DC Peanut butter synthetic control

Install packages and load library.

```
#install.packages('tidyverse')
#install.packages("Synth")
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------
---------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.0       v purrr   0.3.0
## v tibble  2.0.1       v dplyr   0.8.0.1
## v tidyr   0.8.2       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0

## -- Conflicts ------------------------------------------------------------
---------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```
library(Synth)
```

```
## ##
## ## Synth Package: Implements Synthetic Control Methods.

## ## See http://www.mit.edu/~jhainm/software.htm for additional information.
```

Unzip the 'peanutbutter.tgz' file. Read in 2011 movement and store files, merge them based on the store code, and filter for 'F' (food stores) and states North Dakota, Minnesota, Missouri, and D.C.

```
untar("peanutbutter.tgz")

move_11 <-
read_tsv("nielsen_extracts/RMS/2011/Movement_Files/0506_2011/1421_2011.tsv")
```

```
## Parsed with column specification:
## cols(
##   store_code_uc = col_double(),
##   upc = col_character(),
```

```
##   week_end = col_double(),
##   units = col_double(),
##   prmult = col_double(),
##   price = col_double(),
##   feature = col_double(),
##   display = col_double()
## )

stores_11 <-
read_tsv("nielsen_extracts/RMS/2011/Annual_Files/stores_2011.tsv")

## Parsed with column specification:
## cols(
##   store_code_uc = col_double(),
##   year = col_double(),
##   parent_code = col_double(),
##   retailer_code = col_double(),
##   channel_code = col_character(),
##   store_zip3 = col_character(),
##   fips_state_code = col_double(),
##   fips_state_descr = col_character(),
##   fips_county_code = col_double(),
##   fips_county_descr = col_character(),
##   dma_code = col_double(),
##   dma_descr = col_character()
## )

full_11 <- move_11 %>%
  inner_join(stores_11, by ="store_code_uc") %>%
  filter(channel_code == "F",
         fips_state_descr %in% c("ND", "MN", "MO" ,"DC"))

save(full_11, file = 'pb_full_11.RData')

rm(move_11, stores_11, full_11)
```

Perform the previous step for 2012 and 2013.

```
move_12 <-
read_tsv("nielsen_extracts/RMS/2012/Movement_Files/0506_2012/1421_2012.tsv")

## Parsed with column specification:
## cols(
##   store_code_uc = col_double(),
##   upc = col_character(),
##   week_end = col_double(),
##   units = col_double(),
##   prmult = col_double(),
##   price = col_double(),
##   feature = col_double(),
```

```
##   display = col_double()
## )

stores_12 <-
read_tsv("nielsen_extracts/RMS/2012/Annual_Files/stores_2012.tsv")

## Parsed with column specification:
## cols(
##   store_code_uc = col_double(),
##   year = col_double(),
##   parent_code = col_double(),
##   retailer_code = col_double(),
##   channel_code = col_character(),
##   store_zip3 = col_character(),
##   fips_state_code = col_double(),
##   fips_state_descr = col_character(),
##   fips_county_code = col_double(),
##   fips_county_descr = col_character(),
##   dma_code = col_double(),
##   dma_descr = col_character()
## )

full_12 <- move_12 %>%
  inner_join(stores_12, by ="store_code_uc") %>%
  filter(channel_code == "F",
         fips_state_descr %in% c("ND", "MN", "MO" ,"DC"))

save(full_12, file = 'pb_full_12.RData')

rm(move_12, stores_12, full_12)

move_13 <-
read_tsv("nielsen_extracts/RMS/2013/Movement_Files/0506_2013/1421_2013.tsv")

## Parsed with column specification:
## cols(
##   store_code_uc = col_double(),
##   upc = col_character(),
##   week_end = col_double(),
##   units = col_double(),
##   prmult = col_double(),
##   price = col_double(),
##   feature = col_double(),
##   display = col_double()
## )

stores_13 <-
read_tsv("nielsen_extracts/RMS/2013/Annual_Files/stores_2013.tsv",
                      col_types = list(col_double(), col_double(),
                      col_double(), col_double(), col_character(),
                      col_character(), col_double(), col_character(),
```

```
                      col_double(), col_character(), col_double(),
col_character()))

full_13 <- move_13 %>%
  inner_join(stores_13, by = "store_code_uc") %>%
  filter(channel_code == "F",
         fips_state_descr %in% c("ND", "MN", "MO", "DC"))

save(full_13, file = 'pb_full_13.RData')

rm(move_13, stores_13)
```

