PP4RS | R Module

Slot 5

Dora Simon

07.09.2018

Outline of the R-Module

Slot 1: Intro & Data Types

Slot 2: Conditionals and Functions & Loops

Slot 3: Read in Data

Slot 4: Data Manipulation

Slot 5: Regressions

Slot 6: Graphs

Slot 7: knitR

Now: Regressions

Regressions

Notation

- regular letters (i.e. X, Y) = generally used to denote **observed** variables
- Greek letters (i.e. μ, σ) = generally used to denote **unknown** variablesthat we are trying to estimate
- $X_1, X_2, \ldots, X_n$ describes n data points
- $\bar{X}$, $\bar{Y}$ = observed means for random variables X and Y
- $\hat{\beta}_0$, $\hat{\beta}_1$ = estimators for true values of β_0 and β_1

Basic Regression Model:

$$Y_i = \beta_0 + \beta_1 X_i + u_i$$

Functions for Regressions

Functions for Regressions

- Linear regression: `lm`

```
my_ols<-lm(y ~ x_1 + x_2, data = my_dataframe)
```

Functions for Regressions

- Linear regression: `lm`

```
my_ols<-lm(y ~ x_1 + x_2, data = my_dataframe)
```

- IV regression: `ivreg`

```
# "z" is the instrument
my_iv<-ivreg(y ~ x_1 + x_2 | z, data = my_dataframe)
```

Functions for Regressions

Functions for Regressions

- Pooled OLS: `plm`

```
my_pooled<-plm(y ~ x_1 + x_2, data = my_dataframe, model="pooling")
```

Functions for Regressions

- Pooled OLS: `plm`

```
my_pooled<-plm(y ~ x_1 + x_2, data = my_dataframe, model="pooling")
```

- Within: `plm`

```
# You have to put your data in panel format:
# Need a time and an index variable
plm_dataframe<-plm.data(my_dataframe, index=c(country, year))
my_within<-plm(y ~ x_1 + x_2, data = plm_dataframe, model="within")
```

Functions for Regressions

- Pooled OLS: `plm`

```
my_pooled<-plm(y ~ x_1 + x_2, data = my_dataframe, model="pooling")
```

- Within: `plm`

```
# You have to put your data in panel format:
# Need a time and an index variable
plm_dataframe<-plm.data(my_dataframe, index=c(country, year))
my_within<-plm(y ~ x_1 + x_2, data = plm_dataframe, model="within")
```

- Random Effects: `plm`

```
# Also need the plm data here!
my_re<-plm(y ~ x_1 + x_2, data = plm_dataframe, model="random")
```

Standard Errors

So far, we only regressed and did not adjust the standard errors.

Standard Errors

So far, we only regressed and did not adjust the standard errors. Assume we have a linear regression

```
my_ols<-lm(y ~ x_1 + x_2, data = my_dataframe)
```

Standard Errors

So far, we only regressed and did not adjust the standard errors. Assume we have a linear regression

```
my_ols<-lm(y ~ x_1 + x_2, data = my_dataframe)
```

We have to define robust standard errors

```
ols_vcovmatrix <- vcovHC(my_ols, type = "HC0" )
ols.robust.se   <- sqrt(diag(model.olsrobust ))
```

Standard Errors

So far, we only regressed and did not adjust the standard errors. Assume we have a linear regression

```
my_ols<-lm(y ~ x_1 + x_2, data = my_dataframe)
```

We have to define robust standard errors

```
ols_vcovmatrix <- vcovHC(my_ols, type = "HC0" )
ols.robust.se   <- sqrt(diag(model.olsrobust ))
```

For clustered standard errors:

```
plm_dataframe<-plm.data(my_dataframe, index=c(country, year))
my_within<-plm(y ~ x_1 + x_2, data = plm_dataframe, model="within")
```

Standard Errors

So far, we only regressed and did not adjust the standard errors. Assume we have a linear regression

```
my_ols<-lm(y ~ x_1 + x_2, data = my_dataframe)
```

We have to define robust standard errors

```
ols_vcovmatrix <- vcovHC(my_ols, type = "HC0" )
ols.robust.se   <- sqrt(diag(model.olsrobust ))
```

For clustered standard errors:

```
plm_dataframe<-plm.data(my_dataframe, index=c(country, year))
my_within<-plm(y ~ x_1 + x_2, data = plm_dataframe, model="within")
```

```
within_vcovmatrix_cluster <- vcovHC(my_within, type = "HC0",
                                    cluster="group")
within.robust.se   <- sqrt(diag(ols_vcovmatrix_cluster ))
```

Easier: felm function from lfe package

Similar to Stata's `areg`: Better way of estimating fixed effects models.

