Instituto Tecnológico Autónomo de México

Econometría Aplicada I

Exámen Final: Parte Inidividual

Profesor: Irving Arturo de Lira Salvatierra

Alumno: Marco Antonio Ramos Juárez

142244

Ciudad de México 2020

Índice

```
#Activo que se muestre el códgio
knitr::opts_chunk$set(echo = TRUE)
```

```
#packageurl <- "https://cran.r-project.org/src/contrib/Archive/rowr/rowr_1.1.3.tar.gz"
#install.packages(packageurl, repos=NULL, type="source")
```

Ambiente

```
#Desactivo la notación científica
options(scipen=999)

# Primero importo los paquetes que se usaran.
library(MASS)
library(rv)
```

```
## Warning: package 'rv' was built under R version 3.5.2
```

```
library(dplyr)
library(sandwich)
library(quantreg)
```

```
## Warning: package 'quantreg' was built under R version 3.5.2
```

```
## Warning: package 'SparseM' was built under R version 3.5.2
```

```
library(ggplot2)
library(MLmetrics)
library(rowr)
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 3.5.2
```

I.

```r
# Primero creo mi base de datos con variables aleatorias que provienen de una distribución nor
set.seed(12346)
p <- 100  ## normal multivaraida de p dimensiones
set.seed(0); X <- matrix(runif(p * p), p, p)  ## Esta es la matriz de rango completo
COV <- crossprod(X)  ## t(X) %*% X
mu <- rep(0, p)  ## Vector de medias 0
x <- mvrnorm(5000, mu, COV)  ## murnorm(sample.size, mean, covariance)
x<-as.data.frame(x)


#Creo una función para bautizar mis columnas como xi
colRename<-function(x){
  for(i in 1:ncol(x)){
    colnames(x)[i] <- paste("x",i,sep="")
  }
  return(x)
}
x<-colRename(x)
```

II.

```r
#Creo el vector y y lo anexo a la base de datos x
y<-rcauchy(n = 5000, location = -2, scale = 1)
x<-cbind(x,y)
```

III.

```r
#Primero preparo un data frame con yt+1 en la misma fila que xt. (solo quito la primera observ

mylist<-y[-1]
newelem <- 0
y_2 <- c(mylist, newelem)
x<-cbind(x,y_2)
```

```r
#Aimismo quito la ultima observación en t=5000 pues ya no tiene un y_t+1 con el cuál podaos re
x_clean<-x[-c(5000),]


#Segundo, calculo las regresiones cuantílicas para cada variable
coefs_raw<-lapply(seq_len(ncol(x_clean)),function(i) {coef(rq(x_clean$y_2~x_clean[,i],tau=.2))

# Extraigo solamente los coeficientes de los objetos generados
coefs <- data.frame(matrix(unlist(coefs_raw), nrow=length(coefs_raw), byrow=T))

#Edito el data frame para que esté ordenado
coefs<-coefs %>% mutate (Var=paste("x",1:102, sep = ""))
colnames(coefs) <- paste(c("B0","B1","Var"))


#nota, la variable x101 y x102 en realidad se refiere a la regresión de y_2 con y y consigo mi
#Por ello el data frame final de este inciso es el siguiente

respuesta_3<-coefs[-c(101:102),]
```

IV.

```r
#Primero preparo un data frame donde pueda encontrar para cada t, todas las xs y las bs.
x_transpose <- as.data.frame(t(as.matrix(x_clean)))
cross<-cbind(x_transpose,coefs)
#Quitamos las variables extra de tal manera que podamos hacer un lapply limpio
cross$Var <- NULL
cross$B0 <- NULL
cross<-cross[-c(101:102),]

#Segundo, calculo las regresiones para cada t
coefs_raw_t<-lapply(seq_len(ncol(cross)),function(i) {coef(lm(cross[,i]~cross$B1))})

# Extraigo solamente los coeficientes de los objetos generados
coefs_t <- data.frame(matrix(unlist(coefs_raw_t), nrow=length(coefs_raw_t), byrow=T))
```

```r
#Edito el data frame para que esté ordenado
coefs_t<-coefs_t %>% mutate (Var=paste("t",1:5000, sep = ""))
colnames(coefs_t) <- paste(c("alpha","efe_1","T"))

# Elimino la ultima fila que en realidad es la regresión de B1 con sigo mismo
respuesta_4<-coefs_t[-c(5000),]
```

