ECON 370 Final Project - Group 1

SD:2, 5, 17, 19, 28

2022-11-8

==

Set-up

==

```
library(readr)
library(mosaic)

## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2

##
## The 'mosaic' package masks several functions from core packages in order
to add
## additional features.  The original behavior of these functions should not
be affected by this.

##
## Attaching package: 'mosaic'

## The following objects are masked from 'package:dplyr':
##
##     count, do, tally

## The following object is masked from 'package:Matrix':
##
##     mean

## The following object is masked from 'package:ggplot2':
##
##     stat

## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var
```

```
## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum

library(data.table)

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

library("xlsx")
library(openxlsx)

##
## Attaching package: 'openxlsx'

## The following objects are masked from 'package:xlsx':
##
##     createWorkbook, loadWorkbook, read.xlsx, saveWorkbook, write.xlsx

library(mvtnorm)
library(tidyverse)

## ── Attaching packages
## ───────────────────────────────────────
## tidyverse 1.3.2 ──

## ✓ tibble  3.1.8     ✓ stringr 1.4.0
## ✓ tidyr   1.2.0     ✓ forcats 0.5.1
## ✓ purrr   0.3.4
## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✕ data.table::between()      masks dplyr::between()
## ✕ mosaic::count()            masks dplyr::count()
## ✕ purrr::cross()             masks mosaic::cross()
## ✕ mosaic::do()               masks dplyr::do()
## ✕ tidyr::expand()            masks Matrix::expand()
## ✕ dplyr::filter()            masks stats::filter()
## ✕ data.table::first()        masks dplyr::first()
## ✕ ggstance::geom_errorbarh() masks ggplot2::geom_errorbarh()
## ✕ dplyr::lag()               masks stats::lag()
## ✕ data.table::last()         masks dplyr::last()
## ✕ tidyr::pack()              masks Matrix::pack()
## ✕ mosaic::stat()             masks ggplot2::stat()
## ✕ mosaic::tally()            masks dplyr::tally()
```

```
## ✖ purrr::transpose()         masks data.table::transpose()
## ✖ tidyr::unpack()            masks Matrix::unpack()
```

```
library(stringr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(dplyr)
library(usa)
```

```
## The 'usa' package masks the state datasets included in base R:
## * state.abb
## * state.area
## * state.center
## * state.division
## * state.name
## * state.region
## Objects are similar in class and content but updated and expanded.
##
## Attaching package: 'usa'
##
## The following objects are masked from 'package:datasets':
##
##     state.abb, state.area, state.center, state.division, state.name,
##     state.region
```

```
library(fixest)
```

===

Question 1 - Download Data

===

Note: r eval = FALSE prevents code chunk from running when data is already downloaded (the required “switch” in our code). Delete eval = FALSE for first run of code to download files.

```
for (i in 1998:2010){

url1 <- paste("http://www.nber.org/hcris/265-94/rnl_rpt265_94_", i ,
".csv",sep = '')
url2 <- paste("http://www.nber.org/hcris/265-94/rnl_nmrc265_94_", i,
"_long.csv", sep = '')
url3 <- paste("http://www.nber.org/hcris/265-94/rnl_alpha265_94_", i,
"_long.csv", sep = '')

destfile <- paste("./hcris_raw/rnl_rpt265_94_", i , ".csv",sep = '')
download.file(url1,destfile)
destfile <- paste("./hcris_raw/rnl_nmrc265_94_", i, "_long.csv", sep = '')
download.file(url2,destfile)
destfile <- paste("./hcris_raw/rnl_alpha265_94_", i, "_long.csv", sep = '')
download.file(url3,destfile)

}
```

