Reproducing Grieco & McDevitt (2017)

Paul Schrimpf

Due: February 6th, 2018

In this problem set, we will attempt to reproduce the main results of Grieco and McDevitt (2017).

> Problem 1: http://tryr.codeschool.com/ is an interactive introduction to R. Please work through it if you have not used R before. If you're already familiar with R, then you can skip this.

If you're new to R, here is some advice about additional tools to use. R itself comes with either no GUI nor text-editor (on Linux) or a basic GUI with a limited text editor, (on Windows and I'd guess on Mac, but I don't know). There are numerous programs that provide a nicer way of working with R. The most popular is RStudio. It gives nice syntax highlighting, easier debugging, etc; it is somewhat similar to Matlab's GUI. A potentially useful, but not essential, tool for this assignment is the rmarkdown package. It lets you combine R code and text into a single document and produces nice looking output in multiple formats. I often use it for preliminary data work that I'm just looking at or sharing with coauthors and still making changes frequently. For more complete papers, I prefer to keep my code and the text of the paper separate.

1 Explore the data

I downloaded the data for this problem set from https://dialysisdata.org/content/dialysis-facility-repo
As in Grieco and McDevitt (2017) the data comes from Dialysis Facility Reports (DFRs) created under contract to the Centers for Medicare and Medicaid Services (CMS). However, there are some differences. Most notably, this data covers 2006-2014, instead of 2004-2008 as in Grieco and McDevitt (2017).

We will begin our analysis with some exploratory statistics and figures. There are at least two reasons for this. First, we want to check for any anomalies in the data, which may indicate an error in our code, our understanding of the data, or the data itself. Second, we should try to see if there are any striking patterns in the data that deserve extra attention.

The script downloadDialysisData.R downloads, combines, and cleans the data from https://dialysisdata.org/content/dialysis-facility-report-data. You can run this script yourself if you'd like, but it will take some time and download 825M of data. The result of the script is dialysisFacilityReports.Rdata, an R data file containing most of the variables used by Grieco and McDevitt (2017).

Load the data into R by typing

```
load("dialysisFacilityReport.R")
```

You could enter this directly into R's command line, but it is always better to write your commands in a script (or rmarkdown document) and run the script. You are bound to make mistakes, and it will be easier to fix them if you save your commands in a file instead of entering them one by one.

Your workspace should now contain a data frame named "dialysis." Data frames are how R stores data. A data frame is basically a matrix where the columns represent variables and the rows are observations. The variables/columns have names. To list the names of a data frame, run

```
names(df)
```

The meanings of these variables are listed in Table 1.

The raw data contains information on many variables in each of the previous 4 years. Staffing variables with no suffic are staff as of January 31, year as reported in year + 1. Staffing variables with ".l1" are staff as of January 31, year - 1 as reported in year + 1. If there were no reporting errors, the .l1 variables would equal the lag of the ones without .l1. However, you might find that this is not the case.

As explained in downloadDialysisData.R, data collected in year Y has information on most variables in years Y-1, Y-2, Y-3, and Y-4. However, for some variables and survey years, only information in years Y-2, Y-3, Y-4 is included. For such variables, at year Y-1, I use the value reported in survey year Y if it is available. If not, I use the value reported in survey year Y+1. The variables ending with ".p3" instead use the convention to use use Y-2 values if available and the Y-1 ones if not. Again, if there were no reporting errors the variables with and without .p3 would be the same.