Bind full_11,12,13 into one tbl, 'full_11_12_13'. Overwrite the variable week_end to be in year-month-date format. Create sales variable, which is the number of units sold multiplied by the price.

```
load('pb_full_11.RData')
load('pb_full_12.RData')

full_11_12_13 <- full_11 %>%
  bind_rows(full_12) %>%
  bind_rows(full_13) %>%
  mutate(week_end = ymd(week_end), sales = units * price)

rm(full_11, full_12, full_13)
```

Read in products master file.

```
products <- read_tsv('products.tsv', quote = "")

## Parsed with column specification:
## cols(
##   upc = col_character(),
##   upc_ver_uc = col_double(),
##   upc_descr = col_character(),
##   product_module_code = col_double(),
##   product_module_descr = col_character(),
##   product_group_code = col_double(),
##   product_group_descr = col_character(),
##   department_code = col_double(),
##   department_descr = col_character(),
##   brand_code_uc = col_double(),
##   brand_descr = col_character(),
##   multi = col_double(),
##   size1_code_uc = col_double(),
##   size1_amount = col_double(),
##   size1_units = col_character(),
##   dataset_found_uc = col_character(),
##   size1_change_flag_uc = col_double()
## )
```

Merge products file and full_11_12_13.

```
full_11_12_13 <- full_11_12_13 %>%
  left_join(products, by = 'upc')

rm(products)
```

Filter the full dataset for data pertaining to the retailers in DC. Display the retailer codes in DC.

```
dc_data <- full_11_12_13 %>%
  filter(fips_state_descr == 'DC')

treated_stores <- dc_data %>% filter(retailer_code==842) %>%
distinct(store_code_uc)

control_stores <- dc_data %>% filter(retailer_code!=842) %>%
distinct(store_code_uc)
```

Assign munfacturer for each brand. Organize data by stores and date, add sales across UPCs.

```
dc_data <- dc_data %>%
  mutate(manuf_name = substr(dc_data$brand_descr, 1, 3)) %>%
  filter(manuf_name %in% c('SKI', "JIF", "SMU", "SIM", "SAN", "ADA", "CTL"))
%>%
  mutate(actual_manuf = if_else(manuf_name == 'CTL', 'CTL',
                        if_else(manuf_name == 'SKI', 'UNI', 'SMU'))) %>%
  mutate(store_code_uc=as.numeric(store_code_uc))

dc_data <- dc_data %>% group_by(store_code_uc, week_end) %>%
summarize(sales=sum(sales)) %>%
  arrange(store_code_uc, week_end)

dc_data <- as.data.frame(dc_data)
```

Check store codes to see how many weeks they have. Stores with less than 157 weeks of observations are dropped from the data. Then we arrange by 'store_code_uc' and 'week_end', then create a 'n_week_end' variable to represent the date numerically.

```
weeks <- unique(dc_data$week_end)

codes <- unique(dc_data$store_code_uc)

drops <- c()

for (i in seq_along(codes)) {

  single_store <- dc_data %>% filter(store_code_uc==codes[i])
```