===

Question 2 - Cleaning The Data

===

Read in raw data

```
# Returns indices of dataframe with matching variable code for subsetting
parse = function(data, match){
  vars = grepl(match, data)
  indices = c()
  for (i in 1:length(vars)){
    if (vars[i]){
      indices = c(indices,i)
```

```r
    }
  }
  indices
}

variable_codes <- read.csv("variable_codes.csv")

# 0 pad lines and columns to match desired character count for key
for (i in 1:nrow(variable_codes)){
  variable_codes$line[i] = str_pad(variable_codes$line[i], 5, pad = "0")
  variable_codes$column[i] = str_pad(variable_codes$column[i], 4, pad = "0")
}

# Generate variable code keys
variable_codes$key = paste(variable_codes$worksheet, variable_codes$line,
variable_codes$column, sep = '')

# Fix typo error in data
variable_codes$key[35] = "S000001001020300"

# Regular expression to parse keys
toMatch = paste(variable_codes$key, collapse = "|")

# Read in all CSVs
for (i in 1998:2010){

  rpt_name = paste("rpt", i, sep = '')
  nmrc_name = paste("nmrc", i, sep = '')
  alpha_name = paste("alpha", i, sep = '')

  rpt <- read_csv(paste("hcris_raw/rnl_rpt265_94_", i , ".csv",sep = ''),
show_col_types = F)
  nmrc <- read_csv(paste("./hcris_raw/rnl_nmrc265_94_", i, "_long.csv", sep =
''), show_col_types = F)
  alpha  <- read_csv(paste("./hcris_raw/rnl_alpha265_94_", i, "_long.csv",
sep = ''), show_col_types = F)

  nmrc$key = paste(nmrc$wksht_cd, nmrc$line_num, nmrc$clmn_num, sep = '')
  alpha$key = paste(alpha$wksht_cd, alpha$line_num, alpha$clmn_num, sep = '')

  # Parse alpha and nmrc data for matching variable codes
  indices = parse(nmrc$key, toMatch)
  nmrc_parsed = nmrc[indices,]
  indices = parse(alpha$key, toMatch)
  alpha_parsed = alpha[indices,]

  assign(rpt_name, rpt)
  assign(nmrc_name, nmrc_parsed)
  assign(alpha_name, alpha_parsed)
```

```
}

# Clean up environment
rm(rpt, nmrc, alpha, rpt_name, nmrc_name, alpha_name, alpha_parsed,
nmrc_parsed)
```

Reformatting The Raw Data

```
# Takes a long alpha dataframe, merges variable codes, and returns wide alpha
dataframe
long_to_wide_alpha = function(data){
  data$key <- dplyr::recode(
  data$key,
  !!!setNames(as.character(idkey$new), idkey$original)
)
  setDT(data)
  wide = dcast(data, rpt_rec_num~key, value.var = "alphnmrc_itm_txt")
  return(wide)
}

# Takes a long nmrc dataframe, merges variable codes, and returns wide nmrc
dataframe
long_to_wide_nmrc = function(data){
  data$key <- dplyr::recode(
  data$key,
  !!!setNames(as.character(idkey$new), idkey$original)
)
  setDT(data)
  wide = dcast(data, rpt_rec_num~key, value.var = "itm_val_num")
  return(wide)
}

# Generate key to merge variable names into dataframe
idkey <- data.frame("original" = variable_codes$key, "new" =
variable_codes$variable)

# Declare alpha and nmrc lists to iterate through
alpha_list = list(alpha1998, alpha1999, alpha2000, alpha2001, alpha2002,
alpha2003,
                  alpha2004, alpha2005, alpha2006, alpha2007, alpha2008,
alpha2009,
                  alpha2010)
nmrc_list = list(nmrc1998, nmrc1999, nmrc2000, nmrc2001, nmrc2002, nmrc2003,
                 nmrc2004, nmrc2005, nmrc2006, nmrc2007, nmrc2008, nmrc2009,
                 nmrc2010)

# Convert data from all years from long to wide
for(i in 1998:2010){
  j = i - 1997
```

```
  nmrc_wide_name = paste("nmrc_wide_", i, sep = '')
  alpha_wide_name = paste("alpha_wide_", i, sep = '')

  nmrc_wide = long_to_wide_nmrc(as.data.frame(nmrc_list[j]))
  alpha_wide = long_to_wide_alpha(as.data.frame(alpha_list[j]))

  assign(nmrc_wide_name, nmrc_wide)
  assign(alpha_wide_name, alpha_wide)
  j = j+1
}

# Add in missing epo_cost and state columns for 1998 (to match # of columns)
alpha_wide_1998$state = NA
alpha_wide_1998 <- subset(alpha_wide_1998, select=c(1:9, 11, 10))
nmrc_wide_1998$epo_cost = NA
nmrc_wide_1998 <- subset(nmrc_wide_1998, select = c(1:6, 26, 7:25))

wide_alpha_list = list(alpha_wide_1998, alpha_wide_1999, alpha_wide_2000,
alpha_wide_2001,
                       alpha_wide_2002, alpha_wide_2003, alpha_wide_2004,
alpha_wide_2005,
                       alpha_wide_2006, alpha_wide_2007, alpha_wide_2008,
alpha_wide_2009,
                       alpha_wide_2010)

wide_nmrc_list = list(nmrc_wide_1998, nmrc_wide_1999, nmrc_wide_2000,
nmrc_wide_2001,
                      nmrc_wide_2002, nmrc_wide_2003, nmrc_wide_2004,
nmrc_wide_2005,
                      nmrc_wide_2006, nmrc_wide_2007, nmrc_wide_2008,
nmrc_wide_2009,
                      nmrc_wide_2010)

rpt_list = list(rpt1998[c(1,13,14)], rpt1999[c(1,13,14)],
rpt2000[c(1,13,14)], rpt2001[c(1,13,14)],
                rpt2002[c(1,13,14)], rpt2003[c(1,13,14)],
rpt2004[c(1,13,14)], rpt2005[c(1,13,14)],
                rpt2006[c(1,13,14)], rpt2007[c(1,13,14)],
rpt2008[c(1,13,14)], rpt2009[c(1,13,14)],
                rpt2010[c(1,13,14)])

# Merge all years
for(i in 1998:2010){
  j = i - 1997
  merged_year_name = paste("merged", i, sep = '')

  # merge alpha and nmrc
  merged_year = merge(as.data.frame(wide_nmrc_list[j]),
                      as.data.frame(wide_alpha_list[j]), by = "rpt_rec_num")
```

```
  # merge fy_bgn_dt and fy_end_dt from rpt data
  merged_year = merge(merged_year, rpt_list[j], by = "rpt_rec_num")

  # add year variable
  merged_year$year = i

  assign(merged_year_name, merged_year)
  j = j+1
}

# Write all files to CSVs in hcris_cleaned directory
fwrite(merged1998, file = "hcris_cleaned/hcris_1998.csv")
fwrite(merged1999, file = "hcris_cleaned/hcris_1999.csv")
fwrite(merged2000, file = "hcris_cleaned/hcris_2000.csv")
fwrite(merged2001, file = "hcris_cleaned/hcris_2001.csv")
fwrite(merged2002, file = "hcris_cleaned/hcris_2002.csv")
fwrite(merged2003, file = "hcris_cleaned/hcris_2003.csv")
fwrite(merged2004, file = "hcris_cleaned/hcris_2004.csv")
fwrite(merged2005, file = "hcris_cleaned/hcris_2005.csv")
fwrite(merged2006, file = "hcris_cleaned/hcris_2006.csv")
fwrite(merged2007, file = "hcris_cleaned/hcris_2007.csv")
fwrite(merged2008, file = "hcris_cleaned/hcris_2008.csv")
fwrite(merged2009, file = "hcris_cleaned/hcris_2009.csv")
fwrite(merged2010, file = "hcris_cleaned/hcris_2010.csv")
```

