POLI 30 D: Political Inquiry

TA Sessions

Lab 05 | R Plots and R Data Analysis I

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

Recap: In the Lab sessions, you learned:

- How to install R and R Studio on your computer.
- How to do basic math operations in R.
- How to do basic vector and data.frame operations in R.
- How to install packages and work with R Markdown.

Great job!

- Do you have any questions about these contents?

Plan for Lab 05

- Factors
- Booleans and logical tests
- Lists
- Tables
- Intro to plots

Factors

Factors

- There is another type of variable that you learned little about: `factor`.
- Factors are great when you want to build tables because it takes a number and adds a mask to it.
- Raw content:

```
voted <- c(0,0,1,0,1) # 0=no and 1=yes
table(voted)
## voted
## 0 1
## 3 2
```

Factors

- Factor:

```
voted_factor <- factor(voted,
                       levels = c(0,1),
                       labels = c('No', 'Yes'))
table(voted_factor)
## voted_factor
##  No Yes
##   3   2
```

Factor

- The side-effect is that you cannot take means of it anymore, but you can still do `prop.tables`:

```
prop.table(table(voted_factor))
## voted_factor
##  No Yes
## 0.6 0.4
```

- And the mean would be equal to 0.4.

Booleans and Logics

Booleans

- On programming, we frequently perform logical operations.
- For instance, suppose you have a vector of ages:

```
ages <- c(40, 20, 64, 20, 26, 21, 29, 32)
ages
## [1] 40 20 64 20 26 21 29 32
```

Booleans

- Which ages are above 30?

```
ages
## [1] 40 20 64 20 26 21 29 32
ages > 30
## [1]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE
```

- Note the `TRUE` and `FALSE`. These are the so-called Boolean values.

Booleans

- Which ages are equal to 20?

```
ages
## [1] 40 20 64 20 26 21 29 32
ages == 20
## [1] FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE
```

- Note the ==. This is a **comparison operator**. For an assignment, you can use <- or =.

Booleans

- What are the comparison operators?

operator	operation
>	bigger than
>=	bigger than or equal
<	less than
<=	less than or equal
==	equal
!=	different (not equal)
!	not
|	or
&	and

Booleans

- And we can use the operators for subsetting:

```
ages[!(ages == 20)] # same as ages[ages != 20]
## [1] 40 64 26 21 29 32
```

- **Your turn**: Select ages above 25 and below 50. Hint: First, create the logical operation that selects those; then, use the logic operation to subset.

Lists

Lists

- All objects we have seen so far are of a fixed type.
- This makes it hard for you to store a text and a number in a placeholder without messing up with either the text or the number.
- Lists can hold any object without changing its type.
- But they are harder to operate.

Lists

Example:

```
mylist <- list('numbers' = c(1,2,3),
               'ages' = c(19,20,33),
               'text' = 'My name is Tom')
mylist
## $numbers
## [1] 1 2 3
##
## $ages
## [1] 19 20 33
##
## $text
## [1] "My name is Tom"
```

Lists

To index a list, do:

```
mylist[[1]]
## [1] 1 2 3
mylist$text
## [1] "My name is Tom"
```

And to add to a list:

```
mylist$newelement = 3
```

- We will not use them much, but it is nice that you know they exist.

Tables

Tables

- We know you already saw a few of those, but still, let's talk about them.
- Tables count the number of elements and display the label and the frequency of the elements.
- Let's load the voting dataset:

```
voting <- read.csv("https://raw.githubusercontent.com/umber
```

Tables

- And a table of messages looks like this:

```
table(voting$message)
##
##     no    yes
## 191243  38201
```

- The syntax is always like this: `table(dataframename$variablename)`.
- **Your turn**: Make a table of the variable `voted`.

