POLI 30 D: Political Inquiry

TA Sessions

Lab 08 | R Plots and R Data Analysis IV

Before we start

Announcements:

- GitHub page: **https://github.com/umbertomig/POLI30Dpublic**
- Piazza forum: The link in the slides needs to be fixed. Check with instructors for an alternative link.

Before we start

Announcements: Final Exam

- The best way to study for the **final exam** is to:
  1. Revise the lectures' content from lecture one until the last lecture. All will be on.
  2. Make sure you understand how to run the code and how to interpret results.
  3. Revise the content from the homework. They are a good clue regarding the format of the exam.
  4. If you cannot do it, then explain with words how you would do it. Explain in detail.
     - This helps us to give you partial credit.

Before we start

Recap: In the Lab sessions so far, you learned:

- How to install R and R Studio on your computer.
- How to do basic and advanced operations with vectors and data frames.
- How to install packages and work with R Markdown.
- How to create plots and how to do data analysis.

Great job!

- Do you have any questions about these contents?

Plan for Lab 08

- Group-by and summarize
- A bit more recoding
- Dealing with missing data
- Extract random samples from data
- Playing with random variables
- Full summary of a regression

Getting started

Getting started

- To get started, we need to load the datasets we will need in the lab.
- We also need to load the `tidyverse` package, which has all the R functions we use.

```
library(tidyverse)
## -- Attaching packages --------------------------------------- tidyverse 1.3.
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts(
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Getting started - Education expenditure data

```r
educexp <- read.csv("https://raw.githubusercontent.com/umbertomig/POLI30Dpublic
head(educexp)
##   education income young urban states
## 1       189   2824 350.7   508     ME
## 2       169   3259 345.9   564     NH
## 3       230   3072 348.5   322     VT
## 4       168   3835 335.3   846     MA
## 5       180   3549 327.1   871     RI
## 6       193   4256 341.0   774     CT
```

Getting started - Chile survey data

```
chile <- read.csv("https://raw.githubusercontent.com/umbertomig/POLI30Dpublic/m
head(chile)
##   region population sex age education income statusquo vote   logpop    logi
## 1      N     175000   M  65         P  35000   1.00820    Y 12.07254 10.4631
## 2      N     175000   M  29        PS   7500  -1.29617    N 12.07254  8.9226
## 3      N     175000   F  38         P  15000   1.23072    Y 12.07254  9.6158
## 4      N     175000   F  49         P  35000  -1.03163    N 12.07254 10.4631
## 5      N     175000   F  23         S  35000  -1.10496    N 12.07254 10.4631
## 6      N     175000   F  28         P   7500  -1.04685    N 12.07254  8.9226
```

Getting started - Voting

```
voting <- read.csv("https://raw.githubusercontent.com/umbertomig/POLI30Dpublic/
head(voting)
##   birth message voted
## 1  1981      no     0
## 2  1959      no     1
## 3  1956      no     1
## 4  1939     yes     1
## 5  1968      no     0
## 6  1967      no     0
```

Group-by and Summarize

Group-by and Summarize

- Suppose we want to find the average age by region.
- This is pretty straightforward when using group-by and summarize:
  - First, we group our results by the `region`
  - Then, we summarize the `age`
- It will create one average (or whatever stat we ask for) for each region group.
- Syntax:

```
dat %>% group_by(groupvar) %>%

  summarize(stat1 = calcs(vars1), etc)
```

Group-by and Summarize

- What operations are available?

Function	**Operation**
`first()`	First value of a vector
`last()`	Last value of a vector
`nth()`	Nth value of a vector
`n()`	Number of values in a vector
`n_distinct()`	Number of distinct values in a vector
`min()`	Minimum value in a vector
`max()`	Maximum value in a vector
`mean()`	Mean of a vector
`median()`	Median of a vector
`var()`	Variance of a vector
`sd()`	Standard deviation of a vector

Group-by and Summarize

- Example: Find the maximum age by region in the Chile Survey data.

```
chile %>% group_by(region) %>%
  summarize(maximumage = max(age))
## # A tibble: 5 x 2
##   region maximumage
##   <chr>       <int>
## 1 C              70
## 2 M              68
## 3 N              70
## 4 S              70
## 5 SA             NA
```

- Note the `NA`. We are going to learn how to deal with those today.

Group-by and Summarize

- Suppose you want to check the vote of the oldest person in each region.