V.

```r
#Prepar una base de datos con las efes relacionadas con las ys
cross_final<-cbind(coefs_t,y_2)
cross_final<-cross_final[-c(5000),]




#Segundo, calculo las regresiones cuantílicas para cada variable
final_reg_quant<- rq(y_2~efe_1, data = cross_final,tau=.2)
final_reg_quant
```

```
## Call:
## rq(formula = y_2 ~ efe_1, tau = 0.2, data = cross_final)
##
## Coefficients:
##  (Intercept)        efe_1
## -3.414556746  0.003455476
##
## Degrees of freedom: 4999 total; 4997 residual
```

VI.

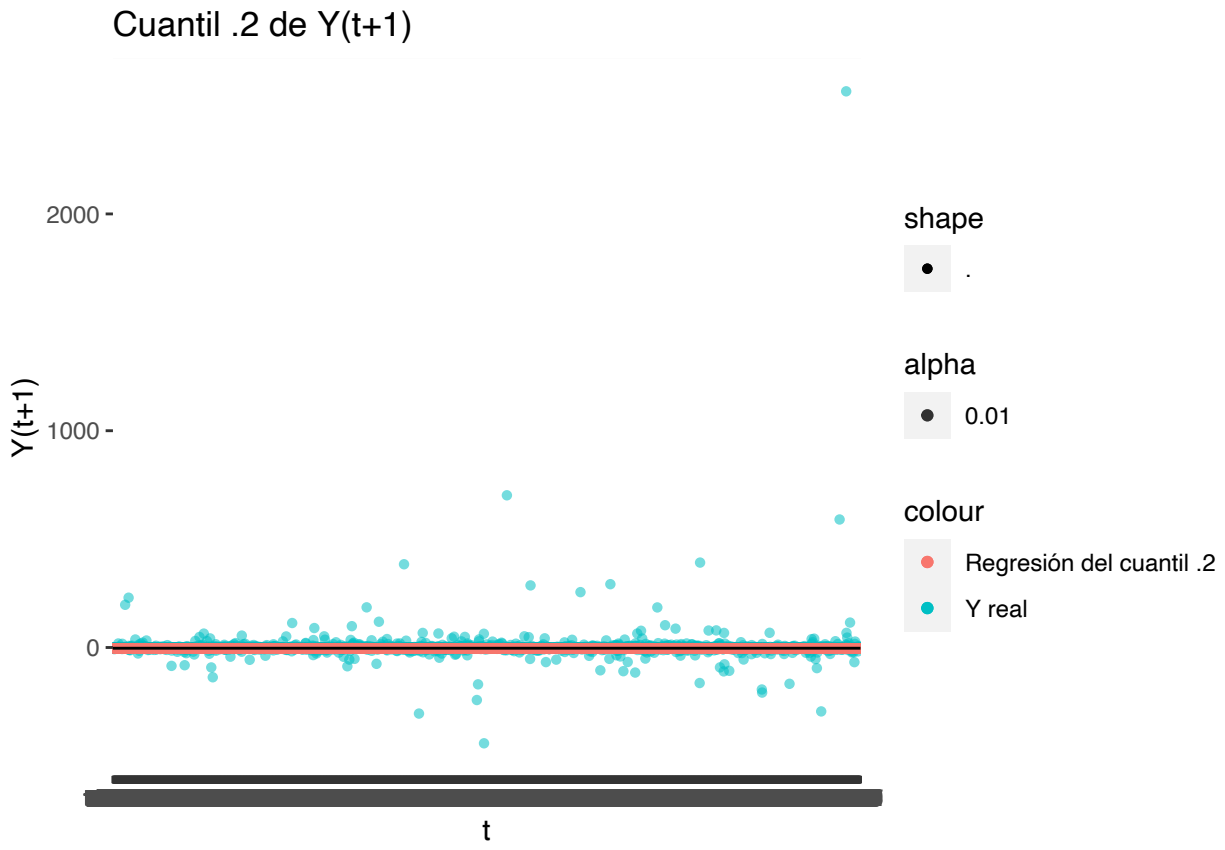
La f indica la relación de cada x_t con cada $y_(t+1)$ ajustada a x_t.