Cleaning The Reformatted Data

Set-up

rbind 13 reformatted datasets

```
hcris_data <- read_csv("hcris_cleaned/hcris_1998.csv", show_col_types = F)

# rbind all 13 datasets
for (i in 1999:2010){
  current = paste("hcris_cleaned/hcris_", i, ".csv", sep = '')
  to_bind <- read_csv(current, show_col_types = F)
  hcris_data <- rbind(hcris_data, to_bind)
}

# Rename first column and cast prvdr_num as numeric
colnames(hcris_data)[1] = "report_number"
hcris_data$prvdr_num = as.numeric(hcris_data$prvdr_num)

# Take head of fully reformatted data
head(hcris_data)
```

```
## # A tibble: 6 × 39
##   report_number avg_da…¹ avg_s…² avg_w…³ dialy…⁴ dialy…⁵ epo_c…⁶ epo_n…⁷
```

```
epo_r…⁸
##             <dbl>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
<dbl>
## 1               7        6     4.5       3      65       1      NA  -1.01e6 -
158587
## 2              23        6     4         3      NA       1      NA  -2.13e5
-32631
## 3              44        6     4         3      15       1      NA  -5.23e5
-80538
## 4              53        3     4.5       3      40       1      NA  -1.37e5
-21319
## 5              62        6     4.5       3      40       1      NA  -5.13e5
-78169
## 6              80        6     4         3      10       1      NA  -1.25e4
-1372
## # … with 30 more variables: epo_total <dbl>, lab_services <dbl>,
## #   non_medicare_sessions <dbl>, non_medicare_sessions_indirect <dbl>,
## #   num_machines_regular <dbl>, num_machines_standby <dbl>, supplies
<dbl>,
## #   total_costs_hd_benefits <dbl>, total_costs_hd_drugs <dbl>,
## #   total_costs_hd_housekeeping <dbl>, total_costs_hd_labs <dbl>,
## #   total_costs_hd_machines <dbl>, total_costs_hd_other <dbl>,
## #   total_costs_hd_salaries <dbl>, total_costs_hd_supplies <dbl>, …
## # ℹ Use `colnames()` to see all variable names
```