Not all variables used Grieco and McDevitt (2017) are included here. Some variables will need to be transformed to be comparable to what is in the paper. For example, net investment in stations in year t is the difference between the number of stations in year $t+1$ and year in t.

```
#' Create lags for panel data.
#'
#' This function creates lags (or leads) of panel data variables.
#' Input data should be sorted by i, t --- e.g.
#' df <- df[order(i,t),]
#' @param x Vector or matrix to get lags of.
#' @param i unit index
#' @param t time index
#' @param lag How many time periods to lag. Can be negative if leading
#' values are desired.
#' @return Lagged copy of x.
panel.lag <- function(x, i, t, lag=1) {
  if (!identical(order(i,t),1:length(i))) {
    stop("inputs not sorted.")
  }
  if (is.matrix(x)) {
    return(apply(x,MARGIN=2,FUN=function(c) { panel.lag(c,i,t,lag) }))
  }
  if (length(i) != length(x) || length(i) != length(t) ) {
    stop("Inputs not same length")
  }
  if (lag>0) {
    x.lag <- x[1:(length(x)-lag)]
    x.lag[i[1:(length(i)-lag)]!=i[(1+lag):length(i)] ] <- NA
    x.lag[t[1:(length(i)-lag)]+lag!=t[(1+lag):length(i)] ] <- NA
    val <- (c(rep(NA,lag),x.lag))
  } else if (lag<0) {
    lag <- abs(lag)
    x.lag <- x[(1+lag):length(x)]
    x.lag[i[1:(length(i)-lag)]!=i[(1+lag):length(i)] ] <- NA
    x.lag[t[1:(length(i)-lag)]+lag!=t[(1+lag):length(i)] ] <- NA
```

Table 1: Variable definitions

Variable	Definition
provfs	provided identifier
year	year of data
city	city of provider
name	provider name
state	state of provider
network	network number*
chain.name	name of chain if provider is part of one
profit.status	whether for profit or non-profit
comorbities	average number of patient comorbidities
hemoglobin	average patient hemoglobin level
std.mortality	standardized mortality ratio
std.hosp.days	standardized hospitalization days
std.hosp.admit	standardized hospitalization admittance rate
pct.septic	percent of patients hospitalized due to septic infection
n.hosp.admt	number of hospitalizations
stations	number of dialysis stations
total.staff	total staff
dieticiansFT	full-time renal dieticians
dieticiansPT	part-time renal dieticians
nurseFT	full-time nurses ($>$ 32 hours/week)
nursePT	part-time nurses ($<$ 32 hours/week)
ptcareFT	full-time patient care technicians
ptcarePT	part-time patient care technicians
social.workerFT	full-time social workers
social.workerPT	part-time social workers
patient.months	number of patient-months treated during the year
pct.fistula	the percentage of patient months in which the patient received dialysis through arteriove
pct.female	percent of female patients
patient.age	average age of patients
patient.esrd.years	average number of years patients have had end stage renal disease
treatment.type	types of treatment provided at facility
inspect.date	date of most recent inspection
inspect.result	result of most recent inspection
inspect.cfc.cites	number of condition for coverage deficiencies in most recent inspection
inspect.std.cites	number of standard deficiencies in most recent inspection
days.since.inspection	days since last inspection

* I am unsure of the meaning of these variables. You could try checking the data guides and/or dictionaries on https://dialysisdata.org/content/dialysis-facility-report-data to find out.

```
    val <- (c(x.lag,rep(NA,lag)))
  } else { # lag=0
    return (x)
  }
  if (class(x)=="Date" & class(val)=="numeric") {
    stopifnot(0==as.numeric(as.Date("1970-01-01")))
    val <- as.Date(val, origin="1970-01-01")
  }
  return(val)
}
dialysis <- dialysis[order(dialysis$provfs, dialysis$year), ]
dialysis$change.stations <- panel.lag(dialysis$stations, dialysis$provfs,
                                      dialysis$year, lag=-1) - dialysis$stations
```

Net hiring can similarly be created. State inspection rates can be created as

```
dialysis$inspected.this.year <- (dialysis$days.since.inspection>=0 &
                                 dialysis$days.since.inspection<365)
dialysis$state.inspect.rate <- with(dialysis,
                                    ave(inspected.this.year,state,
                                        year, FUN=function(x) {mean(x,na.rm=TRUE)})
                                        )
```

Chain dummies and number of other facilities in the same city could also be created.