VII.

```
cross_final<-cross_final %>%
  mutate( final_reg_quant = predict(final_reg_quant))

plot_data<-(subset(cross_final, subset=(cross_final$y_2 <= quantile(cross_final$y_2 , 0.20))))
```

```
 q2<-quantile(cross_final$y_2 , 0.20)
```

```
ggplot(cross_final,aes(T,y_2,colour="Y real",shape=".", alpha=.01))+ geom_point() +geom_point(a
```



VIII.

```
#Para hacer este ejercicio agrupo los pasos que realicé en una función.
```

```
proceso <- function(muestra) {
#Calculo las regresiones cuantílicas para cada variable
coefs_raw<-lapply(seq_len(ncol(muestra)),function(i)          {coef(rq(muestra$y_2~muestra[,i]

coefs <- data.frame(matrix(unlist(coefs_raw), nrow=length(coefs_raw), byrow=T))
#Extraigo los coeficientes
coefs <- data.frame(matrix(unlist(coefs_raw), nrow=length(coefs_raw), byrow=T))
#Edito el data frame para que esté ordenado
coefs<-coefs %>% mutate (Var=paste("x",1:102, sep = ""))
colnames(coefs) <- paste(c("B0","B1","Var"))

#PARTE B
# Preparo un data frame donde pueda encontrar para cada t, todas las xs y las bs.
x_transpose <- as.data.frame(t(as.matrix(muestra)))
cross<-cbind(x_transpose,coefs)
#Quitamos las variables extra de tal manera que podamos hacer un lapply limpio
cross$Var <- NULL
cross$B0 <- NULL
cross<-cross[-c(101:102),]

#Calculo las regresiones para cada t
coefs_raw_t<-lapply(seq_len(ncol(cross)),function(i) {coef(lm(cross[,i]~cross$B1))})

#Extraigo solamente los coeficientes de los objetos generados
coefs_t <- data.frame(matrix(unlist(coefs_raw_t), nrow=length(coefs_raw_t), byrow=T))
#Edito el data frame para que esté ordenado
colnames(coefs_t) <- paste(c("alpha","efe_1"))
#Elimino el último renglon que es la regresión de una variable consigo misma
coefs_t <- coefs_t[-nrow(coefs_t),]

#PARTE C
#Prepar una base de datos con las efes relacionadas con las ys
cross_final<-cbind(coefs_t,muestra$y_2)
#Segundo, calculo las regresiones cuantílicas para cada variable
final_reg_quant<- rq(muestra$y_2~efe_1, data = cross_final,tau=.2)
```

```
   return(final_reg_quant)
}
```

```
#Ahora simplemente divido la base de datos en dos muestras
#Y aplico la función creada.

seccion_a <- x_clean[1:4000,]
```

Para la base de datos completa:

Para la submuestra con las primeros 4000 observaciones:

```
coef(proceso(seccion_a))
```

```
## (Intercept)       efe_1
##  -3.4153624   0.0049586
```

Para la submuestra con las últimas 1000 (en realidad 999) observaciones:

En cuanto a los MSE:

```
#En primer lugar, creo una columna de valores predecidos
seccion_a<-seccion_a %>%
  mutate( proceso_a = predict(proceso(seccion_a)))
```

MSE de la submuestra:

```
MSE(seccion_a$proceso_a, seccion_a$y_2)
```

```
## [1] 2202.864
```

