Descriptive Analysis

R for Data Analysis

DIME Analytics
The World Bank | WB Github
April 2025

Introduction

Initial Setup

If You Attended Session 2 | If You Did Not Attend Session 2

1. Copy/paste the following code into a new RStudio script:

```
install.packages("usethis")
library(usethis)
usethis::use_zip(
    "https://github.com/worldbank/dime-r-training/archive/main.zip",
    cleanup = TRUE
)
```

2. A new RStudio environment will open. Use this for the session today.

Table of contents

Workflows for outputs, reports, and papers

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("modelsummary") # to export easy descriptive tables
install.packages("fixest")       # easy fixed effects regressions
install.packages("huxtable")     # easy regression tables
install.packages("openxlsx")     # export tables to Excel format
install.packages("estimatr")     # backend calculations for balance tables
```

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

Setting the stage

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

Tip: Use `here`, as we saw in the data wrangling session.

```
# 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", "Colorado", "Connecticut", "Delaware", "Flo
## $ state2   <chr> "AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "FL", "GA", "HI", "ID", "IL", "IN", "IA", "KS", "KY"
## $ region   <fct> South, West, West, South, West, West, NE, South, South, South, West, West, N Cntrl, N Cntrl, N Cntrl
## $ pop      <int> 3893888, 401851, 2718215, 2286435, 23667902, 2889964, 3107576, 594338, 9746324, 5463105, 964691, 943
## $ poplt5   <int> 296412, 38949, 213883, 175592, 1708400, 216495, 185188, 41151, 570224, 414935, 77848, 93531, 842241,
## $ pop5_17  <int> 865836, 91796, 577604, 495782, 4680558, 592318, 637731, 125444, 1789412, 1231195, 197735, 213134, 240
## $ pop18p   <int> 2731640, 271106, 1926728, 1615061, 17278944, 2081151, 2284657, 427743, 7386688, 3816975, 689108, 6372
## $ pop65p   <int> 440015, 11547, 307362, 312477, 2414250, 247325, 364864, 59179, 1687573, 516731, 76150, 93680, 126188
## $ popurban <int> 2337713, 258567, 2278728, 1179556, 21607606, 2329869, 2449774, 419819, 8212385, 3409081, 834592, 5097
## $ medage   <dbl> 29.3, 26.1, 29.2, 30.6, 29.9, 28.6, 32.0, 29.8, 34.7, 28.7, 28.4, 27.6, 29.9, 29.2, 30.0, 30.1, 29.1
## $ death    <int> 35305, 1604, 21226, 22676, 186428, 18925, 26005, 5123, 104190, 44230, 4849, 6753, 102230, 47300, 2634
## $ marriage <int> 49018, 5361, 30223, 26513, 210864, 34917, 26048, 4437, 108344, 70638, 11856, 13428, 109823, 57853, 27
## $ divorce  <int> 26745, 3517, 19908, 15882, 133541, 18571, 13488, 2313, 71579, 34743, 4438, 6596, 50997, 40006, 11854
```

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 1 ⌨

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

00:45

Exploring a dataset

