R dpylr project

Text

Description automatically generated

You'll start by finding this summary for the entire dataset: the fraction of all votes in their history that were "yes". Note that within your call to summarize(), you can use n() to find the total number of votes and mean(vote == 1) to find the fraction of "yes" votes

**Summarizing by year**

The summarize() function is especially useful because it can be used within *groups*.

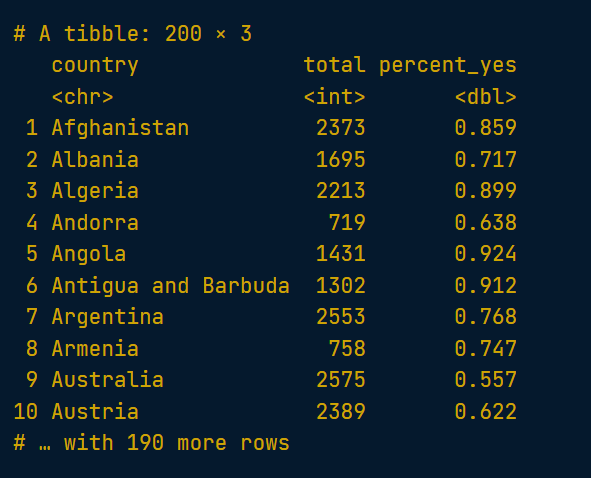
For example, you might like to know how much the average "agreeableness" of countries changed from year to year. To examine this, you can use group\_by() to perform your summary not for the entire dataset, but within each year.

Table

Description automatically generated with low confidence

# Summarizing by country

In the last exercise, you performed a summary of the votes within each year. You could instead summarize() within each country, which would let you compare voting patterns between countries.



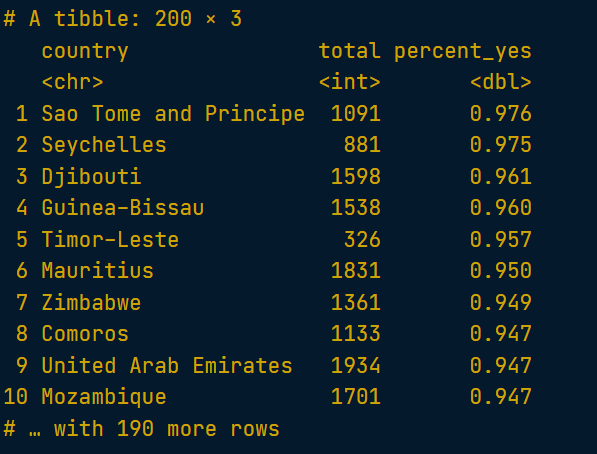
Diagram

Description automatically generated

# Sorting by percentage of "yes" votes

Now that you've summarized the dataset by country, you can start examining it and answering interesting questions.

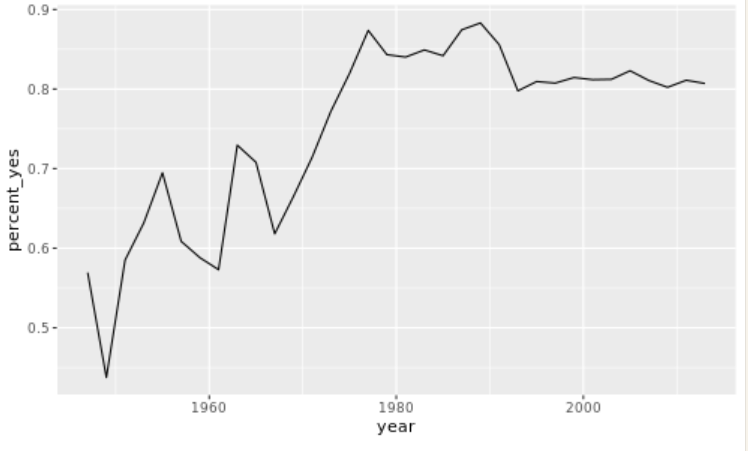
For example, you might be especially interested in the countries that voted "yes" least often, or the ones that voted "yes" most often.



# Plotting a line over time

In the last chapter, you learned how to summarize() the votes dataset by year, particularly the percentage of votes in each year that were "yes".

