro, do ADF. Sem reportar os resultados, os comandos são (já com a saída do resumo dos resultados):

```
summary(ur.df(lprs, type = c("trend"), lags = 10, selectlags = c("BIC")))
```

```
summary(ur.ers(lprs, type = c("DF-GLS"), model = c("constant", "trend"), lag.max = 8))
```

```
summary(ur.kpss(lprs, type = c("tau"), lags = c("long"), use.lag = NULL))
```

```
summary(ur.pp(lprs, type = c("Z-alpha"), model = c("trend"), lags = c("long"), use.lag = NULL))
```

O leitor pode consultar a documentação do pacote para compreender as opções do comando. Partiremos do pressuposto, para fins didáticos, de que as séries envolvidas possuem apenas uma raiz unitária<sup>56</sup>.

Tendo em mente os gráficos das séries, pode-se testar se existe raiz unitária na presença de quebra estrutural. Por exemplo, o teste de Zivot-Andrews para uma quebra endógena é facilmente implementado no mesmo pacote. Usando doze defasagens e testando a possibilidade de que a quebra possa ocorrer na tendência e no intercepto, temos

<sup>56</sup> Outra opção seria usar o pacote *forecast* e o comando *ndiffs* como vimos anteriormente.



que apenas a série de preços para Minas Gerais possui uma provável quebra na 34ª observação. Os comandos e resultados estão a seguir.

```
> zars<-ur.za(lprs, model = c("both"), lag=12)
> zamg<-ur.za(lpmg, model = c("both"), lag=12)
> zapr<-ur.za(lppr, model = c("both"), lag=12)
> zasp<-ur.za(lpsp, model = c("both"), lag=12)
> zago<-ur.za(lpgo, model = c("both"), lag=12)
```

Por exemplo, veja o resultado para *lpmg*, dividido em duas partes.

```
> summary(zamg)

#####
# Zivot-Andrews Unit Root Test #
#####

Call:
lm(formula = testmat)

Residuals:
    Min       1Q   Median       3Q      Max
-0.088244 -0.011986 -0.001152  0.015407  0.047742

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.3291352   0.0727375   4.525 2.29e-05 ***
y.l1         0.6361183   0.0714716   8.900 2.86e-13 ***
trend        0.0003472   0.0009441    0.368  0.7141
y.d11        0.2258067   0.1018491    2.217  0.0297 *
y.d12        0.2423400   0.1023624    2.367  0.0206 *
y.d13        0.0956245   0.1014924    0.942  0.3492
y.d14        0.0594849   0.1019013    0.584  0.5612
y.d15       -0.0262456   0.1029951   -0.255  0.7996
y.d16        0.0402855   0.0969229    0.416  0.6789
y.d17       -0.1847940   0.0968763   -1.908  0.0604 .
y.d18        0.0686489   0.0958340    0.716  0.4761
y.d19        0.2020695   0.0950415    2.126  0.0369 *
y.d110       0.0260240   0.0980914    0.265  0.7915
y.d111       0.0511987   0.0968087    0.529  0.5985
y.d112      -0.0249194   0.0970657   -0.257  0.7981
du           0.0838216   0.0188563   4.445 3.07e-05 ***
dt           0.0007867   0.0010327    0.762  0.4486
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Com a quebra identificada ao final desta saída na 34ª posição, conforme o recorte abaixo (que tem, note, uma interseção com a figura anterior).

```

y.d11      0.2258067  0.1018491  2.217  0.0297 *
y.d12      0.2423400  0.1023624  2.367  0.0206 *
y.d13      0.0956245  0.1014924  0.942  0.3492
y.d14      0.0594849  0.1019013  0.584  0.5612
y.d15     -0.0262456  0.1029951 -0.255  0.7996
y.d16      0.0402855  0.0969229  0.416  0.6789
y.d17     -0.1847940  0.0968763 -1.908  0.0604 .
y.d18      0.0686489  0.0958340  0.716  0.4761
y.d19      0.2020695  0.0950415  2.126  0.0369 *
y.d110     0.0260240  0.0980914  0.265  0.7915
y.d111     0.0511987  0.0968087  0.529  0.5985
y.d112    -0.0249194  0.0970657 -0.257  0.7981
du          0.0838216  0.0188563  4.445  3.07e-05 ***
dt          0.0007867  0.0010327  0.762  0.4486
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

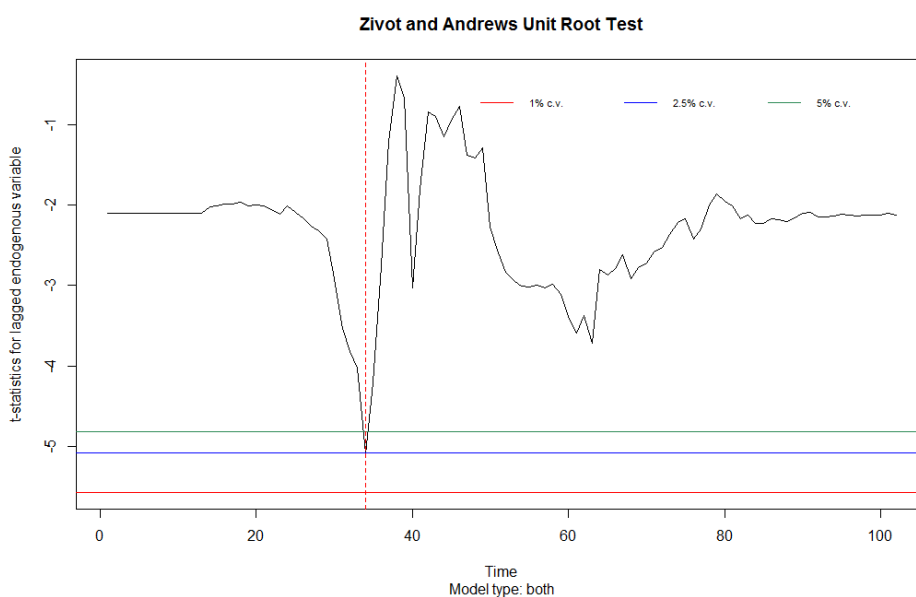
Residual standard error: 0.02576 on 73 degrees of freedom
(13 observations deleted due to missingness)
Multiple R-squared:  0.9805,    Adjusted R-squared:  0.9763
F-statistic: 229.9 on 16 and 73 DF,  p-value: < 2.2e-16

Teststatistic: -5.0913
critical values: 0.01= -5.57 0.05= -5.08 0.1= -4.82

Potential break point at position: 34

```

Note que a mesma aparece significativa a 5%. Podemos visualizar os gráficos dos testes. Como exemplo, apresento apenas o relativo a Minas Gerais.



Inicialmente, geramos a matriz de dados em logaritmos.

```
logsleite=ts.intersect(log(lpmg), log(lprs), log(lpsp), log(lppr), log(lpgo))
```

Por meio do pacote *vars*, inicialmente, selecionamos o número de defasagens do VAR para a base logsleite, iniciando com um máximo de doze defasagens.

