Descriptive Analysis

R for Stata Users

Luiza Andrade, Rob Marty, Rony Rodriguez-Ramirez, Luis Eduardo San Martin, Leonardo Viotti
DIME |The World Bank

24 May 2022

Table of contents

Workflows for outputs

Not reproducible

Anything that requires

Copy-pasting

Manual formatting after exported

Reproducible

R Markdown: dynamic document containing code and text that is exported directly from R into PDF, HTML, Word, Power Point and other formats

LaTeX: typesetting system used for scientific publications that automatically reloads tables and figures every time the document is rendered

Setting the stage

Load the packages that we will use today

```
# Install new packages
install.packages("skimr")
install.packages("lfe")
install.packages("huxtable")
install.packages("openxlsx")
```

```
# Load packages
library(here)
library(tidyverse)
library(modelsummary)
library(lfe)
library(huxtable)
library(openxlsx)
```

Setting the stage

Load the data that we will use today: Stata's `census` dataset

```
# Load data
 census <-
  read_rds(
    here(
      "DataWork",
      "DataSets",
      "Final",
      "census.rds"
    )
  )
```

02:00

Taking a peek at the data

```
glimpse(census)
```

```
## Rows: 50
## Columns: 13
## $ state    <chr> "Alabama", "Alaska", "Arizona", "Arkansas", "California", "Co~
## $ state2   <chr> "AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "FL", "GA", "~
## $ region   <fct> South, West, West, South, West, West, NE, South, South, South~
## $ pop      <int> 3893888, 401851, 2718215, 2286435, 23667902, 2889964, 3107576~
## $ poplt5   <int> 296412, 38949, 213883, 175592, 1708400, 216495, 185188, 41151~
## $ pop5_17  <int> 865836, 91796, 577604, 495782, 4680558, 592318, 637731, 12544~
## $ pop18p   <int> 2731640, 271106, 1926728, 1615061, 17278944, 2081151, 2284657~
## $ pop65p   <int> 440015, 11547, 307362, 312477, 2414250, 247325, 364864, 59179~
## $ popurban <int> 2337713, 258567, 2278728, 1179556, 21607606, 2329869, 2449774~
## $ medage   <dbl> 29.3, 26.1, 29.2, 30.6, 29.9, 28.6, 32.0, 29.8, 34.7, 28.7, 2~
## $ death    <int> 35305, 1604, 21226, 22676, 186428, 18925, 26005, 5123, 104190~
## $ marriage <int> 49018, 5361, 30223, 26513, 210864, 34917, 26048, 4437, 108344~
## $ divorce  <int> 26745, 3517, 19908, 15882, 133541, 18571, 13488, 2313, 71579,~
```

Quick summary statistics

Exploring a dataset

`summary(x, digits)`

Equivalent to Stata's `codebook`. Its arguments are:

- **x:** the object you want to summarize, usually a vector or data frame
- *digits:* the number of decimal digits to be displayed

Exercise ⌨

Use the `summary()` function to describe the `census` data frame.

00:45

Exploring a dataset

```
summary(census)
```

```
##     state               state2               region        pop          
##  Length:50          Length:50          NE     : 9   Min.   :  401851  
##  Class :character   Class :character   N Cntrl:12   1st Qu.: 1169218  
##  Mode  :character   Mode  :character   South  :16   Median : 3066433  
##                                        West   :13   Mean   : 4518149  
##                                                     3rd Qu.: 5434033  
##                                                     Max.   :23667902  
##      poplt5            pop5_17            pop18p             pop65p       
##  Min.   :  35998   Min.   :  91796   Min.   :  271106   Min.   :  11547  
##  1st Qu.:  98831   1st Qu.: 257949   1st Qu.:  823702   1st Qu.: 118660  
##  Median : 227468   Median : 629654   Median : 2175130   Median : 370495  
##  Mean   : 326278   Mean   : 945952   Mean   : 3245920   Mean   : 509503  
##  3rd Qu.: 361321   3rd Qu.:1143292   3rd Qu.: 3858173   3rd Qu.: 580087  
##  Max.   :1708400   Max.   :4680558   Max.   :17278944   Max.   :2414250  
##     popurban            medage          death            marriage      
##  Min.   :  172735   Min.   :24.20   Min.   :  1604   Min.   :  4437  
##  1st Qu.:  826651   1st Qu.:28.73   1st Qu.:  9087   1st Qu.: 14840  
##  Median : 2156905   Median :29.75   Median : 26177   Median : 36279  
```

Summarizing continuous variables

- `summary()` can also be used with a single variable.
- When used with continuous variables, it works similarly to `summarize` in Stata.
- When used with categorical variables, it works similarly to `tabulate`.

