Introduction to Regression in R for Policy Analysis

Chisom Obasih

2025-10-29

This worksheet presents a basic introduction to regression analysis in R, specifically for policy analysis. This tutorial was developed by Chisom Obasih, in collaboration with Kristen Scotti, for the course 84-703: Employing Qualitative and Quantitative Methods in Policy Analysis at Carnegie Mellon University's Institute for Strategy and Technology (CMIST), however, it is welcomed to be used by anyone interested in regression analysis of political policy data using R.

In this lesson you will:

- Familiarize yourself with the Correlates of State Policy Project (cspp) dataset (Lucas & McCrain, 2020).
- Visualize data relationships using base R code
- Learn the syntax for regression modeling using base R code
- Understand the regression model output and learn to report regression outcomes and uncertainty

0. Before the lesson

You will need to have downloaded R and RStudio to complete these exercises. Please do this before coming to class if you don't already have R and RStudio on your machine.

If the hyperlinks don't work, copy and paste these links into your browser. R: https://cran.r-project.org/. RStudio: https://www.rstudio.com/products/rstudio/download/

1. Install and load necessary packages

0. To perform these exercises, you will need the following R packages. Installing R packages only needs to happen once per machine. Uncomment the line of code and run this code chunk **if you do not already have these packages installed.**

Protip: You can highlight a chunk of text within a code chunk and press Cmd+Shift+C on Mac or Cntl+Shift+C on Windows to comment or uncomment multiple lines.

```
# install.packages(c('tidyverse', 'cspp'))
```

1. Loading R packages needs to happen every time you start RStudio. Run the code chunk below to load the required packages.

```r
library(tidyverse)  # package with a lot of useful utilities that helps with data
↪  cleaning
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.1     v stringr   1.5.2
## v ggplot2   4.0.0     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```r
library(cspp)  # package that contains the dataset
```

```
## Please cite:
##
## Caleb Lucas and Joshua McCrain (2020). cspp: A Package for The Correlates of State
Policy Project Data.
## R package version 0.3.3.
##
## Grossmann, M., Jordan, M. P. and McCrain, J. (2021) 'The Correlates of State Policy
and the Structure of State Panel Data,'
## State Politics & Policy Quarterly. Cambridge University Press, pp. 1-21. doi:
10.1017/spq.2021.17.
##
## You are using the version of the Correlates Dataset stored in your local copy of
csppData. Running `library(csppData)` will print your version number.
```

```r
# R setting: preserve the printing of decimal form and switch to scientific
# notation after 5 decimal places
options(scipen = 5)
```

2. Basic use of the cspp dataset

The cspp dataset has two primary functions (and a number of other helper functions that we are not concerned with yet). These are the `get_var_info` and `get_cspp_data` functions. These functions allow you to access part or all of the two dataframes that comprise the dataset: the codebook for the variables in the state policy data and the state policy data itself, respectively.

First, take a look at the data itself by loading it into a dataframe.

```r
# running the function without any arguments returns the entire dataframe
all_data <- get_cspp_data()

# see the structure of the first 20 columns of the dataframe
glimpse(all_data[1:20])
```

```
## Rows: 6,171
## Columns: 20
## $ st            <chr> "AK", "AK", "AK", "AK", "AK", "AK", "AK", "AK", "AK", "A~
## $ stateno       <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,~
## $ state         <chr> "Alaska", "Alaska", "Alaska", "Alaska", "Alaska", "Alask~
## $ state_fips    <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,~
## $ state_icpsr   <dbl> 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, ~
## $ year          <dbl> 1900, 1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 19~
## $ popdensity    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ popfemale     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ pctpopfemale  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ popmale       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ pctpopmale    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ popunder5     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ pctpopunder14 <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ pop5to17      <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ pop18to24     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ pctpop15to24  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ pop25to44     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ pctpop25to44  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ pop45to64     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ pctpop45to64  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

Alaska doesn't have much or any data from what we can see between 1900-1919, except for variables denoting state labels and year, also known as **'panel variables'**).

In RStudio, run the below code chunk and take 30 seconds to browse the full dataframe.

```
# open the entire dataframe in a new tab in RStudio
View(all_data)
```

You'll notice that each row contains policy data for a state for a particular year. For the 50 states + Washington DC (51 data groups), there are rows for data across the years 1900-2020 (121 years), resulting in 6171 rows. Not every row contains data for every variable ('NA'), but it's important to know when data is unavailable versus having the data point be completely missing (e.g., it's better to have all columns be filled with 'NA' for Alaska in 1900 than it is to not have the row in the dataset at all). You'll also notice that there's over 3000 columns (*'variables'*) of policy data, which is wildly overwhelming.

The codebook, accessed by the `get_var_info` function, describes each variable, the years for which data is available for any state, and the source of the data. This also includes information on units of numerical data units and coding/meaning of categorical or binary data. Load the codebook into a dataframe to examine the variables available in the state policy data.

```
# running the function without any arguments returns the entire dataframe
all_variables <- get_var_info() %>%
    # see the structure of the dataframe
glimpse()
```

```
## Rows: 3,014
## Columns: 14
## $ variable        <chr> "poptotal", "popdensity", "popfemale", "pctpopfemale",~
## $ years           <chr> "1900-2019", "1975-1999", "1994-2000, 2002-2010", "201~
## $ short_desc      <chr> "Population total", "Population density", "Female popu~
## $ long_desc       <chr> "Total population per state", "Number of people per sq~
```

```
## $ sources         <chr> "U.S. Census Bureau (http://www.census.gov/)\r\nOrigin~
## $ category        <chr> "demographics", "demographics", "demographics", "demog~
## $ plaintext_cite  <chr> "Stateminder: A data visualization project from George~
## $ bibtex_cite     <chr> "@misc{stateminderNA,\n  title={Stateminder.org},\n  a~
## $ plaintext_cite2 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ bibtex_cite2    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ plaintext_cite3 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ bibtex_cite3    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ plaintext_cite4 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ bibtex_cite4    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
```

```
# Protip: The pipe operator (`%>%`) is a useful tool for clean coding it allows
# us to use the output of the function to the left of the pipe as an input to
# the function after the pipe without having to retype information
# `all_variables <- get_var_info() %>% glimpse()` is equivalent to the two
# lines of code: `all_variables <- get_var_info()` `glimpse(all_variables)`.
```

The codebook has only 14 columns (much more manageable) and 3014 rows. The number of rows almost matches the number of columns in the state data; only the panel variables (`st`, `stateno`, `state`, `state_fips`, `state_icpsr`, `year`) are excluded from the codebook.

Run the below chunk in RStudio and take another 30 seconds to browse the full dataframe. In the search box, type in some key words (e.g., 'women') to see which variables come up.

```
# open the dataframe in a new tab in RStudio
View(all_variables)
```

Each variable is categorized into one of 16 data categories: demographic, economic/fiscal, government, elections, policy/ideology, criminal justice, education, healthcare, welfare, rights/anti-discrimination, environment, drugs/alcohol, gun control, labor, transportation, regulation.

(For more detailed variable information, see the full codebook: https://ippsr.msu.edu/sites/default/files/CorrelatesCodebook.pdf)

We'll use the state data function `get_cspp_data` and codebook function `get_var_info` together to filter the data into more manageable chunks to do some analysis.

To get variables, we'll use the `get_var_info` function and several search arguments. You can get a list of variables by searching for key terms in the name (`var_name = c("key", "words")`), in the name, descriptions, or source (`related_to = c("key", "words")`), or by category (`category = c("key", "words")`). Variables in the `all_variables` dataframe that match the search terms will be filtered into a dataframe that we can use to filter the `all_data` dataframe. Saving the chosen variables in a dataframe is useful to have a reference to go back to check the variable descriptions.