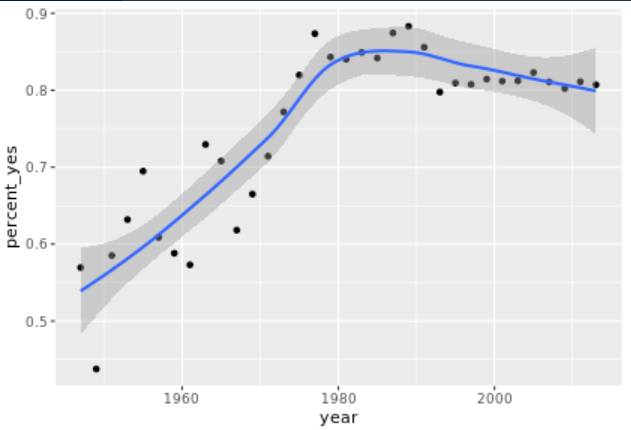
You'll now use the ggplot2 package to turn your results into a visualization of the percentage of "yes" votes over time.



# Other ggplot2 layers

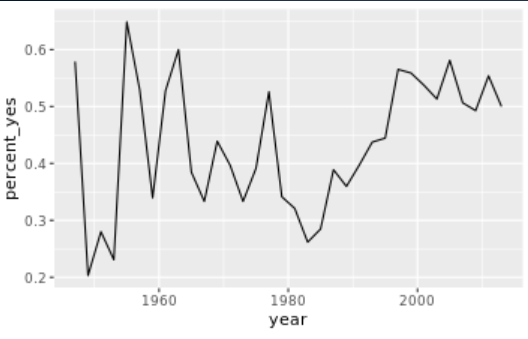
A line plot is one way to display this data. You could also choose to display it as a scatter plot, with each year represented as a single point. This requires changing the layer (i.e. geom\_line() to geom\_point()).

You can also add additional layers to your graph, such as a smoothing curve with geom\_smooth().



# Summarizing by year and country

You're more interested in trends of voting within specific countries than you are in the overall trend. So instead of summarizing just by year, summarize by both year and country, constructing a dataset that shows what fraction of the time each country votes "yes" in each year.



# Plotting multiple countries

Plotting just one country at a time is interesting, but you really want to compare trends between countries. For example, suppose you want to compare voting trends for the United States, the UK, France, and India.

You'll have to filter to include all four of these countries and use another aesthetic (not just x- and y-axes) to distinguish the countries on the resulting visualization. Instead, you'll use the color aesthetic to represent different countries.

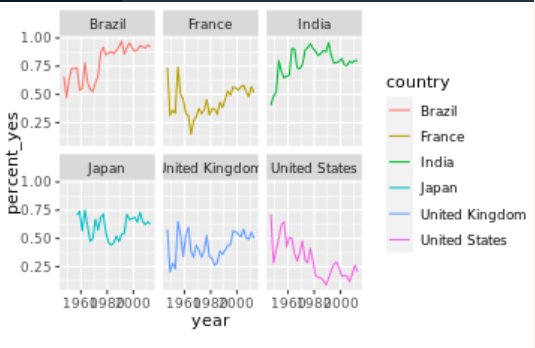
Chart, line chart

Description automatically generated

# Faceting the time series

Now you'll take a look at six countries. While in the previous exercise you used color to represent distinct countries, this gets a little too crowded with six.

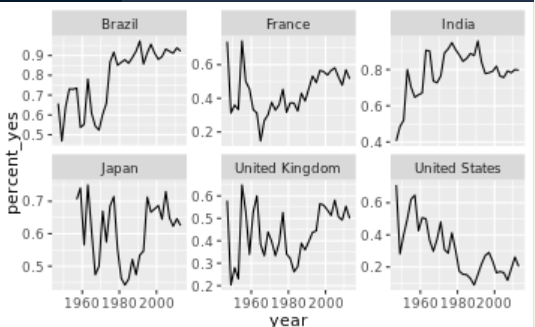
Instead, you will facet, giving each country its own sub-plot. To do so, you add a facet\_wrap() step after all of your layers.