IX.

```
#Creo 1000 bases de datos correspondientes para cada ventana
for (i in 0:999){
 nam <- paste("D", i, sep = "")
 assign(nam, x_clean[(1+i):(4000+i),])}
```

```r
#En esta sección se realiza la estimación de ventana, sin embargo no pude ejecutar un lapply a
a<-lapply(list(D0,D1,D2,D3,D4,D5,D6,D7,D8,D9,D10,D11,D12,D13,D14,D15,D16,D17,D18,D19,D20,D21,D

b<-lapply(list(D100,D101,D102,D103,D104,D105,D106,D107,D108,D109,D110,D111,D112,D113,D114,D115

c<-lapply(list(D300,D301,D302,D303,D304,D305,D306,D307,D308,D309,D310,D311,D312,D313,D314,D315
```

```r
# Extraigo los coeficientes de cada regresión
coefa <- data.frame(matrix(unlist(a), nrow=length(a), byrow=T))
coefb <- data.frame(matrix(unlist(b), nrow=length(b), byrow=T))
coefc <- data.frame(matrix(unlist(c), nrow=length(c), byrow=T))
```

```r
#Anexo las efes de la base de datos para calcular la y predecida

bindcoefs <- bind_rows(coefa, coefb, coefc)
colnames(bindcoefs) <- paste(c("gamma_0","gamma_1"))
append<-cross_final[4900:4999,]
final_a<-cbind(append,bindcoefs)
```

```
## Warning in data.frame(..., check.names = FALSE): row names were found from a
## short variable and have been discarded
```

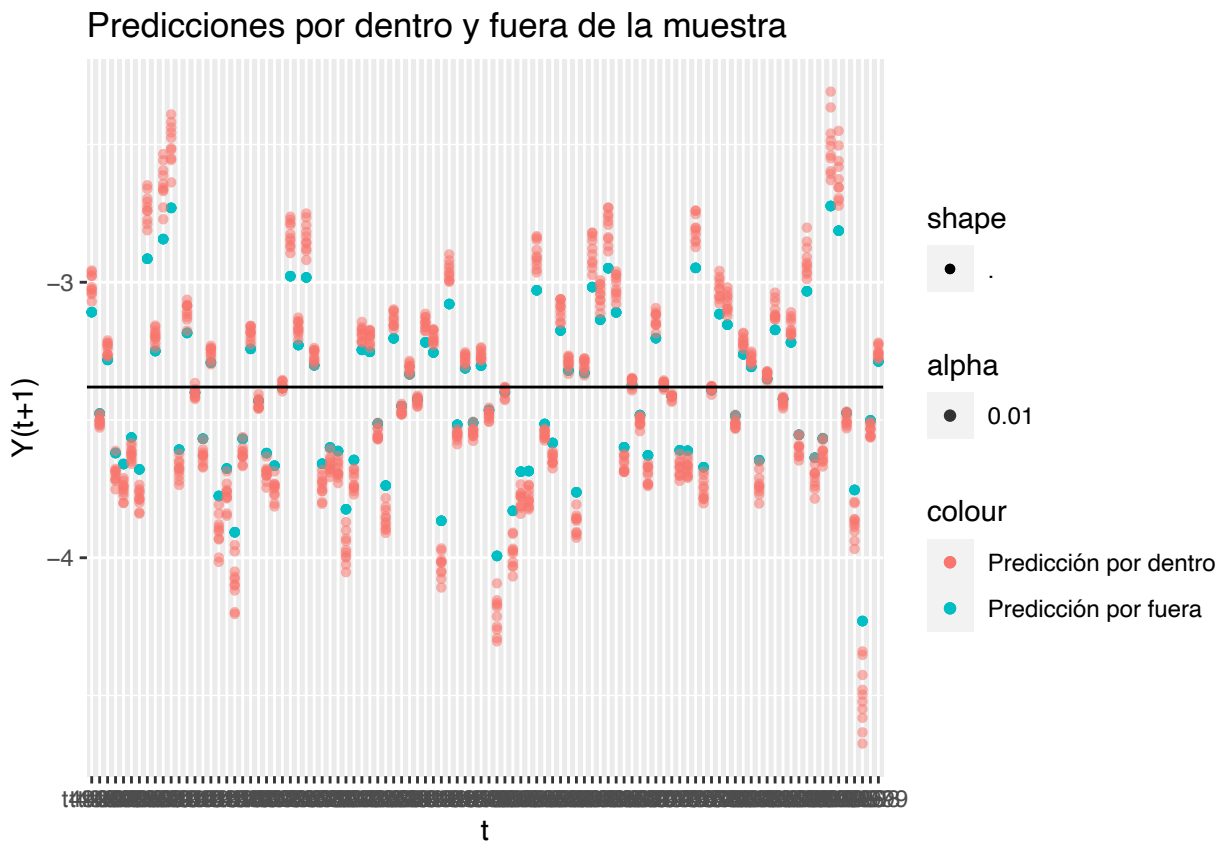
```r
final_a<-final_a%>%
          mutate(predicted=(gamma_0+gamma_1*efe_1))
write.csv(final_a,"final_a.csv")
```

```r
#Anexo las efes de la base de datos para calcular la y predecida
final_a<-final_a%>%
          mutate(predicted_error=((y_2-predicted)^2))

# El MSE de esta predicción es:
mean(final_a$predicted_error)
```

```
## [1] 852.452
```

