Introduction to "R"
Recap and model estimation

Arndt Leininger

@a_leininger
arndt.leininger@fu-berlin.de

29 September 2019

Exercise and recap

Exercise and recap

Good morning!

Continue your work on `hands-on/03_datawrangling/hands-on1.R`.

If you're done, you can begin solving the take-home exercise: `exercise/R_exercise_1_questions.pdf`. You can write your solutions into `exercise/R_exercise_1.R`.

Program

Sunday, 29 September 2019

10:00h - 11:15h Recap and Model Estimation

11:15h - 11:30h break

11:30h - 13:00h Data Visualization

13:00h - 14:00h Lunch break

14:00h - 16:30h Data and Model Visualization

OLS Regression

- Regression models are functions that are fed an equation and data
- Further options are possible but optional
- The dependent variable is seperated by a tilde from the independent variables
- Equation: `dv ~ iv`
- No need for $ operator in the equation

```
library(foreign)
d <- read.dta('../data/EUsuppDK.dta')

lm(left_right ~ age, d)
```

OLS Regression

```
library(foreign)
d <- read.dta('../data/EUsuppDK.dta')

lm(left_right ~ age, d)
```

```
##
## Call:
## lm(formula = left_right ~ age, data = d)
##
## Coefficients:
## (Intercept)          age
##    5.087151     0.007995
```

Formulae

A complete overview of formulae in R

https://ww2.coastal.edu/kingw/statistics/R-tutorials/formulae.html

Model objects

- You can save the output of the `lm()` function just like with any other function.

```
m1 <- lm(left_right ~ age, d)
```

- `lm()` is for linear models, i.e. OLS

Model objects

- Once you save an estimated model as object you can always access it to obtain model statistics.

```
summary(m1)  # estimation results

coef(m1)  # coefficients

vcov(m1)  # Variance-Covariance Matrix

predict(m1)  # Predicted values

resid(m1)  # Residuals
```

Model objects

```
summary(m1)  # estimation results
```

```
##
## Call:
## lm(formula = left_right ~ age, data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.687 -1.303 -0.351  1.515  4.649
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.087151   0.157297  32.341   <2e-16 ***
## age         0.007995   0.003327   2.403   0.0164 *
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.787 on 966 degrees of freedom
##   (33 observations deleted due to missingness)
```

Model objects

```
coef(m1)  # coefficients
```

```
## (Intercept)         age
## 5.087151366 0.007995059
```

Model objects

```r
predict(m1)  # Predicted values
```

```
##        1        2        3        4        5        6
## 5.414949 5.422944 5.215072 5.303018 5.279033 5.223067
##        7        8        9       10       11       12
## 5.231062 5.295023 5.295023 5.303018 5.231062 5.414949
##       13       14       15       16       17       18
## 5.311013 5.303018 5.279033 5.239057 5.319008 5.279033
##       19       20       21       22       23       24
## 5.430939 5.422944 5.327003 5.247053 5.438934 5.446929
##       25       26       27       28       29       30
## 5.430939 5.271038 5.255048 5.263043 5.454924 5.462919
##       31       32       33       34       35       36
## 5.470914 5.311013 5.319008 5.327003 5.438934 5.334998
##       37       38       39       40       41       42
## 5.334998 5.271038 5.223067 5.478909 5.446929 5.342993
##       43       44       45       46       47       48
## 5.454924 5.350988 5.358983 5.486904 5.462919 5.494899
##       49       50       51       52       53       54
```

Model objects

```r
resid(m1)  # Residuals
```

```
##          1          2          3          4          5
## -0.4149488 -0.4229438 -0.2150723  1.6969820  2.7209672
##          6          7          8          9         10
## -1.2230674 -0.2310624 -0.2950229 -0.2950229 -0.3030180
##         11         12         13         14         15
## -0.2310624  1.5850512  0.6889870  1.6969820 -3.2790328
##         16         17         18         19         20
##  1.7609425  2.6809919  2.7209672 -1.4309389 -0.4229438
##         21         22         23         24         25
## -1.3270031 -0.2470525  1.5610660 -0.4469290  1.5690611
##         26         27         28         29         30
## -3.2710377 -2.2550476 -3.2630427 -0.4549241  1.5370809
##         31         32         33         34         35
## -0.4709142 -0.3110130  1.6809919 -3.3270031  2.5610660
##         36         37         38         39         40
##  0.6650018 -0.3349982 -3.2710377 -0.2230674  2.5210907
##         41         42         43         44         45
##  1.5530710 -1.3429932  0.5450759 -0.3509883  2.6410166
```

Model objects

```
class(m1)
```

```
## [1] "lm"
```

```
objects(m1)
```

```
##  [1] "assign"        "call"          "coefficients"
##  [4] "df.residual"   "effects"       "fitted.values"
##  [7] "model"         "na.action"     "qr"
## [10] "rank"          "residuals"     "terms"
## [13] "xlevels"
```

Interactions

- Interactions can be specified as follows
  - `var1*var2` = `var1 + var2 + var1:var2`
  - `var1:var2` is simply the interaction term

```
m_i <- lm(left_right ~ sex*age, d)
# or
m_i <- lm(left_right ~ sex + age + sex:age, d)
```

Polynomials

```
lm(left_right ~ age + I(age^2), d) # second-order polynomial

lm(left_right ~ age + age^2, d) # does not work
```

- `?I()`: "Change the class of an object to indicate that it should be treated 'as is'."

