Universidad de Buenos Aires
Facultad de Ciencias Económicas

Carrera de Actuario.
Cátedra de Análisis Numérico.
Prof. Titular: Javier García Fronti.

Curso modalidad Virtual

Prof. Adjunto: R. Darío Bacchini

Melisa Elfenbaum

Análisis predictivo con métodos de aprendizaje automático

Implementaciones de ML en R

1. **Modelado de temas sobre datos de Twitter**
   - Datos de Twitter
   - Modelo: Latent Dirichlet Allocation
2. **Clasificación de créditos**
   - Set de datos de créditos
   - Modelo: Regresión Logística
3. **Agrupamiento de acciones**
   - Datos de acciones
   - Modelo: K-means

Modelado de temas sobre datos de Twitter

- **Datos:**
  - Obtener las claves de Twitter para acceder a la API
- **Librerías en R:**
  - sqldf, ggplot2, wordcloud, RColorBrewer
  - twitteR, tm, syuzhet, topicmodels
- **Documentación:**
  - API de Twitter: https://developer.twitter.com/en/docs/twitter-api
  - Método de ML: http://www.cse.cuhk.edu.hk/irwin.king/_media/presentations/latent_dirichlet_allocation.pdf

Modelado de temas sobre datos de Twitter

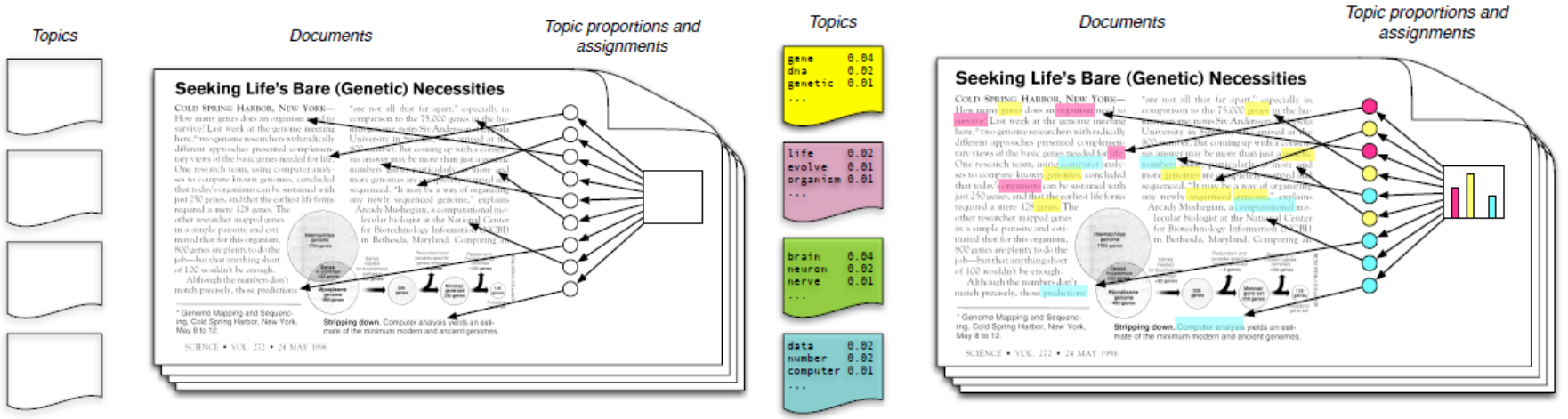
Latent Dirichlet Allocation (LDA)

- Permite descubrir los temas que ocultos en una gran colección de documentos no estructurados.
- Sencillo y ampliamente utilizado en la actualidad, disponible en diversos lenguajes de programación como R, Matlab, Java y Python.
- Trata cada documento como una mezcla de temas y cada tema como una mezcla de palabras y asigna un conjunto de temas ponderados a cada uno de los documentos

Modelado de temas sobre datos de Twitter

Latent Dirichlet Allocation (LDA)

- Dadas las palabras y los documentos que son observables, el objetivo es inferir la estructura de temas latente, ajustando de forma iterativa la importancia relativa de los temas en los documentos y las palabras en los temas.