```

Loading required package: lmtest
> VARselect(logsleite, lag.max=12, type="both")
$selection
AIC(n)   HQ(n)   SC(n) FPE(n)
   12      2     1     2

$criteria
          1          2          3          4          5
AIC(n) -3.455609e+01 -3.503275e+01 -3.464328e+01 -3.472382e+01 -3.480466e+01
HQ(n)  -3.416648e+01 -3.436485e+01 -3.369709e+01 -3.349935e+01 -3.330190e+01
SC(n)  -3.359038e+01 -3.337724e+01 -3.229797e+01 -3.168872e+01 -3.107976e+01
FPE(n)  9.843322e-16  6.149280e-16  9.209950e-16  8.726472e-16  8.403724e-16
          6          7          8          9         10
AIC(n) -3.479610e+01 -3.475420e+01 -3.484630e+01 -3.515176e+01 -3.541747e+01
HQ(n)  -3.301504e+01 -3.269485e+01 -3.250866e+01 -3.253583e+01 -3.252326e+01
SC(n)  -3.038140e+01 -2.964970e+01 -2.905201e+01 -2.866767e+01 -2.824359e+01
FPE(n)  9.043213e-16  1.034457e-15  1.072228e-15  9.394186e-16  9.079473e-16
          11         12
AIC(n) -3.548394e+01 -3.603781e+01
HQ(n)  -3.231144e+01 -3.258701e+01
SC(n)  -2.762026e+01 -2.748433e+01
FPE(n)  1.156454e-15  1.003496e-15

```

Diante da diversidade de resultados, o exercício prosseguirá com duas defasagens. A saída parcial do teste do máximo autovalor de Johansen encontra-se a seguir (com duas defasagens e *dummies* sazonais).

```

> leite.vecm_eigen <- ca.jo(logsleite, ecdet=c("trend"),type="eigen", k=2, spec="longru
n",season=12)
> summary(leite.vecm_eigen)

#####
# Johansen-Procedure #
#####

Test type: maximal eigenvalue statistic (lambda max) , with linear trend in cointegrati
on

Eigenvalues (lambda):
[1] 4.101880e-01 2.915460e-01 2.081621e-01 1.493599e-01 2.552224e-02 1.120161e-17

values of teststatistic and critical values of test:

      test 10pct  5pct  1pct
r <= 4 |  2.61 10.49 12.25 16.26
r <= 3 | 16.34 16.85 18.96 23.65
r <= 2 | 23.57 23.11 25.54 30.34
r <= 1 | 34.81 29.12 31.46 36.65
r = 0  | 53.32 34.75 37.52 42.36

Eigenvectors, normalised to first column:
(These are the cointegration relations)

log.lnma..12 log.lnrs..12 log.lnsp..12 log.lnpr..12 log.lnqa..12

```

Usando 1% como critério, analisamos sequencialmente o teste começando da hipótese de que não há cointegração e decidimos por um vetor. A 5%, teríamos a evidência de que seriam dois vetores.

O teste do traço também pode ser estimado e encontra-se a seguir e, a 1%, a evidência é a de que existiriam, no máximo, dois vetores de cointegração<sup>57</sup>.

```
> leite.vecm_tr <- ca.jo(logsleite, ecdet=c("trend"),type="trace", k=2, spec="longrun",
season=12)
> summary(leite.vecm_tr)

#####
# Johansen-Procedure #
#####

Test type: trace statistic , with linear trend in cointegration

Eigenvalues (lambda):
[1] 4.101880e-01 2.915460e-01 2.081621e-01 1.493599e-01 2.552224e-02 1.120161e-17

values of teststatistic and critical values of test:

      test 10pct  5pct  1pct
r <= 4 |   2.61 10.49 12.25 16.26
r <= 3 |  18.95 22.76 25.32 30.45
r <= 2 |  42.52 39.06 42.44 48.45
r <= 1 |  77.33 59.14 62.99 70.05
r = 0  | 130.66 83.20 87.31 96.58

Eigenvectors, normalised to first column:
(These are the cointegration relations)

      log.lpmg..l2 log.lprs..l2 log.lpsp..l2 log.lppr..l2 log.lpgo..l2
log.lpmg..l2  1.000000000  1.000000000  1.000000000  1.000000000  1.000000000
log.lprs..l2 -0.254407448 -0.515722228 -0.4374463369  1.089193370 -0.470656867
```

O passo seguinte é estimar o VECM com duas relações de cointegração. Optei por seguir com as *dummies sazonais* (você as verá como “sd1, sd2, ..., sd11” na saída a seguir).

<sup>57</sup> No teste do maior autovalor (máximo autovalor), a hipótese nula é a de que existem “r” relações de cointegração contra a alternativa de que existem “r + 1”. Já no teste do traço, a hipótese nula é a de que existem “no máximo r relações de cointegração” e a alternativa é a de que existem “no máximo r + 1 relações de cointegração”.

```

> leite_duas<-cajorls(leite.vecm_eigen, r=2)
> leite_duas
$rlm

call:
lm(formula = substitute(form1), data = data.mat)

Coefficients:
          log.lpmg..d  log.lprs..d  log.lpsp..d  log.lppr..d  log.lpgo..d
ect1          -0.0295937    0.2834592    0.1668958    0.2855710    0.2266958
ect2           0.0152138   -0.1286230   -0.0892883   -0.1148808   -0.0687937
constant       0.0059134   -0.0299966   -0.0163176   -0.0274555   -0.0179235
sd1          -0.0014674    0.0045055   -0.0144352   -0.0303163   -0.0078403
sd2          -0.0106708    0.0046376   -0.0180788   -0.0534789   -0.0181460
sd3          -0.0063818   -0.0140101   -0.0115046   -0.0376185   -0.0199795
sd4          -0.0134473    0.0178936   -0.0087681   -0.0224432   -0.0002042
sd5          -0.0012276    0.0217859   -0.0117914   -0.0337024   -0.0368365
sd6          -0.0079594    0.0336057   -0.0221464   -0.0121471   -0.0543989
sd7          -0.0146533    0.0066111   -0.0190166   -0.0360627   -0.0305116
sd8           0.0131122    0.0433585    0.0119991   -0.0079002    0.0056550
sd9           0.0015761    0.0176888   -0.0137884   -0.0397468   -0.0052050
sd10          -0.0041502    0.0283508    0.0007195    0.0099472   -0.0302194
sd11           0.0081397    0.0365967   -0.0050336    0.0138921   -0.0229861
log.lpmg..d11  -0.1392349    0.3670917    0.3738500    0.4965890    0.2542332
log.lprs..d11  0.2073981   -0.0509907    0.0840258    0.1328169    0.1791627
log.lpsp..d11  0.3567109    0.5341485   -0.1488094    0.5176070    0.6192992

```

Na parte inferior desta tela, temos os vetores de cointegração estimados.

```

$beta
          ect1          ect2
log.lpmg..12  1.000000000  0.000000000
log.lprs..12  0.000000000  1.000000000
log.lpsp..12 10.969770408 24.28138924
log.lppr..12 -4.349075187 -8.66891648
log.lpgo..12 -6.555309647 -13.21721565
trend.12     -0.008118012 -0.02316313

```

Observando o vetor de cointegração, temos que a média do preço do leite pasteurizado em Minas Gerais, possui uma forte relação negativa com a média de São Paulo. Leitura similar pode ser feita para o segundo vetor. O comando `summary(leite_duas$rlm)` nos dá as saídas das equações individuais do VECM com a significância dos parâmetros estimados. Segundo estes resultados, o preço de Minas Gerais seria exógeno aos dois vetores de cointegração. Análise similar pode ser feita para as demais equações.

Testes de diagnóstico podem ser feitos sem grandes dificuldades. Para os testes ARCH e de normalidade (multivariado) <sup>58</sup>, deve-se, em caso de uso do pacote *vars*, converter o VECM em um VAR inicialmente. O comando é `vec2var` e os resultados seguem abaixo.

<sup>58</sup> Conforme destacado por Pfaff (2008), no caso do teste Jarque-Bera multivariado, o resultado é dependente da ordenação das variáveis.