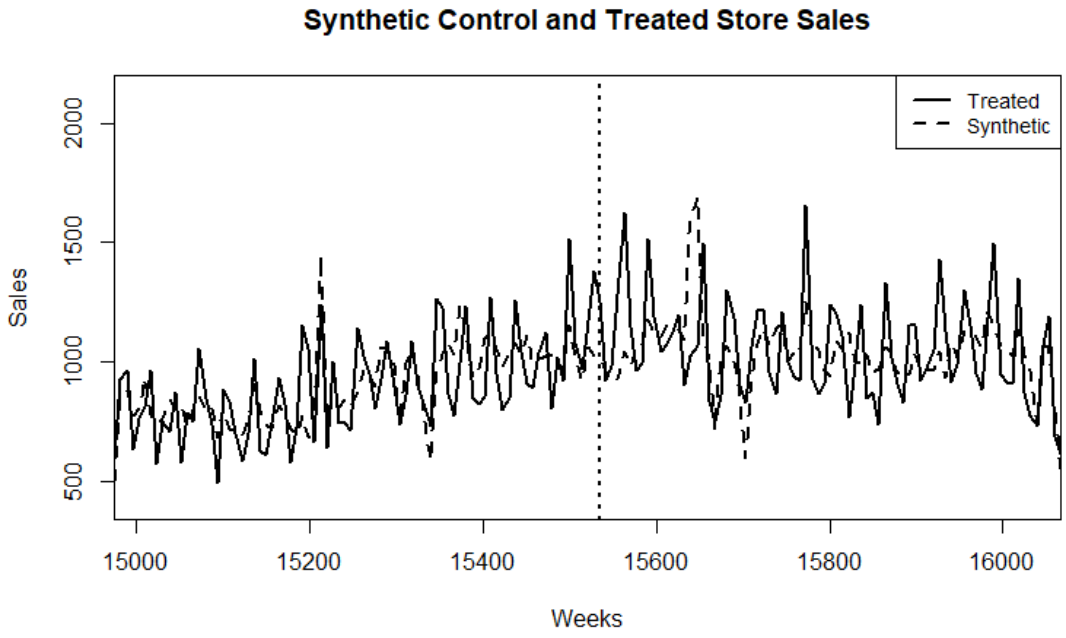
```
  single_store_vec <- unique(single_store$week_end)

  if (length(single_store_vec) < length(weeks)) {

    drops <- c(drops, codes[i])

  }

}

dc_data <- dc_data %>% filter(!store_code_uc %in% drops) %>%
arrange(store_code_uc, week_end) %>%
  mutate(n_week_end=as.numeric(week_end))

control_stores <- control_stores %>% filter(!store_code_uc %in% drops)
```

Use data prep to create weights for synthetic control.