Summarizing continuous variables

Exercise ⌨

Use the `summary()` function to display summary statistics for a continuous variable in the `census` data frame.

00:45

Summarizing continuous variables

Exercise ⌨

Use the `summary()` function to display summary statistics for a continuous variable in the `census` data frame.

```
summary(census$pop)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   401851  1169218  3066433  4518149  5434033 23667902
```

Summarizing categorical variables

`table()`

Equivalent to `tabulate` in Stata, creates a frequency table. Its main arguments are vectors to be tabulated.

Exercise ⌨

Use the `table()` function to display frequency tables for:

1. The variable `region` in the `census` data frame
2. The variables `region` and `state` in the `census` data frame, simultaneously

01:00

Summarizing categorical variables

One way tabulation

```
table(census$region)
```

```
##
##       NE N Cntrl    South     West
##        9       12       16       13
```

Summarizing continuous variables

Two way tabulation

```
table(census$state, census$region)
```

```
##
##                NE N Cntrl South West
##   Alabama       0       0     1    0
##   Alaska        0       0     0    1
##   Arizona       0       0     0    1
##   Arkansas      0       0     1    0
##   California    0       0     0    1
##   Colorado      0       0     0    1
##   Connecticut   1       0     0    0
##   Delaware      0       0     1    0
##   Florida       0       0     1    0
##   Georgia       0       0     1    0
##   Hawaii        0       0     0    1
##   Idaho         0       0     0    1
##   Illinois      0       1     0    0
##   Indiana       0       1     0    0
```

Descriptives tables

Descriptives tables

What if you want to...

- ...export the summary statistics to another software?
- ...customize which statistics to display?
- ...format the table?

Well, then you will need to go beyond base R

- There are many packages that can be used both for displaying and exporting summary statistics
- Today we will show you a combination of two packages: `modelsummary` and `huxtable`
- We chose this combination because together, they can perform all the tasks we are interested in
- In fact, `modelsummary` can perform most of them by itself -- with the exception of exporting formatted tables to Excel

Exploring datasets with *modelsummary*

The package *modelsummary* contains a family of functions called `datasummary` which can be used to create different types of summary statistics tables. These include:

- `datasummary_skim`, to create descriptive statistics tables
- `datasummary_balance`, to create balance tables
- `datasummary_correlation`, to create a correlation table
- `datasummary_crosstab`, to create a twoway tabulation
- `datasummary`, to create customized descriptive statistics tables

Exploring datasets with *modelsummary*

`datasummary_skim()`

- **data:** the data set to be summarized, the only required argument
- *output:* the type of output desired
- *...*: additional options allow for formatting customization, such as including notes and titles

```
datasummary_skim(
  data,
  type = "numeric",
  output = "default",
  histogram = TRUE,
  title = NULL,
  notes = NULL,
  ...
)
```

Exploring datasets with *modelsummary*

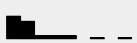
Exercise ⌨

Use `datasummary_skim()` to create a descriptive statistics table for the `census` data

00:45

Exploring datasets with *modelsummary*

```
datasummary_skim(census)
```

	Unique (#)	Missing (%)	Mean	SD	Min	Median	Max	
pop	50	0	4518149.4	4715037.8	401851.0	3066433.0	23667902.0	
poplt5	50	0	326277.8	331585.1	35998.0	227467.5	1708400.0	
pop5_17	50	0	945951.6	959372.8	91796.0	629654.0	4680558.0	
pop18p	50	0	3245920.1	3430531.3	271106.0	2175130.0	17278944.0	
pop65p	50	0	509502.8	538932.4	11547.0	370495.0	2414250.0	
popurban	50	0	3328253.2	4090177.9	172735.0	2156905.0	21607606.0	
medage	37	0	29.5	1.7	24.2	29.8	34.7	
death	50	0	39474.3	41742.3	1604.0	26176.5	186428.0	
marriage	50	0	47701.4	45130.4	4437.0	36279.0	210864.0	
divorce	50	0	23679.4	25094.0	2142.0	17112.5	133541.0	