Question 1

Drop observations with missing prvdr_num

```
hcris_data <- hcris_data[complete.cases(hcris_data$prvdr_num), ]
```

Question 2

Take absolute value for cost variables

```
hcris_data$epo_cost = abs(hcris_data$epo_cost)
hcris_data$epo_net_cost = abs(hcris_data$epo_net_cost)
hcris_data$epo_rebates = abs(hcris_data$epo_rebates)
```

Question 3

Replace NAs with 0 for epo_rebates

```
hcris_data$epo_rebates[is.na(hcris_data$epo_rebates)] = 0
```

Question 4

Clean epo variables

```
##### 4a - epo_cost #####
indices = is.na(hcris_data$epo_cost) & hcris_data$epo_rebates == 0 &
!is.na(hcris_data$epo_net_cost)
```

```
hcris_data$epo_cost[indices] = hcris_data$epo_net_cost[indices]

##### 4b - epo_cost #####
indices = is.na(hcris_data$epo_cost) & hcris_data$epo_rebates != 0 &
!is.na(hcris_data$epo_net_cost)
hcris_data$epo_cost[indices] = hcris_data$epo_net_cost[indices] +
hcris_data$epo_rebates[indices]

##### 4c - epo_cost and epo_net_cost #####
hcris_data <- hcris_data %>%
  mutate(epo_cost = ifelse(is.na(epo_cost) &
                             epo_rebates == 0 &
                             is.na(epo_net_cost),0,epo_cost),
         epo_net_cost = ifelse(is.na(epo_cost) &
                             epo_rebates == 0 &
                             is.na(epo_net_cost),0,epo_net_cost))

##### 4d - Cost data left as missing, nothing to do #####

##### 4e - epo_net_cost #####
indices = !is.na(hcris_data$epo_cost) & is.na(hcris_data$epo_net_cost)
hcris_data$epo_net_cost[indices] = hcris_data$epo_cost[indices] -
hcris_data$epo_rebates[indices]
```

Question 5

Switch epo_cost and epo_net_cost for relevant observations

```
hcris_data <- hcris_data %>%
  mutate(epo_cost = ifelse((epo_cost < epo_net_cost),epo_net_cost,epo_cost),
         epo_net_cost = ifelse((epo_cost <
epo_net_cost),epo_cost,epo_net_cost))
```

Question 6

Fix prvdr_num error

```
index = which(hcris_data$prvdr_num == 322664)
hcris_data$prvdr_num[index] = 342664
```

Question 7

Clean dates

```
hcris_data = mutate(hcris_data, fy_bgn_dt = mdy(fy_bgn_dt))
hcris_data = mutate(hcris_data, fy_end_dt = mdy(fy_end_dt))
hcris_data = mutate(hcris_data, report_start_date = mdy(report_start_date))
hcris_data = mutate(hcris_data, report_end_date = mdy(report_end_date))
```

Question 8

Remove extraneous variables

```r
hcris_data$report_start_date = NULL
hcris_data$report_end_date = NULL
```

Question 9

Clean zip codes

```r
##### 9a - Trim whitespace and get substring #####
# Trim trailing & leading whitespace
hcris_data=as.data.frame(apply(hcris_data,2,trimws))
# Remove alphabets and special characters
hcris_data$zip_code=gsub("[[:alpha:]-]", "", hcris_data$zip_code)
# Trim whitespace again - command above left some zip codes with
leading/trailing white space
hcris_data=as.data.frame(apply(hcris_data,2,trimws))
# Slice zip codes to first 5 digits of string
hcris_data$zip_code=substr(hcris_data$zip_code,1,5)

##### 9b + 9c - Clean zip codes #####
 hcris_dataZipClean=as.data.frame(c())# blank data frame
#loop through each unique provider number
for(i in unique(hcris_data$prvdr_num)){
  #select zip code variable from ith provider number
   testcase=hcris_data%>%filter(prvdr_num==i) %>%
   dplyr::select(zip_code)
  # see if there is only 1 zip code associated with the provider
  if(nrow(unique(na.omit(testcase)))==1){
    hcris_dataZipClean=rbind(hcris_dataZipClean,hcris_data %>%
    filter(prvdr_num==i) %>%
      #if there is only 1 zip code, replace all NA zip code values for the
ith provider with it's unique zip code and append to data frame
  mutate(zip_code=rep(na.omit(zip_code)[1], sum(prvdr_num==i|prvdr_num!=i))))
    } else{
    hcris_dataZipClean=rbind(hcris_dataZipClean,hcris_data %>%
    filter(prvdr_num==i))
    #if there is more than 1 zip code, do nothing but add data to clean data
frame so in end
    #you get all original data back with cleaned zips for NAs that can be
unambiguously replaced
    }
 }
hcris_data=hcris_dataZipClean
```