Modelado de temas sobre datos de Twitter

- **Paquetes**

```
#Extraccion de datos de Twitter
library(twitteR)
#Consultas sql sobre los dataframe
library(sqldf)
options(sqldf.driver = "SQLite")
#Limpieza y mineria de textos
library(tm)
#Analisis de sentimientos
library(syuzhet)
```

Modelado de temas sobre datos de Twitter

- **Paquetes**

```
#Nube de palabras
library(wordcloud)
#Colores
library(RColorBrewer)
#Graficos
library(ggplot2)
#Modelado de temas
library(topicmodels)
```

Modelado de temas sobre datos de Twitter

- **Recolección de datos con la API de Twitter**

```
consumer_key <- ""
consumer_secret <- ""
access_token <- ""
access_secret <- ""

#Conexion a la API
setup_twitter_oauth(consumer_key, consumer_secret,
access_token, access_secret)

#searchTwitter: busqueda de tweets
tweet_corpus <- searchTwitter('#hashtag',  n = 10000,
lang = 'en', resultType = 'recent')

#Eliminacion de retweets
tweet_corpus <- strip_retweets(tweet_corpus)
```

Modelado de temas sobre datos de Twitter

- **Parámetros de la búsqueda de tweets**

  https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets

q	Búsqueda dentro de los tweets de palabras clave y usuarios
geocode	Ubicación de los usuarios en un radio determinado de la latitud/longitud dada
lang	Restringe tweets al idioma dado (de acuerdo al código ISO 639-1)
result_type	El tipo de resultados de búsqueda: recent (resultados más recientes), popular (resultados más populares) y mixed (recientes y populares). El valor predeterminado es "mixed".
count	El número de tweets hasta un máximo de 100. El valor predeterminado es 15.
until	Devuelve los tweets generados antes de la fecha indicada. La fecha debe tener el formato AAAA-MM-DD
since_id	Devuelve resultados con un ID mayor que (más recientes) el ID especificado.
max_id	Devuelve resultados con un ID menor que (más antiguos) o igual que el ID especificado.

Modelado de temas sobre datos de Twitter

- **Otras funciones de la API de Twitter**

```
#userTimeline: publicaciones de un usuario
tweets_usuario <- userTimeline('user',  n = 1000)
tweets_usuario_df <- twListToDF(tweets_usuario)

#getTrends: busqueda de tendencias por localidad
localidades <- availableTrendLocations()
tendencias_arg <- getTrends(woeid = 23424747)
tendencias_mundo <- getTrends(woeid = 1)
```

Modelado de temas sobre datos de Twitter

- **Almacenamiento de los datos**

```
#Data frame
tweet_df <- twListToDF(tweet_corpus)

#Almacenamiento en un csv
write.csv(tweet_df, 'C:/datos_twitter.csv')

#Lectura del csv con los tweets descargados
tweet_df <- read.csv("C:/datos_twitter.csv")
tweets <- iconv(tweet_df$text, to = "ASCII")
```

Modelado de temas sobre datos de Twitter

- **Análisis de sentimiento**

```
#Sentimientos y emociones con distintos metodos
sent_syuzhet <- get_sentiment(tweets, method="syuzhet")
sent_bing <- get_sentiment(tweets, method="bing")
sent_afinn <- get_sentiment(tweets, method="afinn")
sent_nrc <- get_sentiment(tweets, method="nrc")
nrc_data <- get_nrc_sentiment(tweets)

#Se agrega el analisis de sentimiento al dataframe
tweet_df <- cbind(tweet_df, sent_syuzhet, sent_bing,
sent_afinn, sent_nrc, nrc_data)

#Sentimientos y emociones en distintos idiomas
nrc_data <- get_nrc_sentiment(tweets, language="spanish")
```

Modelado de temas sobre datos de Twitter

- **Análisis de sentimiento**

```
#Cant de tweets con sentimientos positivos/negativos
sqldf()
sentimientos <- sqldf("select sum(case when sent_syuzhet> 0
then 1 else 0 end) as Cantidad,  'pos_syuzhet' as Sentimiento
from tweet_df union select sum(case when sent_syuzhet< 0 then
1 else 0 end) ,'neg_syuzhet' from tweet_df      union select
sum(case when sent_bing> 0 then 1 else 0 end) ,'pos_bing'
from tweet_df union select sum(case when sent_bing< 0 then 1
else 0 end), 'neg_bing' from tweet_df union select sum(case
when sent_afinn> 0 then 1 else 0 end), 'pos_afinn' from
tweet_df union select sum(case when sent_afinn< 0 then 1 else
0 end), 'neg_afinn' from tweet_df union select sum(case when
sent_nrc> 0 then 1 else 0 end), 'pos_nrc' from tweet_df union
select sum(case when sent_nrc< 0 then 1 else 0 end),'neg_nrc'
from tweet_df")
```

