Tutorial 4: Regression Model Estimation

Anh Le (anh.le@duke.edu)

September 18, 2015

Agenda

1. Create data frames
2. Subset data frames
3. Estimate a linear model with `lm()`
4. Tips and tricks

- Prefix your objects in R (and related `TAB` tricks, i.e. arguments within function, variables within a data frame)
- `fig.height()`, `fig.width()`
- Code length <= 80 (RStudio > Tools > Options > Code)

1. Create data frames

```
my_dataframe <- data.frame(var1 = c(11, 12, 13),
                           var2 = c(21, 22, 23),
                           var3 = c("a", "b", "c"))
my_dataframe
```

```
##   var1 var2 var3
## 1   11   21    a
## 2   12   22    b
## 3   13   23    c
```

2. Subset data frames

All subsetting can be done with the following construct: `my_dataframe[?1 , ?2]`

The first question mark (`?1`) refers to which rows we want. The second question mark (`?2`) refers to which columns we want.

How to indicate to R which rows / columns we want? Multiple ways:

1. Use rows / columns index

```
my_dataframe[1, 2]
```

```
## [1] 21
```

```
my_dataframe[1:2, 2]
```

```
## [1] 21 22
```

```
my_dataframe[1:2, ]
```

```
##   var1 var2 var3
## 1   11   21    a
## 2   12   22    b
```

Rapid fire quiz

```
my_dataframe[2:3, ]
my_dataframe[ , 1:2]
my_dataframe[1:2, 2:3]

my_dataframe[c(1, 3), ]
my_dataframe[c(1, 3, 2), ]
```

2. Use rows / columns name

```
my_dataframe[ , "var2"]
```

```
## [1] 21 22 23
```

Rapid fire quiz:

```
my_dataframe[ , c("var1", "var3")]
my_dataframe[c(2, 3), c("var1", "var3")]
```

3. Use a condition

```
my_dataframe[c(TRUE, TRUE, FALSE), ]
```

```
##   var1 var2 var3
## 1   11   21    a
## 2   12   22    b
```

```
my_dataframe[, c(TRUE, FALSE, TRUE)]
```

```
##   var1 var3
## 1   11    a
## 2   12    b
## 3   13    c
```

Of course this is not tenable for a large data frame. So we have this very useful trick:

```
my_dataframe[my_dataframe$var1 < 13, ]
```

```
##   var1 var2 var3
## 1   11   21    a
## 2   12   22    b
```

This works because `my_dataframe$var1 < 13` actuall returns `c(TRUE, TRUE, FALSE)` (vectorized operation in the wild!). Indeed:

```
my_dataframe$var1 < 13
```

```
## [1]  TRUE  TRUE FALSE
```

Rapid fire quiz:

```
my_dataframe[my_dataframe$var2 == 22, ]
my_dataframe[my_dataframe$var2 == 25, ]
```

4. Use a combination of condition

```
my_dataframe[my_dataframe$var1 > 10 & my_dataframe$var2 > 21, ]
```

```
##   var1 var2 var3
## 2   12   22    b
## 3   13   23    c
```

```
my_dataframe[my_dataframe$var1 > 10 | my_dataframe$var2 > 21, ]
```

```
##   var1 var2 var3
## 1   11   21    a
## 2   12   22    b
## 3   13   23    c
```

3. Estimate a linear model with `lm()`

In this section, I'll demo a (simplified) pipeline of steps in doing regression analysis with real data.

Download and clean data

```
library(WDI)
```

```
## Loading required package: RJSONIO
```

```
d_2010 <- WDI(indicator = c("NY.GDP.PCAP.CD", "SP.DYN.IMRT.IN", "SH.MED.PHYS.ZS"),
              start = 2010, end = 2010, extra = TRUE)
# d_2010[d_2010$region != "Aggregates", ]
```

There are a lot of unwanted columns. What if I just want `country`, `year`, and the three variables of interest (`NY.GDP.PCAP.CD`, `SP.DYN.IMRT.IN`, `SH.MED.PHYS.ZS`)? (Hint: subsetting)

```
d_2010 <- d_2010[d_2010$region != "Aggregates",
       c("country", "year", "NY.GDP.PCAP.CD", "SP.DYN.IMRT.IN", "SH.MED.PHYS.ZS")]
```

