I am currently working on a package for the R programming language; its initial goal was to simply  
distribute the data used in the Card and Krueger 1994 paper that you can read  
[here](http://davidcard.berkeley.edu/papers/njmin-aer.pdf) (PDF warning). However, I decided that I  
would add code to perform diff-in-diff.

In my [previous blog post](https://www.brodrigues.co/blog/2019-04-28-diffindiff_part1/) I showed  
how to set up the structure of your new package. In this blog post, I will only focus on getting  
Card and Krueger’s data and prepare it for distribution. The next blog posts will focus on writing  
a function to perform difference-in-differences.

If you want to distribute data through a package, you first need to use the usethis::use\_data\_raw()  
function (as shown in part 1).

This creates a data-raw folder, and inside you will find the DATASET.R script. You can edit this  
script to prepare the data.

First, let’s download the data from Card’s website, unzip it and load the data into R. All these  
operations will be performed from R:

library(tidyverse)

tempfile\_path <- tempfile()

download.file("http://davidcard.berkeley.edu/data\_sets/njmin.zip", destfile = tempfile\_path)

tempdir\_path <- tempdir()

unzip(tempfile\_path, exdir = tempdir\_path)

To download and unzip a file from R, first, you need to define where you want to save the file. Because  
I am not interested in keeping the downloaded file, I use the tempfile() function to get a temporary  
file in my /tmp/ folder (which is the folder that contains temporary files and folders in a GNU+Linux  
system). Then, using download.file() I download the file, and save it in my temporary file. I then  
create a temporary directory using tempdir() (the idea is the same as with tempfile()), and use  
this folder to save the files that I will unzip, using the unzip() function. This folder now contains  
several files:

check.sas

codebook

public.csv

read.me

survey1.nj

survey2.nj

check.sas is the SAS script Card and Krueger used. It’s interesting, because it is quite simple,  
quite short (170 lines long) and yet the impact of Card and Krueger’s research was and has been  
very important for the field of econometrics. This script will help me define my own functions.  
codebook, you guessed it, contains the variables’ descriptions. I will use this to name the columns  
of the data and to write the dataset’s documentation.

public.csv is the data. It does not contain any column names:

46 1 0 0 0 0 0 1 0 0 0 30.00 15.00 3.00 . 19.0 . 1 . 2 6.50 16.50 1.03 1.03 0.52 3 3 1 1 111792 1 3.50 35.00 3.00 4.30 26.0 0.08 1 2 6.50 16.50 1.03 . 0.94 4 4

49 2 0 0 0 0 0 1 0 0 0 6.50 6.50 4.00 . 26.0 . 0 . 2 10.00 13.00 1.01 0.90 2.35 4 3 1 1 111292 . 0.00 15.00 4.00 4.45 13.0 0.05 0 2 10.00 13.00 1.01 0.89 2.35 4 4

506 2 1 0 0 0 0 1 0 0 0 3.00 7.00 2.00 . 13.0 0.37 0 30.0 2 11.00 10.00 0.95 0.74 2.33 3 3 1 1 111292 . 3.00 7.00 4.00 5.00 19.0 0.25 . 1 11.00 11.00 0.95 0.74 2.33 4 3

56 4 1 0 0 0 0 1 0 0 0 20.00 20.00 4.00 5.00 26.0 0.10 1 0.0 2 10.00 12.00 0.87 0.82 1.79 2 2 1 1 111492 . 0.00 36.00 2.00 5.25 26.0 0.15 0 2 10.00 12.00 0.92 0.79 0.87 2 2

61 4 1 0 0 0 0 1 0 0 0 6.00 26.00 5.00 5.50 52.0 0.15 1 0.0 3 10.00 12.00 0.87 0.77 1.65 2 2 1 1 111492 . 28.00 3.00 6.00 4.75 13.0 0.15 0 2 10.00 12.00 1.01 0.84 0.95 2 2

62 4 1 0 0 0 0 1 0 0 2 0.00 31.00 5.00 5.00 26.0 0.07 0 45.0 2 10.00 12.00 0.87 0.77 0.95 2 2 1 1 111492 . . . . . 26.0 . 0 2 10.00 12.00 . 0.84 1.79 3 3

Missing data is defined by . and the delimiter is the space character. read.me is a README file.  
Finally, survey1.nj and survey2.nj are the surveys that were administered to the fast food  
restaurants’ managers; one in February (before the raise) and the second one in November  
(after the minimum wage raise).