```
# compile list of variables that have 'education' in the name
name_education <- get_var_info(var_names = "education")
# view the first six rows and first three columns
head(name_education[1:3])
```

```
##                          variable     years
## 1                   exp_education 2012-2016
## 2 interstate_compact_on_educationa 2008-2017
## 3            compact_for_education 1965-2017
## 4                   pro_education 1973-2012
```

```
## 5                   anti_education 1973-2012
## 6                neutral_education 1973-2012
##                                                                 short_desc
## 1                           General expenditure, by function: education
## 2 Helps military children transfer academic credits between institutions
## 3                                Information center on education matters
## 4                                                  Pro-education opinion
## 5                                                 Anti-education opinion
## 6                                               Neutral education opinion
```

```
# compile list of variables that have 'pollu' or 'disease' anywhere in the
# name, description, or source
desc_pollu_disease <- get_var_info(related_to = c("pollu", "disease"))
# view the first six rows and first three columns
head(desc_pollu_disease[1:3])
```

```
##             variable                       years
## 1          ig_health 1980, 1990, 1997-1999, 2007
## 2             ig_env 1980, 1990, 1997-1999, 2007
## 3 newbornheartscreen                  2011-2017
## 4        health_rank                  1992-2012
## 5              hmbkd                 2006, 2008
## 6            hmblyme                 2006, 2008
##                                                                 short_desc
## 1          Health care practice, delivery, and disease interest groups
## 2          Environmental preservation and conservation interest groups
## 3 Coverage for newborn screening for critical congenital heart disease
## 4                                                 Overall health ranking
## 5                                                         Kidney disease
## 6                                                           Lyme disease
```

```
# compile list of variables in the 'welfare' category
category_transportation <- get_var_info(categories = "transportation")
# view the first six rows and first three columns
head(category_transportation[1:3])
```

```
##             variable     years
## 1   allpass_seatbelt 1985-2017
## 2         bikehelmet 1989-2017
## 3 frontpass_seatbelt 1985-2017
## 4       helmetlaw_17 1980-2017
## 5    helmetlaw_learn 1980-2017
## 6     helmetlaw_pass 1980-2017
##
short_desc
## 1                                                          Requiring all passengers
to wear a seatbelt
## 2                                                                      Requiring
helmet for riding bike
## 3                                                        Requiring front
passengers to wear seatbelts
## 4                                              Motorcyclists 17 and under
required to wear a helmet
```

```
## 5 Those with instructional permits or licenses less than 1 year old required to wear
helmet on motorcycle
## 6                                                                    Helmet required for
motorcycle passenger
```

```
# compile list of variables that have 'pct' in the name AND 'femal' anywhere in
# the name or description AND in the categories of 'government' or 'healthcare'
pct_female_gov_health <- get_var_info(var_names = "pct", related_to = "femal", categories
↪  = c("government",
    "healthcare"))
# view the first six rows and first three columns
head(pct_female_gov_health[1:3])
```

```
##               variable                       years
## 1         pctfemaleleg 1975, 1977, 1979, 1981-2019
## 2  employer_female_pct                   2008-2017
## 3 uninsured_female_pct                   2008-2017
## 4  medicaid_female_pct                   2008-2017
##                                           short_desc
## 1          Percent of state legislators that are female
## 2 Percentage of women with employer- sponsored coverage
## 3                  Percentage of women with no coverage
## 4               Percentage of women covered by Medicaid
```

In RStudio, run the below code chunk to view all of these variable dataframes in new tabs.

```
# open dataframes in new tabs in RStudio
View(name_education)
View(desc_pollu_disease)
View(category_transportation)
View(pct_female_gov_health)
```

We'll use the `variable` column of the filtered dataframes, through the `$` operator to get the actual data from the `all_data` dataframe. You can return the data for these variables for all states and all years, or further filter the data by state and year. You can also filter for the specified variables as well as all the variables of a category using the `var_category` argument.

```
# return data for all states and all years for the variables in name_education
education_data <- get_cspp_data(vars = name_education$variable)

# show the last six rows note: we didn't use this line of code with the pipe
# (%>%) operator, because it would have SAVED the last six rows of the selected
# data, but we just want to SEE the last six rows
tail(education_data)
```

```
## # A tibble: 6 x 23
##   st    stateno state   state_fips state_icpsr  year exp_education
##   <chr>   <dbl> <chr>        <dbl>       <dbl> <dbl>         <int>
## 1 WY         50 Wyoming         56          68  2015       2010117
## 2 WY         50 Wyoming         56          68  2016       1982651
## 3 WY         50 Wyoming         56          68  2017            NA
## 4 WY         50 Wyoming         56          68  2018            NA
```

```
## 5 WY         50 Wyoming         56          68  2019            NA
## 6 WY         50 Wyoming         56          68  2020            NA
## # i 16 more variables: interstate_compact_on_educationa <dbl>,
## #   compact_for_education <dbl>, pro_education <dbl>, anti_education <dbl>,
## #   neutral_education <dbl>, education <int>,
## #   education_corporal_punishment_ba <int>, education_librarysystem <int>,
## #   education_schoolfordeaf <int>, education_teacher_cert_elementar <int>,
## #   education_teacher_cert_hs <int>, w_education_biblereading <int>,
## #   w_education_moment_of_silence <int>, ...
```

```
# return data for all states for the years 1990-2005 for the variables in
# desc_pollu_disease
pollu_disease_data <- get_cspp_data(vars = desc_pollu_disease$variable, years =
↪  1990:2005) %>%
   glimpse()
```

```
## Rows: 816
## Columns: 17
## $ st                                <chr> "AK", "AK", "AK", "AK", "AK", "AK", "~
## $ stateno                           <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ state                             <chr> "Alaska", "Alaska", "Alaska", "Alaska~
## $ state_fips                        <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ state_icpsr                       <dbl> 81, 81, 81, 81, 81, 81, 81, 81, 81, 8~
## $ year                              <dbl> 1990, 1991, 1992, 1993, 1994, 1995, 1~
## $ ig_health                         <int> 12, NA, NA, NA, NA, NA, NA, 25, 24, 2~
## $ ig_env                            <int> 9, NA, NA, NA, NA, NA, NA, 3, 3, 3, N~
## $ newbornheartscreen                <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ health_rank                       <int> NA, NA, 30, 29, 29, 27, 25, 23, 26, 2~
## $ hmbkd                             <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ hmblyme                           <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ transboundary_pollution_reciproc  <dbl> 0, 0, 0, NA, NA, NA, NA, NA, NA, NA, ~
## $ environment_air_pollution_contro  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ pollution                         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ pollutionconcentration            <dbl> NA, NA, NA, 12.45390, 10.03090, 10.28~
## $ penaltyfac                        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
```

```
# return data for Delaware, New York, and New Jersey (state abbreviations only)
# for all years for the variables in category_transportation, and additionally
# add all variables from the labor category note: both of these categories
# could have been filtered by the `categories` function of the `get_var_info`
# function and saved into a dataframe, like above, or filtered with the
# `var_category` argument directly in the `get_cspp_data` function, or a
# combination, like below
tristate_transportation_data <- get_cspp_data(states = c("DE", "NY", "NJ"), vars =
↪  category_transportation$variable,
   var_category = "labor")

# show last six rows of data
tail(tristate_transportation_data)
```

```
## # A tibble: 6 x 142
##   st    stateno state    state_fips state_icpsr  year antiinj fairemp fairtrade
##   <chr>   <dbl> <chr>         <dbl>       <dbl> <dbl>   <int>   <int>     <int>
```