Tables

- If we want a proportions table, we can add `prop.table` to a table:

```
prop.table(table(voting$message))
##
##        no       yes
## 0.8335062 0.1664938
```

- The syntax is always like this: `prop.table(table(dataframename$variablename))`.
- **Your turn**: Make a proportions table of the variable `voted`.
- **Your turn**: Make a proportions table of the variable `birth`. Does it look good?

Intro to Plots

Plots of a single variable

- For plots, we will use a package called `ggplot2`.
- Here is a good cheat sheet. This is a great thing to print and has close by when creating plots.
- `ggplot2` is based on the `grammar of graphs`. But what is this?

Plots of a single variable

- In abstract, the grammar of graphs is a decomposition of plots in its main features.
- In essence, every plot has:

1. A dataset
2. A coordinated system
3. A geometric shape

- And different plots are different compositions of these three key ingredients.

Plots of a single variable

- Histograms are great plots for continuous variables. The syntax is:

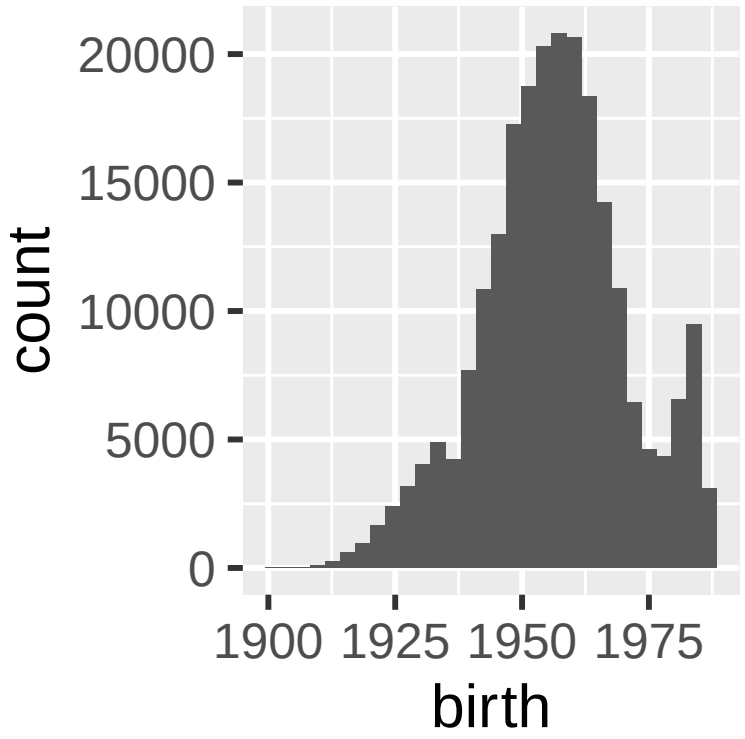
```
ggplot(data = dataset, aes(x = variable_x_name)) +
  geom_histogram()
```

- Note the components in action:
  - `data` adds the dataset;
  - `aes(x =...)` adds the coordinated system;
  - `geom_histogram()` adds the geometric shape.