```
chile %>% group_by(region) %>% arrange(desc(age)) %>%
  summarize(voteoldest = first(vote),
            ageoldest = first(age))
## # A tibble: 5 x 3
##   region voteoldest ageoldest
##   <chr>  <chr>          <int>
## 1 C      Y                 70
## 2 M      U                 68
## 3 N      Y                 70
## 4 S      U                 70
## 5 SA     Y                 70
```

- Y stands for a vote for Pinochet, and U stands for undecided.

- **Question**: How about the vote of the youngest person in each of the regions?

Recoding variables

Recoding variables

- Create a binary variable that is one when the person has PS schooling (some college or more).

```
chile2 <- chile %>%
  mutate(postsec = ifelse(education == 'PS', 1, 0))
chile2 %>% select(education, postsec) %>% head(4)
##   education postsec
## 1         P       0
## 2        PS       1
## 3         P       0
## 4         P       0
```

- When using `ifelse`, the syntax is:

```
    ifelse(test, val_if_T, val_if_F)
```

- Note that we work with vectors in the logical test.

Recoding variables

- Create a binary variable that is one when the person is older than 40.

```
chile3 <- chile %>%
  mutate(olderthan40 = ifelse(age > 40, 1, 0))
chile3 %>% select(age, olderthan40) %>% head(4)
##   age olderthan40
## 1  65           1
## 2  29           0
## 3  38           0
## 4  49           1
```

- **Your turn:** What is the proportion of people over 40 years old by region?

Recoding variables

- Let us say you want to create a binary indicator of whether the person lives in the `North` or `Central` regions.

- `ifelse()` with `%in%` gets this done:

```r
chile4 <- chile %>%
  mutate(NCind = ifelse(region %in% c('N', 'C'), 1, 0))
chile4 %>% count(NCind)
##   NCind    n
## 1     0 1778
## 2     1  922
```

- Note the `%in%` operator. This is the operator of choice when we make two or more comparisons!

- **Your turn:** Adapt the code above to add also the South region.

Recoding variables

- Suppose we want to recode a continuous variable into a discrete one.
- Suppose we want to recode age into three groups:
  - From youngest to 30 years old
  - From 31 to 60 years old
  - Older than 60 years old
- To do that, we use the function `cut`. We do three things:
  - `labels = c('lab1', 'lab2, ...)`
  - `breaks = c(-Inf, break1, break2, ..., Inf)`
  - `right = (T or F)`: Add the right break to it?!

Recoding variables

- It is simple to use:

```r
chile5 <- chile %>%
  mutate(agecat =
           cut(age, breaks = c(-Inf, 30, 60, Inf),
               labels = c('<= 30', '31 to 60', '> 60'),
               right = T))
chile5 %>% count(agecat)
##      agecat    n
## 1     <= 30 1002
## 2 31 to 60 1425
## 3      > 60  272
## 4      <NA>    1
```

- Note `-Inf` and `Inf`. They stand for $-\infty$ and ∞ ☺

Missing Values

Missing Values

- Note this `NA` that keeps popping up. This is the way we tell R we have missing data.
- Missing data stands for data that we have no idea is
  - It could be errors in typing up the data
  - It could be that you do not know
  - It could be that a respondent stopped the interview
  - It could be that the person refused to answer a question
- For all these reasons, the key is to understand that it may mess up our analysis.

Missing Data

```
chile %>%
  group_by(region) %>%
  summarize(avgsquo = mean(statusquo), nobs = n())
## # A tibble: 5 x 3
##   region avgsquo  nobs
##   <chr>    <dbl> <int>
## 1 C       NA       600
## 2 M        0.287   100
## 3 N        0.136   322
## 4 S       NA       718
## 5 SA      NA       960
```

- Note the `NA`s. These are the missing values.

Missing Data

- Most functions in R have a way to deal with it. In `mean`, we add the `na.rm = TRUE` to fix:

```
chile %>%
  group_by(region) %>%
  summarize(avgsquo = mean(statusquo, na.rm = T), nobs = n())
## # A tibble: 5 x 3
##   region avgsquo  nobs
##   <chr>    <dbl> <int>
## 1 C      -0.0298   600
## 2 M       0.287    100
## 3 N       0.136    322
## 4 S       0.165    718
## 5 SA     -0.180    960
```

Missing Data

- We can remove missing using `na.omit`. It removes the missing values and returns a *clean* dataset.

