Tarea Examen Final :: Estadística Computacional

Marcos Olguín Martínez

5 de diciembre de 2015

Instrucciones

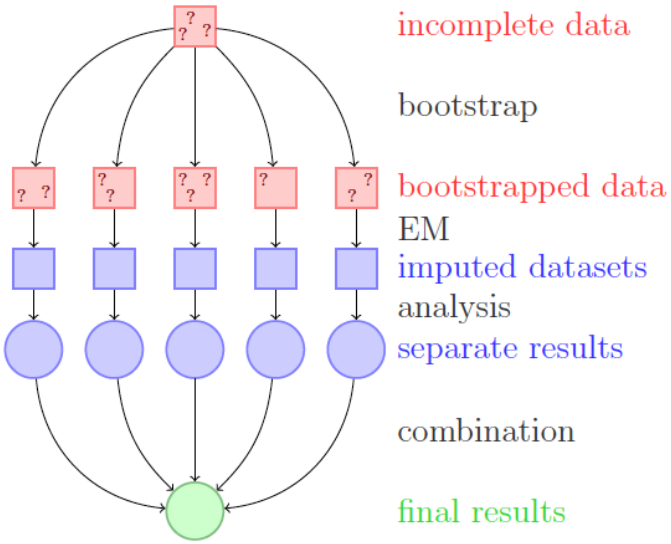
En la clase se vió el algoritmo EM condicional para datos normales multivariados, y se implementó la metodología de **Amelia II** para imputar datos faltantes en una base de datos. Amelia II es un método de *state-of-the-art* para imputación de datos basado en los métodos de la clase, se implementa un método EM condicional generalizado con bootstrapping para hacer imputación de datos eficientemente.

Para este examen se utilizó la base de `"datos_politicos.dta"` de comunidad ITAM.

Se implementó Amelia II como sigue:

a. Usando los datos completos, calcula el vector de medias y matriz de covarianzas.
b. Llena los huecos faltantes simulando de una normal con la media y varianza que le corresponde según el inciso anterior.
c. Calcula la logverosimilitud completa con estos datos. Tienes que usar una normal multivariada con el vector de medias y matriz de covarianzas del paso a).
d. Hasta que haya convergencia, actualiza variable por variable. En un paso de actualización tienes que hacer una regresión de la variable en cuestión respecto a las demás y luego sustituir un dato faltante con su valor predicho. En cada iteración tienes que calcular el vector de medias y matriz de covarianzas y volver a calcular la logverosimilitud hasta que converja.
e. Habiendo terminado la imputación, haz un modelo **logit** para predecir el sistema de gobierno con las demás variables y guarda en una variable los resultados del modelo, así como el éxito de predicción.
f. Haz **pooling** para presentar los datos juntos.

En resumen, la documentación de Amelia II indica que el paquete funciona de acuerdo al siguiente diagrama:

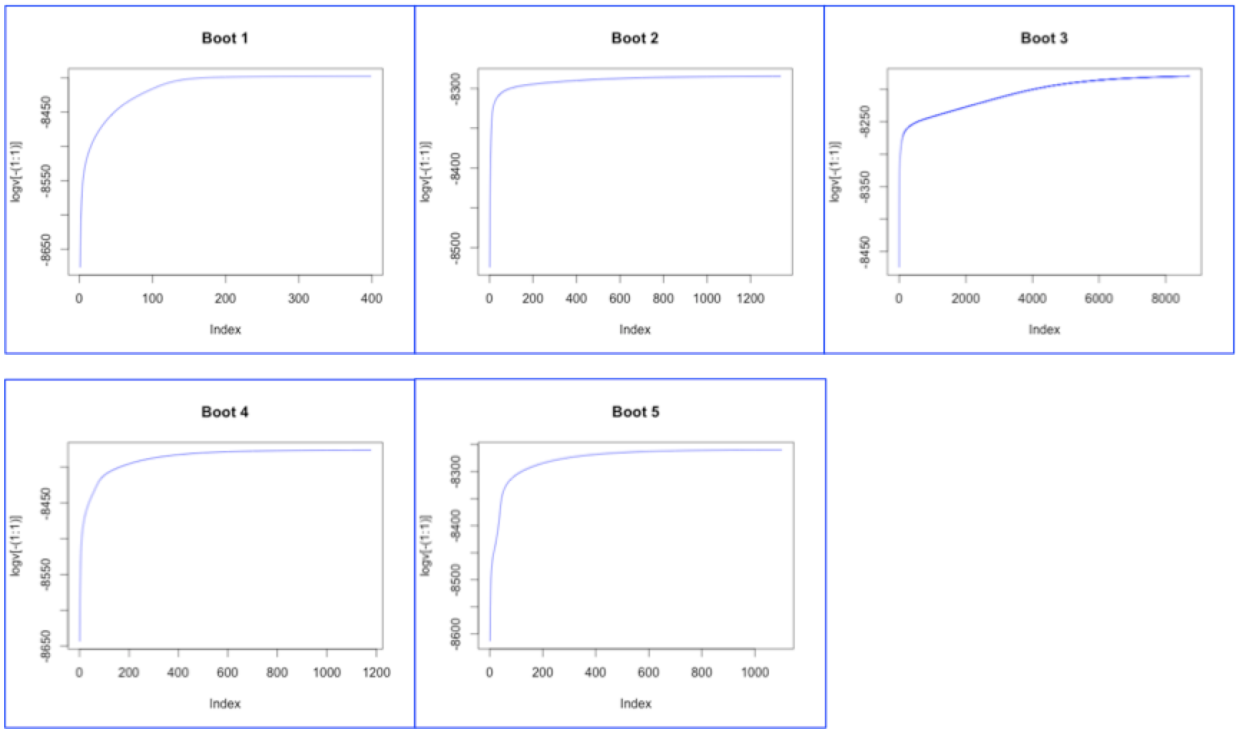


Reporte de resultados

El código tarda un poco en ejecutarse, en promedio como 30 minutos, considerando un **m=5** y con una computadora con 16 Gb en Ram, en equipos más pequeños tardaría un poco más. Sin embargo, la documentación de la metodología indica que es mucho más rapido que otros programas que hacen tareas similares, lo cual lo hace un programa sumamente competitivo.

```
#[1] "Boots 1"
#[1] "iteraciones totales: 399"
#[1] "Boots 2"
#[1] "iteraciones totales: 1339"
#[1] "Boots 3"
#[1] "iteraciones totales: 8719"
#[1] "Boots 4"
#[1] "iteraciones totales: 1178"
#[1] "Boots 5"
#[1] "iteraciones totales: 1102"
```

Lás gráficas que se generaron fueron las siguientes:



Las predicciones son suficientemente buenas, aquí mostramos una tabla en donde podemos ver la proporción de éxito para cada una de las bases imputadas:

Boot	Proporción de éxito
1	0.7682927
2	0.8292683
3	0.8536585
4	0.8414634
5	0.8353659

Como se puede ver, mientras más iteraciones haya tenido la tabla, mayor proporción de éxito tiene en la predicción del sistema de gobierno (0, democracia; 1, autocracia).

La tabla de las betas estimadas para cada variable es la siguiente:

Variables	Betas	SD
level	-0.00014	0.00035
open	0.00523	0.01501
g	-0.09918	0.24394
strikes	-0.10921	1.48225
govpwt	0.03118	0.10863
ls	0.00975	0.08200
invpwt	0.14194	0.13933
fertil	-0.04056	3.24329
lfagric	0.08938	0.23305
popg	0.59493	1.44442
femsec	0.00927	0.03842
EnergProd	0.00000	0.00001

Este es un gran método de múltiple imputación, muy sencillo de usar y bastante eficiente.

Código

El código que se utilizó fue el siguiente:

```
#### Datos

X <- subset(data, select = list.depend)

for (j in 1:ncol(X)) X[,j] <- as.numeric(X[,j])
row.names(X) <- data$name
X.comp <- X[complete.cases(X),]
nrow(X.comp)

View(round(cor(X.comp),2))

data.full <- data.frame(Y[complete.cases(X)], X.comp)

names(data.full)[1] <- "Y"


nrows <- nrow(X)
ncols <- ncol(X)
m <- 5
tol <- 1e-3
res <- list()
imputed.sets <- list()
pred.success <- numeric(m)

for (rep in 1:m) {
    print(paste("Boots",rep))
```

```
    samp <- sample(1:nrows,nrows,replace=TRUE)
    Xb <- X[samp,]
    M <- is.na(Xb)
    Sigma <- cov(Xb[complete.cases(Xb),])
    sd.vec <- sqrt(diag(Sigma))
    mu <- apply(Xb[complete.cases(Xb),],2,mean)
    for(i in 1:nrows) for(j in 1:ncols) if(M[i,j]) Xb[i,j] <- rnorm(1, mu[j], sd.vec[j])
    logv <- sum(apply(Xb, 1, function(row) log(dmvnorm(row,mu,Sigma))))
    ###-Iteracion-
    iter <- 1
    repeat{
        ####-Valor de la verosimilitud-
        for(j in 1:ncols){
            ind <- as.matrix(Xb[,j],ncol=1)
            dep <- as.matrix(Xb[,-j])
            mod <- lm(ind~dep)
            pred <- predict(mod)
            for(k in 1:nrows) if (M[k,j]) Xb[k,j] <- pred[k]
            #Xb[M[,j],j]<-pred[M[,j]]
        }
        ####-Nueva matriz de covarianza-
        Sigma <- cov(Xb)
        mu <- apply(Xb,2,mean)
        logv[iter+1] <- sum(apply(Xb, 1, function(row) log(dmvnorm(row,mu,Sigma))))
        if (abs(logv[iter+1]-logv[iter]) < tol) break
        iter <- iter+1
    }
    print(paste("iteraciones totales:", iter))
    imputed.sets[[rep]] <- Xb
    ####-Grafica-
    plot(logv[-(1:1)], type="l", col="blue", main = paste("Boot",rep))
    ####-Modelo-
    data.full <- data.frame(Y[samp], Xb)
    names(data.full)[1] <- "Y"
    res[[rep]] <- glm(Y~., data = data.full, family = "binomial")
    guess <- round(predict(res[[rep]], type="response"))
    pred.success[rep] <- sum(guess==data.full$Y)/(nrows)
}

pred.success
```

Saludos cordiales.