X.

```
ggplot(final_a,aes(T,final_reg_quant,colour="Predicción por fuera",shape=".", alpha=.01))+ geo
```



XI.

Para esta sección simplemente realizaremos un ejercicio similar: dividiremos la base de datos en dos partes, una con 4000 observaciones y una con mil observaciones. Nuestro objetivo es predecir las y_{t+1} de los ultimos 1000 periodos (en realidad 999) periodos de tal manera que la ventana sea de 4000. Para eso realizaremos primero primero una regresion con la primer submuestra que luego utilizaremos para predecir las últimas 999 y_{t+1} (análisis dentro de la muestra) para posteriormente realizar el análisis fuera de la muestra.

```
#Estimación del modelo dentro de la base de datos:
drop <- c("y","proceso_a")
seccion_a_beta = seccion_a[,!(names(seccion_a) %in% drop)]
```

```
linearmodel<-lm(y_2 ~ .,seccion_a_beta)
seccion_b <- x_clean[4000:4999,]
seccion_b_beta = seccion_b[,!(names(seccion_a) %in% drop)]
pred_fuera<-predict(linearmodel,seccion_b_beta)

##Estimación del modelo dentro de la base de datos de fuera:

drop_b <- c("y_2")
ventana_beta = x_clean[,!(names(x_clean) %in% drop)]

#Para calcular las regresiones de ventana movil, aprovecho las 1000 bases de datos que ya creé

d<-lapply(list(D0,D1,D2,D3,D4,D5,D6,D7,D8,D9,D10,D11,D12,D13,D14,D15,D16,D17,D18,D19,D20,D21,D

e<-lapply(list(D500,D501,D502,D503,D504,D505,D506,D507,D508,D509,D510,D511,D512,D513,D514,D515

#Ahora extraigo los coeficientes para continuar con las predicciones
coefd <- data.frame(matrix(unlist(d), nrow=length(d), byrow=T))
coefe <- data.frame(matrix(unlist(e), nrow=length(d), byrow=T))
bindcoefs <- bind_rows(coefd, coefe)

ventana_beta$y_2<-NULL

ventana_beta<-ventana_beta%>% mutate(B0=1)%>%
  select(B0, everything())
ventana_beta<-ventana_beta[4900:4999,]

#Finalmente realizo la predicción del modelo "fuera"
predict_final<-data.frame(
 Map(function(x,y) if(all(is.numeric(x),is.numeric(y))) x * y else x, ventana_beta, bindcoefs)
)
pred_dentro<-rowSums(predict_final)

comparacion<-as.data.frame(cbind(pred_fuera,pred_dentro,seccion_b$y_2))
colnames(comparacion) <- paste(c("fuera","dentro","real"))
```

XII.

```
# Para resolver si cuál modelo mejor simplemente calculamos los MSE:

comparacion<-comparacion%>%
          mutate(error_fuera=((real-fuera)^2))%>%
          mutate(error_dentro=((real-dentro)^2))

# El MSE de esta predicción para el análisis fuera:
mean(comparacion$error_fuera)
```

```
## [1] 490.0803
```

```
# El MSE de esta predicción para el análisis dentro:
mean(comparacion$error_dentro)
```

```
## [1] 486.9803
```

```
#Comparado con el del modelo de factores
mean(final_a$predicted_error)
```

```
## [1] 852.452
```

```
#El modelo lineal tiene un error medio más pequeño por lo que parece mejor.
```

XIII.