```

> leite_testes<-vec2var(leite.vecm_eigen,r=2)
> arch.test(leite_testes)

      ARCH (multivariate)

data:  Residuals of VAR object leite_testes
Chi-squared = 1175.6, df = 1125, p-value = 0.1433

> normality.test(leite_testes)
$JB

      JB-Test (multivariate)

data:  Residuals of VAR object leite_testes
Chi-squared = 26.629, df = 10, p-value = 0.00298

$Skewness

      Skewness only (multivariate)

data:  Residuals of VAR object leite_testes
Chi-squared = 5.6351, df = 5, p-value = 0.3434

$Kurtosis

      Kurtosis only (multivariate)

data:  Residuals of VAR object leite_testes
Chi-squared = 20.994, df = 5, p-value = 0.0008121

```

A análise dos resultados fica por sua conta, ok?

Outro teste importante para a análise de cointegração é a de estabilidade dos parâmetros. No caso do pacote *vars*, testes de mudança estrutural só estão disponíveis para vetores autoregressivos sem cointegração (VAR). Para o caso de VECM, o teste disponível é o de Lütkepohl, Saikkonen & Trenkler (2004) *apud* Pfaff (2008), que estima uma quebra (endógena) na média do processo analisado e, em seguida, estima o VECM ajustando os dados pela quebra. A seguir, apresentamos o resultado deste procedimento. O comando é *cajolst()*.

```

> leite_duas_st<-cajolst(logsleite,k=2)
> summary(leite_duas_st)

#####
# Johansen-Procedure #
#####

Test type: trace statistic , with linear trend in shift correction

Eigenvalues (lambda):
[1] 0.49705087 0.39706978 0.22721158 0.15556124 0.05867109

values of teststatistic and critical values of test:

      test 10pct  5pct  1pct
r <= 4 |   5.76  5.42  6.79 10.04
r <= 3 |  20.36 13.78 15.83 19.85
r <= 2 |  41.04 25.93 28.45 33.76
r <= 1 |  74.81 42.08 45.20 51.60
r = 0  | 115.57 61.92 65.66 73.12

Eigenvectors, normalised to first column:
(These are the cointegration relations)

```

A saída parcial nos mostra um resultado distinto do anterior. Agora, a 1%, teríamos evidência de quatro relações de cointegração.

O teste de exogeneidade fraca pode ser realizado por meio do pacote *urca*. Por exemplo, testamos a seguir a exogeneidade fraca dos preços do RS. Inicialmente fornecemos a matriz das velocidades de ajustamento:

```
A1 <- matrix( c (1,0,0,0,0, 0,0,1,0,0, 0,0,0,1,0 ,0,0,0,0,1),nrow=5, ncol=4)
```

Em seguida fazemos o teste:

```
H41 <-summary (alrtest( z = leite.vecm_tr, A = A1 , r = 2) )
```

Os resultados, por sua vez, são:



```
> A1 <- matrix( c (1,0,0,0,0, 0,0,1,0,0, 0,0,0,1,0 ,0,0,0,0,1),
+ nrow=5, ncol=4)
> H41 <-summary (alrtest( z = leite.vecm_tr, A = A1 , r = 2) )
> H41
```

```
#####
# Johansen-Procedure #
#####
```

Estimation and testing under linear restrictions on beta

The VECM has been estimated subject to:  
beta=H\*phi and/or alpha=A\*psi

	[,1]	[,2]	[,3]	[,4]
[1,]	1	0	0	0
[2,]	0	0	0	0
[3,]	0	1	0	0
[4,]	0	0	1	0
[5,]	0	0	0	1

Eigenvalues of restricted VAR (lambda):  
[1] 0.3991 0.2439 0.2082 0.0255 0.0000 0.0000

The value of the likelihood ratio test statistic:  
8.44 distributed as chi square with 2 df.  
The p-value of the test statistic is: 0.01

Eigenvectors, normalised to first column  
of the restricted VAR:

	[,1]	[,2]
RK.log.lpmg..12	1.0000	1.0000
RK.log.lprs..12	0.1175	0.0631
RK.log.lpsp..12	6.0473	-1.1496
RK.log.lppr..12	-2.7470	-0.2923
RK.log.lpgo..12	-3.9440	-0.0796
RK.trend.12	-0.0038	0.0028

weights w of the restricted VAR:

	[,1]	[,2]
[1,]	-0.0024	-0.0923
[2,]	0.0000	0.0000
[3,]	-0.0334	0.1844
[4,]	0.0737	0.1751
[5,]	0.1387	0.0877

O leitor pode verificar que a hipótese nula de exogeneidade fraca em RS não é rejeitada a 0.01 (1%). Em outras palavras, RS deve ser endógeno<sup>59</sup>.

Os pacotes *vars* e *urca*, conjuntamente, possibilitam-nos ainda a estimação de um VECM estrutural com as correspondentes funções resposta ao impulso e decomposições de

<sup>59</sup> Testes sobre o vetor de cointegração podem ser feitos por meio do comando *blrtest*. Outros testes possíveis são com restrições conjuntas nas matrizes de ajustamento e dos coeficientes de cointegração (*ablrtest*) e com conhecimento parcial do vetor de cointegração (*bh5lrtest*).

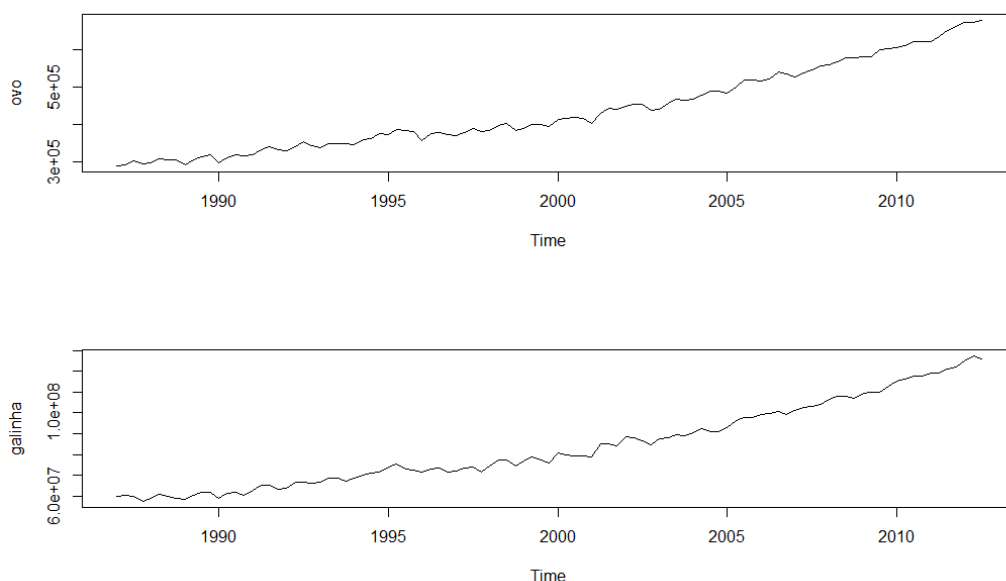


variância. Não prosseguiremos com este exemplo porque seria necessário conhecer melhor o mercado de leite para que tivéssemos análises mais elaboradas. O leitor é convidado a aperfeiçoar esta análise<sup>60</sup>.

## 9.2. Quem veio primeiro: o ovo ou a galinha?

Uma pergunta filosófica interessante – e divertida – diz respeito a quem veio primeiro: o ovo ou a galinha. Esta pergunta foi explorada, pela primeira vez, por meio do teste de causalidade de Granger, por Thurman & Fisher (1988). Recentemente, Shikida, Araujo Jr & Figueiredo (2011) replicaram a pergunta para o Brasil. Enquanto nos EUA, ovos precedem galinhas, para o Brasil, haveria bicausalidade.

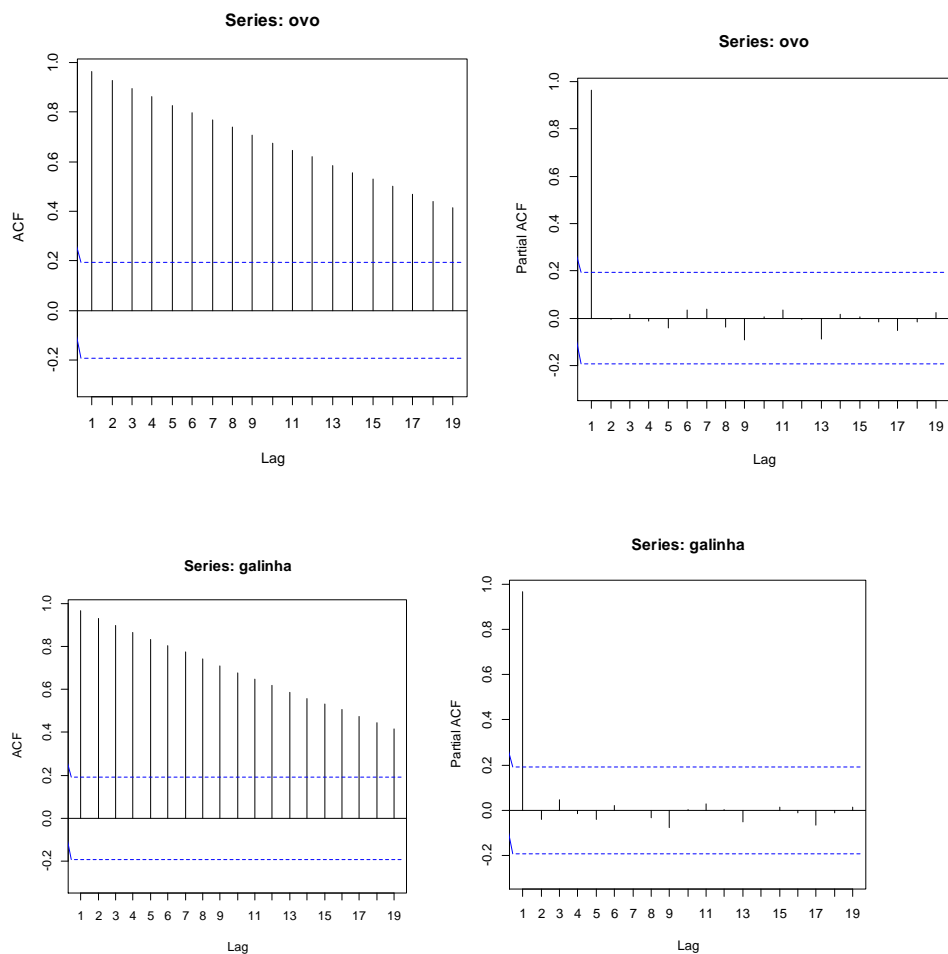
O tom divertido de ambos os artigos sugere que o uso do R pode transformar o último tópico destes encontros em algo mais interessante. Nosso objetivo é utilizar o pacote *var* para testar a causalidade entre ovos e galinhas. Inicialmente, vejamos as séries. Ovos estão em mil dúzias e o número de galinhas poedeiras são medidas por cabeças. Os dados são do IBGE<sup>61</sup>.



Fazendo uso dos comandos *Acf* e *Pacf* (pacote *forecast*) obtemos os correlogramas das mesmas.

<sup>60</sup> Você pode usar também o pacote *tsDyn* para análises similares a estas. Outro pacote interessante, mas apenas para o uso com duas variáveis, ou seja, com o mecanismo de correção de erros de Engle-Granger é o *egcm*.

<sup>61</sup> <http://www.sidra.ibge.gov.br/bda/pecua/default.asp?z=t&o=24&i=P>.



Antes de prosseguirmos, transformamos ambas as séries em logaritmos. Poderíamos fazer alguns testes de raiz unitária, já vistos anteriormente, mas vamos recorrer, novamente, aos comandos *ndiffs* e *nsdiffs* do mesmo pacote, afim de resumir nosso trabalho em termos do número de diferenciações para estacionarizar as séries. Os resultados apontam para apenas uma diferenciação.

