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

# ============================================

# 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)  
 }  
 }  
 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

hcris\_data$report\_start\_date = NULL  
hcris\_data$report\_end\_date = NULL

### Question 9

Clean zip codes

##### 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)  
# Cast to numeric  
hcris\_data$zip\_code = as.numeric(hcris\_data$zip\_code)  
  
##### 9b + 9c - Clean zip codes #####  
 hcris\_dataZipClean=as.data.frame(c())  
 for(i in unique(hcris\_data$prvdr\_num)){  
 testcase=hcris\_data%>%filter(prvdr\_num==i) %>%  
 select(zip\_code)  
 if(nrow(unique(na.omit(testcase)))==1){  
 hcris\_dataZipClean=rbind(hcris\_dataZipClean,hcris\_data %>%  
 filter(prvdr\_num==i) %>%  
 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))  
 }  
 }

### Question 10

Clean missing states

# 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 84040 3.00 4.00 3.00  
## 2 84041 3.00 4.00 3.00  
## 3 71340 6.00 4.00 3.00  
## 4 83296 6.00 4.79 3.00  
## 5 83312 6.00 4.50 3.00  
## 6 137791 6.00 4.50 3.00  
## dialyser\_reuse\_times dialyzer\_type epo\_cost epo\_net\_cost epo\_rebates  
## 1 29 1 188098 188098 0  
## 2 29 1 329642 329642 0  
## 3 26 1 153317 153317 0  
## 4 24 1 457764 457764 0  
## 5 25 1 446168 446168 0  
## 6 <NA> 1 63603 63603 0  
## epo\_total lab\_services non\_medicare\_sessions non\_medicare\_sessions\_indirect  
## 1 17922 <NA> 470 <NA>  
## 2 30254000 <NA> 536 <NA>  
## 3 16718 <NA> 1452 <NA>  
## 4 61460 <NA> 2793 <NA>  
## 5 59978 <NA> 3817 <NA>  
## 6 7700 <NA> 106 <NA>  
## num\_machines\_regular num\_machines\_standby supplies total\_costs\_hd\_benefits  
## 1 17 2 <NA> 56486  
## 2 19 2 <NA> 105211  
## 3 21 2 <NA> 32936  
## 4 20 2 <NA> 124969  
## 5 22 2 <NA> 125627  
## 6 12 2 <NA> 10770  
## total\_costs\_hd\_drugs total\_costs\_hd\_housekeeping total\_costs\_hd\_labs  
## 1 40069 117244 1611  
## 2 70369 247982 2731  
## 3 8221 150929 2428  
## 4 130622 311862 16943  
## 5 18369 269171 <NA>  
## 6 3557 32110 1900  
## total\_costs\_hd\_machines total\_costs\_hd\_other total\_costs\_hd\_salaries  
## 1 11614 42873 389516  
## 2 18589 50513 645684  
## 3 54179 52359 182378  
## 4 45471 432983 608435  
## 5 128129 274042 512477  
## 6 60483 38362 85242  
## total\_costs\_hd\_supplies total\_treatments\_hd total\_treatments\_pd  
## 1 113576 4752 <NA>  
## 2 244183 6248 <NA>  
## 3 70115 3637 <NA>  
## 4 217523 1092500 <NA>  
## 5 218153 13364 <NA>  
## 6 28391 1449 <NA>  
## certification\_date chain\_identity chain\_indicator ever\_hospital\_based  
## 1 01/01/1981 <NA> <NA> N  
## 2 01/12/1993 <NA> <NA> N  
## 3 11/01/2000 <NA> <NA> N  
## 4 04/09/1999 Other Y N  
## 5 03/25/1997 DaVita Y N  
## 6 08/03/2000 <NA> <NA> N  
## facility\_name prvdr\_num state zip\_code fy\_bgn\_dt  
## 1 AUSTIN DIAGNOSTIC CLINIC-SAN MARCOS 452522 TX 78758 1999-01-01  
## 2 AUSTIN DIAGNOSTIC CLINIC 452650 TX 78758 1999-01-01  
## 3 GUAM DIALYSIS CENTER INC. 652501 CA 90831 2000-11-01  
## 4 CORPUS CHRISTI DIALYSIS CENTER 452800 TX 78405 1999-10-31  
## 5 MESA VISTA DIALYSIS #372 452758 TX 79902 2000-01-01  
## 6 KIDNEY CARE OF ACADIANA - NEW IBERIA 192636 LA 70560 2000-08-03  
## fy\_end\_dt year chain\_id  
## 1 1999-09-09 1999 NA  
## 2 1999-09-09 1999 NA  
## 3 2000-12-31 2000 NA  
## 4 2000-12-31 2000 1  
## 5 2000-12-31 2000 2  
## 6 2000-12-31 2000 NA

# ============================================

# Question 3 - Analysis

# ============================================