```
## 1 NY         32 New York         36          13  2015      NA      NA        NA
## 2 NY         32 New York         36          13  2016      NA      NA        NA
## 3 NY         32 New York         36          13  2017      NA      NA        NA
## 4 NY         32 New York         36          13  2018      NA      NA        NA
## 5 NY         32 New York         36          13  2019      NA      NA        NA
## 6 NY         32 New York         36          13  2020      NA      NA        NA
## # i 133 more variables: miglab <int>, right2work <dbl>, sals <int>,
## #   gagexem <int>, gcomp <dbl>, gcompcov <dbl>, gcompfnd <int>, gcompjob <int>,
## #   gminraw <dbl>, gminwag <dbl>, gunion <dbl>, gcompman <int>, gdisab <int>,
## #   geadl <int>, gleave <int>, gnonc <int>, gosh <dbl>, gprev <int>,
## #   gprivins <int>, grtw <int>, gsbexem <int>, gselfins <int>, gsfund <int>,
## #   gverif <int>, labor_age_discrimination <int>,
## #   labor_antiinjunction_laws <int>, ...
```

```
# return data for California and Texas for the years 1990-2000 and 2008-2018
# for the variables in pct_female_gov_health
CA_TX_female_gov_health_data <- get_cspp_data(states = c("CA", "TX"), years =
↪  c(1990:2000,
    2008:2018), vars = pct_female_gov_health$variable) %>%
    glimpse()
```

```
## Rows: 44
## Columns: 10
## $ st                   <chr> "CA", "CA", "CA", "CA", "CA", "CA", "CA", "CA", "~
## $ stateno              <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5~
## $ state                <chr> "California", "California", "California", "Califo~
## $ state_fips           <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6~
## $ state_icpsr          <dbl> 71, 71, 71, 71, 71, 71, 71, 71, 71, 71, 71, 71, 7~
## $ year                 <dbl> 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1~
## $ pctfemaleleg         <dbl> 15.8, 17.5, 18.3, 22.5, 24.2, 20.8, 20.0, 22.5, 2~
## $ employer_female_pct  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 57, 5~
## $ uninsured_female_pct <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 21, 2~
## $ medicaid_female_pct  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 11, 1~
```

Now take a minute on your own to find some variables and filter the dataset to a manageable size based on information that interests you. Remove any arguments that you don't need (e.g., if you don't want specific categories for the variables, remove the `categories` argument from `get_var_info`, or if you want the data for all states, remove the `states` argument from `get_cspp_data`, or if you don't want variables from another category, remove the `var_category` argument from `get_cspp_data`). Feel free to change the names of the dataframes `var` and `var_data` to something more descriptive.

```
var <- get_var_info(var_names = c("", ""), related_to = c("", ""), categories = c("",
    ""))

var_data <- get_cspp_data(vars = var$variable, states = c("", ""), years = c("",
    ""), var_category = c("", ""))
```

3. Cleaning data and visualizing data relationships

Let's look at some variables in the education policy category. In this example, we're interested in the percent of students that drop out from 9th-12th grade, a variable called `eddropoutrate`. Let's see if the high school

dropout rate across states is related to the adoption of a universal pre-K policy (binary variable), whether the state adopted educational television (binary variable), whether there is a ban on corporal punishment in schools (binary variable), dollars spent on instruction per student (continuous variable), and pupil-to-teacher ratio (continuous variable).

```
# get the variables of interest by name
education_vars <- get_var_info(var_names = c("universalprek", "edutv",
↪  "education_corporal_punishment_ba",
   "edinstruct_expend_pstud", "pupilteachratio", "eddropoutrate"))

# view first three columns of the variable
education_vars[1:3]
```

```
##                           variable     years
## 1                            edutv 1913-2010
## 2 education_corporal_punishment_ba 1970-2014
## 3                     universalprek 1995-2017
## 4          edinstruct_expend_pstud 1986-2008
## 5                     eddropoutrate 2001-2009
## 6                   pupilteachratio 1987-2010
##                                     short_desc
## 1                          Educational television
## 2           Ban on corporal punishment in schools
## 3                       Universal pre-K education
## 4 Educational instruction expenditure per student
## 5    Percent school dropout rate 9th- 12th  grade
## 6                          Pupil-to-teacher ratio
```

```
# return data based on chosen variables and the years that overlap for which
# data is available for all variables
education_data <- get_cspp_data(vars = education_vars$variable, years = 2001:2008)

# view first 10 rows of the dataframe
head(education_data, n = 10)
```

```
## # A tibble: 10 x 12
##    st    stateno state state_fips state_icpsr  year edutv education_corporal_p~1
##    <chr>   <dbl> <chr>      <dbl>       <dbl> <dbl> <int>                   <int>
##  1 AK          2 Alas~          2          81  2001     0                       1
##  2 AK          2 Alas~          2          81  2002     0                       1
##  3 AK          2 Alas~          2          81  2003     0                       1
##  4 AK          2 Alas~          2          81  2004     0                       1
##  5 AK          2 Alas~          2          81  2005     0                       1
##  6 AK          2 Alas~          2          81  2006     0                       1
##  7 AK          2 Alas~          2          81  2007     0                       1
##  8 AK          2 Alas~          2          81  2008     0                       1
##  9 AL          1 Alab~          1          41  2001     1                       0
## 10 AL          1 Alab~          1          41  2002     1                       0
## # i abbreviated name: 1: education_corporal_punishment_ba
## # i 4 more variables: universalprek <dbl>, edinstruct_expend_pstud <int>,
## #   eddropoutrate <dbl>, pupilteachratio <dbl>
```

```r
str(education_data)
```

```
## tibble [408 x 12] (S3: tbl_df/tbl/data.frame)
##  $ st                                : chr [1:408] "AK" "AK" "AK" "AK" ...
##  $ stateno                           : num [1:408] 2 2 2 2 2 2 2 2 1 1 ...
##  $ state                             : chr [1:408] "Alaska" "Alaska" "Alaska" "Alaska"
...
##  $ state_fips                        : num [1:408] 2 2 2 2 2 2 2 2 1 1 ...
##  $ state_icpsr                       : num [1:408] 81 81 81 81 81 81 81 81 41 41 ...
##  $ year                              : num [1:408] 2001 2002 2003 2004 2005 ...
##  $ edutv                             : int [1:408] 0 0 0 0 0 0 0 0 1 1 ...
##  $ education_corporal_punishment_ba: int [1:408] 1 1 1 1 1 1 1 1 0 0 ...
##  $ universalprek                     : num [1:408] 0 0 0 0 0 0 0 0 0 0 ...
##  $ edinstruct_expend_pstud           : int [1:408] 5617 5740 5829 6262 6562 7021 8528
8599 3692 3812 ...
##  $ eddropoutrate                     : num [1:408] 8.2 8.1 7.6 7 8.2 8 7.3 7.3 4.1 3.7
...
##  $ pupilteachratio                   : num [1:408] 16.9 16.7 16.6 17.2 17.1 16.8 16.8
17.2 15.4 15.8 ...
```

```r
# Uncomment and run this line of code in RStudio to view in a new tab
# View(education_data)
```

Some rows of the dataframe are missing data for certain variables. Additionally, some variables that should be binary categorical (`edutv`, `education_corporal_punishmen_ba`, `universalprek`) are being read by R as integer or numerical variables rather than factor variables (what R calls categorical variables). Below, we'll clean the dataset to remove the rows with any missing data, change variable class to be factors as necessary, as well as rename some of the longer variable names to something shorter.