1.1 Descriptive statistics

Show some summary statistics with

```
summary(dialysis)
```

The builtin summary command is easy to use, but it does not quite provide all the information that we might want. For example, it does not show the standard deviation of each variable. We can calculate the standard deviation of a single variable with

```
sd(dialysis$patient.months, na.rm=TRUE)
```

or, we could calculate the standard deviation of all variables using the apply command,

```
apply(dialsys, 2, FUN=function(x) { sd(x, na.rm=TRUE) })
```

We can use the apply command to create something similar to part of Table 1 of Grieco and McDevitt (2017).

```
dialysis$for.profit <- dialysis$profit.status=="For Profit"
var.names <- c("days.since.inspection","for.profit")
tab1 <- t(apply(dialysis[,var.names], 2,
              function(input) {
                x <- as.numeric(input)
                c(mean(x,na.rm=TRUE), sd(x,na.rm=TRUE), sum(!is.na(x)))
              }))
colnames(tab1) <- c("Mean", "St. Dev.", "N")
tab1
```

There are multiple R packages for converting R matrices (like tab1) into a nicely formatted table. The stargazer package provides latex and html tables. Install it by typing

```
install.packages("stargazer")
```

You only need to install it once, but you need to load it in R session before you use it. Load it with

```
library(stargazer)
```

You could make a latex table with

```
stargazer(tab1,type="latex")
```

There are other packages for other formats. Here is a list of a few.

Problem 2: Create a table or tables containing similar information as Tables 1, 2, and 3 of Grieco and McDevitt (2017). Not all variables from the paper are available here, but include what you can. You could also choose to display additional summary statistics or variables. Comment on any large differences from the paper or other anomalies.

I did not make any intentional mistakes while creating the dialysis data frame from the raw data in the dialysis facility reports, but I am also not certain that there are not errors or omissions. If you suspect there is a problem, you can look at the downloadDialysisData.R code and the documentation at https://dialysisdata.org/content/dialysis-facility-report-data to figure out what might be wrong, and change downloadDialysisData.R to fix the problem.

1.2 Descriptive plots

Let's make some plots of the data. Here's a histogram of patient months

```
hist(dialysis$patient.months)
```

Here's a scatter plot of total staff and patient months

```
plot(x=dialysis$total.staff, y=dialysis$patient.months)
```

The builtin R plotting commands are convenient, but the ggplot2 package can create nicer looking figures. Creating nicer figures is not without a cost; the syntax for ggplot2 is far more verbose than the builtin plotting commands.

The following uses ggplot2 to show the state inspection rates vs year.

```
library(ggplot2) ## install.packages("ggplot2")
## plot state inspection rates vs year
fig.df <- aggregate(inspected.this.year ~ state*year, data=dialysis, FUN=mean)
fig <- ggplot(data=fig.df, aes(x=year, y=inspected.this.year,
                               colour=state)) +
  geom_point() +
  theme_minimal()
print(fig)
```

The plotly package can convert ggplot2 figures into interactive webpages. Among other things, it lets you hover over any point in a scatter plot and see the underlying data.

```
library(plotly) ## instack.packages("plotly")
ggplotly(fig)
```

Problem 3: Create scatter plots of output, labor, capital, and quality. Consider creating other exploratory plots as well. Try to be creative. Are there any strange patterns or other obvious problems with the data?

1.3 Infection rate and incentive shifters

The command for regression in R is "lm", which stands for linear model. For fixed effects regression, there is "felm". Here's an example:

```
library(lfe)
tab4.col1 <- felm(pct.septic ~ I(days.since.inspection/365) +
                    patient.age + pct.female + patient.esrd.years + pct.fistula +
                    comorbidities + hemoglobin | provfs, clustervar="provfs",
                  data=dialysis)
summary(tab4.col1)
```

Problem 4: Reproduce results similar to table 4 of Grieco and McDevitt (2017). Not all columns with be reproducible since our data does not have referrals, or HSAs. Comment on notable differences.

2 Production function estimation

Problem 5: Reproduce columns 2,3, 5, and 6 of table 5. Use the lm() command for OLS and felm() for fixed effects.