```
> ndiffs(lovo)
[1] 1
> nsdiffs(lovo,m=4)
[1] 0
> ndiffs(lgalinha)
[1] 1
> nsdiffs(lgalinha,m=4)
[1] 0
> |
```

Criamos então a matriz “granger” (o comando é: *granger=ts.intersect(lovo, lgalinha)*). Antes de testar qualquer causalidade, vejamos o número de defasagens para as séries em nível.

```

> VARselect(granger, lag.max=4, type="both")
$selection
AIC(n)  HQ(n)  SC(n)  FPE(n)
      4      4      3      4

$criteria
              1              2              3              4
AIC(n) -1.647613e+01 -1.668599e+01 -1.683932e+01 -1.689120e+01
HQ(n)  -1.639128e+01 -1.655872e+01 -1.666963e+01 -1.667908e+01
SC(n)  -1.626642e+01 -1.637143e+01 -1.641991e+01 -1.636693e+01
FPE(n)  6.991138e-08  5.668845e-08  4.864990e-08  4.622188e-08

```

Diante disso, assumimos que o VAR em diferenças deve ter três defasagens. Eis o comando:

```
granger1<-VAR(dgranger, p=3,type="both")
```

Vejamos os resultados.

```

> dgranger=ts.intersect(diff(love),diff(lgalinha))
> granger1<-VAR(dgranger, p=3,type="both")
> summary(granger1)

```

VAR Estimation Results:

```

=====
Endogenous variables: diff.love., diff.lgalinha.
Deterministic variables: both
Sample size: 99
Log Likelihood: 566.218
Roots of the characteristic polynomial:
0.8873 0.8873 0.6259 0.4738 0.4738 0.4709
Call:
VAR(y = dgranger, p = 3, type = "both")

```

Estimation results for equation diff.love.:

```

=====
diff.love. = diff.love..l1 + diff.lgalinha..l1 + diff.love..l2 + diff.lgalinha..l2 + di
ff.love..l3 + diff.lgalinha..l3 + const + trend

```

	Estimate	Std. Error	t value	Pr(> t )	
diff.love..l1	-7.559e-01	1.420e-01	-5.322	7.33e-07	***
diff.lgalinha..l1	5.568e-01	1.477e-01	3.769	0.000291	***
diff.love..l2	-5.250e-01	1.447e-01	-3.627	0.000473	***
diff.lgalinha..l2	2.927e-01	1.548e-01	1.890	0.061943	.
diff.love..l3	-2.206e-01	1.336e-01	-1.651	0.102190	
diff.lgalinha..l3	1.955e-02	1.563e-01	0.125	0.900760	
const	9.878e-03	4.022e-03	2.456	0.015952	*
trend	8.343e-05	6.691e-05	1.247	0.215596	

```

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01827 on 91 degrees of freedom
Multiple R-Squared:  0.4387,    Adjusted R-squared:  0.3955
F-statistic: 10.16 on 7 and 91 DF,  p-value: 2.445e-09

Estimation results for equation diff.lgalinha.:
=====
diff.lgalinha. = diff.lovo..l1 + diff.lgalinha..l1 + diff.lovo..l2 + diff.lgalinha..l2
+ diff.lovo..l3 + diff.lgalinha..l3 + const + trend