Question 10

Clean missing states

```r
# Helper function to pull state codes from zips
zipz=usa::zipcodes
 getstate = function(zipvec){
   codes=c()
```

```
   for (i in zipvec){
     if(is.na(i)==FALSE){
    Temp= zipz%>% filter(zip.code==i)%>%select(state)
codes=c(codes,Temp$state[1])
     }else{
  codes=c(codes,NA)
}
   }
  codes
 }

# Separating data set to only fix ones with state missing
nostatefixed=hcris_data %>% filter(is.na(state))%>%
mutate(state=getstate(zipvec = zip_code))
hastate=hcris_data %>% filter(is.na(state)==F)

# Combine data sets back together
hcris_data=rbind(hastate,nostatefixed)
```

Question 11

Cleaning chain_identity

```
# Regular expressions for string parsing
f_regex = "^FEN|^FER|^FES|^FR4|^FRE|^FRR|^FRS|^\\bDRES\\B"
d_regex = "^DAC|^DAN|^DAV|^DAT|^DV"

# Indicator variables for iteration
hcris_data$is_fresenius = grepl(f_regex, hcris_data$chain_identity,
ignore.case = TRUE)
hcris_data$is_davita = grepl(d_regex, hcris_data$chain_identity, ignore.case
= TRUE)

# Default - when chain_indicator == 0 (not chain)
hcris_data$chain_id = 0

# Iterate through data to clean chain_identity
for (i in 1:nrow(hcris_data)){
  if (is.na(hcris_data$chain_identity[i])){
    hcris_data$chain_id[i] = NA
  }
  # Fresenius chain
  else if (hcris_data$is_fresenius[i]){
    hcris_data$chain_id[i] = 3
    hcris_data$chain_identity[i] = "Fresenius"
  }
  # Davita chain
  else if (hcris_data$is_davita[i]){
    hcris_data$chain_id[i] = 2
    hcris_data$chain_identity[i] = "DaVita"
```

```
  }
  # not Fresenius or Davita, but has a chain indicator/non-empty
chain_identity (other)
  else if (!is.na(hcris_data$chain_indicator[i])
           & hcris_data$chain_indicator[i] == "Y" &
           !is.na(hcris_data$chain_identity[i]) &
           hcris_data$chain_identity[i] != ""){
    hcris_data$chain_id[i] = 1
    hcris_data$chain_identity[i] = "Other"
  }
  else{ # not Fresenius/Davita/other (chain_id == 0)
    hcris_data$chain_identity[i] = "Not chain"
  }
}

# Drop indicator variables
hcris_data$is_fresenius = NULL
hcris_data$is_davita = NULL
```