Rename columns:

```
colnames(d_2010)
```

```
## [1] "country"        "year"           "NY.GDP.PCAP.CD" "SP.DYN.IMRT.IN"
## [5] "SH.MED.PHYS.ZS"
```

```
colnames(d_2010)[3:5] <- c('gdppc', 'infant_mortality', 'number_of_physician')
colnames(d_2010)
```

```
## [1] "country"             "year"                "gdppc"
## [4] "infant_mortality"    "number_of_physician"
```

Log gdp per capita

```
d_2010$log_gdppc <- log(d_2010$gdppc)
```

Remove missing data

```
d_2010 <- na.omit(d_2010)
```

Build a linear model

```
lm(infant_mortality ~ log_gdppc, data = d_2010)
```

```
##
## Call:
## lm(formula = infant_mortality ~ log_gdppc, data = d_2010)
##
## Coefficients:
## (Intercept)    log_gdppc
##      134.53       -12.78
```

```
m1 <- lm(infant_mortality ~ log_gdppc, data = d_2010)
summary(m1)
```

```
##
## Call:
## lm(formula = infant_mortality ~ log_gdppc, data = d_2010)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -25.258  -8.809  -0.596   6.510  50.661
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 134.5287     6.0710   22.16   <2e-16 ***
## log_gdppc   -12.7846     0.6978  -18.32   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.85 on 148 degrees of freedom
## Multiple R-squared:  0.694,  Adjusted R-squared:  0.6919
## F-statistic: 335.7 on 1 and 148 DF,  p-value: < 2.2e-16
```

```
m2 <- lm(infant_mortality ~ log_gdppc + number_of_physician, data = d_2010)
summary(m2)
```

```
##
## Call:
## lm(formula = infant_mortality ~ log_gdppc + number_of_physician,
##     data = d_2010)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -25.281  -7.954  -1.578   5.957  50.889
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         122.0253     7.4063  16.476  < 2e-16 ***
## log_gdppc           -10.7678     0.9881 -10.898  < 2e-16 ***
## number_of_physician  -2.6479     0.9388  -2.821  0.00546 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.56 on 147 degrees of freedom
## Multiple R-squared:  0.7097, Adjusted R-squared:  0.7058
## F-statistic: 179.7 on 2 and 147 DF,  p-value: < 2.2e-16
```

Extract result from the model

`str()` (stands for `structure`) is used to look into the structure of an object in `R`, see what it contains.

```
str(m1)
```

```
## List of 12
##  $ coefficients : Named num [1:2] 134.5 -12.8
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "log_gdppc"
##  $ residuals    : Named num [1:150] 3.33 6.29 21.7 -13.39 -15.55 ...
##   ..- attr(*, "names")= chr [1:150] "6" "7" "8" "10" ...
##  $ effects      : Named num [1:150] -305.9 235.4 22.2 -13.5 -15.6 ...
##   ..- attr(*, "names")= chr [1:150] "(Intercept)" "log_gdppc" "" "" ...
##  $ rank         : int 2
##  $ fitted.values: Named num [1:150] -0.829 1.005 53.404 28.195 31.65 ...
##   ..- attr(*, "names")= chr [1:150] "6" "7" "8" "10" ...
##  $ assign       : int [1:2] 0 1
##  $ qr           :List of 5
##   ..$ qr   : num [1:150, 1:2] -12.2474 0.0816 0.0816 0.0816 0.0816 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:150] "6" "7" "8" "10" ...
##   .. .. ..$ : chr [1:2] "(Intercept)" "log_gdppc"
##   .. ..- attr(*, "assign")= int [1:2] 0 1
##   ..$ qraux: num [1:2] 1.08 1.09
##   ..$ pivot: int [1:2] 1 2
##   ..$ tol  : num 1e-07
##   ..$ rank : int 2
##   ..- attr(*, "class")= chr "qr"
##  $ df.residual  : int 148
##  $ xlevels      : Named list()
##  $ call         : language lm(formula = infant_mortality ~ log_gdppc, data = d_2010)
##  $ terms        :Classes 'terms', 'formula'  language infant_mortality ~ log_gdppc
##   .. ..- attr(*, "variables")= language list(infant_mortality, log_gdppc)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "infant_mortality" "log_gdppc"
##   .. .. .. ..$ : chr "log_gdppc"
##   .. ..- attr(*, "term.labels")= chr "log_gdppc"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(infant_mortality, log_gdppc)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "infant_mortality" "log_gdppc"
##  $ model        :'data.frame':   150 obs. of  2 variables:
##   ..$ infant_mortality: num [1:150] 2.5 7.3 75.1 14.8 16.1 13 3.6 4.1 33.9 6.4 ...
##   ..$ log_gdppc       : num [1:150] 10.59 10.44 6.35 8.32 8.05 ...
##   ..- attr(*, "terms")=Classes 'terms', 'formula'  language infant_mortality ~ log_gdppc
##   .. .. ..- attr(*, "variables")= language list(infant_mortality, log_gdppc)
##   .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:2] "infant_mortality" "log_gdppc"
##   .. .. .. .. ..$ : chr "log_gdppc"
##   .. .. ..- attr(*, "term.labels")= chr "log_gdppc"
##   .. .. ..- attr(*, "order")= int 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(infant_mortality, log_gdppc)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. ..- attr(*, "names")= chr [1:2] "infant_mortality" "log_gdppc"
##  - attr(*, "class")= chr "lm"
```