```r
# save the cleaned data in a new dataframe
education_data_clean <- education_data %>%
    # rename some variables using the syntax newname = oldname
rename(punishmentban = education_corporal_punishment_ba, spendperstudent =
↪  edinstruct_expend_pstud) %>%
    # change the necessary variables into factors
mutate(edutv = as.factor(edutv), punishmentban = as.factor(punishmentban), universalprek
↪  = as.factor(universalprek)) %>%
    # remove any rows that have any missing data, i.e. any rows for which any
    # of the variables contain 'NA'
na.omit()

# compare the number of rows of the original education data vs. the cleaned
# version to see that 27 rows were removed
nrow(education_data)  # 408
```

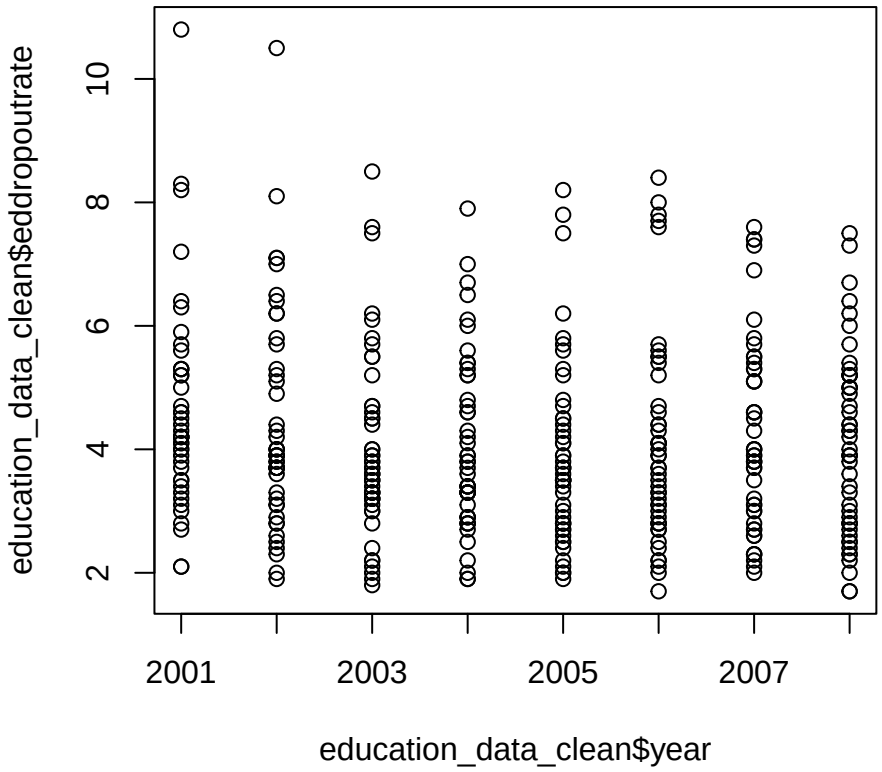
```
## [1] 408
```

```r
nrow(education_data_clean)  # 381
```

```
## [1] 381
```

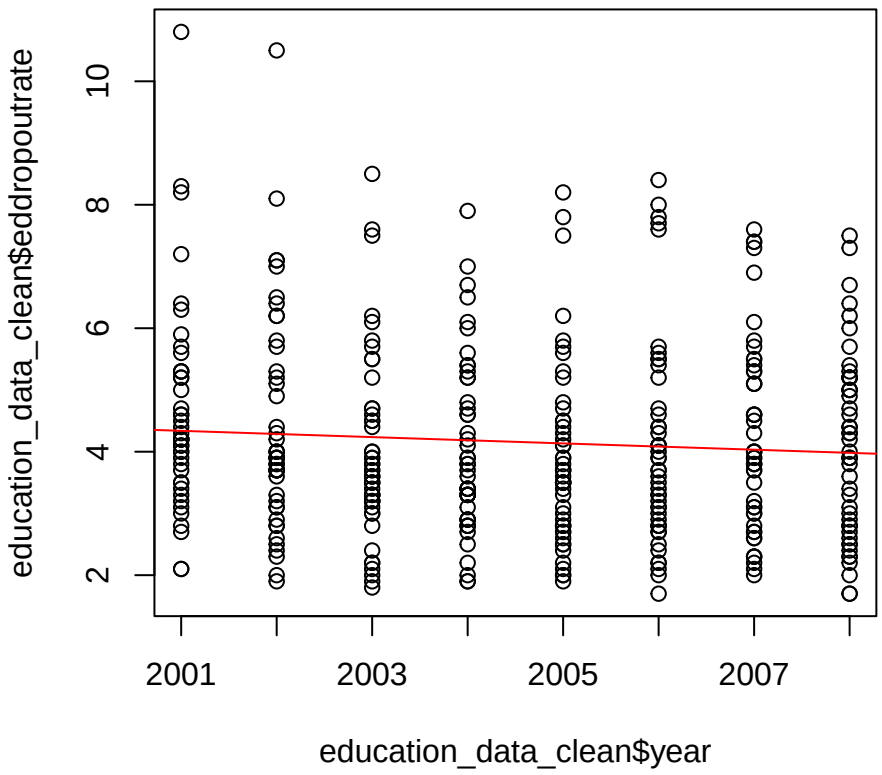
Visualizing data using base R code is as simple as using the `plot` function. All you need to supply is the x and y values by calling the specific variable of interest in our specified dataframe using the `$` operator. The `plot` function can automatically read the type of variable and plots the data accordingly. The plot below visualizes the relationship between year and high school dropout rate

```
# plot high school dropout rate over time as a scatterplot
plot(x = education_data_clean$year, y = education_data_clean$eddropoutrate)
```



We can also plot the line that represents dropout rate as a function of time by adding a regression line over the data. This is done by using the `abline` function (as in, $y = a + bx$) with the linear model `lm` function that figures out the line that best fits the data of dropout rate as a function . The syntax of the `lm` function is $y \sim x$.

```
# plot high school dropout rate over time as a scatterplot with a regression
# line
plot(x = education_data_clean$year, y = education_data_clean$eddropoutrate) +
↪  abline(lm(education_data_clean$eddropoutrate ~
   education_data_clean$year), col = "red")
```

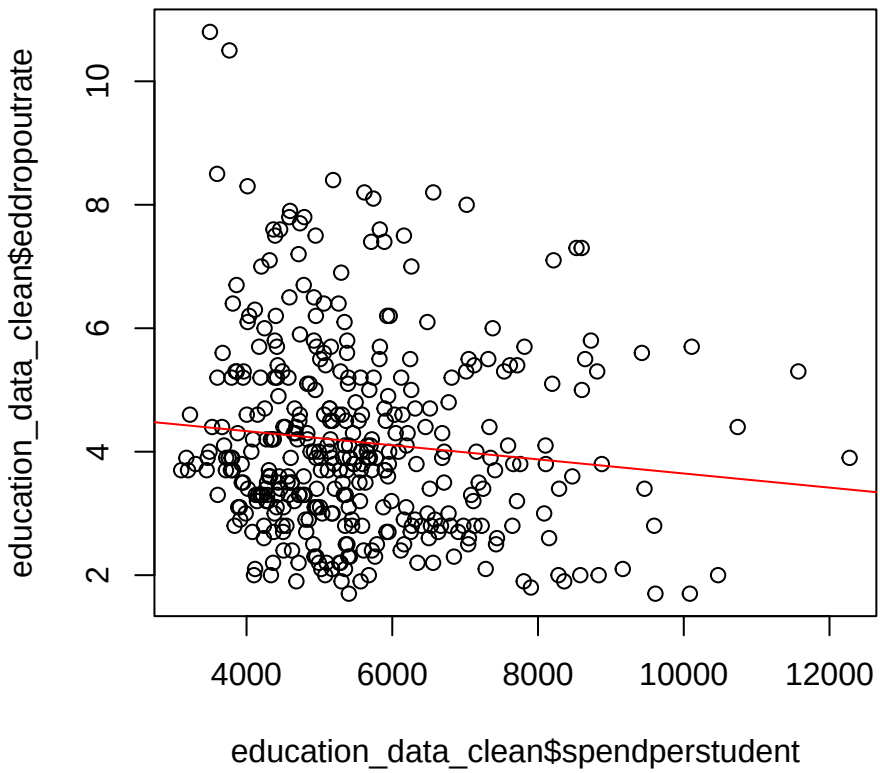