# Faceting with free y-axis

In the previous plot, all six graphs had the same axis limits. This made the changes over time hard to examine for plots with relatively little change.

Instead, you may want to let the plot choose a different y-axis for each facet.



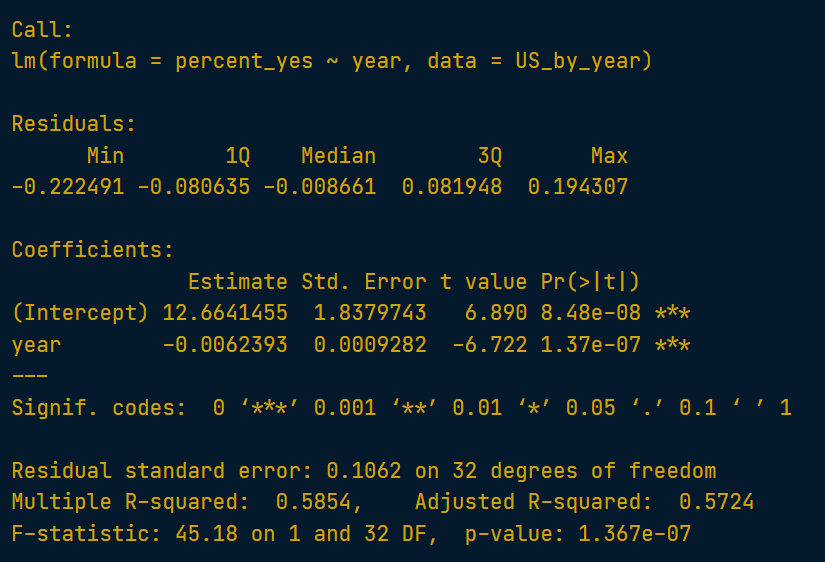
# Choose your own countries

The purpose of an exploratory data analysis is to ask questions and answer them with data. Now it's your turn to ask the questions.

You'll choose some countries whose history you are interested in and add them to the graph. If you want to look up the full list of countries, enter by\_country$country in the console.

Calendar

Description automatically generated



Text

Description automatically generated with medium confidence

# Nesting a data frame

Right now, the by\_year\_country data frame has one row per country-vote pair. So that you can model each country individually, you're going to "nest" all columns besides country, which will result in a data frame with one row per country. The data for each individual country will then be stored in a **list column** called data.

# List columns

This "nested" data has an interesting structure. The second column, data, is a **list**, a type of R object that hasn't yet come up in this course that allows complicated objects to be stored within each row. This is because each item of the data column is itself a data frame.

tibble: 200 × 2

country data

<chr> <list>

1 Afghanistan <tibble [34 × 3]>

2 Argentina <tibble [34 × 3]>

3 Australia <tibble [34 × 3]>

4 Belarus <tibble [34 × 3]>

5 Belgium <tibble [34 × 3]>

6 Bolivia, Plurinational State of <tibble [34 × 3]>

7 Brazil <tibble [34 × 3]>

8 Canada <tibble [34 × 3]>

9 Chile <tibble [34 × 3]>

10 Colombia <tibble [34 × 3]>

# Performing linear regression on each nested dataset

Now that you've divided the data for each country into a separate dataset in the data column, you need to fit a linear model to each of these datasets.

The map() function from purrr works by applying a formula to each item in a list, where . represents the individual item. For example, you could add one to each of a list of numbers:

map(numbers, ~ 1 + .)