Question 12

Remake chain_indicator

```
# Get indices
chains = hcris_data$chain_identity == "Fresenius" |
  hcris_data$chain_identity == "DaVita" |
  hcris_data$chain_identity == "Other"

not_chains = hcris_data$chain_identity == "Not chain"

chain_NA = is.na(hcris_data$chain_identity)

# Remake chain_indicator variable
hcris_data$chain_indicator[chains] = "Y"
hcris_data$chain_indicator[not_chains] = "N"
hcris_data$chain_indicator[chain_NA] = NA

# Take head of fully cleaned data
head(hcris_data)

##   report_number avg_days_open_per_week avg_session_time
avg_weekly_sessions
## 1         77118                   6.00             5.00
3.00
## 2         90396                   6.00             4.50
3.00
## 3        100506                   6.00             4.97
3.00
## 4        107317                   6.00             5.03
3.00
```

```
## 5       113663                  6.00             4.90
3.00
## 6       132899                  6.00             4.84
3.00
##   dialyser_reuse_times dialyzer_type epo_cost epo_net_cost epo_rebates
## 1                 <NA>             1  1601536      1398755      202781
## 2                 <NA>             1  1892776      1627766      265010
## 3                 <NA>             1  1473729      1126346      347383
## 4                 <NA>             1  1523757      1024134      499623
## 5                 <NA>             1  1317838       901779      416059
## 6                 <NA>             1  1395610       956593      439017
##   epo_total lab_services non_medicare_sessions
non_medicare_sessions_indirect
## 1    177836        11004                  2641
<NA>
## 2    194842         8465                  2768
<NA>
## 3    127309         5281                  2582
<NA>
## 4    141798        16460                  3002
<NA>
## 5    116698        12746                  3390
<NA>
## 6    142040        16628                  4503
<NA>
##   num_machines_regular num_machines_standby supplies
total_costs_hd_benefits
## 1                   32                    1     4310
126467
## 2                   32                    1     6356
146393
## 3                   32                    2     3933
145501
## 4                   32                    2     3363
176813
## 5                   32                    2     4551
219567
## 6                   32                    3     3626
246582
##   total_costs_hd_drugs total_costs_hd_housekeeping total_costs_hd_labs
## 1                 3094                      286347                1004
## 2                  886                      295169                1230
## 3                 1239                      350332                1230
## 4                 1203                      382231                1093
## 5                 1500                      420750                1094
## 6                18597                      410178                1359
##   total_costs_hd_machines total_costs_hd_other total_costs_hd_salaries
## 1                  108344               406898                  516206
## 2                  117353               487192                  563261
## 3                  103969               524546                  687103
```

```
## 4                  126041                652248                  770281
## 5                  151590                604410                  819273
## 6                  143281                638107                  712335
##   total_costs_hd_supplies total_treatments_hd total_treatments_pd
## 1                  255037               13252                <NA>
## 2                  221577               13376                <NA>
## 3                  269978               11883                <NA>
## 4                  364023               12829                <NA>
## 5                  294223               13845                <NA>
## 6                  307046               14895                <NA>
##   certification_date chain_identity chain_indicator ever_hospital_based
## 1         03/01/1977      Fresenius               Y                   N
## 2         03/01/1977      Fresenius               Y                   N
## 3         03/01/1977      Fresenius               Y                   N
## 4         03/01/1977      Fresenius               Y                   N
## 5         03/01/1977      Fresenius               Y                   N
## 6         03/01/1977      Fresenius               Y                   N
##         facility_name prvdr_num state zip_code  fy_bgn_dt  fy_end_dt
year
## 1          CAPITOL CITY     12500    AL    36104 2004-01-01 2004-12-31
2004
## 2 CAPITOL CITY DIALYSIS     12500    AL    36104 2005-01-01 2005-12-31
2005
## 3      FMC CAPITAL CITY     12500    AL    36104 2006-01-01 2006-12-31
2006
## 4      FMC CAPITAL CITY     12500    AL    36104 2007-01-01 2007-12-31
2007
## 5      FMC CAPITAL CITY     12500    AL    36104 2008-01-01 2008-12-31
2008
## 6      FMC CAPITAL CITY     12500    AL    36104 2009-01-01 2009-12-31
2009
##   chain_id
## 1        3
## 2        3
## 3        3
## 4        3
## 5        3
## 6        3
```

==

Question 3 - Analysis

==

For the analysis portion of our project, we performed three analyses: we examined the relationship between whether a provider was a chain and drug cost, the relationship between whether a provider was a chain and cost of lab services, and the relationship between whether a provider was a chain and the accuracy of data reporting as represented by the number of missing values for individual provider observations. These analysis choices were inspired by several statements in the supporting paper, "How Acquisitions Affect Firm Behavior and Performance: Evidence from the Dialysis Industry". In this paper, it is explained that multiple differences in drug cost were observed when comparing chain vs. non-chain providers. Specifically for EPOGEN, it was found that EPOGEN dosage increased 129% when previously independent provider facilities were acquired by large chains. Additionally, the paper supports that "large chains… may have lower average costs due to volume discounts for pharmaceuticals as well as centralized clinical laboratories". Because of these two facts, we determined that analyzing the changes in EPOGEN cost would be a worthwhile analysis to analyze the difference in chain vs. non-chain provider behavior. For the cost of lab services, the evidence described in the paper explains that chains were able to decrease lab services costs by having "centralized clinical laboratories" and discounts on large volume pharmaceuticals which can decrease the average drug costs. This inspired us to include the relationship between chain vs. non-chain and lab services in our analysis as well. Lastly, our decision to examine the accuracy of data collection and reporting for chains vs. non-chains from source material support that highlighted improvements in data quality after providers changed from chain to non-chain.