Modelado de temas sobre datos de Twitter

- **Análisis de sentimiento**

```
#Cant de tweets con sentimientos positivos/negativos
graf_sentimientos<-ggplot(data=sentimientos,
aes(x=reorder(Sentimiento, -Cantidad), y=Cantidad,
fill=Sentimiento)) +
geom_bar(stat="identity") +coord_flip()+
geom_text(aes(label=Cantidad),  color="black",
size=3.5)
graf_sentimientos
```

Modelado de temas sobre datos de Twitter

- **Análisis de sentimiento**

```
#Cantidad de tweets con distintas emociones
sqldf()
emociones <- sqldf("select sum(joy) as Cantidad,
'Alegria' as Emocion from tweet_df
union select sum(fear), 'Miedo' from tweet_df
union select sum(anger), 'Ira' from tweet_df
union select sum(sadness),  'Tristeza' from tweet_df
union select sum(trust), 'Confianza' from tweet_df
union select sum(surprise), 'Sorpresa' from tweet_df
union select sum(disgust), 'Disgusto' from tweet_df
union select sum(anticipation), 'Anticipacion' from
tweet_df")
```

Modelado de temas sobre datos de Twitter

- **Análisis de sentimiento**

```
#Cantidad de tweets con distintas emociones
graf_emociones<-ggplot(data=emociones,
aes(x=reorder(Emocion, -Cantidad), y=Cantidad,
fill=Emocion)) +
geom_bar(stat="identity") +coord_flip()+
geom_text(aes(label=Cantidad),  color="black",
size=3.5)
graf_emociones
```

Modelado de temas sobre datos de Twitter

- **Limpieza de datos**

```
#Se cargan los datos como un corpus
corpus <- Corpus(VectorSource(tweets))
removeURL <- function(x) gsub("http[[:alnum:]]*", "", x)
toSpace <- function (x , pattern ) gsub(pattern, " ", x)
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, content_transformer(toSpace), "/")
corpus <- tm_map(corpus, content_transformer(toSpace), "@")
corpus <- tm_map(corpus, content_transformer(toSpace), "\\|")
corpus <- tm_map(corpus, content_transformer(removeURL))
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, stemDocument)
corpus <- tm_map(corpus, removeWords, stopwords('english'))
corpus <- tm_map(corpus, removeWords, c("this", "the", "for",
"you", "appl", "get", "iphon"))
```

Modelado de temas sobre datos de Twitter

- **Análisis exploratorio: frecuencia y nube de palabras**

```
#Matriz de terminos
tdm <- TermDocumentMatrix(corpus)

#Frecuencias
term.freq <- rowSums(as.matrix(tdm))
term.freq <- subset(term.freq, term.freq >=50)
df <- data.frame(term = names(term.freq), freq =
term.freq)
graf_freq_palabras <- ggplot(df, aes(x=term, y=freq)) +
geom_bar(stat = "identity", fill="#f70388") +
xlab("Palabras") + ylab("Cant") +coord_flip()
graf_freq_palabras

#Nube de palabras
wordcloud(corpus, max.words = 100, random.order =
FALSE, scale=c(2,.5), col = brewer.pal(8, "Dark2"))
```

Modelado de temas sobre datos de Twitter

- **Modelado de temas**

```
#Matriz de documentos
dtm <- as.DocumentTermMatrix(tdm)

#Metodo Latent Dirichlet Allocation (LDA)
doc.lengths <- apply(dtm , 1, sum)
dtm <- dtm[doc.lengths >0,]
k <- 3
iter <- 20000
lda <- LDA(dtm, k = k, method="Gibbs",
control=list(iter = iter))

#Los 10 terminos mas importantes de cada grupo
lda.terms <- as.matrix(terms(lda,10))
lda.terms
```

Clasificación de créditos

- **Datos:**
  - Dataset público: https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29
- **Librerías en R:**
  - caret
  - ggplot2
  - Hmisc

Clasificación de créditos

Regresión logística

- Cuando los valores que se desean predecir toman valores discretos se realiza un proceso de clasificación en el cual se asigna cada una de las observaciones a una categoría o clase.
- Si se tienen que identificar 2 clases, el problema es de clasificación binaria y si se tienen que identificar más de 2 clases, el problema es de clasificación múltiple.

Clasificación de créditos

Regresión logística

- Uno de los métodos de clasificación binaria más utilizados y más populares en la actualidad
- Se modela la probabilidad de que la respuesta pertenezca a una categoría particular utilizando la función logística, en el que esa respuesta solo puede tomar dos valores, donde se define a 0 como la clase negativa y a 1 como la clase positiva y se asigna la clase más probable.