```
## integer(0)
```

The regression line, in red, shows a slight negative correlation between year and dropout rate, indicating that dropout rate seems to decline over the years 2001-2008, but the line is rather flat.

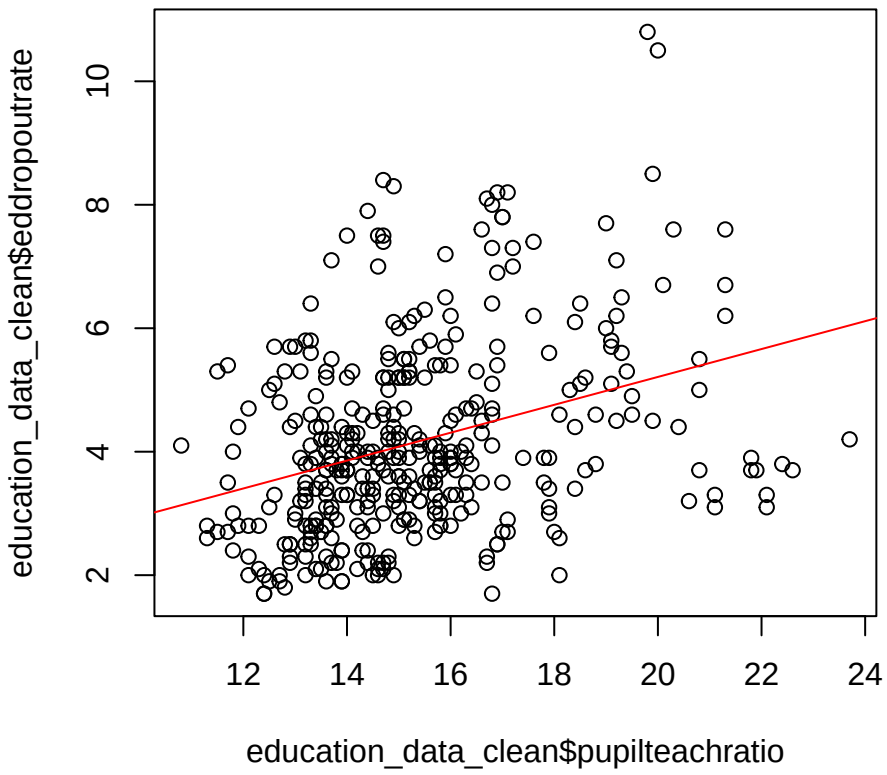
The below code outputs two plots that visualize the data relationships for dropout rate as a function of instructional expenditure per student and pupil-to-teacher ratio, respectively. These two variables seem more promising than year in explaining change in dropout rate across states and time. Spend per student has a negative correlation with dropout rate, suggesting that high school dropout rates decrease as expenditure per student increases, while student-teacher ratio has a positive relationship, suggesting that as as student-teacher ratio increases (more students assigned to a single teacher), the dropout rate increases.

```
# plot the relationship between instructional expenditure per student and high
# school dropout rate as a scatterplot with a regression line
plot(x = education_data_clean$spendperstudent, y = education_data_clean$eddropoutrate) +
    abline(lm(education_data_clean$eddropoutrate ~ education_data_clean$spendperstudent),
        col = "red")
```




```
## integer(0)
```

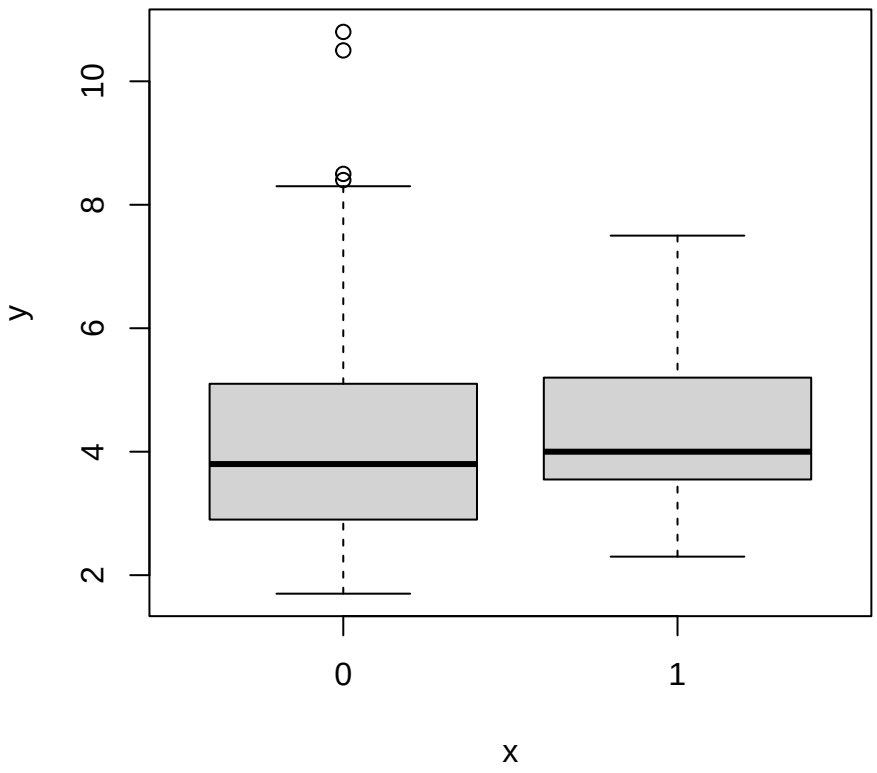
```
# plot the relationship between pupil-to-teacher ratio and high school dropout
# rate as a scatterplot with a regression line
plot(x = education_data_clean$pupilteachratio, y = education_data_clean$eddropoutrate) +
    abline(lm(education_data_clean$eddropoutrate ~ education_data_clean$pupilteachratio),
        col = "red")
```



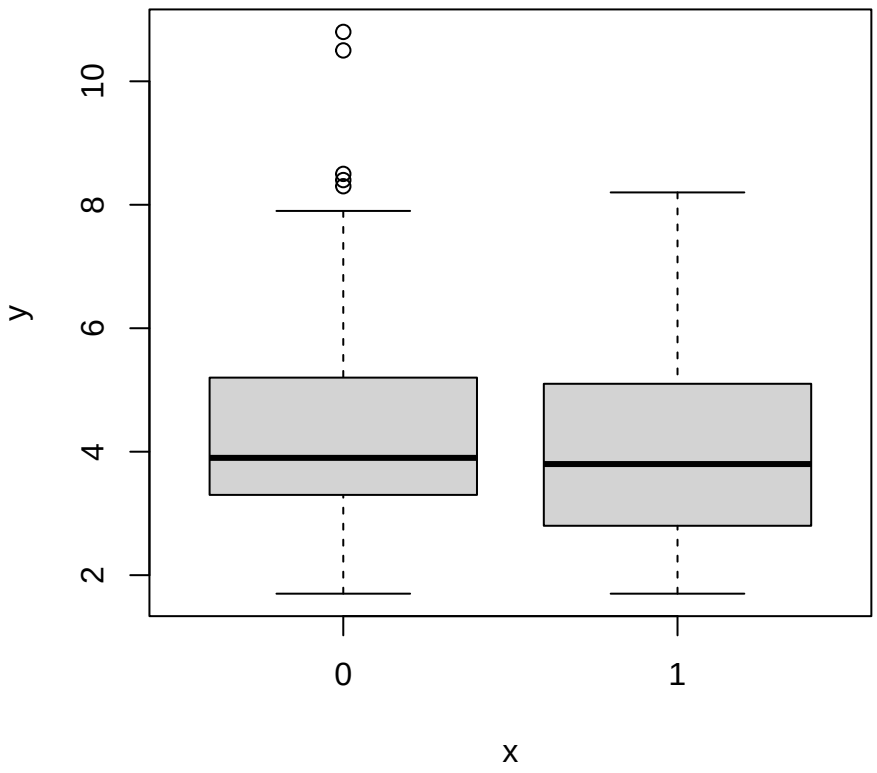