Examining Lab Service Costs, Chain vs. Not Chain

```
summary(feols(as.numeric(total_costs_hd_labs)~factor(chain_indicator)+as.nume
ric(total_treatments_hd)+as.numeric(lab_services)|factor(year)+factor(prvdr_n
um), data=hcris_data, cluster=~prvdr_num))
```

```
## NOTE: 32,291 observations removed because of NA values (LHS: 8,146, RHS:
31,852).
```

```
## OLS estimation, Dep. Var.: as.numeric(total_costs_hd_labs)
## Observations: 16,670
## Fixed-effects: factor(year): 13,  factor(prvdr_num): 2,396
## Standard-errors: Clustered (prvdr_num)
##                                       Estimate  Std. Error  t value
Pr(>|t|)
## factor(chain_indicator)Y        -482.29101794 214.3525768 -2.24999
2.4540e-02
## as.numeric(total_treatments_hd)    0.00000738   0.0000017  4.33973
```

```
1.4858e-05
## as.numeric(lab_services)          -0.12260319   0.0288950 -4.24306
2.2888e-05
##
## factor(chain_indicator)Y        *
## as.numeric(total_treatments_hd) ***
## as.numeric(lab_services)        ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 2,902.5     Adj. R2: 0.558347
##                 Within R2: 0.082576
```

To examine the difference in lab services costs by chain vs. not chain, we created a fixed-effects OLS model to examine the influence the chain_indicator variable on total lab costs. We included the variables total_treatments_hd and lab_services to improve the accuracy of the model, as the number of treatments and the the number of lab services have a direct relationship with total lab costs. We also included fixed effects for year and provider number in the regression to allow us to control for changes over time and different provider averages, which essentially isolated our regression so that we were looking at how costs changed for providers that switched from chain to non-chain rather than simply examining the average epo_cost for all the chains in the data vs. non-chains. Our model estimates that the cost of lab services for chains is $482.29 lower than non-chains. This value is statistically significant at the 5% level with a p-value of 2.4540e-02. This supports the statement in the paper that chain providers have lower lab costs because of centralized laboratories. As providers switched from non-chain to chain, they were able to lower lab costs by utilizing these centralized laboratories as opposed to using alternative methods, like having to outsource lab work to a third-party company.

Examining Drug (EPOGEN) Costs, Chain vs. Not Chain

```
summary(feols(as.numeric(epo_cost)~factor(chain_indicator)+as.numeric(total_t
reatments_hd)+as.numeric(epo_rebates)|factor(year)+factor(prvdr_num),
data=hcris_data, cluster=~prvdr_num))
```

```
## NOTE: 8,676 observations removed because of NA values (LHS: 33, RHS:
8,657).
```

```
## OLS estimation, Dep. Var.: as.numeric(epo_cost)
## Observations: 40,285
## Fixed-effects: factor(year): 13,  factor(prvdr_num): 4,847
## Standard-errors: Clustered (prvdr_num)
##                                      Estimate   Std. Error  t value
Pr(>|t|)
## factor(chain_indicator)Y        -27355.508992 11288.226853 -2.42337
0.0154138
## as.numeric(total_treatments_hd)      0.001219     0.000426  2.86190
0.0042291
## as.numeric(epo_rebates)              1.430376     0.030578 46.77730 <
2.2e-16
##
```

```
## factor(chain_indicator)Y        *
## as.numeric(total_treatments_hd) **
## as.numeric(epo_rebates)         ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 729,223.5     Adj. R2: 0.490068
##                   Within R2: 0.024231
```

To examine the difference in drug costs, specifically of EPOGEN, by chain vs. not chain, we created a fixed-effects OLS model to examine the influence the chain_indicator variable on EPOGEN costs (epo_cost). We included the variables total_treatments_hd and epo_rebates to improve the fit of our model, as the number of treatments at a provider and the amount of rebates the provider was able to acquire would have a relationship with the cost of EPOGEN. We included fixed effects for year and provider number in this model as well. Our model estimates that the cost of drug services for chains is $27,355 lower than non-chains. This value is statistically significant at the 5% level with a p-value of 0.015. Based on the paper, this is most likely because chains are able to benefit from economies of scale when purchasing pharmaceuticals. Chains also may have contracts with certain pharmaceutical companies that allow them to purchase product at a lower cost on the assumption that many providers from the chain will purchase from the same company.