              Estimate Std. Error t value Pr(>|t|)
diff.lovo..l1  -2.463e-01  1.352e-01  -1.822   0.0717 .
diff.lgalinha..l1 -1.014e-01  1.406e-01  -0.721   0.4726
diff.lovo..l2   -6.716e-02  1.377e-01  -0.488   0.6270
diff.lgalinha..l2 -3.111e-01  1.474e-01  -2.111   0.0375 *
diff.lovo..l3    2.071e-01  1.272e-01   1.628   0.1069
diff.lgalinha..l3 -2.546e-01  1.488e-01  -1.711   0.0905 .
const           8.156e-03  3.828e-03   2.131   0.0358 *
trend           1.033e-04  6.367e-05   1.622   0.1083
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

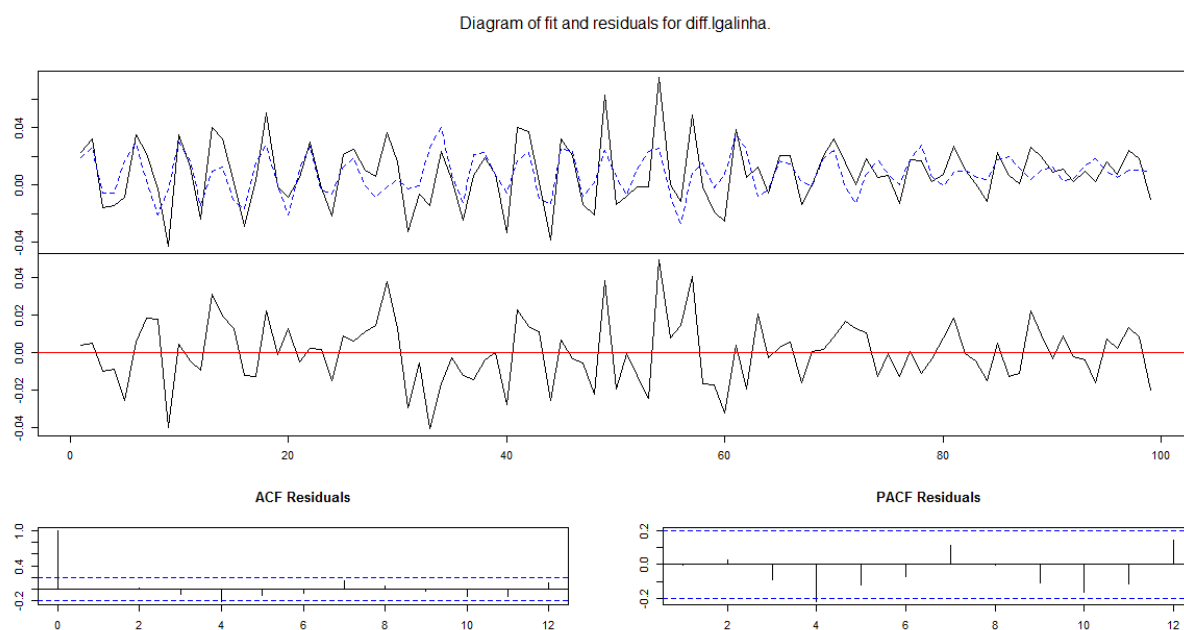
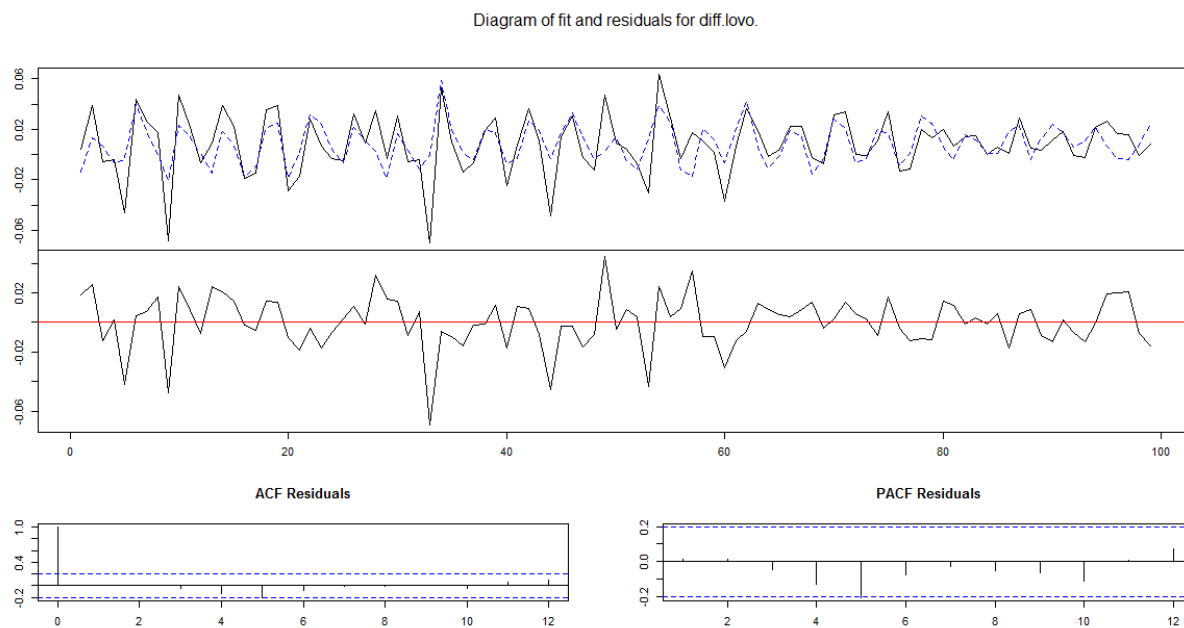
Residual standard error: 0.01739 on 91 degrees of freedom
Multiple R-Squared:  0.3939,    Adjusted R-squared:  0.3473
F-statistic:  8.45 on 7 and 91 DF,  p-value: 6.212e-08

Covariance matrix of residuals:
              diff.lovo. diff.lgalinha.
diff.lovo.    0.0003338    0.0002392
diff.lgalinha. 0.0002392    0.0003023

Correlation matrix of residuals:
              diff.lovo. diff.lgalinha.
diff.lovo.    1.000        0.753
diff.lgalinha. 0.753        1.000

```

Antes de prosseguirmos com conclusões acerca da causalidade, é necessário verificar alguns testes. Usando o comando *plot(granger1)* obtemos os resíduos.



Como já vimos os testes de diagnóstico e de estabilidade, deixarei por sua conta reproduzir o exercício.

A leitura dos resultados do VAR não parece nos dar resultados muito promissores em termos de causalidade (possivelmente não há causalidade em qualquer direção). Uma outra forma de se fazer o teste é usar o comando *grangertest* do pacote *lmtest*, que retorna, entretanto, um resultado bem menos detalhado. Para efeitos de comparação, aí vai.

```

> grangertest(diff(lover)~diff(lgalinha), order=3)
Granger causality test

Model 1: diff(lover) ~ Lags(diff(lover), 1:3) + Lags(diff(lgalinha), 1:3)
Model 2: diff(lover) ~ Lags(diff(lover), 1:3)
      Res.Df Df      F    Pr(>F)
1         92
2         95 -3  6.957 0.0002854 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> grangertest(diff(lgalinha)~diff(lover), order=3)
Granger causality test

Model 1: diff(lgalinha) ~ Lags(diff(lgalinha), 1:3) + Lags(diff(lover), 1:3)
Model 2: diff(lgalinha) ~ Lags(diff(lgalinha), 1:3)
      Res.Df Df      F    Pr(>F)
1         92
2         95 -3  3.6654 0.01515 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
~ |

```

A hipótese nula, no primeiro modelo, é a de que galinhas não Granger-causam ovos, o que é rejeitado a 1%. Para a hipótese de que ovos não Granger-causam galinhas, a 1%, não é rejeitada. Ou seja, a 1%, galinhas “causam” ovos, e não o contrário. Entretanto, a 10%, teríamos bicausalidade.

## Anexo 6 – Ovos e Galinhas e a causalidade de Granger no pacote vars

No pacote *vars*, o teste pode ser feito pelo comando *causality*. No caso deste exemplo, como geramos a variável em diferença de forma indireta, temos que ter cuidado ao indicá-la. Eis os comandos e as saídas. Repare na grafia da variável cuja causalidade se pretende testar.