```
summary(census)
```

```
##     state              state2               region       pop               poplt5           pop5_17           pop18p            pop65p           popurban    
##  Length:50          Length:50          NE      : 9   Min.   :  401851   Min.   :  35998   Min.   :  91796   Min.   :  271106   Min.   :  11547   Min.   :  172735  
##  Class :character   Class :character   N Cntrl:12   1st Qu.: 1169218   1st Qu.:  98831   1st Qu.: 257949   1st Qu.:  823702   1st Qu.: 118660   1st Qu.:  826651  
##  Mode  :character   Mode  :character   South  :16   Median : 3066433   Median : 227468   Median : 629654   Median : 2175130   Median : 370495   Median : 2156905  
##                                        West   :13   Mean   : 4518149   Mean   : 326278   Mean   : 945952   Mean   : 3245920   Mean   : 509503   Mean   : 3328253  
##                                                     3rd Qu.: 5434033   3rd Qu.: 361321   3rd Qu.:1143292   3rd Qu.: 3858173   3rd Qu.: 580087   3rd Qu.: 3403450  
##                                                     Max.   :23667902   Max.   :1708400   Max.   :4680558   Max.   :17278944   Max.   :2414250   Max.   :21607606  
##      medage          death           marriage         divorce      
##  Min.   :24.20   Min.   :  1604   Min.   :  4437   Min.   :  2142  
##  1st Qu.:28.73   1st Qu.:  9087   1st Qu.: 14840   1st Qu.:  6898  
##  Median :29.75   Median : 26177   Median : 36279   Median : 17113  
##  Mean   :29.54   Mean   : 39474   Mean   : 47701   Mean   : 23679  
##  3rd Qu.:30.20   3rd Qu.: 46533   3rd Qu.: 57338   3rd Qu.: 27987  
##  Max.   :34.70   Max.   :186428   Max.   :210864   Max.   :133541
```

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 2 ⌨

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

00:45

Summarizing continuous variables

Exercise 2 ⌨

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

`tabyl(x, ...)`

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

- **x:** the object you want to summarize, usually a vector or data frame
- ... additional options as show_na, or show_missing_levels.

Exercise 3 ⌨

Use the `tabyl()` 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

```
census %>%
  tabyl(region)
```

region	n	percent
NE	9	0.18
N Cntrl	12	0.24
South	16	0.32
West	13	0.26

Summarizing categorical variables

Two way tabulation

```
census %>%
  tabyl(state, region)
```

state	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

Descriptives tables

Descriptives tables

What if you want to...

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

Well, then you will need a few more packages

- 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, output, ....)`

- **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 4 ⌨

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

00:45

Exploring datasets with *modelsummary*

```
datasummary_skim(census)
```

	Unique	Missing Pct.	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 all variables by default.
- To summarize only categorical variables, use the argument `type`

```
datasummary_skim(census %>% select(region), type = "categorical")
```

region	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, data, output, ...)`

- **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 5 ⌨

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

> **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	401851.00	23667902.00
death	50	39474.26	41742.35	26176.50	1604.00	186428.00
marriage	50	47701.40	45130.42	36279.00	4437.00	210864.00
divorce	50	23679.44	25094.01	17112.50	2142.00	133541.00

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	401851.00	23667902.00
poplt5	50	326277.78	331585.14	227467.50	35998.00	1708400.00
pop5_17	50	945951.60	959372.83	629654.00	91796.00	4680558.00
pop18p	50	3245920.06	3430531.31	2175130.00	271106.00	17278944.00
pop65p	50	509502.80	538932.38	370495.00	11547.00	2414250.00
popurban	50	3328253.18	4090177.93	2156905.00	172735.00	21607606.00
medage	50	29.54	1.69	29.75	24.20	34.70
death	50	39474.26	41742.35	26176.50	1604.00	186428.00
marriage	50	47701.40	45130.42	36279.00	4437.00	210864.00
divorce	50	23679.44	25094.01	17112.50	2142.00	133541.00

Balance tables with *modelsummary*

```
# Creating a toy "treatment" variable
census_rct <-
  census %>%
  mutate(
    treatment = as.numeric(runif(n()) > 0.5)
  ) %>%
  select(
    -c(state, state2, region)
  )

# Balance table
datasummary_balance(
  ~ treatment,
  data = census_rct
)
```

Balance tables with *modelsummary*

	0		1			
	Mean	Std. Dev.	Mean	Std. Dev.	Diff. in Means	Std. Error
pop	4098477.0	3389446.4	4822050.1	5520104.8	723573.1	1264044.1
poplt5	292420.0	217669.3	350795.5	396339.6	58375.6	87595.2
pop5_17	846999.3	678908.6	1017606.7	1126412.7	170607.4	256320.8
pop18p	2959057.8	2499648.7	3453647.9	4003187.9	494590.1	922030.0
pop65p	495352.0	469935.8	519749.9	591846.8	24397.9	150316.0
popurban	2937620.9	2777957.9	3611124.8	4854608.6	673503.9	1086343.2
medage	29.8	2.1	29.4	1.4	-0.4	0.5
death	37467.6	34466.4	40927.4	46856.3	3459.8	11501.1
marriage	46564.4	34246.4	48524.8	52200.8	1960.4	12239.7
divorce	22748.0	16681.3	24353.9	30035.1	1605.9	6660.2

Exporting tables

Exporting *modelsummary* table to LaTeX

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

```
# Saving the formula into an object
descriptives <-
  All(census) ~ N + Mean + SD + Median + Min + Max

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

If you an error message saying `Assertion on 'output' failed: Path to file (dirname) does not exist`, create the folder `Output` and subfolder `Raw` in `DataWork`.