```
## integer(0)
```

We can also plot the categorical variables as the x variable using the same function, which produces boxplots. These plots show that these policies doesn't seem to have much of an effect on the mean dropout rate across the years various states have adopted these policies ($x = 1$) and the years various states have no adopted the policy ($x = 0$). that have adopted these policies ($x = 1$), however, the spread of data is generally smaller when the policies are adopted. It is difficult to determine whether these relationships are meaningful just from visualizing the data, which is the purpose of running linear regression models in the next section.

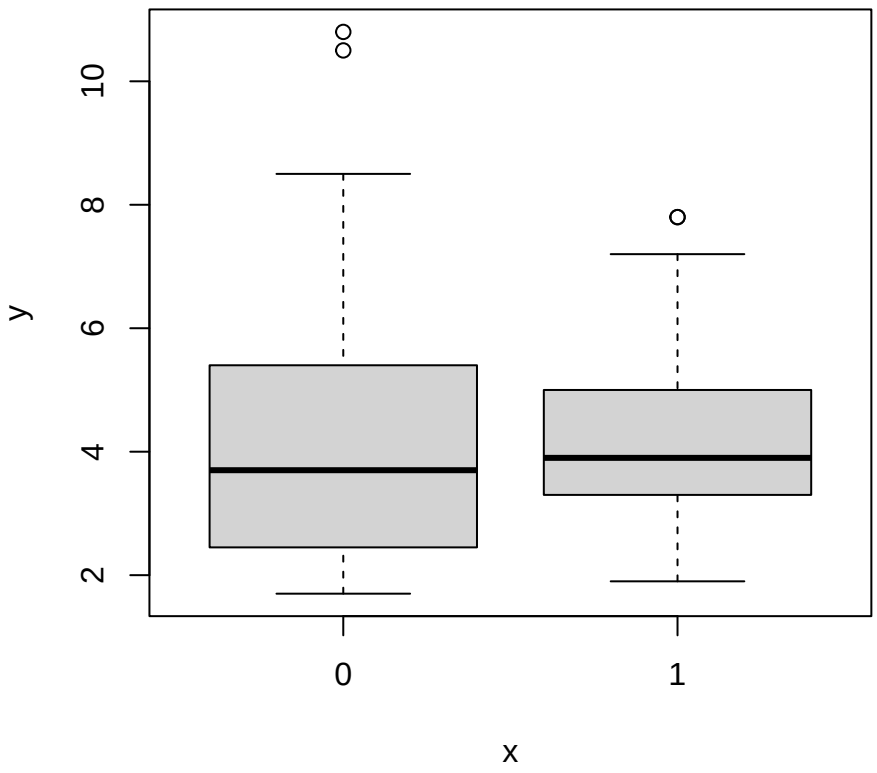
```
# plot the relationship between adoption of universal prek (1 = policy adopted)
# and highschool dropout rate as a boxplot
plot(x = education_data_clean$universalprek, y = education_data_clean$eddropoutrate)
```




```
# plot the relationship between banning corporal punishment in schools (1 =
# banned) and high school dropout rate as a boxplot
plot(x = education_data_clean$punishmentban, y = education_data_clean$eddropoutrate)
```

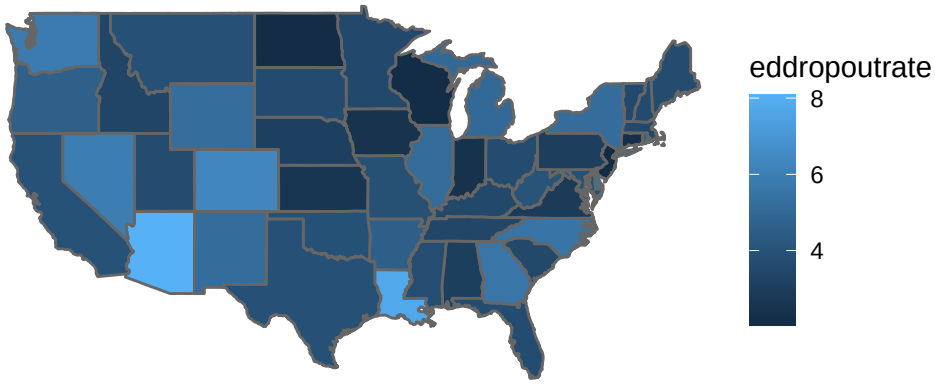