```
chile %>% select(region, statusquo) %>%
  na.omit() %>% group_by(region) %>%
  summarize(avgsquo = mean(statusquo), nobs = n())
## # A tibble: 5 x 3
##   region avgsquo  nobs
##   <chr>    <dbl> <int>
## 1 C      -0.0298   597
## 2 M       0.287    100
## 3 N       0.136    322
## 4 S       0.165    709
## 5 SA     -0.180    955
```

- The complete dataset has 2700 observations.
- After removing the missing in `region` and `statusquo`, it has 2683 observations.

Missing Data

- When you use `na.omit`, you end up with a smaller dataset.
    - This is the way to go if you do not need the removed cases.
- When you use `na.rm = T`, the dataset remains the same:
    - Good, since missingness can be different in different variables.
    - But sometimes, the function has a different pattern. This works for `mean`, but not for `cor`.
    - It works most of the time, though.

Extracting random samples from data

Random samples

- Sometimes, we need to extract random samples from a dataset.
- Examples:
    - Suppose you have the Census in your computer and want to extract random people to survey.
    - Suppose you have a dataset of all students and want to extract a representative sample to run a survey.
    - Suppose you are working with a large dataset, but your computer is old.
- In all these situations, you may extract a sample from your data and work with this sample.

Random samples

- Extract a 10% sample of the data, **without** replacement.

```
set.seed(123456) # change here for a different result
educexp10pct <-
  educexp %>% sample_frac(0.1, replace = F)
educexp10pct %>% head()
##   education income young urban states
## 1       192   3340 358.1   785     CO
## 2       212   3513 382.9   831     HI
## 3       273   3968 348.4   909     CA
## 4       261   4151 326.2   856     NY
## 5       201   2790 412.4   804     UT
educexp10pct %>% dim()
## [1] 5 5
```

- As you can see, the sample has 10% of the cases or 5 cases.

Random samples

- Extract a 10% sample of the data, **with** replacement.

```
educexp10pctwr <-
  educexp %>% sample_frac(0.1, replace = T)
educexp10pctwr %>% head()
##   education income young urban states
## 1       162   2634 389.6   661     LA
## 2       155   3029 369.4   797     TX
## 3       172   3509 354.5   753     OH
## 4       230   3072 348.5   322     VT
## 5       230   3072 348.5   322     VT
educexp10pctwr %>% dim()
## [1] 5 5
```

- With replacement, it draws VT (Vermont) twice.

Random samples

- Extract a 10-case sample of the data, **without** replacement.

```
educexp10case <-
  educexp %>% sample_n(10, replace = F)
educexp10case %>% head()
##   education income young urban states
## 1       201   2790 412.4   804     UT
## 2       191   3191 336.0   805     FL
## 3       215   3688 341.3   726     WA
## 4       247   3742 364.1   766     MD
## 5       230   3072 348.5   322     VT
## 6       149   2380 376.7   476     SC
educexp10case %>% dim()
## [1] 10  5
```

- As you can see, the sample has precisely 10 cases.

Random samples

- Extract a 10-case sample of the data, **with** replacement.

```
educexp10casewr <-
  educexp %>% sample_n(10, replace = T)
educexp10casewr %>% tail(10)
##    education income young urban states
## 1        189   2824 350.7   508     ME
## 2        246   4425 352.1  1000     DC
## 3        130   2081 385.2   445     MS
## 4        225   3957 385.1   809     NV
## 5        215   3688 341.3   726     WA
## 6        209   3363 360.7   659     WI
## 7        262   3341 365.4   664     MN
## 8        134   2322 351.9   500     AR
## 9        234   3265 343.8   572     IO
## 10       247   3742 364.1   766     MD
```

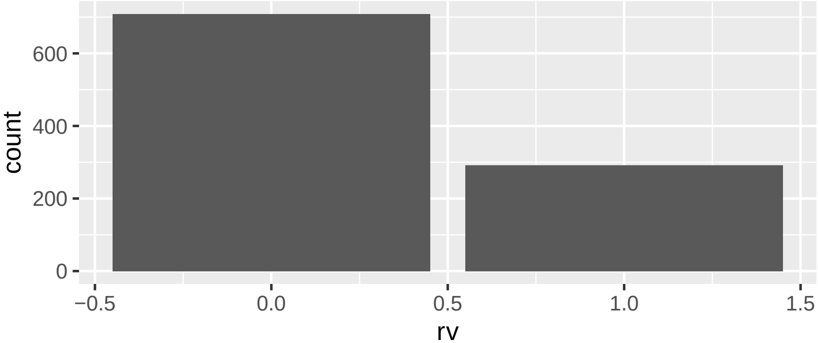
- We were lucky that no state had been drawn twice.

Playing with random variables

Creating Random Bernoulli

- The function `rbinom` gets: Number of cases, *1* (zeros or ones), and `prob = p`.