```

> causality(granger1, cause="diff.lovo.")
$Granger

      Granger causality H0: diff.lovo. do not Granger-cause diff.lgalinha.

data:  VAR object granger1
F-Test = 3.2637, df1 = 3, df2 = 182, p-value = 0.02267

$Instant

      H0: No instantaneous causality between: diff.lovo. and diff.lgalinha.

data:  VAR object granger1
Chi-squared = 35.8217, df = 1, p-value = 2.162e-09

> causality(granger1, cause="diff.lgalinha.")
$Granger

      Granger causality H0: diff.lgalinha. do not Granger-cause diff.lovo.

data:  VAR object granger1
F-Test = 6.0766, df1 = 3, df2 = 182, p-value = 0.000579

$Instant

      H0: No instantaneous causality between: diff.lgalinha. and diff.lovo.

data:  VAR object granger1
Chi-squared = 35.8217, df = 1, p-value = 2.162e-09

```

Repare em “diff.lgalinha.” e “diff.lovo.” que são as leituras de “diff(lgalinha)” e “diff(lovo)”, respectivamente. Repare também que, neste comando, pode-se testar a causalidade instantânea entre variáveis. Aparentemente, não se vê causalidade de qualquer espécie nos resultados. Poderíamos, no caso do teste de causalidade de Granger, usar um estimador robusto da matriz de variância-covariância. Eis os comandos.

```
causality(granger1, cause="diff.lovo.", vcov.=vcovHC(granger1))
```

```
causality(granger1, cause="diff.lgalinha.", vcov.=vcovHC(granger1))
```

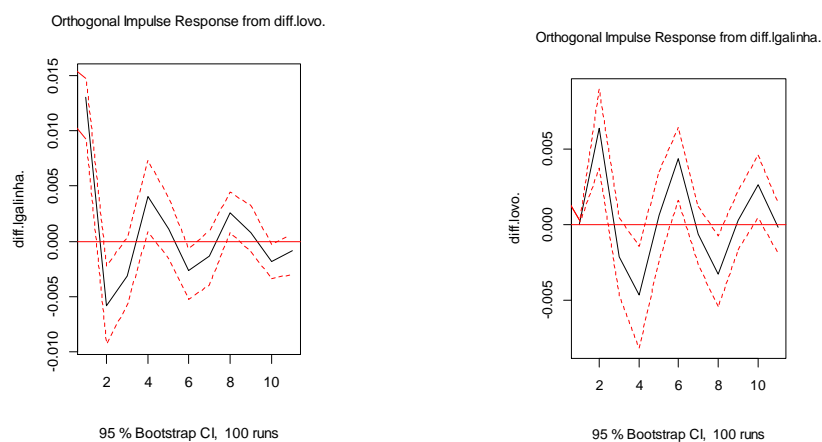
Especificamente, no caso deste exemplo, as conclusões não se alteraram e, portanto, os resultados não são reportados.

Podemos também obter as funções de resposta ao impulso e a decomposição de variância do erro de previsão do VAR.

```

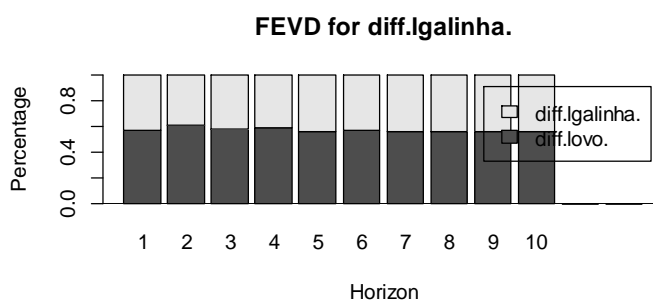
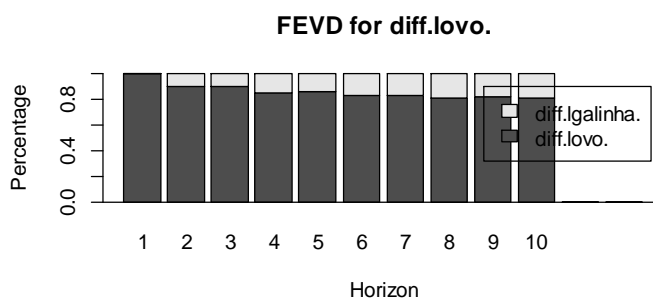
> irf.ovo <-irf(granger1,impulse="diff.lovo.", response="diff.lgalinha.",n.ahead=10,
+ ortho=TRUE)
> plot(irf.ovo)
> irf.galinha <-irf(granger1,impulse="diff.lgalinha.", response="diff.lovo.",n.ahead=10,
+ ortho=TRUE)
> plot(irf.galinha)

```



Para a decomposição:

```
> fevd.granger1<-fevd(granger1, n.ahead=10)
> plot(fevd.granger1, addbars=2)
```



O leitor interessado pode obter mais informações sobre VAR e VECM estruturais na bibliografia indicada. Por enquanto, nosso maior interesse ainda é o de saber se é o ovo ou a galinha o culpado disto tudo. Esperamos que estas notas tenham ajudado na solução desta e de outras questões.



## 10. Salário, Etnia e Educação: qual é a relação? – Uma Introdução à Econometria dos Dados em Painel no R (Prof. Rodrigo N. Fernandez)

Salários e discriminação. Eis aí uma mistura explosiva. Este exemplo se baseia em Vella e Verbeek (1998), com uma amostra de pessoas do sexo masculino ( $n = 545$ ) que trabalharam em cada ano de 1980-87. Considere a seguinte equação para o salário:

$$\log(wage_{it}) = \theta_{it} + \beta_1 educ_{it} + \beta_2 black_{it} + \beta_3 hispan_{it} + \beta_4 exper_{it} + \beta_5 exper_{it}^2 + \beta_6 married_{it} + \beta_7 union_{it} + c_{it} + u_{it}$$

Podemos observar que na equação do salário temos variáveis como educação, etnia, experiência e dados sobre o estado civil destes indivíduos. Primeiramente carregaremos as bibliotecas necessárias:

```
library(foreign)
```

```
library(plm)
```

```
library(lmtest)
```

Em seguida especificaremos o nome do arquivo e carregaremos os dados:

```
arquivo="wagepan.dta"
```

```
diretorio="C:/Users/Rodrigo/Desktop/"
```

```
setwd(diretorio)
```

```
dados = read.dta(file=arquivo,convert,factors=F)
```

Para utilizarmos a biblioteca *plm* referente a dados em painel no R precisaremos transformar nossos dados em um *data frame*:

```
n=nrow(dados)
```

```
dados.plm=pdata.frame(dados,n/8)
```

Primeiramente obtemos o número de linhas da base de dados com o comando `nrow` e em seguida construímos nosso *dataframe* dividindo o total de linhas da nossa base de dados pelo número de cortes no tempo, ou seja 8 anos de 1980 a 1987. Também utilizaremos o comando `attach` para usar apenas o nome das variáveis de acordo com nossa base de dados:

```
attach(dados.plm)
```

O primeiro passo é construirmos nossa equação de regressão e a estimarmos por POLS (mínimos quadrados agrupados):