Problem 6: Estimate α. I recommend using a polynomial to approximate the control function. You can then estimate α from an IV regression. For example,

```
## Create quality measure
```

```
dialysis$quality <- -residuals(lm(pct.septic ~ pct.fistula +
                                    pct.female + patient.age +
                                    patient.esrd.years + hemoglobin +
                                    comorbidities,
                                  data=dialysis, na.action=na.exclude))

## poly() does not work well with NA's so subset the data accordingly
inc <- with(dialysis,stations>0 & labor>0 & patient.months>0 & hiring
            != 0 & !is.na(stations) & !is.na(labor) & !is.na(hiring) &
               invest==0 & !is.na(std.mortality))
deg.phi <- 3
deg.iv <- 3
iv1 <- felm( log(patient.years) ~
             I(hiring>0)*(poly(log(stations), log(labor), hiring,
                         state.inspect.rate, degree=deg.phi ))*(for.profit +
                             comp.factor)
                        | 1 | (quality  ~ std.mortality) | provfs,
                        data=subset(dialysis,inc))

alpha <- coef(iv1)[grep("quality",names(coef(iv1)))]
Phi.step1 <- iv1$fitted.values - alpha*subset(dialysis,inc)$quality
dialysis$Phi <- NA
dialysis[names(Phi.step1),"Phi"] <- Phi.step1
```

You may want to change some aspects of this specification to better match Grieco and McDevitt (2017).

If you'd prefer, you can instead use the 3-step procedure based on Robinson (1988) that Grieco and McDevitt (2017) describe. This procedure first partials out the control function, and then estimates an IV regression using the residuals. There are multiple R packages for local linear regression. I like the "np" package, but beware that it will be somewhat slow for a dataset this large.

Problem 7: Estimate β. To do so, you most write an function that for a given β returns the objective function. You then minimize this function. To compute η given $\beta = b$, you could do the following:

```
eta.func <- function(b,alpha,df,degree) {
  bx <- with(df,log(stations)*b[1] + log(labor)*b[2])
  omega <- df$Phi - bx
  omega.lag <- panel.lag(omega,i=df$provfs,t=df$year)
  yhat <- with(df,log(patient.years) - alpha*quality - bx)
  tmp <- data.frame(yhat=yhat,omega.lag=omega.lag)
  rownames(tmp) <- rownames(df)
  eta <- residuals(lm(yhat ~ poly(omega.lag,degree=degree),
                      na.action=na.exclude,
                      data=subset(tmp, is.finite(yhat) &
                                       is.finite(omega.lag)))
                   )
  tmp$eta <- NA
  tmp[names(eta),"eta"] <- eta
  return(tmp$eta)
}
```

To minimize your objective function, you can use the nloptr package.

```
library(nloptr)
```

```
obj <- function(b) {
  ## You have to write the body of this function
}

nloptr(x0=c(0.2,0.2), eval_f=obj, opts=list(algorithm="NLOPT_LN_BOBYQA",
                                            print_level=3))
## You may want to try changing algorithms and/or initial values
```

Optimization algorithms can fail and are often sensitive to initial values. Since we are minimizing a function of two variables, it may be helpful to plot the objective.

```
library(rgl) ## For 3-d plots

bk <- seq(0,0.8,length.out=15)
bl <- seq(0,0.8,length.out=15)
grid <- expand.grid(b1=bk,
                    b2=bl)
grid$fval <- apply(grid,1,obj)

zlim <- range(grid$fval,na.rm=TRUE)
zlen <- zlim[2] - zlim[1] #+ 1
colorlut <- terrain.colors(100,alpha=1) # height color lookup table
col <- colorlut[ floor(100*((grid$fval-zlim[1])/zlen))+1 ] # assign colors to
    heights for each point
persp3d(bk, bl, matrix((grid$fval),nrow=length(bk),ncol=length(bl)),
        xlab="bk",ylab="bl",zlab="fval", color=col,
        alpha=0.8, shiniess=128)
```

Comment on your results. If your estimates are very different than those in Grieco and McDevitt (2017), speculate as to why. Is it the different data? The changes in estimation details? A bug in your code? Describe how you could isolate the source of the difference.

References

Grieco, Paul LE and Ryan C McDevitt. 2017. "Productivity and Quality in Health Care: Evidence from the Dialysis Industry." Review of Economic Studies (forthcoming) URL http://www.restud.com/wp-content/uploads/2016/08/MS17933manuscript.pdf.