Exploring datasets with *modelsummary*

- *modelsummary* summarizes only numeric variables by default.
- To summarize categorical variables, use the argument `type`:

```
datasummary_skim(census, type = "categorical")
```

data	N	%
NE	9	18.0
N Cntrl	12	24.0
South	16	32.0
West	13	26.0

Exploring datasets with *modelsummary*

You can also customize the variables and statistics to include using a **formula** with the `datasummary()` function.

`datasummary()`

- **formula:** a two-sided formula to describe the table: rows ~ columns
- **data:** the data set to be summarized
- *output:* the type of output desired
- ...: additional options allow for formatting customization

```
datasummary(
  var1 + var2 + var3 ~ stat1 + stat2 + stat3 + stat4,
  data = data
)
```

Exploring datasets with *modelsummary*

Exercise ⌨

Create a table showing the number of observations, mean, standard deviation, minimum, maximum and median value for all the population, number of deaths, number of marriage and number of divorces in the `census` data.

```
datasummary(
  var1 + var2 + var3 ~ stat1 + stat2 + stat3 + stat4,
  data = data
)
```

> **Tip:** some of the allowed statistics are N, Mean, SD, Min, Max, Median, P0, P25, P50, P75, P100, Histogram

01:30

Exploring datasets with *modelsummary*

```
datasummary(
  pop + death + marriage + divorce ~ N + Mean + SD + Median + Min + Max,
  data = census
)
```

	N	Mean	SD	Median	Min	Max
pop	50	4518149.44	4715037.75	3066433.00	401851	23667902
death	50	39474.26	41742.35	26176.50	1604	186428
marriage	50	47701.40	45130.42	36279.00	4437	210864
divorce	50	23679.44	25094.01	17112.50	2142	133541

Exploring datasets with *modelsummary*

```
datasummary(
  All(census) ~ N + Mean + SD + Median + Min + Max,
  data = census
)
```

	N	Mean	SD	Median	Min	Max
pop	50	4518149.44	4715037.75	3066433.00	401851	23667902
poplt5	50	326277.78	331585.14	227467.50	35998	1708400
pop5_17	50	945951.60	959372.83	629654.00	91796	4680558
pop18p	50	3245920.06	3430531.31	2175130.00	271106	17278944
pop65p	50	509502.80	538932.38	370495.00	11547	2414250
popurban	50	3328253.18	4090177.93	2156905.00	172735	21607606
medage	50	29.54	1.69	29.75	24.20	34.70
death	50	39474.26	41742.35	26176.50	1604	186428
marriage	50	47701.40	45130.42	36279.00	4437	210864

Balance tables with *modelsummary*

```
census_rct <-
  census %>%
  mutate(
    treatment = as.numeric(runif(n()) > 0.5)
  ) %>%
  select(
    -starts_with("state")
  )

datasummary_balance(
  ~ treatment,
  data = census_rct
)
```

Balance tables with *modelsummary*

		0		1			
		Mean	**Std. Dev.**	**Mean**	**Std. Dev.**	**Diff. in Means**	**Std. Error**
pop		5757007.7	5646595.8	3176053.0	3015272.8	-2580954.8	1266940.1
poplt5		407722.2	404115.1	238046.3	202748.8	-169675.9	89408.7
pop5_17		1196229.4	1146488.1	674817.3	620154.1	-521412.1	258030.7
pop18p		4153056.1	4104355.7	2263189.3	2196259.6	-1889866.8	921354.7
pop65p		652173.7	634232.4	354942.6	365532.8	-297231.1	145046.3
popurban		4399925.8	5045363.6	2167274.5	2295045.8	-2232651.2	1094775.5
medage		30.2	1.5	28.8	1.7	-1.3	0.4
death		50559.7	48435.0	27465.0	29542.5	-23094.7	11251.4
marriage		62722.3	53703.1	31428.8	25942.3	-31293.5	11788.4
divorce		31411.7	31378.8	15302.8	11423.5	-16108.9	6580.9
		N	Pct.	N	Pct.		
region	NE	6	23.1	3	12.5		
	N Cntrl	6	23.1	6	25.0		
	South	10	38.5	6	25.0		
	West	4	15.4	9	37.5		