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` or `.xlsx`

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-huxtable.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-huxtable.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-multiple-sheets.xlsx"
  ),
  overwrite = TRUE # replace if the file exists
)
```

Exporting tables to different Excel tabs

	A	B	C	D	E	F	G	H
1		N	Mean	SD	Median	Min	Max	
2	pop	50	4518149.44	4715037.75	3066433	401851	23667902	
3	poplt5	50	326277.78	331585.14	227467.5	35998	1708400	
4	pop5_17	50	945951.6	959372.83	629654	91796	4680558	
5	pop18p	50	3245920.06	3430531.31	2175130	271106	17278944	
6	pop65p	50	509502.8	538932.38	370495	11547	2414250	
7	popurban	50	3328253.18	4090177.93	2156905	172735	21607606	
8	medage	50	29.54	1.69	29.75	24.2	34.7	
9	death	50	39474.26	41742.35	26176.5	1604	186428	
10	marriage	50	47701.4	45130.42	36279	4437	210864	
11	divorce	50	23679.44	25094.01	17112.5	2142	133541	
12								
13								
14								
15								

Summary stats | Other sheet | +

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(
    all_of(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 %>%
 # Don't round large numbers
 set_number_format(
   row = everywhere,
   col = 2:ncol(.),
   value = "%9.0f"
   ) %>%
 # Centralize cells in first row
 set_align(1, everywhere, "center") %>%
 # Set a theme for quick formatting
 theme_basic()
```

	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
Number of marriages	50	47701	45130	36279	4437	210864
Number of divorces	50	23679	25094	17113	2142	133541

Export beautified tables

```
# Format table
summary_stats_table %>%
 set_number_format(
   row = everywhere,
   col = 2:ncol(.),
   value = "%9.0f"
   ) %>%
 set_align(1, everywhere, "center") %>%
 theme_basic()

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

Export beautified tables

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	G
1		**N**	**Mean**	**SD**	**Median**	**Min**	**Max**
2	Population	50	4518149	4715038	3066433	401851	23667902
3	Pop, < 5 year	50	326278	331585	227468	35998	1708400
4	Pop, 5 to 17 years	50	945952	959373	629654	91796	4680558
5	Pop, 18 and older	50	3245920	3430531	2175130	271106	17278944
6	Pop, 65 and older	50	509503	538932	370495	11547	2414250
7	Urban population	50	3328253	4090178	2156905	172735	21607606
8	Median age	50	30	2	30	24	35
9	Number of deaths	50	39474	41742	26177	1604	186428
10	Number of marriages	50	47701	45130	36279	4437	210864
11	Number of divorces	50	23679	25094	17113	2142	133541

Other themes to play with

jams

Type	Price	Sugar content
Strawberry	1.90	40.00%
Raspberry	2.10	35.00%
Plum	1.80	50.00%

theme_plain

Type	Price	Sugar content
Strawberry	1.90	40.00%
Raspberry	2.10	35.00%
Plum	1.80	50.00%

theme_basic

Type	Price	Sugar content
Strawberry	1.90	40.00%
Raspberry	2.10	35.00%
Plum	1.80	50.00%

theme_compact

Type	Price	Sugar content
Strawberry	1.90	40.00%
Raspberry	2.10	35.00%
Plum	1.80	50.00%

theme_article

Type	Price	Sugar content
Strawberry	1.90	40.00%
Raspberry	2.10	35.00%
Plum	1.80	50.00%

theme_bright

Type	Price	Sugar content
Strawberry	1.90	40.00%
Raspberry	2.10	35.00%
Plum	1.80	50.00%

theme_grey

Type	Price	Sugar content
Strawberry	1.90	40.00%
Raspberry	2.10	35.00%
Plum	1.80	50.00%

theme_blue

Type	Price	Sugar content
Strawberry	1.90	40.00%
Raspberry	2.10	35.00%
Plum	1.80	50.00%

theme_green

Type	Price	Sugar content
Strawberry	1.90	40.00%
Raspberry	2.10	35.00%
Plum	1.80	50.00%

theme_mondrian

Type	Price	Sugar content
Strawberry	1.90	40.00%
Raspberry	2.10	35.00%
Plum	1.80	50.00%

theme_orange

Type	Price	Sugar content
Strawberry	1.90	40.00%
Raspberry	2.10	35.00%
Plum	1.80	50.00%

theme_striped

Type	Price	Sugar content
Strawberry	1.90	40.00%
Raspberry	2.10	35.00%
Plum	1.80	50.00%

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 6 ⌨

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 `feols` command from package `fixest` allows for more flexibility in model specification

feols(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
- *vcov:* one of "iid", "hetero" (or "HC1"), "cluster", "twoway", "NW" (or "newey_west"), "DK" (or "driscoll_kraay"), or "conley"
- *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
- *cluster:* a list of vectors, a character vector of variable names, a formula or an integer vector specifying how to cluster standard errors
- ...

Running regressions

Formulas for `feols` are more complex, and take the following format: `y ~ x1 + x2 | fe1 + fe2 | x3 ~ iv3`

- `y ~ x1 + x2` takes all the same formulas as `lm`
- `fe1 + fe2` list the variables to be included as fixed effects
- `x3 ~ iv3` uses instrument `iv3` for variable `x3`

Running regressions

Exercise 7 ⌨

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.

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

01:00

Running regressions

Exercise 7 ⌨

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 and using standard errors clustered by state.