A.

Para esta sección se decidió imputar los valores faltantes en la base de datos con el promedio de cada variable.

```
pib <- read.csv("data.csv")
pib$t<-NULL
```

```r
mylist<-y[-1]
newelem <- 0
y_2 <- c(mylist, newelem)
pib<-cbind(pib,y_2)
pib$a<-NULL

#editamos nuestra función de proceso para que contemple regresiones lineales
#Para hacer este ejercicio agrupo los pasos que realice en una función.

proceso_lineal <- function(muestra) {
#Calculo las regresiones cuantílicas para cada variable
coefs_raw<-lapply(seq_len(ncol(muestra)),function(i)          {coef(lm(muestra$y_2~muestra[,i])

coefs <- data.frame(matrix(unlist(coefs_raw), nrow=length(coefs_raw), byrow=T))
#Extraigo los coeficientes
coefs <- data.frame(matrix(unlist(coefs_raw), nrow=length(coefs_raw), byrow=T))
#Edito el data frame para que esté ordenado
coefs<-coefs %>% mutate (Var=paste("x",1:9, sep = ""))
colnames(coefs) <- paste(c("B0","B1","Var"))

#PARTE B
# Preparo un data frame donde pueda encontrar para cada t, todas las xs y las bs.
x_transpose <- as.data.frame(t(as.matrix(muestra)))
cross<-cbind(x_transpose,coefs)
#Quitamos las variables extra de tal manera que podamos hacer un lapply limpio
cross$Var <- NULL
cross$B0 <- NULL

#Calculo las regresiones para cada t
coefs_raw_t<-lapply(seq_len(ncol(cross)),function(i) {coef(lm(cross[,i]~cross$B1))})

#Extraigo solamente los coeficientes de los objetos generados
coefs_t <- data.frame(matrix(unlist(coefs_raw_t), nrow=length(coefs_raw_t), byrow=T))
#Edito el data frame para que esté ordenado
colnames(coefs_t) <- paste(c("alpha","efe_1"))
#Elimino el último renglon que es la regresión de una variable consigo misma
```

```
coefs_t <- coefs_t[-nrow(coefs_t),]

#PARTE C
#Prepar una base de datos con las efes relacionadas con las ys
cross_final<-cbind(coefs_t,muestra$y_2)
#Segundo, calculo las regresiones cuantílicas para cada variable
final_reg<- lm(muestra$y_2~efe_1, data = cross_final)

   return(final_reg)
}

proceso_lineal(pib)
```

```
##
## Call:
## lm(formula = muestra$y_2 ~ efe_1, data = cross_final)
##
## Coefficients:
## (Intercept)        efe_1
##      43.498        2.102
```