```
rv <- rbinom(1000, 1, prob = 0.3)
ggplot(data = data.frame(rv)) + geom_bar(aes(x = rv))
```



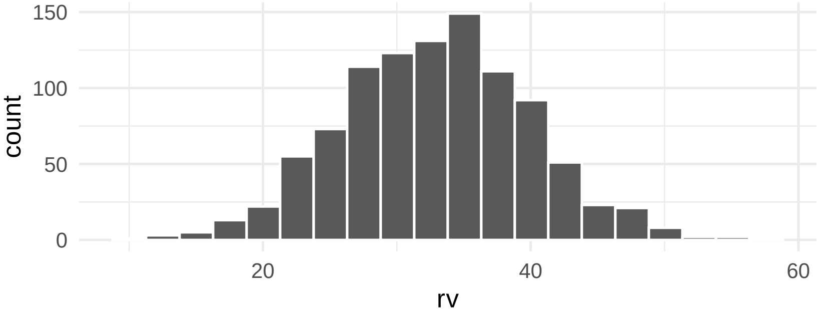
Creating Random Bernoulli

- **Your turn:** Create a variable:
  - 10000 observations with probability $p = 0.23$
  - 2000 observations with probability $p = 0.95$
  - 500 observations with probability $p = 0.41$
- Create a histogram of each of the cooked variables.

Creating Random Normal Variables

- The function rnorm gets: Number of cases, mean, and standard deviation (sd).

```r
rv <- rnorm(1000, mean = 33, sd = 7)
ggplot(data = data.frame(rv)) +
  geom_histogram(aes(x = rv), bins = 20, color = 'white') +
  theme_minimal()
```



Creating Random Normal Variables

- **Your turn:** Create a variable:
  - 10000 observations with mean *3* and standard deviation *5*
  - 2000 observations with mean *35* and standard deviation *10*
  - 500 observations with mean *0* and standard deviation *1*
- Create a histogram of each of the cooked variables.

Checking the full summary of a regression

Regression summary

- We can create a summary of a regression.
- You must wrap the `lm` function around `summary`.
- Check it out for the voting experiment and the expenditure in education in the 1970 US states.
- Couple of things to note:
  1. Lots of statistics. You will learn about the most relevant in class.
  2. You can compute R^2 now for models with more than two variables.

Regression summary

```
summary(lm(voted ~ message, data = voting))
##
## Call:
## lm(formula = voted ~ message, data = voting)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.3780 -0.2966 -0.2966  0.6220  0.7034
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.296638   0.001055  281.05   <2e-16 ***
## messageyes  0.081310   0.002587   31.43   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4616 on 229442 degrees of freedom
## Multiple R-squared:  0.004288,   Adjusted R-squared:  0.004284
## F-statistic: 988.1 on 1 and 229442 DF,  p-value: < 2.2e-16
```

Regression summary

```
summary(lm(education ~ income + young + urban, data = educexp))
##
## Call:
## lm(formula = education ~ income + young + urban, data = educexp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.240 -15.738  -1.156  15.883  51.380
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.868e+02  6.492e+01  -4.418 5.82e-05 ***
## income       8.065e-02  9.299e-03   8.674 2.56e-11 ***
## young        8.173e-01  1.598e-01   5.115 5.69e-06 ***
## urban       -1.058e-01  3.428e-02  -3.086  0.00339 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.69 on 47 degrees of freedom
## Multiple R-squared:  0.6896, Adjusted R-squared:  0.6698
## F-statistic: 34.81 on 3 and 47 DF,  p-value: 5.337e-12
```

Regression summary

- For a prettier table, use the `stargazer` package.
- Stargazer has lots of options and makes plots very pretty. The only one you need to remember:
  - If your R Markdown is a PDF, use `type = 'latex'`
  - If your R Markdown is an HTML, use `type = 'html'`
- Also add the `results = 'asis'` to your code chunk, otherwise, it will look weird.

Regression summary

- To install stargazer:

    1. Install `stargazer`

    2. Load `stargazer`

```
library(stargazer)
##
## Please cite as:
##  Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Stat
##  R package version 5.2.3. https://CRAN.R-project.org/package=stargazer
```

Regression summary

```r
mod <- lm(voted ~ message, data = voting)
stargazer(mod, type = 'latex', header = F, float = F,
          font.size = 'scriptsize', no.space = F,
          dep.var.labels = 'Turnout', style = 'qje',
          covariate.labels = 'Peer-pressure message')
```

	Turnout
Peer-pressure message	0.081***
	(0.003)
Constant	0.297***
	(0.001)
N	229,444
R^2	0.004
Adjusted R^2	0.004
Residual Std. Error	0.462 (df = 229442)
F Statistic	988.067*** (df = 1; 229442)
Notes:	***Significant at the 1 percent level.
	**Significant at the 5 percent level.
	*Significant at the 10 percent level.

Today's Lab

- Group-by and summarize
- A bit more recoding
- Dealing with missing data
- Extract random samples from data
- Playing with random variables
- Full summary of a regression

Next Lab

- Cool things you can do with R

Questions?

See you in the next lab!