- Panel of countires, control for fixed country factors

Easier: felm function from lfe package

Similar to Stata's `areg`: Better way of estimating fixed effects models.

- Panel of countires, control for fixed country factors
- In R: include country as a factor variable, have as many dummy variables as factors

Easier: felm function from lfe package

Similar to Stata's `areg`: Better way of estimating fixed effects models.

- Panel of countires, control for fixed country factors
- In R: include country as a factor variable, have as many dummy variables as factors
- Every country adds an extra column to the data frame!

Easier: felm function from lfe package

Similar to Stata's `areg`: Better way of estimating fixed effects models.

- Panel of countires, control for fixed country factors
- In R: include country as a factor variable, have as many dummy variables as factors
- Every country adds an extra column to the data frame!
- Data transformations that help with one factor
  - use first differences
  - include group means (Mundlak)

Easier: felm function from lfe package

Similar to Stata's `areg`: Better way of estimating fixed effects models.

- Panel of countires, control for fixed country factors
- In R: include country as a factor variable, have as many dummy variables as factors
- Every country adds an extra column to the data frame!

- Data transformations that help with one factor
    - use first differences
    - include group means (Mundlak)

- With multiple factors: Use `felm`: elaboration of group demeaning!
  Technical Details

Felm example

```
# Pooled OLS with lm: Include all the dummies
my_lm <- lm(y ~ x_1 + x_2 +
              factor(country) + factor(year), data = my_dataframe)

# Pooled OLS
my_felm <- felm(y ~ x_1 + x_2 | country | 0 | year,
                data = my_dataframe)
```

Cluster-robust standard errors will be automatically used!

Stargazer

So far, we only regressed stuff. Eventually, you also want to look at your regression. For that, we can use my (second) favorite R package: `stargazer`

Stargazer

So far, we only regressed stuff. Eventually, you also want to look at your regression. For that, we can use my (second) favorite R package: `stargazer`

Usage of the syntax:

```
stargazer(my_ols, my_iv, my_within,
          title="All my regressions",
          se=list(ols.robust.se, NULL, within.robust.se))
```

Stargazer

So far, we only regressed stuff. Eventually, you also want to look at your regression. For that, we can use my (second) favorite R package: `stargazer`

Usage of the syntax:

```
stargazer(my_ols, my_iv, my_within,
          title="All my regressions",
          se=list(ols.robust.se, NULL, within.robust.se))
```

There are a gazillion of options. There is a nice cheatsheet which I use a lot.

Some more realistic examples for using stargazer can be found here.

A note on RStudio

Windows

- you can create a shortcut to RStudio
- you can customize the "Start in" flied

When launched through the shortcut, RStudio will startup within the specified working directory.

Mac

- you can drag a nd drop a folder from the Finder on the RStudio Dock icon

Then, RStudio will startup with the dropped folder as the current working directory.

Linux

- you can run RStudio from the shell

If you have prespecified a working directory in shell, RStudio will startup there.

Another note on RStudio

If you open RStudio and some data is loaded automatically, you can change that in your options.

Tools -> Global options -> Restore .RData into workspace at startup: No!!

In the same window, you can also speficy whether RStudio should ask you to store your .RData at the end of the session. Hadley Wickham would say no ;).

Exercises

Exercises

We will replicate table 2 of the paper "Income and Demovracy" by Acemoglu et al (2008). The original paper can be found here.

First, we will create an R Project together and copy the relevant files to the project folders.

Create an R Project

- Click on "File" and "New Project" in RStudio
- Choose "New Directory"
- Choose "New Project"
- Come up with a name (I came up with "replicating-papers")
- Choose a folder on your computer where your project should be located (**NOT** the common course material folder)
- Create the project

When you open an RProject

- A new R session (process) is started
- The .Rprofile file in the project's main directory (if any) is sourced by R
- The .RData file in the project's main directory is loaded (if project options indicate that it should be loaded).
- The .Rhistory file in the project's main directory is loaded into the RStudio History pane (and used for Console Up/Down arrow command history).
- The current working directory is set to the project directory.
- Previously edited source documents are restored into editor tabs
- Other RStudio settings (e.g. active tabs, splitter positions, etc.) are restored to where they were the last time the project was closed.

When you close an RProject

When you are within a project and choose to either Quit, close the project, or open another project the following actions are taken:

- .RData and/or .Rhistory are written to the project directory (if current options indicate they should be)
- The list of open source documents is saved (so it can be restored next time the project is opened)
- Other RStudio settings (as described above) are saved.
- The R session is terminated.