A little trick

- Put the assignment of a model to an object in a `summary()` call to assign and view results at the same time

```
summary(m2 <- lm(left_right ~ age + I(age^2), d))
```

```
##
## Call:
## lm(formula = left_right ~ age + I(age^2), data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7388 -1.3154 -0.3393  1.5501  4.6607
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.3107878  0.3941342  13.475   <2e-16 ***
## age         -0.0029424  0.0179840  -0.164    0.870
## I(age^2)     0.0001153  0.0001863   0.619    0.536
## ---
## Signif. codes:
```

Predicted values

- `predict()` is a generic function to create predictions from various models

```
predicted_values <- predict(m1)
# but...
d$yhat <- predict(m1)
```

```
## Error in `$<-.data.frame`(`*tmp*`, yhat, value = c(`1` = 5.41
```

```
# Error in `$<-.data.frame`(`*tmp*`, "yhat",
# value = c(5.41494877919278,  :
# replacement has 968 rows, data has 1001
```

Predicted values

```r
d$yhat <- predict(m1, newdata = d)
# no error message because now predictions
# are also made for deleted observations;
# these predictions are obviously NA
```

Hands-on I

Hands-on I

`hands-on/04_modelestimation/hands-on1.R`

Regression tables

Lists

- Lists can contain any kind of objects of any type.
- Note: data.frames can also contain vectors of any of the three types but the vectors are forced to be of the same length.
- Example: One could have a list of differently sized vectors

```
v1 <- c(1, 2, 3)
v2 <- c('a', 'b', 'c', 'd')
alist <- list(v1, v2)
alist
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] "a" "b" "c" "d"
```

Tables

Using packages such as stargazer or texreg we can create nice regression tables.

```
library(stargazer)
```

```
##
## Please cite as:

##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression a

##  R package version 5.2.2. https://CRAN.R-project.org/package=
```

```
library(texreg)
```

```
## Version:  1.36.23
## Date:     2017-03-03
## Author:   Philip Leifeld (University of Glasgow)
##
## Please cite the JSS article in your publications -- see citat
```

Tables

Here's an example in `tex` using `stargazer`.

```
m1 <- lm(mpg ~ cyl, mtcars)
m2 <- lm(mpg ~ cyl + gear, mtcars)

stargazer(list(m1, m2), header = F, float = F,
          font.size = 'tiny', single.row = T)
```

	Dependent variable:	
	mpg	
	(1)	(2)
cyl	-2.876^{***} (0.322)	-2.743^{***} (0.373)
gear		0.652 (0.904)
Constant	37.885^{***} (2.074)	34.659^{***} (4.937)
Observations	32	32
R^2	0.726	0.731
Adjusted R^2	0.717	0.712
Residual Std. Error	3.206 (df = 30)	3.232 (df = 29)
F Statistic	79.561^{***} (df = 1; 30)	39.404^{***} (df = 2; 29)
Note:		*p<0.1; **p<0.05; ***p<0.01

Tables

stargazer provides tables in text, html and tex.

```
# output as text file
stargazer(m1,  type = 'text', out = 'tables/m1.txt')
# output as html file which Word can read
stargazer(m1, type = 'html', out = 'tables/m1.html')
# output as tex, the default
stargazer(m1, out = 'tables/m1.tex')
```

Screenreg

- texreg's screenreg() function is very useful to quickly view some models.

```
screenreg(list(m1, m2))
```

```
##
## ===================================
##                Model 1    Model 2
## -----------------------------------
## (Intercept)   37.88 ***   34.66 ***
##               (2.07)     (4.94)
## cyl           -2.88 ***  -2.74 ***
##               (0.32)     (0.37)
## gear                      0.65
##                           (0.90)
## -----------------------------------
## R^2            0.73       0.73
## Adj. R^2       0.72       0.71
## Num. obs.     32         32
## RMSE           3.21       3.23
```

Tables

texreg provides tables html, tex and to screen.

```
# output as tex file
texreg(m1,  file = 'tables/m1.tex')
# output as html file which Word can read
htmlreg(m1, file = 'tables/m1.html')
# output as tex, the default
screenreg(m1)
```

Regression tables for Word

Via HTML

- Export to HTML using `htmlreg()` (package `texreg`) or `stargazer(..., type = 'html')`
- Then copy and paste to Word
- Or, better, link to the html file from within the Word Document
  - Word: Insert -> Object (dropwdown) -> Text from File -> Insert (dropdown) -> Insert as link; hit F9 to refresh
  - LibreOffice Write: Insert -> Section -> Check option "Link" and choose document; to refresh: Edit -> Links -> click "Update"
  - http://www.techrepublic.com/article/link-to-another-file-in-your-word-document/

Hands-on II

Hands-on II

`hands-on/04_modelestimation/hands-on2.R`