```
reg2 <-
  feols(
    divorce ~ pop + popurban + marriage | region,
    census,
    vcov = cluster ~ state # this defines clustered std errors by state
  )

summary(reg2)
```

Running regressions

```
reg2 <-
  feols(
    divorce ~ pop + popurban + marriage | region,
    census,
    vcov = cluster ~ state
  )

summary(reg2)
```

```
## OLS estimation, Dep. Var.: divorce
## Observations: 50
## Fixed-effects: region: 4
## Standard-errors: Clustered (state)
##          Estimate Std. Error  t value Pr(>|t|)
## pop      0.000395   0.002120 0.186405 0.852897
## popurban 0.003553   0.002278 1.559847 0.125232
## marriage 0.183659   0.104001 1.765939 0.083636 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 6,257.6     Adj. R2: 0.927695
##                 Within R2: 0.935434
```

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`.

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)	(0.104)
N	50	50
R2	0.917	0.937
logLik	-514.766	-508.024
AIC	1039.531	1030.048
*** p < 0.001; ** p < 0.01; * p < 0.05.		

Formatting regression tables

```
huxreg(
  'Model 1' = reg1,
  'Model 2' = reg2,
  coefs = c(
    "Population" = "pop", # Show variable labels instead of names
    "Urban population"  = "popurban",
    "Number of marriages" = "marriage"
  ),
  statistics = c("N. obs." = "nobs"),
  stars = c(`***` = 0.01, `**` = 0.05, `*` = 0.1),
  note = "{stars}\nStandard errors are displayed in parentheses."
) %>%
  add_rows(
    c("Region FE", "No", "Yes"),
    after = 7
  )
```

	Model 1	Model 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)	(0.104)
Region FE	No	Yes
N. obs.	50	50

*** p < 0.01; ** p < 0.05; * p < 0.1
Standard errors are displayed in parentheses.

Exporting regression tables

You can also display other types of computed values with `error_format()`. See the examples below for t-statistics and p-values.