Exporting tables

Exporting *modelsummary* table to LaTeX

To export the tables we created, we can simply use the option `output`:

```
descriptives <-
  All(census) ~ N + Mean + SD + Median + Min + Max

datasummary(
  descriptives,
  data = census,
  output = here( # file path to output file
    "DataWork",
    "Output",
    "Raw",
    "summary-stats.tex"
  )
)
```

Exporting *modelsummary* table

Other valid output formats include:

- `.docx`
- `.pptx`
- `.html`
- `.md`

Exporting *modelsummary* table

Other valid output formats include:

- `.docx`
- `.pptx`
- `.html`
- `.md`
- … but not `.xls`

Exporting *modelsummary* table to Excel

- To export the table to Excel, we will first convert it into an object of type *huxtable*
- `huxtable` is another R package, one that allows not only for exporting tables, but also for extensive customization
- Before getting to the customization part, however, let's export this table:

```
# Create the huxtable object
summary_stats_table <-
  datasummary(
    descriptives,
    data = census,
    output = "huxtable"
  )

# Export it to Excel
quick_xlsx(
  summary_stats_table, # object to be exported
  file = here( # file path to output file
    "DataWork",
    "Output",
    "Raw",
    "summary-stats.xlsx"
  )
)
```

Exporting tables

A similar code can also export the same table to a self-standing LaTeX document

```
# Export to LaTeX
quick_latex(
  summary_stats_table,
  file = here(
    "DataWork",
    "Output",
    "Raw",
    "summary-stats.tex"
  )
)
```

Exporting tables to different Excel tabs

```
# Start a new workbook
wb <- createWorkbook()

# Add one sheet to it
wb <-
  as_Workbook(
    summary_stats_table,
    Workbook = wb,
    sheet = "Summary stats"
  )

# Add another sheet to it
wb <-
  as_Workbook(
    hux("Mock", "table"),
    Workbook = wb,
    sheet = "Other sheet"
  )

# Save the workbook
saveWorkbook(
  wb, # object to be saved
   file = here( # file path to output file
    "DataWork",
    "Output",
    "Raw",
    "summary-stats.xlsx"
  ),
  overwrite = TRUE # replace existing file
)
```

Exporting tables to LaTeX fragment

```
summary_stats_table %>%
  print_latex() %>% # See LaTeX code
  # Save LaTeX code
  capture.output(
    file = here(
      "DataWork",
      "Output",
      "Raw",
      "summary-stats.tex"
    )
  )
```

You will also need to load the required LaTeX packages. To copy the code that creates a preamble with all of them, use this code:

```
report_latex_dependencies()
```

Formatting tables

Beautifying tables

- `huxtable` also allows you to customize table formatting so it can be exported with the same layout to multiple software
- Before we do that, however, we will create a version of the data where the variable names are the Stata labels

```
# Extract variable labels from data frame
labels <- names(census)
names(labels) <- attributes(census)$var.labels

# Rename the variables
census_labelled <-
  census %>%
  rename(
    labels
  )

# Create a labelled summary table
summary_stats_table <-
  datasummary(
    All(census_labelled) ~ N + Mean + SD + Median + Min + Max,
    data = census_labelled,
    output = "huxtable"
  )
```

Beautifying tables

The code below shows the table `summary_stats_table` can be formatted

```
# Format table
 summary_stats_table %>%
   set_header_rows(1, TRUE) %>% # Use first row as table header
   set_header_cols(1, TRUE)  %>%  # Use first column as row header
   set_number_format(everywhere, 2:ncol(.), "%9.0f") %>% # Don't round large numbers
   set_align(1, everywhere, "center") %>% # Centralize cells in first row
   theme_basic() # Set a theme for quick formatting
```