The same happens when you either quit or open another projects.

More background on working with projects can be found here.

TABLE 2—FIXED EFFECTS RESULTS USING FREEDOM HOUSE MEASURE OF DEMOCRACY

	Base sample, 1960–2000								
	Five-year data					Annual data	Ten-year data		Twenty-year data
	Pooled OLS (1)	Fixed effects OLS (2)	Anderson-Hsiao IV (3)	Arellano-Bond GMM (4)	Fixed effects OLS (5)	Fixed effects OLS (6)	Fixed effects OLS (7)	Arellano-Bond GMM (8)	Fixed effects OLS (9)
	Dependent variable is democracy								
Democracy$_{t-1}$	0.706	0.379	0.469	0.489		[0.00]	–0.025	0.226	–0.581
	(0.035)	(0.051)	(0.100)	(0.085)			(0.088)	(0.123)	(0.198)
Log GDP per capita$_{t-1}$	0.072	0.010	–0.104	–0.129	0.054	[0.33]	0.053	–0.318	–0.030
	(0.010)	(0.035)	(0.107)	(0.076)	(0.046)		(0.066)	(0.180)	(0.156)
Hansen *J* test				[0.26]				[0.07]	
AR(2) test				[0.45]				[0.96]	
Implied cumulative effect of income	0.245	0.016	–0.196	–0.252				–0.411	–0.019
	[0.00]	[0.76]	[0.33]	[0.09]				[0.09]	[0.85]
Observations	945	945	838	838	958	2895	457	338	192
Countries	150	150	127	127	150	148	127	118	118
R-squared	0.73	0.80			0.76	0.93	0.77		0.89

Notes: Pooled cross-sectional OLS regression in column 1, with robust standard errors clustered by country in parentheses. Fixed effects OLS regressions in columns 2, 5, 6, 7, and 9, with country dummies and robust standard errors clustered by country in parentheses. Implied cumulative effect of income represents the coefficient estimate of log GDP

Do the regressions

For your convenience, I already did the data manipulation so that the data is ready for you to regress.

1. Read in the tidy data and call the data frame `ajry_df`.
2. Replicate column 1 of the paper using the `lm` function. Adjust your standard errors to be clustered on a country level.
3. Display your regression results with the correct standard errors using `stargazer`.
    - If you have trouble with the output table, explore the `type` option.
    - Change the variable names to something more similar as in the paper (I did not figure out how to have a nice latex-style variable name, but maybe you do)
    - Keep only those statistics that Table 2 also has.
    - You don't have to calculate the "Implied Cumulative Effect of Income"
    - You don't have to have the number of countries
    - It is ok if your R-Squared is different
4. Replicate column 2 of the paper using the `lm` function. Adjust your standard errors to be clustered on a country level.
5. Display your regression results with the correct standard errors using `stargazer` next to your column 1. Keep all settings from column 1 and extend them to column 2 if necessary.

Do the regressions

1. Replicate column 1 of the paper using the `felm` function.
2. Display your regression results with the correct standard errors using `stargazer`. Keep all necessary settings from the previous exercises.
3. Replicate column 2 of the paper using the `felm` function.
4. Display your regression results with the correct standard errors using `stargazer` next to your column 1.

Extra Exercises

If you finished with all regressions, you can go back to the start: I will let you replicate the data manipulation part.

I uploaded the tidy data so that you will for sure get to regress something today, but it is very useful to create the variables yourself.

On the following slides, the instructions will guide you through the data generation process. In the end, your regressions should run the same way as they do with the .rds file before.

The Raw Data

1. Create a folder in your R Project for the raw data.
2. Copy the raw data and the Readme file from folder 17 into your R Project raw data folder.
3. Start an R Script in your R Project folder in which you do the analysis.

Read in and manipulate the data

1. Start the beginning of your script for loading the libraries you will need. As you continue your work, you can add more and more libraries to the beginning.
2. Read in the raw data. We will need only sheet 2 of the excel file.
3. Reorder the data by country and by year.
4. Rename the variable `lrgdpch` to `log_gdp_pc` and the variable `fhpolrigaug` to `freedom_house`.

Create Lagged Variables

1. Group the data by country.
2. Create the following lagged variables, all by year:
    - `lag_log_gdp_pc`
    - `lag_freedom_house`
    - `lag2_nsave`: the second lag of `nsave`
    - `lag_worldincome`
3. Keep only those observations for which `sample==1`
4. Check if your regressions run in the same way as before.