Plots of a single variable

```
ggplot(data = voting, aes(x = birth)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Plots of a single variable

- And we can customize it a lot. For the `geom_histogram()`:
  - `stat =`: Default is `bin`
  - `bins =`: Number of bins (default is 30)
  - `binwidth =`: Size of the bin (don't use when using `bins`)
  - `breaks =`: Vector with custom breaks (overrides `bins` and `binwidth`)
  - `color =`: Changes the contour colors.
  - `fill =`: Changes the color of the shape.
  - `alpha =`: Changes the transparency (between 0 and 1, with 0 being transparent).
  - `aes(y = ..density..)`: Instead of count, show densities (good when we have large datasets).

Plots of a single variable

- For the main plot:

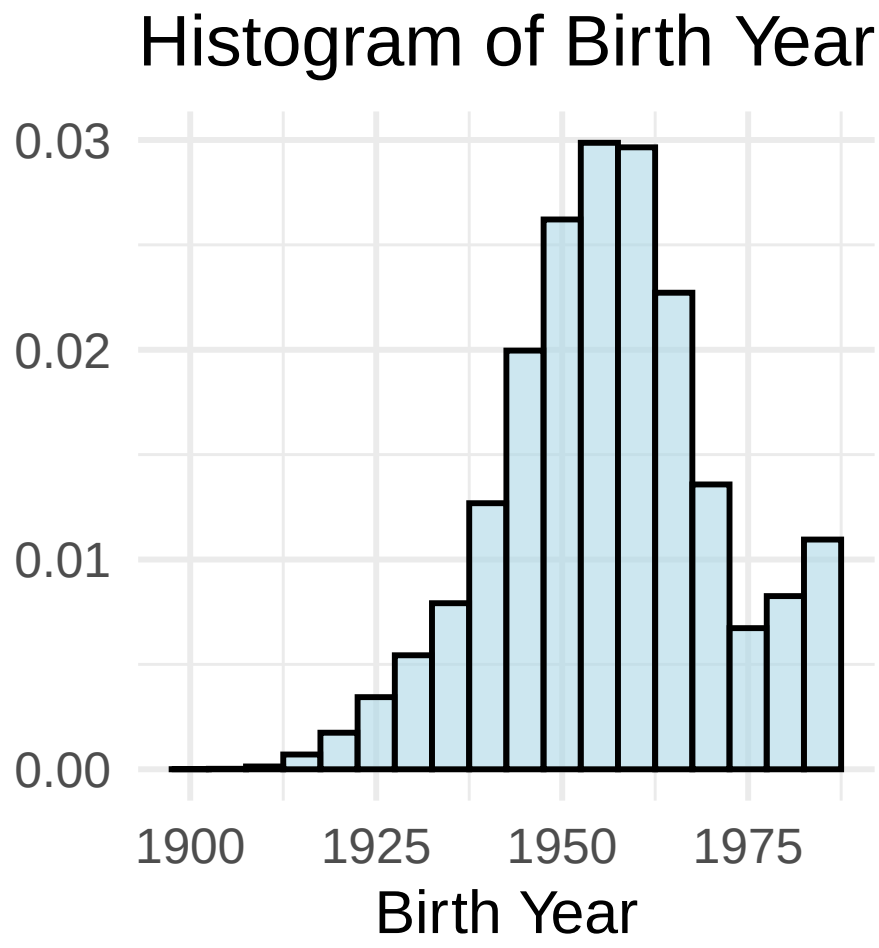
```
plot +
  labs(x = 'X-axis title',
       y = 'y-axis title',
       title = 'plot title',
       subtitle = 'subtitle below the plot')
```

Plots of a single variable

- And the `theme` changes the general look of the plot:
  - `theme_bw()`: Black and white. My personal choice.
  - `theme_gray()`: Gray background.
  - `theme_dark()`: Dark gray background.
  - `theme_minimal()`: Little structure.
  - `theme_void()`: Empty theme.
- Now, let's see a nice plot with lots of customization.

Plots of a single variable

```
ggplot(data = voting, aes(x = birth)) +
  geom_histogram(aes(y = ..density..), binwidth = 5, color = 'black', fill = 'lightblue', alpha = 0.6) +
  labs(title = 'Histogram of Birth Year', x = 'Birth Year', y = '') + theme_minimal()
```



Plots of a single variable

- And for a continuous variable, instead of `geom_histogram`, you could do:
  - `geom_area(stat = 'bin')`: Looks like a big polygon
  - `geom_density()`: Density plot
  - `geom_dotplot()`: Very cool plot with little dots instead of rectangles.
  - `geom_line()`: Line-plot: pretty much connects things with lines.
  - `geom_step()`: Very cool, like a contour of a histogram.
- Try them out and see. Some would need some customizing before it works.

Today's Lab

- Factors
- Booleans and logical tests
- Lists
- Tables
- Intro to plots

Next Lab

- More plots
- Some less obvious data analysis

Questions?

See you in the next lab!