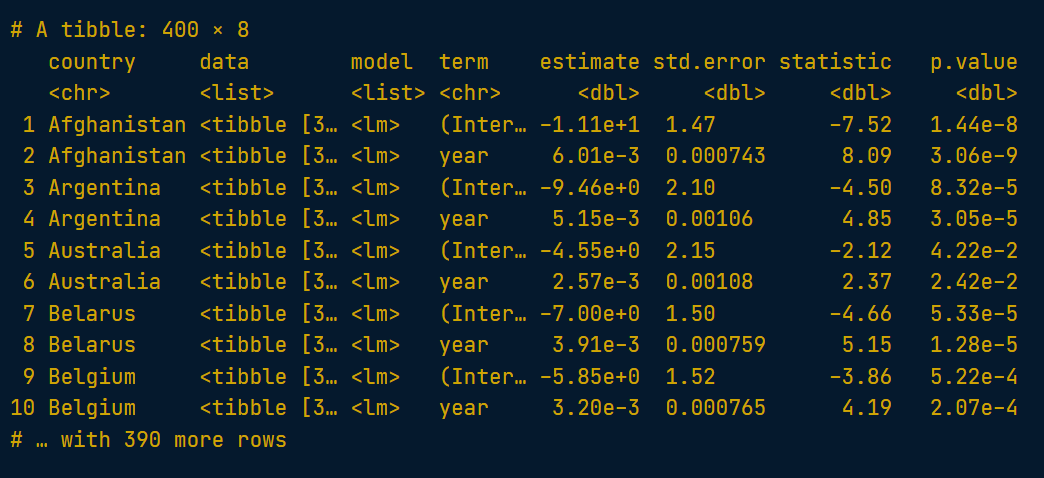
This means that to fit a model to each dataset, you can do:

map(data, ~ lm(percent\_yes ~ year, data = .))

where . represents each individual item from the data column in by\_year\_country. Recall that each item in the data column is a dataset that pertains to a specific country.

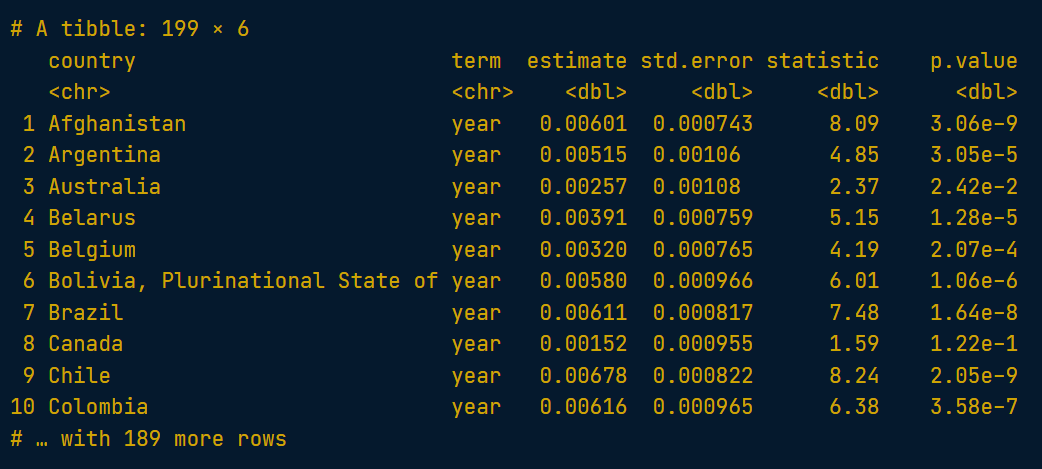
# Unnesting a data frame

You now have a tidied version of each model stored in the tidied column. You want to combine all of those into a large data frame, similar to how you combined the US and UK tidied models earlier. Recall that the unnest() function from tidyr achieves this.



# Filtering model terms

You currently have both the intercept and slope terms for each by-country model. You're probably more interested in how each is changing over time, so you want to focus on the slope terms.