	N	Mean	SD	Median	Min	Max
Population	50	4518149	4715038	3066433	401851	23667902
Pop, < 5 year	50	326278	331585	227468	35998	1708400
Pop, 5 to 17 years	50	945952	959373	629654	91796	4680558
Pop, 18 and older	50	3245920	3430531	2175130	271106	17278944
Pop, 65 and older	50	509503	538932	370495	11547	2414250
Urban population	50	3328253	4090178	2156905	172735	21607606
Median age	50	30	2	30	24	35
Number of deaths	50	39474	41742	26177	1604	186428

Export beautified tables

```
quick_xlsx(
  summary_stats_table,
  file = here(
    "DataWork",
    "Output",
    "Raw",
    "summary-stats-basic.xlsx"
  )
)

quick_latex(
  summary_stats_table,
  file = here(
    "DataWork",
    "Output",
    "Raw",
    "summary-stats-basic.tex"
  )
)
```

Before

	A	B	C	D	E	F
1	skim_varia	Mean	Median	SD	Min	Max
2	pop	4520000	3070000	4720000	402000	23700000
3	poplt5	326000	227000	332000	36000	1710000
4	pop5_17	946000	630000	959000	91800	4680000
5	pop18p	3250000	2180000	3430000	271000	17300000
6	pop65p	510000	370000	539000	11500	2410000
7	popurban	3330000	2160000	4090000	173000	21600000
8	medage	29.5	29.8	1.69	24.2	34.7
9	death	39500	26200	41700	1600	186000
10	marriage	47700	36300	45100	4440	211000
11	divorce	23700	17100	25100	2140	134000

After

	A	B	C	D	E	F
1	**Variable**	**Mean**	**Median**	**SD**	**Min**	**Max**
2	Population	4518149	3066433	4715038	401851	23667902
3	Pop, < 5 year	326278	227468	331585	35998	1708400
4	Pop, 5 to 17 years	945952	629654	959373	91796	4680558
5	Pop, 18 and older	3245920	2175130	3430531	271106	17278944
6	Pop, 65 and older	509503	370495	538932	11547	2414250
7	Urban population	3328253	2156905	4090178	172735	21607606
8	Median age	30	30	2	24	35
9	Number of deaths	39474	26177	41742	1604	186428
10	Number of marriages	47701	36279	45130	4437	210864
11	Number of divorces	23679	17113	25094	2142	133541

Other themes to play with

Aggregating observations

Aggregating observations

- If you want to show aggregated statistics, the function `summarise` is a powerful tool.
- It is similar to `skim` in that it calculates a series of statistics for a data frame.
- However, it does not have pre-defined statistics, so it requires more manual input.
- On the other hand, its output is a regular data frame, so it is also useful to create constructed data sets.
- Its Stata equivalent would be `collapse`

`summarise(.data, ...)`

- **data**: the data frame to be summarized
- **...**: Name-value pairs of summary functions. The name will be the name of the variable in the result.

The "name-value" pairs mentioned under `...` look like this: `new_variable = stat(existing_variable)`, where `stat` takes the same functions as `sfl`

Aggregating observations

```
region_stats <-
  census %>%
  group_by(region) %>%
  summarise(
    `Number of States` = n_distinct(state),
    `Total Population` = sum(pop)
  )
```

region	Number of States	Total Population
NE	9	49135283
N Cntrl	12	58865670
South	16	74734029
West	13	43172490

Aggregating observations

Exercise ⌨

Recreate the `region_stats` data set, now including the average and the standard deviation of the population.

01:30

Aggregating observations

```
region_stats <-
  census %>%
  group_by(region) %>%
  summarise(
    `Number of States` = n_distinct(state),
    `Total Population` = sum(pop),
    `Average Population` = mean(pop),
    `SD of Population` = sd(pop)
  )
```

region	Number of States	Total Population	Average Population	SD of Population
NE	9	49135283	5459476	5925235
N Cntrl	12	58865670	4905473	3750094
South	16	74734029	4670877	3277853
West	13	43172490	3320961	6217177

Aggregating observations

Exercise ⌨

Use `huxtable` to format and export the object `region_stats`.