```
uniao = dados.plm$union

reg = lwage~educ+black+hisp+exper+expersq+married+uniao
```

Os comandos acima constroem a equação de regressão, veja que atribuímos o nome união para a variável *union*, pois ela é uma função do R. Veja um exemplo ilustrativo:

```
x= 1:10

y=10:20

union(x,y)
```

Neste caso fazemos a união dos conjuntos x e y. Agora faremos nossa regressão por POLS:

```
reg.pols=plm(reg,data=dados.plm,model="pooling")

summary(reg.pols)
```

Veja que neste caso teremos um painel balanceado, isto é, com todas as observações (sem lacunas) :

```
> reg.pols=plm(reg,data=dados.plm,model="pooling")
> summary(reg.pols)
Oneway (individual) effect Pooling Model

Call:
plm(formula = reg, data = dados.plm, model = "pooling")

Balanced Panel: n=545, T=8, N=4360

Residuals :
    Min. 1st Qu.  Median 3rd Qu.    Max.
-5.2700 -0.2490   0.0332   0.2960   2.5600

Coefficients :
              Estimate Std. Error t-value Pr(>|t|)
(Intercept) -0.03470569  0.06456900  -0.5375   0.5910
educ         0.09938779  0.00467760  21.2476 < 2.2e-16 ***
black       -0.14384171  0.02355950  -6.1055 1.114e-09 ***
hisp        0.01569798  0.02081119   0.7543   0.4507
exper       0.08917907  0.01011105   8.8200 < 2.2e-16 ***
expersq     -0.00284866  0.00070736  -4.0272 5.742e-05 ***
married     0.10766558  0.01569647   6.8592 7.897e-12 ***
uniao       0.18007257  0.01712053  10.5179 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:    1236.5
Residual Sum of Squares: 1005.8
R-Squared              :  0.18659
Adj. R-Squared       :  0.18624
F-statistic: 142.613 on 7 and 4352 DF, p-value: < 2.22e-16
> |
```

O comando *summary* nos mostra que temos 545 indivíduos ao longo de 8 anos o que nos dá um total de 4360 observações. Além disso, percebemos que o fato de um indivíduo ser hispânico, em média não afeta seu salário de acordo com este modelo. Adicionalmente é importante destacar as principais diferenças entre os modelos de dados em painel:

- Modelos “*Pooled*”: A estimação é feita assumindo que os  $\beta$  são comuns para todos os indivíduos.
- Modelos de Primeira Diferença: (PD): A estimação é feita eliminando os termos que são constantes ao longo do tempo.
- Modelos com efeitos fixos (EF): A estimação é feita assumindo que a heterogeneidade dos indivíduos. Se capta na parte constante, que é diferente de indivíduo para indivíduo, captando as diferenças invariantes no tempo.
- Modelos com efeitos aleatórios (EA): A estimação é feita introduzindo a heterogeneidade dos indivíduos no termo de erro. Esse modelo considera a constante não como um parâmetro fixo, mas como um parâmetro aleatório não observável.

Podemos também fazer a mesma regressão usando a função *lm*:

```
reg.lm=plm(reg,data=dados.plm)
summary(reg.lm)
```

Repare que as diferenças nas estimações são sutis:

```

> reg.lm=lm(reg,data=dados.plm)
> summary(reg.lm)

Call:
lm(formula = reg, data = dados.plm)

Residuals:
    Min       1Q   Median       3Q      Max
-5.2689 -0.2487  0.0332  0.2962  2.5608

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.0347057  0.0645690  -0.537   0.591
educ         0.0993878  0.0046776  21.248 < 2e-16 ***
black       -0.1438417  0.0235595  -6.105 1.11e-09 ***
hisp         0.0156980  0.0208112   0.754   0.451
exper        0.0891791  0.0101110   8.820 < 2e-16 ***
expersq     -0.0028487  0.0007074  -4.027 5.74e-05 ***
married      0.1076656  0.0156965   6.859 7.90e-12 ***
uniao        0.1800726  0.0171205  10.518 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4807 on 4352 degrees of freedom
Multiple R-squared:  0.1866,    Adjusted R-squared:  0.1853
F-statistic: 142.6 on 7 and 4352 DF,  p-value: < 2.2e-16

```

Em seguida fazemos um teste de correlação serial:

*pwtest(reg.pols)*

```

> pwtest(reg.pols)

Wooldridge's test for unobserved individual effects

data: formula
z = 10.6497, p-value < 2.2e-16
alternative hypothesis: unobserved effect

> |

```

Após realizarmos o teste de do efeito individual não observado (correlação serial), detectamos que a presença do efeito individual não observado, isto é,  $c_{it}$  na presente correlação com  $u_{it}$  e desta forma fizemos um teste t de coeficientes usando os erros padrões robustos para este modelo, como segue:

*coeftest(reg.pols,vcov=function(reg.pols)*  
*vcovHC(reg.pols,method="arellano",type="HC1",cluster="group"))*

```

t test of coefficients:

              Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.03470569 0.12000704 -0.2892 0.772444
educ         0.09938779 0.00920092 10.8019 < 2.2e-16 ***
black        -0.14384171 0.05007130 -2.8727 0.004089 **
hisp         0.01569798 0.03916655 0.4008 0.688587
exper        0.08917907 0.01243303 7.1728 8.597e-13 ***
expersq      -0.00284866 0.00086989 -3.2747 0.001066 **
married      0.10766558 0.02606010 4.1314 3.673e-05 ***
uniao        0.18007257 0.02755815 6.5343 7.127e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Usando erros padrões robustos as estatísticas t e os p-valores diminuem, black e expersq são significativas a 1%. O próximo passo é realizarmos a estimação com efeitos fixos e aleatórios. Primeiramente faremos com efeitos aleatórios:

```

reg.re = plm(reg,data=dados.plm,model="random")
summary(reg.re)

```

Vejamos nossos resultados na imagem abaixo:

```

Oneway (individual) effect Random Effect Model
(Swamy-Arora's transformation)

Call:
plm(formula = reg, data = dados.plm, model = "random")

Balanced Panel: n=545, T=8, N=4360

Effects:
              var std.dev share
idiosyncratic 0.1234 0.3513 0.539
individual    0.1053 0.3246 0.461
theta: 0.6426

Residuals :
      Min. 1st Qu.  Median 3rd Qu.    Max.
-4.5800 -0.1450  0.0235  0.1860  1.5400

Coefficients :
              Estimate Std. Error t-value Pr(>|t|)
(Intercept) -0.10746430 0.11070573 -0.9707 0.3317415
educ         0.10122462 0.00891329 11.3566 < 2.2e-16 ***
black        -0.14413068 0.04761483 -3.0270 0.0024843 **
hisp         0.02015107 0.04260112 0.4730 0.6362245
exper        0.11211950 0.00826087 13.5724 < 2.2e-16 ***
expersq      -0.00406885 0.00059183 -6.8751 7.075e-12 ***
married      0.06279510 0.01677285 3.7439 0.0001836 ***
uniao        0.10737886 0.01783001 6.0224 1.861e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

As estimativas de RE são bastante próximas quando estimamos POLS com erros padrões robustos, ainda considerando que o efeito aleatório apresenta erros padrões menores. Agora verificaremos o efeito fixo:

```
reg.fe = plm(reg,data=dados.plm,model="within")
summary(reg.fe)
```

```
Oneway (individual) effect Within Model

Call:
plm(formula = reg, data = dados.plm, model = "within")

Balanced Panel: n=545, T=8, N=4360

Residuals :
      Min.   1st Qu.   Median   3rd Qu.    Max.
-4.17000 -0.12600  0.00925  0.16000  1.47000