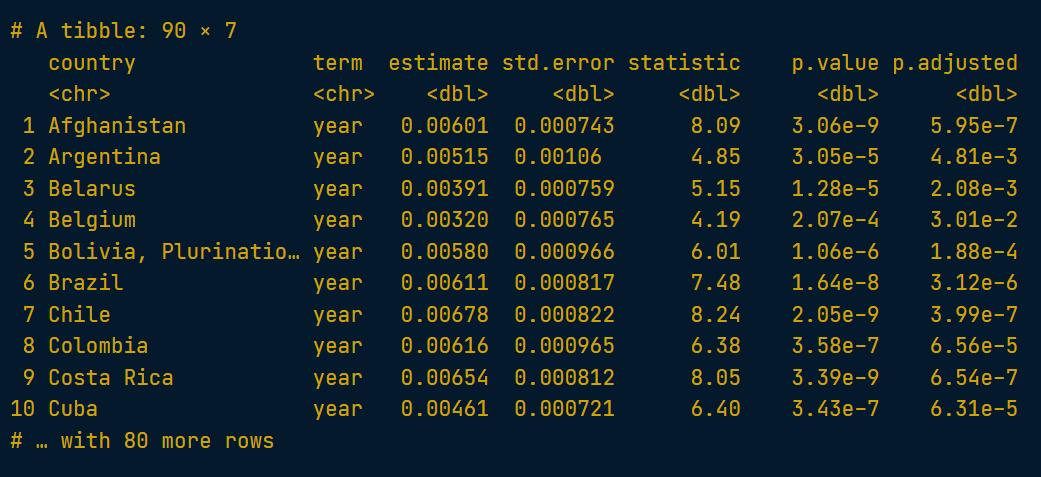


# Filtering for significant countries

Not all slopes are significant, and you can use the p-value to guess which are and which are not.

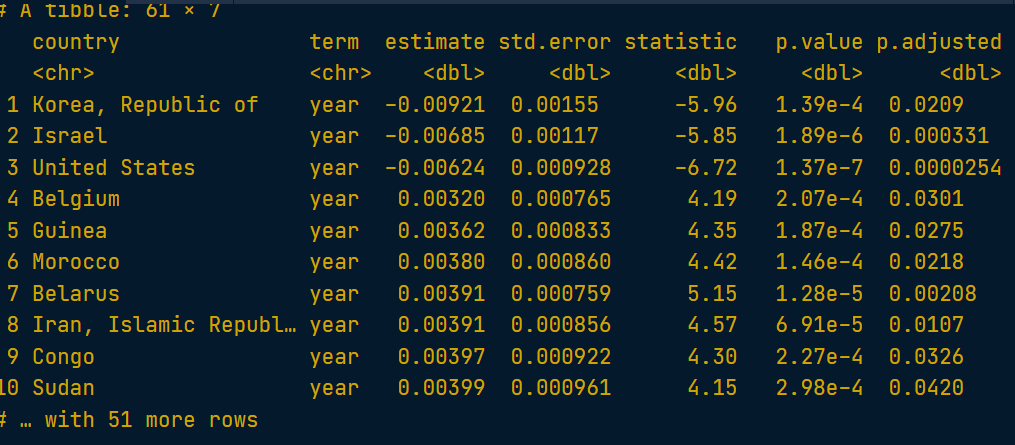
However, when you have lots of p-values, like one for each country, you run into the problem of multiple hypothesis testing, where you have to set a stricter threshold. The **[p.adjust()](https://www.rdocumentation.org/packages/stats/topics/p.adjust" \t "_blank)** function is a simple way to correct for this, where p.adjust(p.value) on a vector of p-values returns a set that you can trust.

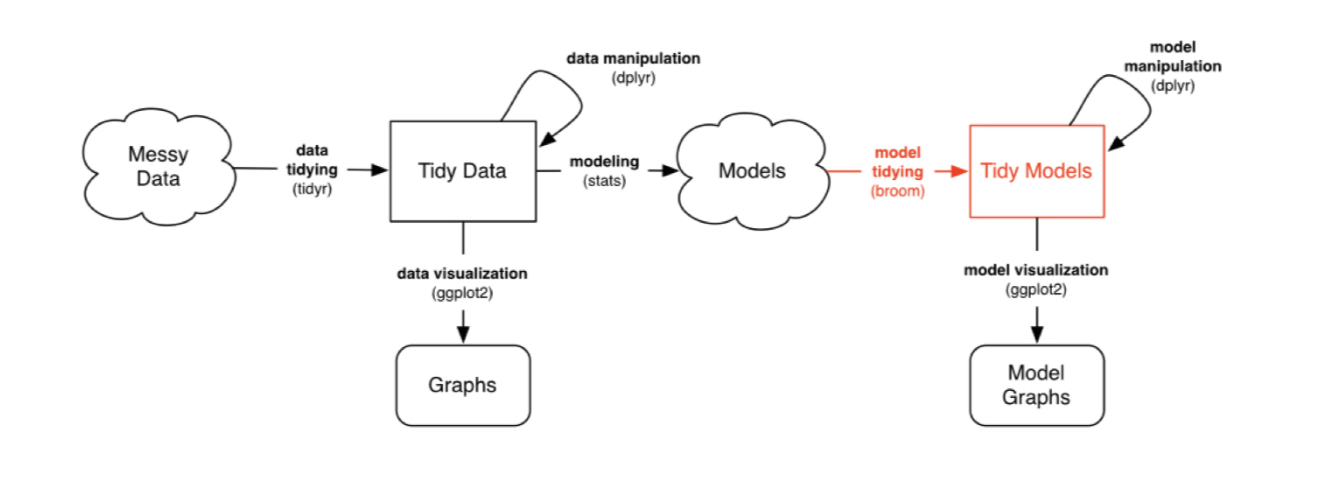
Here you'll add two steps to process the slope\_terms dataset: use a mutate to create the new, adjusted p-value column, and filter to filter for those below a .05 threshold.