```
# plot the relationship between adoption of educational television (1 = policy
# adopted) and high school dropout rate as a boxplot
plot(x = education_data_clean$edutv, y = education_data_clean$eddropoutrate)
```

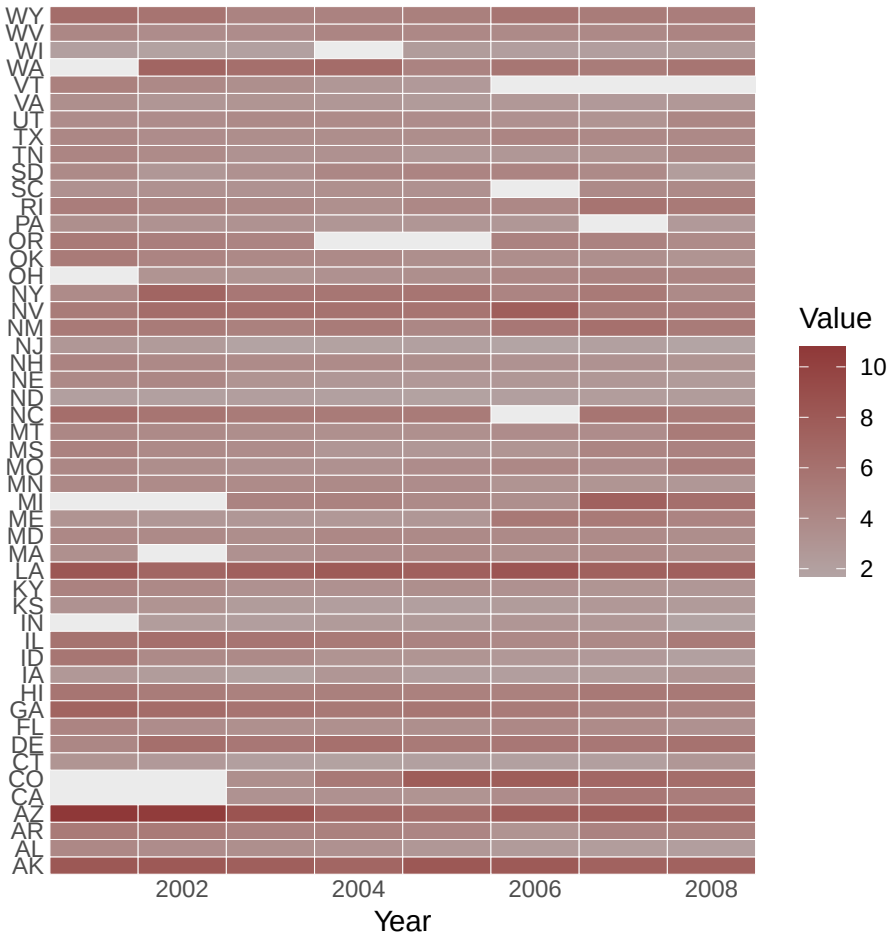


In addition to being able to plot relationships between variables using base R, the `cspp` package also comes with built in visualization functions: `generate_map` to plot the data on a map, `plot_panel` to plot time series data, and `corr_plot` to plot correlation data between continuous variables. The first two are useful if you want to visualize the data by state, and the last one is useful if you want to compare correlations between multiple continuous variables.

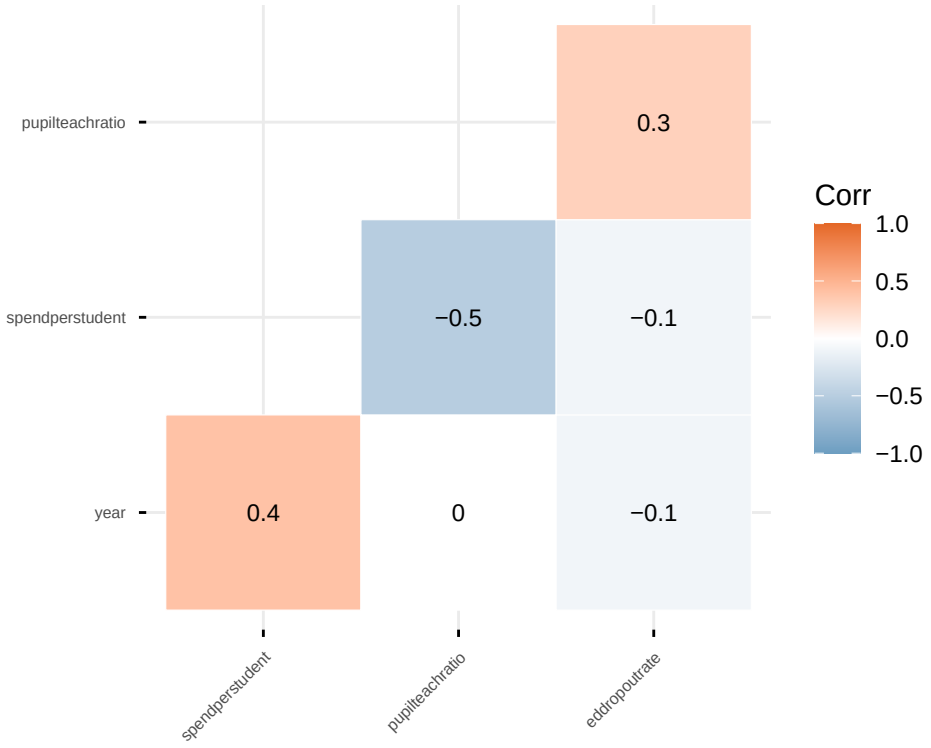
```
# averages over the years included in the specified data
generate_map(cspp_data = education_data_clean, var_name = "eddropoutrate", average_years
↪  = TRUE)
```




```
# plots data across years per state
plot_panel(cspp_data = education_data_clean, var_name = "eddropoutrate")
```




```
# plots correlation matrix between continuous variables
corr_plot(data = education_data_clean, vars = c("year", "spendperstudent",
↪  "pupilteachratio",
    "eddropoutrate"), summarize = FALSE)
```



4. Regression modeling syntax and reporting regression analysis output

Let's finally model the linear relationship between high school dropout rate and the other continuous and categorical variables. Standard linear regression is designed for continuous dependent variables, but can take any number of indepentn continuous and/or categorical variables. Using the same `lm` function we used with the `abline` function during visualization and the same `y ~ x` syntax, we can look at the **slope** of these independent variables, which represents the **effect size** of the variables in relation to the dependent variable. With the `lm` function, we can explicitly specify the dataframe and just use the variable names in the for the `forumla` argument.

```
# run a linear regression model for the effect of expenditure per student, a
# continuous variable, on dropout rate
lm(data = education_data_clean, formula = eddropoutrate ~ spendperstudent)
```

```
##
## Call:
## lm(formula = eddropoutrate ~ spendperstudent, data = education_data_clean)
##
## Coefficients:
```

```
##     (Intercept)  spendperstudent
##       4.7907398       -0.0001144
```

```
# run a linear regression model for the effect of banning corporal punishment
# in schools, a categorical variable, on dropout rate
lm(data = education_data_clean, eddropoutrate ~ punishmentban)
```

```
##
## Call:
## lm(formula = eddropoutrate ~ punishmentban, data = education_data_clean)
##
## Coefficients:
##    (Intercept)  punishmentban1
##         4.3190         -0.3129
```

The direct output of the `lm` function prints the coefficients of the model, so the intercept and the variable slope. For the continuous variable, the intercept represents the predicted value of the y variable when the x variable is 0, and the slope represents the average change in the value of the y variable with each 1-unit increase in the x variable.

For the categorical variable, the same concepts apply, but "0" and "1-unit increase" have different conceptualizations. The intercept represents the mean value of the y variable at the chosen reference level, which R automatically sets according to usually intuitive standards, although it can be manually changed. Because these variables were dummy coded where 0 = 'policy not adopted' and 1 = 'policy adopted', R has automatically set the 0 code as the reference level, although even if the variable were not dummy coded, R will automatically code any factor variable for the purpose of linear regression. The slope, or "1-unit increase" in the x variable, represents the mean value of the y variable when $x = 1$.

Note: In our example, the x variable is listed as "punishmentban1" because that is the name of the variable and the label representing the level that is compared to the reference level. If the labels of the data had been "notadopted" and "adopted", the variable in the regression model output would have read "punishmentbanadopted".

For statistical analysis using linear regression, the values themselves don't mean much. To get the results of the analysis on the regression model, run the `summary` function on the model output.

```
lm(data = education_data_clean, formula = eddropoutrate ~ spendperstudent) %>%
    summary()  # gives the summary of the model run to the left of the pipe (%>%)
    ↪  operator
```

```
##
## Call:
## lm(formula = eddropoutrate ~ spendperstudent, data = education_data_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.4726 -1.1238 -0.3085  0.8887  6.4094
##
## Coefficients:
##                    Estimate  Std. Error t value Pr(>|t|)
## (Intercept)      4.79073979  0.30603593  15.654   <2e-16 ***
## spendperstudent -0.00011439  0.00005336  -2.144   0.0327 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.553 on 379 degrees of freedom
## Multiple R-squared:  0.01198,    Adjusted R-squared:  0.009375
## F-statistic: 4.596 on 1 and 379 DF,  p-value: 0.03268
```

```r
# you can also save the model into a variable and run the summary on the saved
# model
model <- lm(data = education_data_clean, eddropoutrate ~ punishmentban)
summary(model)
```