The next lines import the codebook:

codebook <- read\_lines(file = paste0(tempdir\_path, "/codebook"))

variable\_names <- codebook %>%

`[`(8:59) %>%

`[`(-c(5, 6, 13, 14, 32, 33)) %>%

str\_sub(1, 13) %>%

str\_squish() %>%

str\_to\_lower()

Once I import the codebook, I select lines 8 to 59 using the `[`() function.  
If you’re not familiar with this notation, try the following in a console:

seq(1, 100)[1:10]

## [1] 1 2 3 4 5 6 7 8 9 10

and compare:

seq(1, 100) %>%

`[`(., 1:10)

## [1] 1 2 3 4 5 6 7 8 9 10

both are equivalent, as you can see. You can also try the following:

1 + 10

## [1] 11

1 %>%

`+`(., 10)

## [1] 11

Using the same trick, I remove lines that I do not need, and then using stringr::str\_sub(1, 13)  
I only keep the first 13 characters (which are the variable names, plus some white space characters)  
and then, to remove all the unneeded white space characters I use stringr::squish(), and then  
change the column names to lowercase.

I then load the data, and add the column names that I extracted before:

dataset <- read\_table2(paste0(tempdir\_path, "/public.dat"),

col\_names = FALSE)

dataset <- dataset %>%

select(-X47) %>%

`colnames<-`(., variable\_names) %>%

mutate\_all(as.numeric) %>%

mutate(sheet = as.character(sheet))

I use the same trick as before. I rename the 47th column, which is empty,  
I name the columns with `colnames<-`().

After this, I perform some data cleaning. It’s mostly renaming categories of categorical variables,  
and creating a “true” panel format. Several variables were measured at several points in time. Variables  
that were measured a second time have a “2” at the end of their name. I remove these variables,  
and add an observation data variable. So my data as twice as many rows as the original data, but  
that format makes it way easier to work with. Below you can read the full code:

Click if you want to see the code

dataset <- dataset %>%

mutate(chain = case\_when(chain == 1 ~ "bk",

chain == 2 ~ "kfc",

chain == 3 ~ "roys",

chain == 4 ~ "wendys")) %>%

mutate(state = case\_when(state == 1 ~ "New Jersey",

state == 0 ~ "Pennsylvania")) %>%

mutate(region = case\_when(southj == 1 ~ "southj",

centralj == 1 ~ "centralj",

northj == 1 ~ "northj",

shore == 1 ~ "shorej",

pa1 == 1 ~ "pa1",

pa2 == 1 ~ "pa2")) %>%

mutate(meals = case\_when(meals == 0 ~ "None",

meals == 1 ~ "Free meals",

meals == 2 ~ "Reduced price meals",

meals == 3 ~ "Both free and reduced price meals")) %>%

mutate(meals2 = case\_when(meals2 == 0 ~ "None",

meals2 == 1 ~ "Free meals",

meals2 == 2 ~ "Reduced price meals",

meals2 == 3 ~ "Both free and reduced price meals")) %>%

mutate(status2 = case\_when(status2 == 0 ~ "Refused 2nd interview",

status2 == 1 ~ "Answered 2nd interview",

status2 == 2 ~ "Closed for renovations",

status2 == 3 ~ "Closed permanently",

status2 == 4 ~ "Closed for highway construction",

status2 == 5 ~ "Closed due to Mall fire")) %>%

mutate(co\_owned = if\_else(co\_owned == 1, "Yes", "No")) %>%

mutate(bonus = if\_else(bonus == 1, "Yes", "No")) %>%

mutate(special2 = if\_else(special2 == 1, "Yes", "No")) %>%

mutate(type2 = if\_else(type2 == 1, "Phone", "Personal")) %>%

select(sheet, chain, co\_owned, state, region, everything()) %>%

select(-southj, -centralj, -northj, -shore, -pa1, -pa2) %>%

mutate(date2 = lubridate::mdy(date2)) %>%

rename(open2 = open2r) %>%

rename(firstinc2 = firstin2)

dataset1 <- dataset %>%

select(-ends\_with("2"), -sheet, -chain, -co\_owned, -state, -region, -bonus) %>%

mutate(type = NA\_character\_,

status = NA\_character\_,

date = NA)

dataset2 <- dataset %>%

select(ends\_with("2")) %>%

#mutate(bonus = NA\_character\_) %>%

rename\_all(~str\_remove(., "2"))

other\_cols <- dataset %>%

select(sheet, chain, co\_owned, state, region, bonus)

other\_cols\_1 <- other\_cols %>%

mutate(observation = "February 1992")

other\_cols\_2 <- other\_cols %>%

mutate(observation = "November 1992")

dataset1 <- bind\_cols(other\_cols\_1, dataset1)

dataset2 <- bind\_cols(other\_cols\_2, dataset2)

njmin <- bind\_rows(dataset1, dataset2) %>%

select(sheet, chain, state, region, observation, everything())