# Sorting by slope

Now that you've filtered for countries where the trend is probably not due to chance, you may be interested in countries whose percentage of "yes" votes is changing most quickly over time. Thus, you want to find the countries with the highest and lowest slopes; that is, the estimate column.

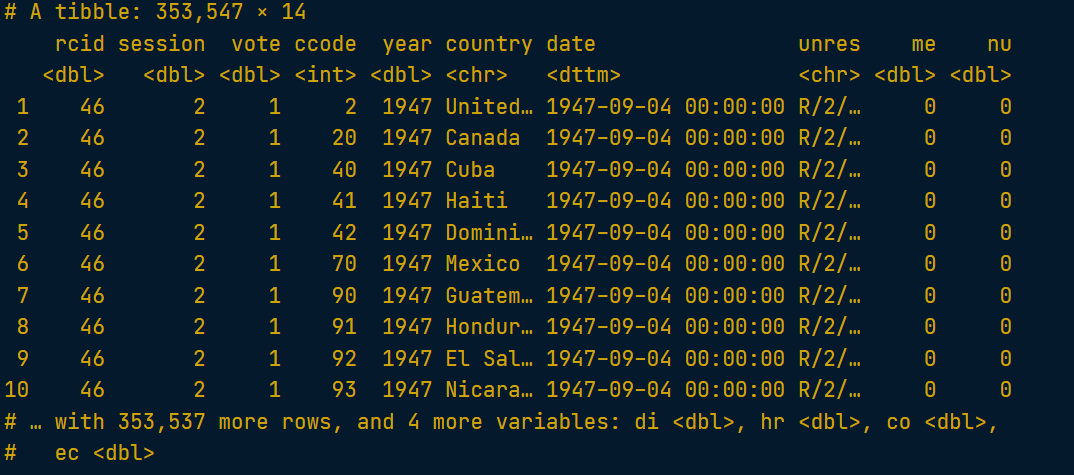




# Joining datasets with inner\_join

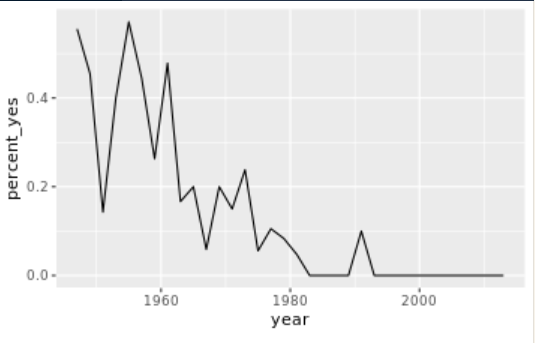
In the first chapter, you created the votes\_processed dataset, containing information about each country's votes. You'll now combine that with the new descriptions dataset, which includes topic information about each country, so that you can analyze votes within particular topics.

To do this, you'll make use of the inner\_join() function from dplyr.