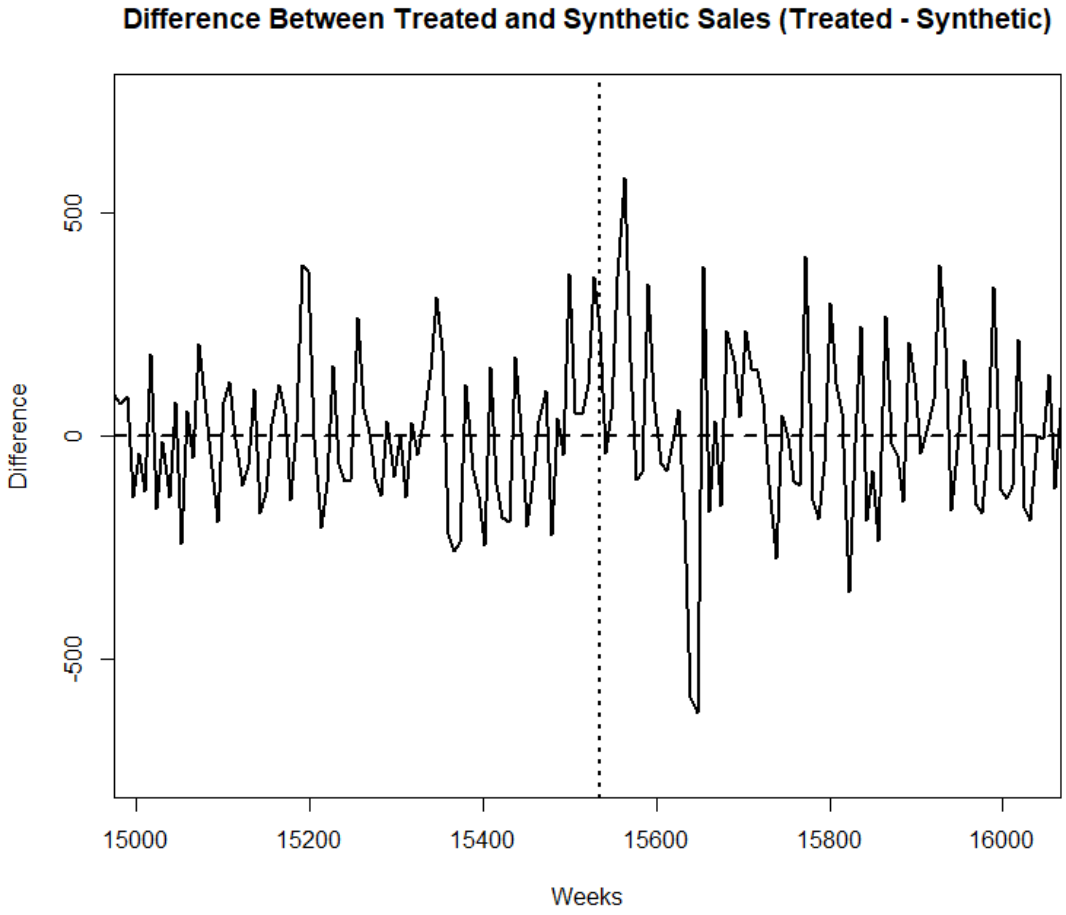
```
dataprep_out <- dataprep(foo=dc_data, predictors="sales", dependent="sales",
unit.variable="store_code_uc",
                         time.variable="n_week_end",
treatment.identifier=5133302,
                         controls.identifier=control_stores$store_code_uc,
                         time.predictors.prior=seq(14975, 15535, by = 7),
time.optimize.ssr=seq(14975, 15535, by = 7),
                         time.plot=seq(14975, 16067, by = 7))

synth_out <-synth(dataprep_out)

##
## X1, X0, Z1, Z0 all come directly from dataprep object.
##
##
## ****************
##  optimization over w weights: computing synthtic control unit
##
##
##
## ****************
## ****************
## ****************
##
## MSPE (LOSS V): 24014.56
##
## solution.v:
##  1
##
## solution.w:
##  0.03670724 0.03469845 0.02984143 0.01159974 0.4612645 0.03238041
```

```
0.03246352 0.03670635 0.03585614 0.03173574 0.036639 0.02044926 0.02551085
0.03672516 0.03472481 0.03640007 0.0367203 0.02957704
```

```
path.plot(dataprep.res = dataprep_out, synth.res = synth_out, Xlab =
c('Weeks'), Ylab = c('Sales'), tr.intake = 15535,
          Main = 'Synthetic Control and Treated Store Sales')
```




```
gaps.plot(dataprep.res = dataprep_out, synth.res = synth_out, Ylab =
'Difference', Xlab = 'Weeks', tr.intake = 15535,
          Main = 'Difference Between Treated and Synthetic Sales (Treated -
Synthetic)')
```




```
synth_tables <- synth.tab(synth.res = synth_out, dataprep.res = dataprep_out)
synth_tables$tab.w
```

```
##         w.weights unit.names unit.numbers
## 1804471     0.037    1804471      1804471
## 3949012     0.035    3949012      3949012
## 5148299     0.030    5148299      5148299
## 1802456     0.012    1802456      1802456
## 2078770     0.461    2078770      2078770
## 1808501     0.032    1808501      1808501
## 1028103     0.032    1028103      1028103
## 5319048     0.037    5319048      5319048
## 2319813     0.036    2319813      2319813
## 3521655     0.032    3521655      3521655
## 5524392     0.037    5524392      5524392
## 1860488     0.020    1860488      1860488
## 1657215     0.026    1657215      1657215
## 1788754     0.037    1788754      1788754
## 1807292     0.035    1807292      1807292
## 7962112     0.036    7962112      7962112
```

```
## 5164822     0.037    5164822      5164822
## 1809307     0.030    1809307      1809307

synth_tables$tab.pred

##       Treated Synthetic Sample Mean
## sales 897.081   897.082     1243.61

dataprep_out$Y1plot