```
huxreg(
  'Model 1' = reg1,
  'Model 2' = reg2,
  error_format = "[{statistic}]",
  # to display t-statistics in brackets
  coefs = c(
    "Population" = "pop",
    "Urban population"  = "popurban",
    "Number of marriages" = "marriage"
  ),
  statistics = c("N. obs." = "nobs")
) %>%
  add_rows(
    c("Region FE", "No", "Yes"),
    after = 7
  )
```

```
huxreg(
  'Model 1' = reg1,
  'Model 2' = reg2,
  error_format = "[{p.value}]",
  # to display p-values in brackets
  coefs = c(
    "Population" = "pop",
    "Urban population"  = "popurban",
    "Number of marriages" = "marriage"
  ),
  statistics = c("N. obs." = "nobs")
) %>%
  add_rows(
    c("Region FE", "No", "Yes"),
    after = 7
  )
```

Exporting regression tables

Exercise 8 ⌨

Export a regression table with the results of your estimations using `lm` and `feols`:

- Use `huxreg` to combine `reg1` and `reg2`.
- Use `quick_xlsx` or `quick_latex` to export the output of `huxreg` to your preferred format.

```
# Combine regression results with huxreg
reg_table <- huxreg(reg1, reg2)

# Export to Excel
quick_xlsx(reg_table,  file = here(
                                "DataWork",
                                "Output",
                                "Raw",
                                "regression_table.xlsx"
```

```
# Or, export to LaTeX
quick_latex(reg_table, file = here(
                                "DataWork",
                                "Output",
                                "Raw",
                                "regression_table.tex")
```

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
- Sample code for tables in R: https://github.com/RRMaximiliano/r-latex-tables-sum-stats
- More sample code for tables in R: https://evalsp20.classes.andrewheiss.com/reference/regtables/
- 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!

Appendix

Appendix - Aggregating observations

Appendix - Aggregating observations

- If you want to show aggregated statistics, the function `summarise` is a powerful tool.
- It is similar to `datasummary` 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`

Appendix - Aggregating observations

`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 = function(existing_variable)`, where possible functions include:

- Center: `mean()`, `median()`
- Spread: `sd()`, `IQR()`, `mad()`
- Range: `min()`, `max()`, `quantile()`
- Count: `n()`, `n_distinct()`

Appendix - 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

Appendix - Aggregating observations

Exercise 9 ⌨

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

01:30

Appendix - 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

Appendix - Aggregating observations

Exercise 9 ⌨

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

02:00

Appendix - 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"
   )
)
```

Appendix - Regression tables with Stargazer

Appendix - Exporting regression tables with Stargazer

- If you need to export regression tables into latex, there is hardly a best option than `stargazer`
- The package `stargazer` uses a command of the with the same name, `stargazer()`, to export beautifully formatted regression tables
- Unfortunately, it doesn't have options to export to Excel. Another type of format it exports is HTML
- See the next slide and check how each argument of `stargazer()` formats the table output

Appendix - Complete latex regression table using

```r
# install.packages("stargazer") # install if needed
library(stargazer)
reg1 <- lm(mpg ~ wt + hp, data = mtcars)
reg2 <- lm(mpg ~ wt + hp + factor(gear), data = mtcars)
reg3 <- lm(qsec ~ wt + hp, data = mtcars)
reg4 <- lm(qsec ~ wt + hp + factor(gear), data = mtcars)

stargazer(reg1,
          reg2,
          reg3,
          reg4,
          title = "Best table ever",
          keep = c('wt', 'hp'),
          covariate.labels = c('Weight',
                               'Horsepower'),
          dep.var.labels = c('Miles per Gallon',
                             '1/4 Mile Time'),
          dep.var.caption = '',
          add.lines = list(c('N Gears FE', 'No', 'Yes', 'No', 'Yes')),
          keep.stat = c('n', 'adj.rsq'),
          header = FALSE,
          notes = 'Standard errors in parentheses')
```

Table 1: Best table ever

	Miles per Gallon		1/4 Mile Time	
	(1)	(2)	(3)	(4)
Weight	−3.878***	−3.239***	0.942***	0.747*
	(0.633)	(0.878)	(0.266)	(0.371)
Horsepower	−0.032***	−0.035***	−0.027***	−0.023***
	(0.009)	(0.013)	(0.004)	(0.005)
N Gears FE	No	Yes	No	Yes
Observations	32	32	32	32
Adjusted R^2	0.815	0.811	0.628	0.616

Note: *p<0.1; **p<0.05; ***p<0.01
Standard errors in parentheses