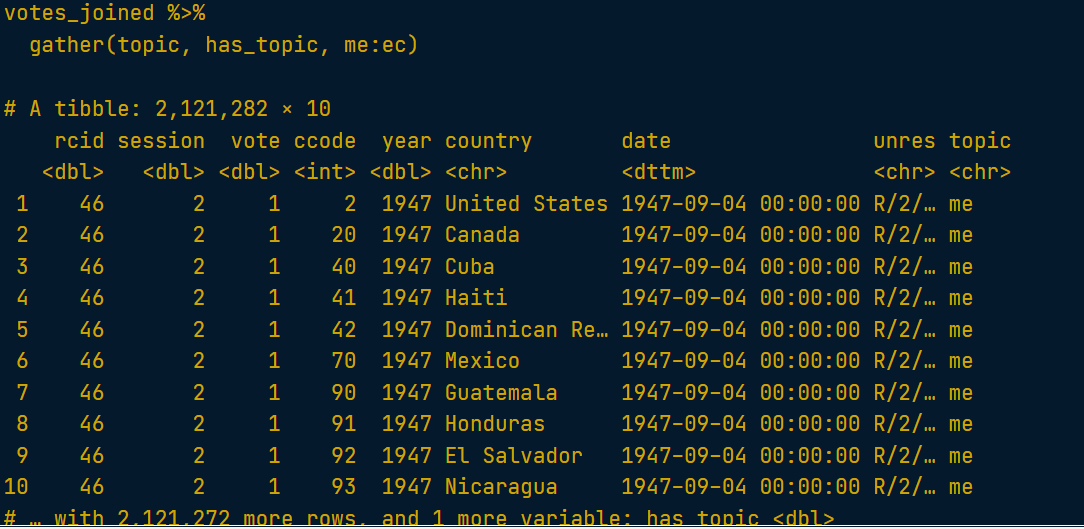
# Visualizing colonialism votes

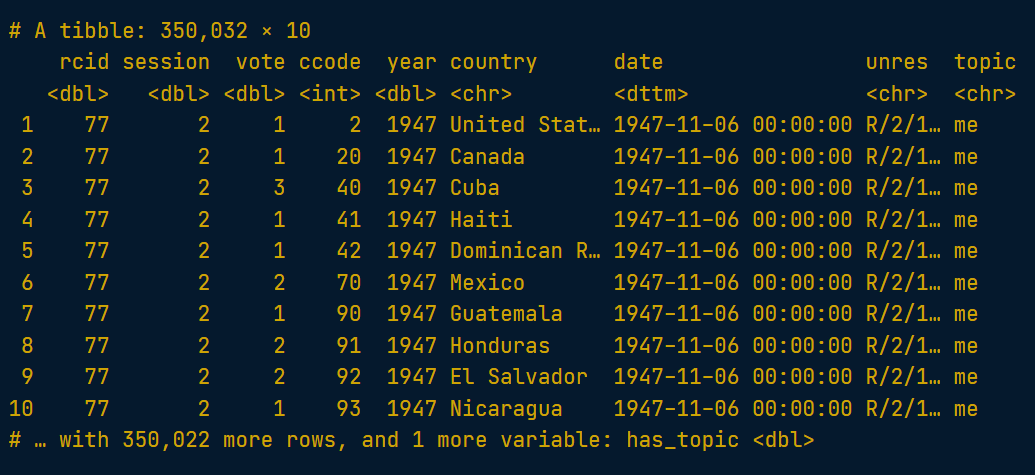
In an earlier exercise, you graphed the percentage of votes each year where the US voted "yes". Now you'll create that same graph, but only for votes related to colonialism.



# Using gather to tidy a dataset

In order to represent the joined vote-topic data in a tidy form so we can analyze and graph by topic, we need to transform the data so that each row has one combination of country-vote-topic. This will change the data from having six columns (me, nu, di, hr, co, ec) to having two columns (topic and has\_topic).