02:00

Aggregating observations

```
region_stats_table <-
  region_stats %>%
  rename(Region = region) %>%
  as_hux %>%
  set_header_cols("Region", TRUE)  %>%
  theme_bright()

quick_xlsx(
  region_stats_table,
   file = here(
     "DataWork",
     "Output",
     "Raw",
     "region-stats.xlsx"
   )
)
```

Ok, can we run some regressions now?!

Running regressions

The base R command for linear regressions is called `lm`

lm(formula, data, subset, weights, ...)

- **formula:** an object of class "formula" containing a symbolic description of the model
- **data:** a data frame containing the variables indicated in the formula
- *subset:* an optional vector specifying a subset of observations to be used in the regression
- *weights:* an optional vector of weights to be used in the regression

Formulas can take three specifications:

- `y ~ x1 + x2` regresses variable `y` on covariates `x1` and `x2`
- `y ~ x1:x2` regresses variable `y` on the interaction of covariates `x1` and `x2`
- `y ~ x1*x2` is equivalent to `y ~ x1 + x2 + x1:x2`

Running regressions

Exercise ⌨

Using the `census` data, run a regression of the number of divorces on population, urban population and number of marriages.

```
lm(y ~ x1 + x2,
   data)
```

01:00

Running regressions

Exercise ⌨

Using the `census` data, run a regression of the number of divorces on population, urban population and number of marriages.

```
reg1 <-
  lm(
    divorce ~ pop + popurban + marriage,
    census
  )
```

Running regressions

- The output of regression commands is a list of relevant information.
- By default, it prints only a small portion of this information.
- The best way to visualize results is to store this list in an object and then access its contents using the function `summary`

Running regressions

```
reg1 <-
  lm(
    divorce ~ pop + popurban + marriage,
    census
  )

summary(reg1)
```

```
##
## Call:
## lm(formula = divorce ~ pop + popurban + marriage, data = census)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -22892.3  -1665.1    796.5   4138.0  17212.2
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.207e+02  1.838e+03   0.066    0.948
## pop         1.044e-03  1.633e-03   0.639    0.526
## popurban    1.954e-03  1.796e-03   1.088    0.282
## marriage    2.587e-01  5.958e-02   4.342  7.7e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7466 on 46 degrees of freedom
## Multiple R-squared:  0.9169,    Adjusted R-squared:  0.9115
## F-statistic: 169.2 on 3 and 46 DF,  p-value: < 2.2e-16
```

Running regressions

The `lfe` command `felm` allows for more flexibility in model specification

felm(formula, data, subset, weights, ...)

- **formula:** an object of class "formula" containing a symbolic description of the model
- **data:** a data frame containing the variables indicated in the formula
- *subset:* an optional vector specifying a subset of observations to be used in the regression
- *weights:* an optional vector of weights to be used in the regression

Formulas for `felm` are more complex, and take the following format: `y ~ x1 + x2 | fe1 + fe2 | (Q|W ~ iv3+iv4) | clu1 + clu2`

- `y ~ x1 + x2` takes all the same formulas as `lm`
- `fe1 + fe2` list the variables to be included as fixed effects
- `(Q|W ~ iv3 + iv4)` uses instruments `iv3` and `iv4` for variables `Q` and `W`
- `clu1 + clu2` indicates that standard errors should be clustered using variables `clu1` and `clu2`

Running regressions

Exercise ⌨

Using the `census` data, run a regression of the number of divorces on population, urban population and number of marriages controlling for region fixed effects.

```
felm(
  y ~ x1 + x2 | fe1 + fe2 | 0 | 0,
  data
)
```

01:00

Running regressions

Exercise ⌨

Using the `census` data, run a regression of divorce on population, urban population and number of marriages controlling for region fixed effects.

```
reg2 <-
  felm(
    divorce ~ pop + popurban + marriage | region | 0 | 0,
    census
  )

summary(reg2)
```

```
##
## Call:
##    felm(formula = divorce ~ pop + popurban + marriage | region |      0 | 0, data = census)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

Running regressions