NAs for Chain vs Not Chain

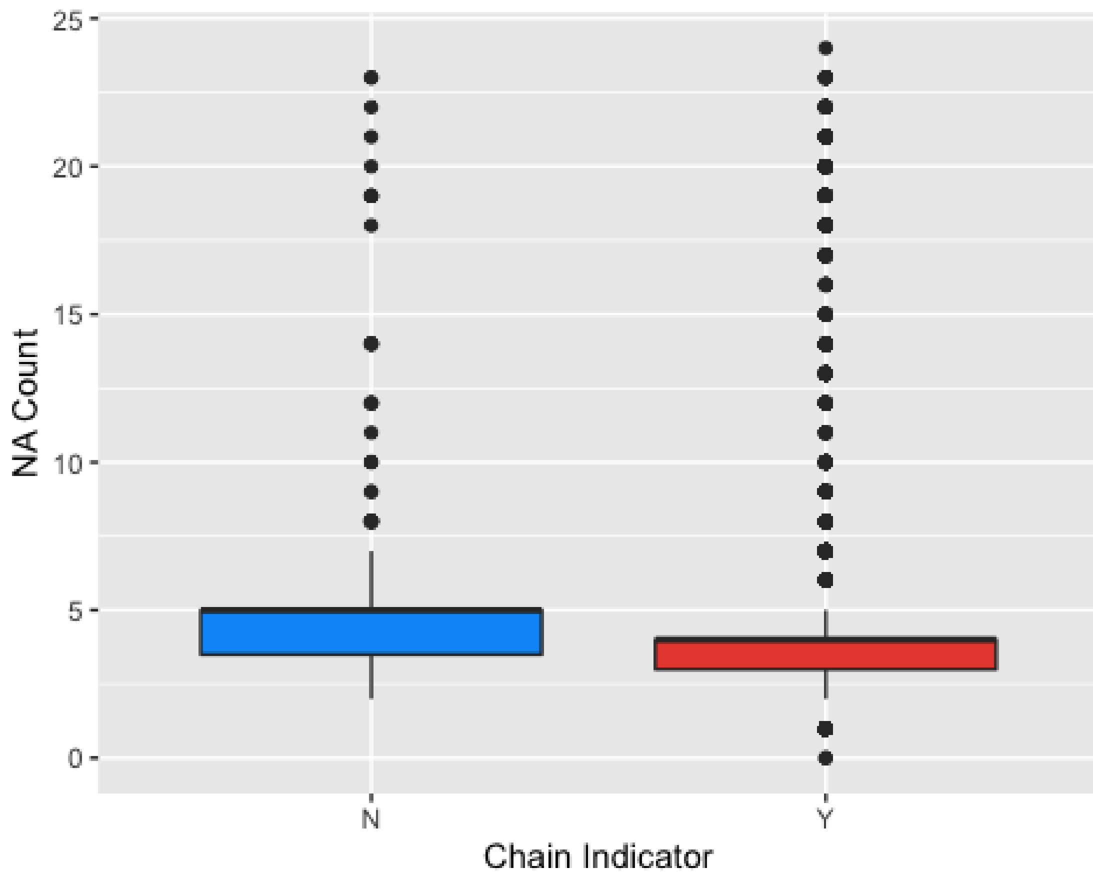
```
hcris_chain = subset(hcris_data, chain_indicator == "Y")
hcris_not_chain = subset(hcris_data, chain_indicator == "N")
hcris_chain$na_count <- apply(hcris_chain, 1, function(x) sum(is.na(x)))
sum(hcris_chain$na_count)/nrow(hcris_chain)
```

```
## [1] 4.077168
```

```
hcris_not_chain$na_count <- apply(hcris_not_chain, 1, function(x)
sum(is.na(x)))
sum(hcris_not_chain$na_count)/nrow(hcris_not_chain)
```

```
## [1] 5.279683
```

```
BothWithNA=rbind(hcris_not_chain,hcris_chain)
ggplot(BothWithNA, aes(x=chain_indicator, y=na_count)) + geom_boxplot(fill =
c("#0099f8", "#e74c3c")) +labs(
    x = "Chain Indicator",
    y = "NA Count")
```



From the source material provided for this project, we know that generally, chain providers have better data collection and organization systems than individual providers. We decided to attempt to analyze this trend by comparing the mean number of missing values for chain providers and non-chain providers. To perform this analysis, we split the original cleaned data set into two separate data sets: one with observations from chain providers, and one with observations from non-chain providers. Then, we created a new column in each data set that contained the number of NA values for each observation. To get the average number of NA values for an observation, we took the sum of each of these columns and divided the sums by the number of observations in each data set, respectively. This gave us an average of about 4 NA values recorded for each observation in the chain data, and an average of about 5 NA values recorded for each observation in the non-chain data. These results follow the background given in the paper, as they show a higher average amount of missing data per provider for non-chain providers than chain providers. In the box plot visualization, we can see that this conclusion is further supported as the plot shows that non-chains providers have a slight tendency to record more NA values than chain provider. If we were to take this analysis further, we would perform a statistical test that would determine whether or not this 1-value difference is statistically significant.