**Recoding the topics**

There's one more step of data cleaning to make this more interpretable. Right now, topics are represented by two-letter codes:

1. **me**: Palestinian conflict
2. **nu**: Nuclear weapons and nuclear material
3. **di**: Arms control and disarmament
4. **hr**: Human rights
5. **co**: Colonialism
6. **ec**: Economic development

So that you can interpret the data more easily, recode the data to replace these codes with their full name. You can do that with dplyr's recode() function, which replaces values with ones you specify:

example <- c("apple", "banana", "apple", "orange")

recode(example,

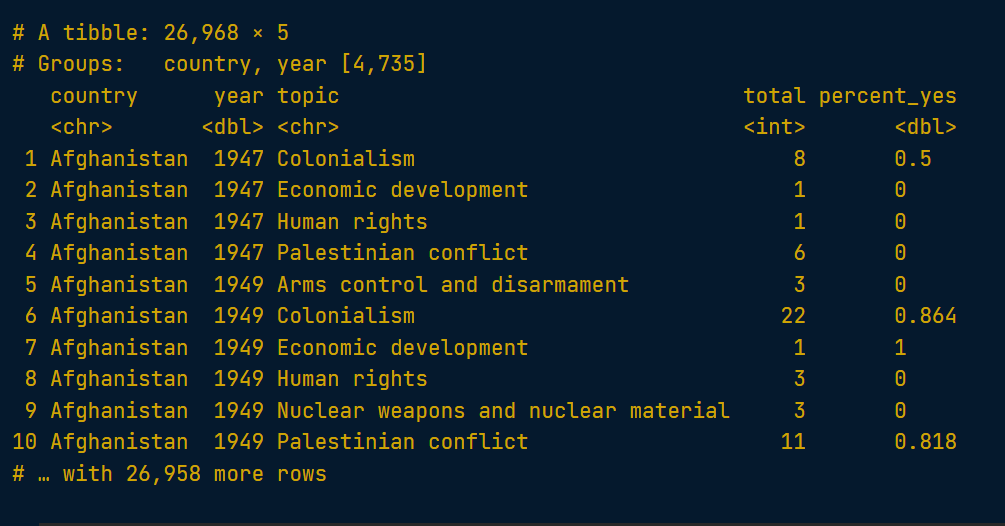
apple = "plum",

banana = "grape")

# Summarize by country, year, and topic

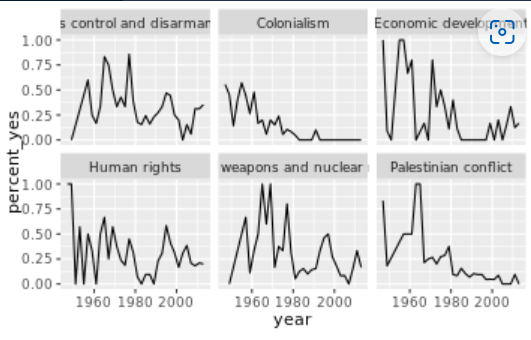
In previous exercises, you summarized the votes dataset by country, by year, and by country-year combination.

Now that you have topic as an additional variable, you can summarize the votes for each combination of country, year, and topic (e.g. for the United States in 2013 on the topic of nuclear weapons.)



# Visualizing trends in topics for one country

You can now visualize the trends in percentage "yes" over time for all six topics side-by-side. Here, you'll visualize them just for the United States.



# Nesting by topic and country

In the last chapter, you constructed a linear model for each country by nesting the data in each country, fitting a model to each dataset, then tidying each model with broom and unnesting the coefficients. The code looked something like this:

country\_coefficients <- by\_year\_country %>%

nest(-country) %>%

mutate(model = map(data, ~ lm(percent\_yes ~ year, data = .)),

tidied = map(model, tidy)) %>%

unnest(tidied)

Now, you'll again be modeling change in "percentage" yes over time, but instead of fitting one model for each country, you'll fit one for each combination of country and topic.



# Interpreting tidy models

Now you have both the slope and intercept terms for each model. Just as you did in the last chapter with the tidied coefficients, you'll need to filter for only the slope terms.

You'll also have to extract only cases that are statistically significant, which means adjusting the p-value for the number of models, and then filtering to include only significant changes.

# Checking models visually

In the last exercise, you found that over its history, Vanuatu (an island nation in the Pacific Ocean) sharply changed its pattern of voting on the topic of Palestinian conflict.

Let's examine this country's voting patterns more closely. Recall that the by\_country\_year\_topic dataset contained one row for each combination of country, year, and topic. You can use that to create a plot of Vanuatu's voting, faceted by topic.

A picture containing line chart

Description automatically generated