```
reg2 <-
  felm(
    divorce ~ pop + popurban + marriage | region | 0 | 0,
    census
  )

summary(reg2)
```

```
##
## Call:
##    felm(formula = divorce ~ pop + popurban + marriage | region |      0 | 0, data = census)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -17919  -3112   -448   3047  13830
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## pop       0.0003951  0.0017881   0.221  0.82615
## popurban 0.0035532  0.0019981   1.778  0.08243 .
## marriage 0.1836593  0.0580271   3.165  0.00285 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6748 on 43 degrees of freedom
## Multiple R-squared(full model): 0.9365   Adjusted R-squared: 0.9277
## Multiple R-squared(proj model): 0.9354   Adjusted R-squared: 0.9264
## F-statistic(full model):105.8 on 6 and 43 DF, p-value: < 2.2e-16
## F-statistic(proj model): 207.7 on 3 and 43 DF, p-value: < 2.2e-16
```

Some notes on regressions

- Whenever a factor is included in the list of covariates, it is treated as a categorical variable, i.e., as if you had written `i.x` in Stata.
- Whenever a boolean is included in the list of covariates, it is treated as a dummy variable, where `TRUE` is `1` and `FALSE` is `0`.
- `felm` also allows for bootstrapping, but this is beyong the scope of this session.

Exporting regression tables

Exporting regression tables

`huxtable` also has a quick wrapper for regression tables

huxreg(...)

- **...**: Models, or a single list of models. Names will be used as column headings.
- *number_format:* Format for numbering. See number_format() for details.
- *stars:* Levels for p value stars.
- *bold_signif:* Where p values are below this number, cells will be displayed in bold.
- *note:* Footnote for bottom cell, which spans all columns.
- *statistics:* A vector of summary statistics to display.
- *coefs:* A vector of coefficients to display. To change display names, name the coef vector: c("Displayed title" = "coefficient_name", ...)

Exporting regression tables

```
huxreg(reg1, reg2)
```

	(1)	(2)
(Intercept)	120.730	
	(1838.216)	
pop	0.001	0.000
	(0.002)	(0.002)
popurban	0.002	0.004
	(0.002)	(0.002)
marriage	0.259 ***	0.184 **
	(0.060)	(0.058)
N	50	50
R2	0.917	0.937
logLik	-514.766	
AIC	1039.531	
*** p < 0.001; ** p < 0.01; * p < 0.05.		

Formatting regression tables

```
huxreg(
  reg1, reg2,
   coefs = c(
     "Population" = "pop", # Show variable labels instead of names
     "Urban population"  = "popurban",
     "Number of marriages" = "marriage"
   ),
   statistics = c("N. obs." = "nobs")) %>%
   add_rows(
     c("Region FE", "No", "Yes"),
     after = 7
   )
```

]

Formatting regression tables

	(1)	(2)
Population	0.001	0.000
	(0.002)	(0.002)
Urban population	0.002	0.004
	(0.002)	(0.002)
Number of marriages	0.259 ***	0.184 **
	(0.060)	(0.058)
Region FE	No	Yes
N. obs.	50	50

*** p < 0.001; ** p < 0.01; * p < 0.05.

References and recommendations

- Econometrics with R https://www.econometrics-with-r.org/index.html
- `modelsummary` documentation: https://vincentarelbundock.github.io/modelsummary/index.html
- Introduction to `huxtable`: https://cran.r-project.org/web/packages/huxtable/vignettes/huxtable.html
- Using `huxtable` for regression tables: https://cran.r-project.org/web/packages/huxtable/vignettes/huxreg.html
- Johns Hopkins Exploratory Data Analysis at Coursera: https://www.coursera.org/learn/exploratory-data-analysis
- Udacity's Data Analysis with R: https://www.udacity.com/course/data-analysis-with-r--ud651

Since we talked about LaTeX so much...

- DIME LaTeX templates and trainings: https://github.com/worldbank/DIME-LaTeX-Templates
- All you need to know about LaTeX: https://en.wikibooks.org/wiki/LaTeX

Thank you!