You can extract the coefficients

```
m1$coefficients
```

```
## (Intercept)   log_gdppc
##   134.52866   -12.78456
```

```
m1$coefficients['(Intercept)']
```

```
## (Intercept)
##    134.5287
```

```
m1$coefficients['log_gdppc']
```

```
## log_gdppc
## -12.78456
```

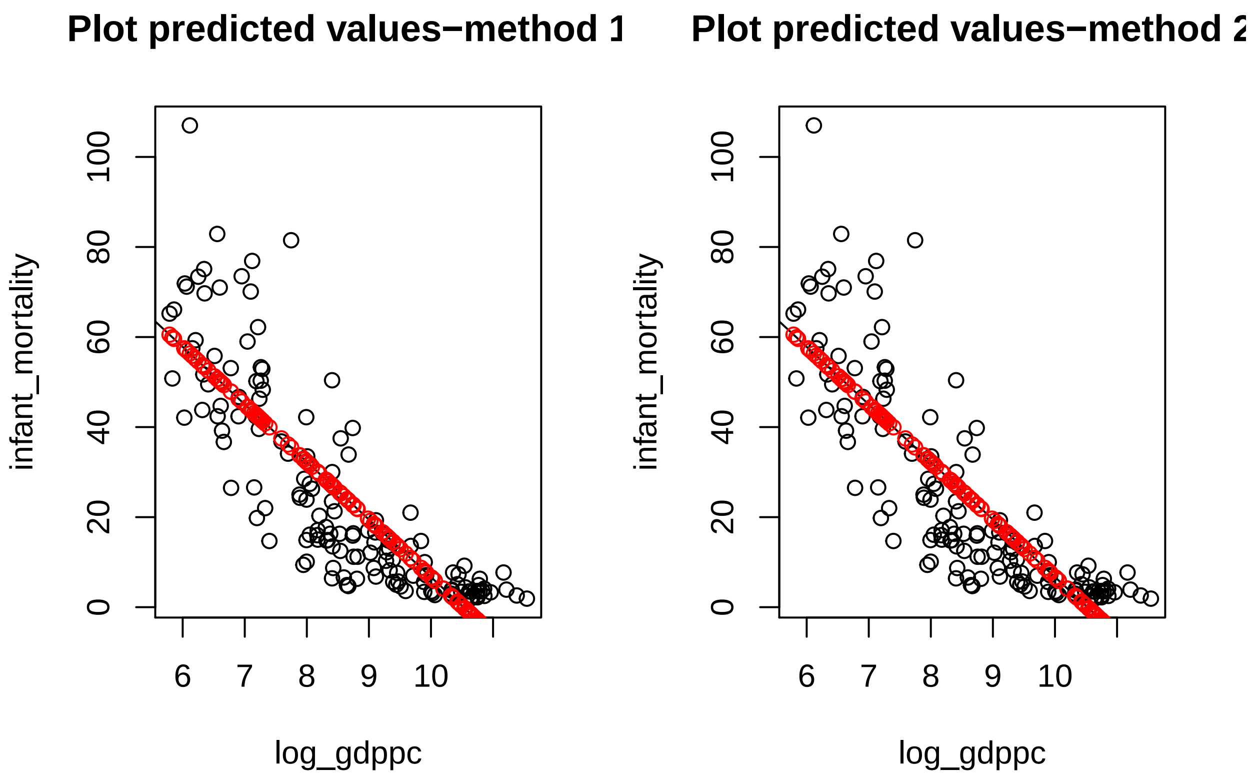
You can also generate predicted / fitted values:

```
d_2010$pred_infant_mortality1 <- predict(m1)
d_2010$pred_infant_mortality2 <- m1$coefficients['(Intercept)'] + m1$coefficients['log_gdppc'] * d_2010
```