B.

```
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 3.5.2
```

```
library(astsa)
library(forecast)
```

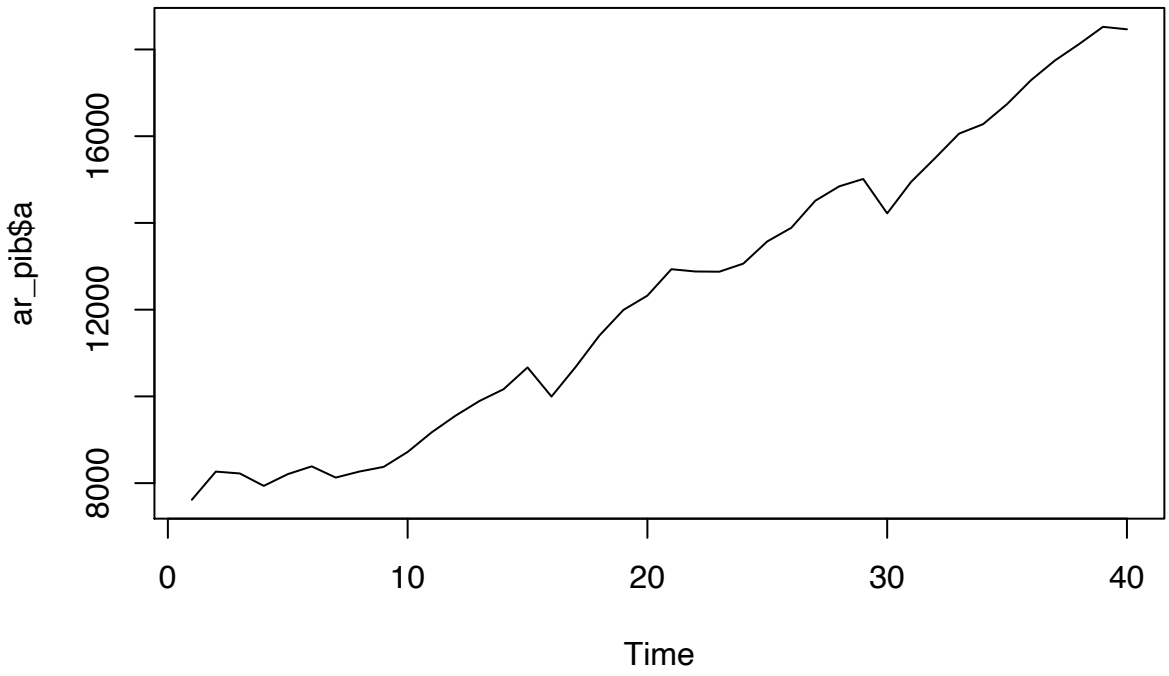
```
##
## Attaching package: 'forecast'

## The following object is masked from 'package:astsa':
##
##     gas
```

```
## The following object is masked from 'package:rv':
##
##     is.constant
```

```
ar_pib <- read.csv("data.csv")
```

```
arr <- ts(ar_pib, frequency=1)
```

```
plot.ts(ar_pib$a)
```




```
arimaModel_1=arima(ar_pib$a, order=c(0,1,2))
arimaModel_1
```

```
##
## Call:
## arima(x = ar_pib$a, order = c(0, 1, 2))
##
## Coefficients:
```

```
##          ma1     ma2
##       0.3162  0.1628
## s.e.  0.1714  0.1232
##
## sigma^2 estimated as 169248:  log likelihood = -290.17,  aic = 586.33
```

C.

De acuerdo a las estadísticas de resumen de los modelos parece que la metodología de factores tiene un mejor valor explicativo derivado de un p value diminuto de menos del 0.0000000000000002.

```
summary(proceso_lineal(pib))
```

```
##
## Call:
## lm(formula = muestra$y_2 ~ efe_1, data = cross_final)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.029  -6.535  -0.371   7.869 197.058
##
## Coefficients:
##              Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 43.498495   0.247565   175.7 <0.0000000000000002 ***
## efe_1        2.102395   0.008574   245.2 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.89 on 4998 degrees of freedom
## Multiple R-squared:  0.9233, Adjusted R-squared:  0.9232
## F-statistic: 6.012e+04 on 1 and 4998 DF,  p-value: < 0.00000000000000022
```

```
summary(arimaModel_1)
```

```
##
## Call:
## arima(x = ar_pib$a, order = c(0, 1, 2))
```

```
##
## Coefficients:
##          ma1     ma2
##       0.3162  0.1628
## s.e.  0.1714  0.1232
##
## sigma^2 estimated as 169248:  log likelihood = -290.17,  aic = 586.33
##
## Training set error measures:
##                     ME     RMSE      MAE      MPE     MAPE     MASE       ACF1
## Training set 182.0245 406.2246 332.4285 1.449493 2.867435 0.854013 -0.2603912
```

D.