```
##
## Call:
## lm(formula = eddropoutrate ~ punishmentban, data = education_data_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6190 -1.1190 -0.3061  0.8810  6.4810
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.3190     0.1146  37.676   <2e-16 ***
## punishmentban1  -0.3129     0.1594  -1.963   0.0504 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.555 on 379 degrees of freedom
## Multiple R-squared:  0.01006,    Adjusted R-squared:  0.007452
## F-statistic: 3.853 on 1 and 379 DF,  p-value: 0.05039
```

The summary provides a lot more information about the linear regression model. The most pertinent information will still be under the "Coefficients" header. The "Estimate" is the values of the intercept and slope of the x variable, followed by the standard error of the estimate, which is the uncertainty of the estimate, the t value, which is the coefficient estimate divided by the standard error, and p-value. A p-value less than 0.05 is typically accepted as significant. For a more detailed breakdown of the model summary output, see this blog: https://feliperego.github.io/blog/2015/10/23/Interpreting-Model-Output-In-R.

When reporting the results of linear regression, it is important to report the value of the coefficient itself along with the standard error and the p-value. For example:

> There was a statistically significant relationship between instructional expenditure per student and high school dropout rate (b = -0.00011, SE = 0.000053, p = 0.032), which suggested that each additional dollar spent per student was associated with a reduction of the dropout rate by 0.00011 percentage points.

> There relationship between the adoption of a policy banning corporal punishment in schools was marginally significant (b = -0.31, SE = 0.16, p = 0.05), suggesting that the banning of corporal punishment in schools was associated with a reduction of the dropout rate by 0.31 percentage points.

The "Multiple R-squared" statistic provides a measure of how well the independent variables explained the variance in the dependent variable. For both of these two models, the R^2 value was around 0.01, meaning that spend per student and punishment ban each predicted about 1% of the variance in high school dropout

rates. This indicates that these aren't very good models, or at the very least, more variables are needed to explain the remaining 99% of variance in the depedent variable.

Multiple linear regression, also known as multivariable regression, involves looking at the effect of multiple independent variables on one dependent variable (not to be confused with multi*variate* regression, which describes a regression model with multiple dependent variables) and is as simple as adding additional variables to the regression formula using the plus sign (+) for additive effects or an asterisk (*) for interactive effects. We'll just look at additive effects for now by first adding another continuous variable.

```
# run and save a linear model looking at the effect of three variables on
# dropout rate
model1 <- lm(data = education_data_clean, eddropoutrate ~ spendperstudent + punishmentban
↪  +
    pupilteachratio)
summary(model1)
```

```
##
## Call:
## lm(formula = eddropoutrate ~ spendperstudent + punishmentban +
##     pupilteachratio, data = education_data_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9869 -1.0735 -0.2723  0.8565  5.5993
##
## Coefficients:
##                    Estimate  Std. Error t value Pr(>|t|)
## (Intercept)     -0.24188899  0.78233133  -0.309   0.7573
## spendperstudent  0.00012761  0.00006732   1.896   0.0588 .
## punishmentban1  -0.34229861  0.18062773  -1.895   0.0589 .
## pupilteachratio  0.25233398  0.03683312   6.851    3e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.467 on 377 degrees of freedom
## Multiple R-squared:  0.1235, Adjusted R-squared:  0.1165
## F-statistic:  17.7 on 3 and 377 DF,  p-value: 9.036e-11
```

Here, we can see that the estimates of both spend per student (0.00013) and punishment ban (-0.34) are different than they were before when they were modeled on their own (-0.0001 and -0.31, respectively). Their p-values also slightly increased. The reason becomes clear when looking at the effect of pupil-to-teacher ratio, which has a p-value of 3.0×10^{-11}. This indicates that the effect of pupil-to-teacher ratio is significant in explaining the variance in the data such that spend per student and punishment ban had a smaller impact on explaining the data than we originally thought. This is also evidence when we look at the R^2 value. Because R^2 always increases when adding more independent variables, we need to look at the "Adjusted R-squared" value, which suggests that the model now explains nearly 12% of the variance in the data. Let's more variables to the model to see how good we can get at explaining the dropout rate data.

```
# run and save a linear model looking at the effect of all 5 variables on
# dropout rate
model2 <- lm(data = education_data_clean, formula = eddropoutrate ~ spendperstudent +
    punishmentban + pupilteachratio + universalprek + edutv)
summary(model2)
```

```
##
## Call:
## lm(formula = eddropoutrate ~ spendperstudent + punishmentban +
##     pupilteachratio + universalprek + edutv, data = education_data_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9481 -1.0349 -0.2538  0.8234  5.6054
##
## Coefficients:
##                    Estimate  Std. Error t value Pr(>|t|)
## (Intercept)     -0.19759668  0.79847480  -0.247   0.8047
## spendperstudent  0.00011104  0.00006862   1.618   0.1065
## punishmentban1  -0.31451813  0.18410042  -1.708   0.0884 .
## pupilteachratio  0.25271375  0.03716699   6.799 4.16e-11 ***
## universalprek1   0.34711113  0.23579933   1.472   0.1418
## edutv1          -0.02517943  0.16660990  -0.151   0.8800
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.467 on 375 degrees of freedom
## Multiple R-squared:  0.1285, Adjusted R-squared:  0.1169
## F-statistic: 11.06 on 5 and 375 DF,  p-value: 5.922e-10
```

```
# run and save a linear model looking at the effect of all 5 variables and time
# on dropout rate
model3 <- lm(data = education_data_clean, formula = eddropoutrate ~ spendperstudent +
    punishmentban + pupilteachratio + universalprek + edutv + year)
summary(model3)
```

```
##
## Call:
## lm(formula = eddropoutrate ~ spendperstudent + punishmentban +
##     pupilteachratio + universalprek + edutv + year, data = education_data_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7117 -0.9961 -0.2073  0.8844  5.2603
##
## Coefficients:
##                    Estimate  Std. Error t value Pr(>|t|)
## (Intercept)     203.8210200  76.0015924   2.682  0.00765 **
## spendperstudent   0.0002160   0.0000785   2.752  0.00621 **
## punishmentban1   -0.4586705   0.1903274  -2.410  0.01644 *
## pupilteachratio   0.2748350   0.0377729   7.276 2.04e-12 ***
## universalprek1    0.3684209   0.2340065   1.574  0.11624
## edutv1           -0.0651381   0.1659170  -0.393  0.69484
## year             -0.1021883   0.0380654  -2.685  0.00759 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.455 on 374 degrees of freedom
## Multiple R-squared:  0.145,  Adjusted R-squared:  0.1313
## F-statistic: 10.57 on 6 and 374 DF,  p-value: 7.499e-11
```

These two models show that without the year variable, `model2`, pupil-to-teacher is the only significant predictor of the dropout data, and the model is able to explain almost 12% of the variance in the data (adjusted R-squared = 0.1169). Interestingly, with the year variable in `model3`, spend per student and punishment ban become significant predictors as well (it is safe to ignore the intercept estimate, as it would be interpreted as the mean dropout rate for year 0, i.e., 2000 years ago). We see that `model3` also explains the data a little better, being able to explain about 13% of the data (adjusted R-squared = 0.1313).

But how can we tell if the difference between explaining a little under 12% of the data to a little over 13% of the data is a meaningful difference?

There is another way to compare models instead of just looking at the R^2 values. Running the `anova` function on linear models that were run on the same data runs a statistical test to compare the amount of variance explained by each model (this does not work on models run on different data, including if the model automatically filtered out missing data). The output compares each model to the one immediately above it, and if the *p*-value is greater than 0.05, the model did not significantly do better in explaining the variance of the data than the model immediately above it in the list, but if the *p*-value is less 0.05, we can reject the null hypothesis and have confidence that the model explains the data significantly better than the model it was compared to. Models that better explain the data generally have lower "RSS" values (which you can think of being analogous to the Estimate value from the linear regression model output).

```
anova(model1, model2, model3)
```

```
## Analysis of Variance Table
##
## Model 1: eddropoutrate ~ spendperstudent + punishmentban + pupilteachratio
## Model 2: eddropoutrate ~ spendperstudent + punishmentban + pupilteachratio +
##     universalprek + edutv
## Model 3: eddropoutrate ~ spendperstudent + punishmentban + pupilteachratio +
##     universalprek + edutv + year
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1    377 811.45
## 2    375 806.75  2    4.7005 1.1105 0.330465
## 3    374 791.50  1   15.2517 7.2068 0.007586 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results of this comparison indicate that `model3` is indeed the model that best explains the data among the other models we ran today.

References

Caleb Lucas and Joshua McCrain (2020). cspp: A Package for The Correlates of State Policy Project Data. R package version 0.3.3. https://github.com/IPPSR/cspp