Now we can use them for other things, e.g plotting

```
par(mfrow = c(1, 2))
plot(infant_mortality ~ log_gdppc, data = d_2010, main = "Plot predicted values-method 1")
abline(a = m1$coefficients['(Intercept)'], b = m1$coefficients['log_gdppc'])
points(d_2010$log_gdppc, d_2010$pred_infant_mortality1, col = 'red')

plot(infant_mortality ~ log_gdppc, data = d_2010, main = "Plot predicted values-method 2")
abline(a = m1$coefficients['(Intercept)'], b = m1$coefficients['log_gdppc'])
points(d_2010$log_gdppc, d_2010$pred_infant_mortality2, col = 'red')
```



Report the model in a nice, journal-ready format

The `stargazer` library takes your model objects and generates tables in LaTeX. This package has a lot of customizing options, which you'll explore in the homework.

```
library(stargazer)
```

```
##
## Please cite as:

##  Hlavac, Marek (2015). stargazer: Well-Formatted Regression and Summary Statistics Tables.

##  R package version 5.2. http://CRAN.R-project.org/package=stargazer
```

```
# LaTeX code that you can copy paste into LateX
stargazer(m1, m2)
```

```
##
## % Table created by stargazer v.5.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvar
## % Date and time: Mon, Sep 19, 2016 - 02:15:55 PM
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lcc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
##  & \multicolumn{2}{c}{\textit{Dependent variable:}} \\
## \cline{2-3}
## \\[-1.8ex] & \multicolumn{2}{c}{infant\_mortality} \\
## \\[-1.8ex] & (1) & (2)\\
## \hline \\[-1.8ex]
##  log\_gdppc & $-$12.785$^{***}$ & $-$10.768$^{***}$ \\
##   & (0.698) & (0.988) \\
##   & & \\
##  number\_of\_physician &  & $-$2.648$^{***}$ \\
##   &  & (0.939) \\
##   & & \\
##  Constant & 134.529$^{***}$ & 122.025$^{***}$ \\
##   & (6.071) & (7.406) \\
##   & & \\
## \hline \\[-1.8ex]
## Observations & 150 & 150 \\
## R$^{2}$ & 0.694 & 0.710 \\
## Adjusted R$^{2}$ & 0.692 & 0.706 \\
## Residual Std. Error & 12.850 (df = 148) & 12.558 (df = 147) \\
## F Statistic & 335.667$^{***}$ (df = 1; 148) & 179.699$^{***}$ (df = 2; 147) \\
## \hline
## \hline \\[-1.8ex]
## \textit{Note:}  & \multicolumn{2}{r}{$^{*}$p$<$0.1; $^{**}$p$<$0.05; $^{***}$p$<$0.01} \\
## \end{tabular}
## \end{table}
```

```
# If using knir, use the option results='asis'
stargazer(m1, m2)
```

% Table created by stargazer v.5.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Mon, Sep 19, 2016 - 02:15:55 PM

Table 1:

	Dependent variable:	
	infant_mortality	
	(1)	(2)
log_gdppc	-12.785^{***}	-10.768^{***}
	(0.698)	(0.988)
number_of_physician		-2.648^{***}
		(0.939)
Constant	134.529^{***}	122.025^{***}
	(6.071)	(7.406)
Observations	150	150
R^2	0.694	0.710
Adjusted R^2	0.692	0.706
Residual Std. Error	12.850 (df = 148)	12.558 (df = 147)
F Statistic	335.667^{***} (df = 1; 148)	179.699^{***} (df = 2; 147)
Note:		*p<0.1; **p<0.05; ***p<0.01