The line I would like to comment is the following:

dataset %>%

select(-ends\_with("2"), -sheet, -chain, -co\_owned, -state, -region, -bonus)

This select removes every column that ends with the character “2” (among others). I split the data  
in two, to then bind the rows together and thus create my long dataset. I then save the data  
into the data/ folder:

usethis::use\_data(njmin, overwrite = TRUE)

This saves the data as an .rda file. To enable users to read the data by typing data("njmin"),  
you need to create a data.R script in the R/ folder. You can read my data.R script below:

Click if you want to see the code

#' Data from the Card and Krueger 1994 paper \*Minimum Wages and Employment: A Case Study of the Fast-Food Industry in New Jersey and Pennsylvania\*

#'

#' This dataset was downloaded and distributed with the permission of David Card. The original

#' data contains 410 observations and 46 variables. The data distributed in this package is

#' exactly the same, but was changed from a wide to a long dataset, which is better suited for

#' manipulation with \*tidyverse\* functions.

#'

#' @format A data frame with 820 rows and 28 variables:

#' \describe{

#' \item{\code{sheet}}{Sheet number (unique store id).}

#' \item{\code{chain}}{The fastfood chain: bk is Burger King, kfc is Kentucky Fried Chicken, wendys is Wendy's, roys is Roy Rogers.}

#' \item{\code{state}}{State where the restaurant is located.}

#' \item{\code{region}}{pa1 is northeast suburbs of Phila, pa2 is Easton etc, centralj is central NJ, northj is northern NJ, southj is south NJ.}

#' \item{\code{observation}}{Date of first (February 1992) and second (November 1992) observation.}

#' \item{\code{co\_owned}}{"Yes" if company owned.}

#' \item{\code{ncalls}}{Number of call-backs. Is 0 if contacted on first call.}

#' \item{\code{empft}}{Number full-time employees.}

#' \item{\code{emppt}}{Number part-time employees.}

#' \item{\code{nmgrs}}{Number of managers/assistant managers.}

#' \item{\code{wage\_st}}{Starting wage ($/hr).}

#' \item{\code{inctime}}{Months to usual first raise.}

#' \item{\code{firstinc}}{Usual amount of first raise (\$/hr).}

#' \item{\code{bonus}}{"Yes" if cash bounty for new workers.}

#' \item{\code{pctaff}}{\% of employees affected by new minimum.}

#' \item{\code{meals}}{Free/reduced priced code.}

#' \item{\code{open}}{Hour of opening.}

#' \item{\code{hrsopen}}{Number of hours open per day.}

#' \item{\code{psode}}{Price of medium soda, including tax.}

#' \item{\code{pfry}}{Price of small fries, including tax.}

#' \item{\code{pentree}}{Price of entree, including tax.}

#' \item{\code{nregs}}{Number of cash registers in store.}

#' \item{\code{nregs11}}{Number of registers open at 11:00 pm.}

#' \item{\code{type}}{Type of 2nd interview.}

#' \item{\code{status}}{Status of 2nd interview.}

#' \item{\code{date}}{Date of 2nd interview.}

#' \item{\code{nregs11}}{"Yes" if special program for new workers.}

#' }

#' @source \url{http://davidcard.berkeley.edu/data\_sets.html}

"njmin"

I have documented the data, and using roxygen2::royxgenise() to create the dataset’s documentation.

The data can now be used to create some nifty plots:

ggplot(njmin, aes(wage\_st)) + geom\_density(aes(fill = state), alpha = 0.3) +

facet\_wrap(vars(observation)) + theme\_blog() +

theme(legend.title = element\_blank(), plot.caption = element\_text(colour = "white")) +

labs(title = "Distribution of starting wage rates in fast food restaurants",

caption = "On April 1st, 1992, New Jersey's minimum wage rose from $4.25 to $5.05. Source: Card and Krueger (1994)")

## Warning: Removed 41 rows containing non-finite values (stat\_density).

In the next blog post, I am going to write a first function to perform diff and diff, and we will  
learn how to make it available to users, document and test it!