##       5133302
## 14975  588.81
## 14982  926.88
## 14989  965.48
## 14996  634.20
## 15003  764.45
## 15010  812.04
## 15017  963.42
## 15024  574.88
## 15031  749.78
## 15038  706.55
## 15045  871.62
## 15052  577.97
## 15059  789.88
## 15066  750.94
## 15073 1057.05
## 15080  847.44
## 15087  741.03
## 15094  490.78
## 15101  883.52
## 15108  837.77
## 15115  699.38
## 15122  583.09
## 15129  701.77
## 15136 1013.36
## 15143  627.44
## 15150  611.76
## 15157  747.25
## 15164  931.21
## 15171  825.22
## 15178  576.71
## 15185  725.09
## 15192 1156.60
## 15199 1059.07
## 15206  664.44
## 15213 1239.72
## 15220  643.42
## 15227  999.53
## 15234  746.53
## 15241  742.68
## 15248  712.40
```

```
## 15255 1139.40
## 15262 1026.55
## 15269  949.28
## 15276  802.57
## 15283  928.94
## 15290 1086.30
## 15297  894.17
## 15304  739.78
## 15311  858.59
## 15318 1085.07
## 15325  906.18
## 15332  829.53
## 15339  730.09
## 15346 1266.87
## 15353 1225.76
## 15360  868.84
## 15367  773.61
## 15374  997.19
## 15381 1232.58
## 15388  849.53
## 15395  826.70
## 15402  852.78
## 15409 1271.72
## 15416  950.29
## 15423  802.10
## 15430  845.58
## 15437 1255.18
## 15444 1045.30
## 15451  908.89
## 15458  889.15
## 15465 1036.04
## 15472 1126.02
## 15479  806.70
## 15486 1019.68
## 15493  924.55
## 15500 1515.34
## 15507 1080.81
## 15514  968.14
## 15521 1175.11
## 15528 1380.98
## 15535 1250.53
## 15542  921.91
## 15549  983.57
## 15556 1280.75
## 15563 1622.91
## 15570 1149.09
## 15577  966.79
## 15584  993.55
## 15591 1517.50
## 15598 1191.88
```

```
## 15605 1043.02
## 15612 1073.78
## 15619 1122.03
## 15626 1195.28
## 15633  904.05
## 15640 1025.22
## 15647 1068.54
## 15654 1494.99
## 15661  833.50
## 15668  748.89
## 15675  862.98
## 15682 1302.23
## 15689 1182.36
## 15696  900.37
## 15703  830.16
## 15710 1062.34
## 15717 1217.09
## 15724 1219.56
## 15731  959.24
## 15738  865.89
## 15745 1208.80
## 15752 1000.97
## 15759  938.39
## 15766  923.34
## 15773 1656.75
## 15780  929.22
## 15787  864.95
## 15794  906.92
## 15801 1238.92
## 15808 1205.70
## 15815 1111.07
## 15822  770.42
## 15829  986.41
## 15836 1236.96
## 15843  845.89
## 15850  871.14
## 15857  736.59
## 15864 1329.18
## 15871 1021.59
## 15878  911.44
## 15885  830.38
## 15892 1153.34
## 15899 1153.55
## 15906  919.70
## 15913  991.71
## 15920 1049.71
## 15927 1426.76
## 15934 1131.07
## 15941  913.37
## 15948  987.69
```

```
## 15955 1298.19
## 15962 1156.15
## 15969  950.33
## 15976  884.77
## 15983 1184.54
## 15990 1496.15
## 15997  946.07
## 16004  915.06
## 16011  912.30
## 16018 1351.48
## 16025  872.08
## 16032  775.60
## 16039  731.96
## 16046 1067.16
## 16053 1193.10
## 16060  691.90
## 16067  615.59
```