Coefficients :
              Estimate Std. Error t-value Pr(>|t|)
exper    0.11684669   0.00841968  13.8778 < 2.2e-16 ***
expersq -0.00430089   0.00060527  -7.1057 1.422e-12 ***
married  0.04530332   0.01830968   2.4743  0.01339 *
uniaio   0.08208713   0.01929073   4.2553 2.138e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:    572.05
Residual Sum of Squares: 470.2
R-Squared      : 0.17804
Adj. R-Squared : 0.15563
F-statistic: 206.375 on 4 and 3811 DF, p-value: < 2.22e-16
> |
```

Com este modelo eliminamos os termos que são constantes no tempo, incluindo o termo constante que pertence ao resíduo. Ademais, notamos que variável *exper* (experiência) é importante, pois conforme a adquirimos a tendência é que nosso salário aumente em média 11 pontos percentuais. As variáveis *marriage* e *union* quanto realizamos estimação por efeitos fixos perdem poder na explicação do salário, mas continuam sendo estatisticamente significantes.

Dentro deste contexto, podemos ainda realizar um teste para checarmos qual dentre os modelos de efeitos fixos e aleatório se mostra mais adequado para nosso problema. Podemos realizar o teste de Hausmann de duas formas, sendo que a primeira analisa individualmente se as variáveis possuem alguma variação entre *i* e *t*:

```
wi = plm(reg,data=dados.plm,effect="individual",model="within")
re = plm(reg,data=dados.plm,effect="individual",model="random")
```

*phptest(wi, re)*

E o modo convencional:

*phptest(reg.fe, reg.re)*

#### Hausman Test

```
data: reg
chisq = 31.4515, df = 4, p-value = 2.476e-06
alternative hypothesis: one model is inconsistent
```

Fazendo os dois testes, podemos concluir que o modelo de efeitos fixos é o mais adequado para nossa estimação. Apenas para apresentarmos o comando, vamos estimar este modelo na forma de primeira diferença:

*reg.fd = plm(reg, data=dados.plm, model="fd")*

*summary(reg.fd)*

Eis os resultados:

```
Oneway (individual) effect First-Difference Model

Call:
plm(formula = reg, data = dados.plm, model = "fd")

Balanced Panel: n=545, T=8, N=4360

Residuals :
    Min. 1st Qu.  Median 3rd Qu.    Max.
-4.5800 -0.1460 -0.0127  0.1330  4.8400

Coefficients :
              Estimate Std. Error t-value Pr(>|t|)
(intercept)  0.1157500  0.0195867   5.9096 3.729e-09 ***
expersq      -0.0038824  0.0013863  -2.8005 0.005128 **
married       0.0381377  0.0229283   1.6633 0.096325 .
uniao        0.0427878  0.0196575   2.1767 0.029566 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:    751.19
Residual Sum of Squares: 748.04
R-Squared      : 0.0042035
Adj. R-Squared : 0.0041991
F-statistic: 5.36242 on 3 and 3811 DF, p-value: 0.0011047
~ |
```

Por fim, calculamos alguns testes de diagnóstico para dados em painel:

*pooltest(reg.pols, reg.fe)*

*plmtest(reg.pols, effect="twoways", type="ghm")*

*pbsytest(reg,data=dados.plm)*

```
> pooltest(reg,pols,reg.fe)

      F statistic

data:  reg
F = 8.0242, df1 = 541, df2 = 3811, p-value < 2.2e-16
alternative hypothesis: unstability

> plmtest(reg,pols,effect="twoways",type="ghm")

Lagrange Multiplier Test - two-ways effects (Gourieroux, Holly and Monfort)

data:  reg
chisq = 3216.735, df = 2, p-value < 2.2e-16
alternative hypothesis: significant effects

> pbsytest(reg,data=dados.plm)

      Bera, Sosa-Escudero and Yoon locally robust test

data:  formula
chisq = 218.972, df = 1, p-value < 2.2e-16
alternative hypothesis: AR(1) errors sub random effects
```

Em suma, o primeiro comando realiza um teste de Chow que verifica o agrupamento dos dados. O segundo é um teste de multiplicador de Lagrange para efeitos temporais ou individuais e o último testa a correlação serial dos resíduos. Observe que nosso modelo de efeito fixo, aparentemente está bem ajustado.



## Bibliografia Parcial

Adkins, L. *Using gretl for Principles of Econometrics*, 4<sup>th</sup> edition. (versão 1.041, 01/02/2013, online).

Arai, M. (2004). A Brief Guide to R for Beginners in Econometrics. (online).

Bollen, K.A. & Jackman, R.W. (1990). Regression Diagnostics: an expository treatment of outliers and influential cases. In: Fox, John; and Long, J. Scott (eds.); *Modern Methods of Data Analysis* (pp. 257-91). Newbury Park, CA: Sage.

Calógeras, P. (1960) [1910] *A Política Monetária do Brasil*. Companhia Editora Nacional, Rio de Janeiro.

Enders, W. (2004) *Applied Econometric Time Series*, Wiley & Sons, 2<sup>nd</sup> edition.

Epple, D. & McCallum, B.T. (2005) Simultaneous Equation Econometrics: The Missing Example. *Economics Inquiry*, v. 44, n.2, pp.374-84.

Fox, J. & Weisberg, S. (2011) *An R Companion to Applied Regression*. SAGE, 2<sup>nd</sup> edition.

Gujarati, D.N. & Porter, D.C. (2011) *Econometria Básica*. McGraw-Hill/Bookman, 5<sup>a</sup> edição.

Heiss, Florian (2016) *Using R for Introductory Econometrics*. Disponível em <http://www.URfIE.net>.

Hill, R.C.; Griffiths, W.E. & Lim, G.C. (2011) *Principles of Econometrics*, 4<sup>th</sup> edition. John Wiley & Sons.

Hyndman, R.J.; Koehler, A.B.; Ord, J.K. & Snyder, R.D. (2008) *Forecasting with Exponential Smoothing*. Springer-Verlag.

Johnston, J. (1991) *Econometric Methods*. McGraw-Hill.

Kleiber, C. & Zeileis, A. (2008) *Applied Econometrics with R*. Springer-Verlag.

Lundholm, M. (2011) *Introduction to R's time series facilities*. (online).

Pfaff, B. (2008) *Analysis of Integrated and Cointegrated Time Series with R*. Springer-Verlag.

Shikida, C.D.; Araujo Jr, A.F. de & Figueiredo, E.A. (2011). Ovos, galinhas: revisitando um dilema secular a partir de dados brasileiros. *Boletim Economia e tecnologia*, v. 26, p. 161-168.

Shikida, C.D. & Cortes, G.S. (2013) *Early Financial Crises in an Infant Republic: An Empirical Analysis of the Encilhamento in Brazil (1889-1906)*. Artigo apresentado no X Congresso da Associação Brasileira de Pesquisadores em História Econômica.

Shikida, C.D.; Nogueira Jr., R.P. & Araujo Jr., A.F. de. (2011) Structural Breaks in Exchange Rate Regimes in Brazil. *Economics Bulletin*, v.31, n.2, p.1748-1756.

Shikida, C.D.; Araujo Jr., A.F. de & Faria, Fernanda R. F. de. (s.d.) *Plebiscito e Criação de Novos Estados: o caso do Pará*. (mimeo)

Shumway, R.H. & Stoffer, D.S. (2011) *Time Series Analysis and its Applications with R Examples*. Springer-Verlag, 3<sup>rd</sup> edition.

Stock, J.H. & Watson, M.W. (2010) *Introduction to Econometrics*. Addison-Wesley, 3<sup>rd</sup> edition.

Thurman, W.N. & Fisher, M.E. (1988) Chickens, Eggs, and Causality, or Which Came First? *American Journal of Agricultural Economics*, v.70, n.2, May/88, pp. 237-8.