Clasificación de créditos

- **Paquetes**

```
#caret (Classification And REgression Training)
library(caret)
#Graficos
library(ggplot2)
#Para cortar una variable numerica en intervalos
library(Hmisc)
```

- **Carga de datos**

```
german_credit =
read.table("http://archive.ics.uci.edu/ml/machine-
learning-databases/statlog/german/german.data")
```

Clasificación de créditos

- **Renombramiento de columnas y datos**

```
#Nombres de las columnas

colnames(german_credit) = c("chk_acct", "duration",
"credit_his", "purpose", "amount", "saving_acct",
"present_emp", "installment_rate", "sex", "other_debtor",
"present_resid", "property", "age", "other_install",
"housing", "n_credits", "job", "n_people", "telephone",
"foreign", "response")

#Valores de clases (1 = Bueno, 2 = Malo)

german_credit$response  <-
replace(as.character(german_credit$response),
german_credit$response == "2", "Malo")

german_credit$response  <-
replace(as.character(german_credit$response),
german_credit$response == "1", "Bueno")
```

Clasificación de créditos

- **Análisis preliminar de los datos**

```
#Cantidad de creditos por clase

table(german_credit$response)

#Grafico Credit amount y Duration, color por clase

qplot(amount, duration, color=response,
data=german_credit)

#Proporciones de variables por cada clase

prop.table(table(german_credit$foreign,
german_credit$response),1)

prop.table(table(german_credit$housing,
german_credit$response),1)
```

Clasificación de créditos

- **Análisis preliminar de los datos**

```
#Proporciones de variables por cada clase
#Intervalos para las edades
cutAge<-cut2(german_credit$age, g=5)
tabla <-table(cutAge, german_credit$response)
prop.table(tabla, 1)
#Intervalos para la duracion de los creditos
cutDuration<-cut2(german_credit$duration, g=5)
tabla <-table(cutDuration, german_credit$response)
prop.table(tabla, 1)
```

Clasificación de créditos

- **Separación de datos para entrenar y testear**

```
#CreateDataPartition: training (80%) y test (20%)

inTrain <- createDataPartition(german_credit$response,
p=0.8, list=FALSE)

training <- german_credit[ inTrain, ]

testing <- german_credit[ -inTrain, ]
```

- **Regresión logística**

```
#Modelo con todos los predictores

mod_fit <- train(response ~ .,  data=training,
method="glm", family="binomial")

mod_fit
```

Clasificación de créditos

- **Regresión logística**

```
#Predicciones con datos de testeo
pred <- predict(mod_fit, newdata=testing)
table(pred)
#Comparacion de predicciones con las clases reales
tabla <-table(pred,testing$response)
tabla
prop.table(tabla, 1)
```

Clasificación de créditos

```
#Modelo con menos variables

mod_fit <-  train(response ~ chk_acct + duration +
property+ credit_his + amount + age + job+ saving_acct
+ purpose + sex + other_install + installment_rate,
data=training, method="glm", family="binomial")

mod_fit

#Predicciones con datos de testeo

pred = predict(mod_fit, newdata=testing)

table(pred)

#Comparacion de predicciones con las clases reales

tabla <-table(pred,testing$response)

prop.table(tabla, 1)
```

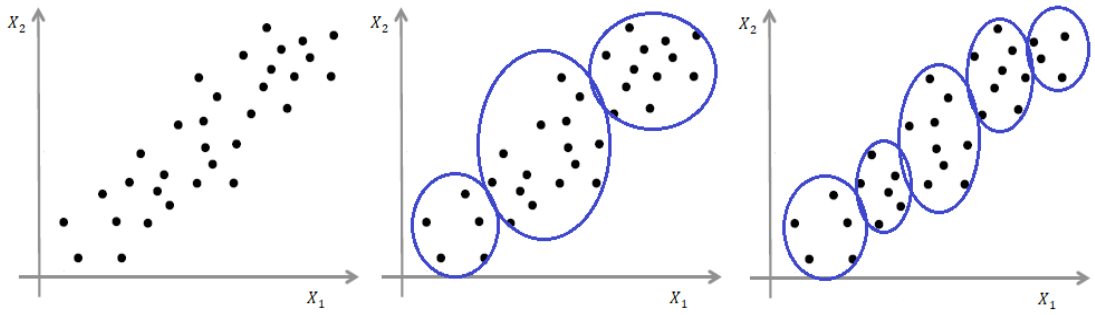
Agrupamiento de acciones

- **Datos:**
  - Datos de acciones descargados desde Yahoo Finance
- **Librerías en R:**
  - quantmod
  - caret
  - ggplot2
  - xlsx

Agrupamiento de acciones

K-means

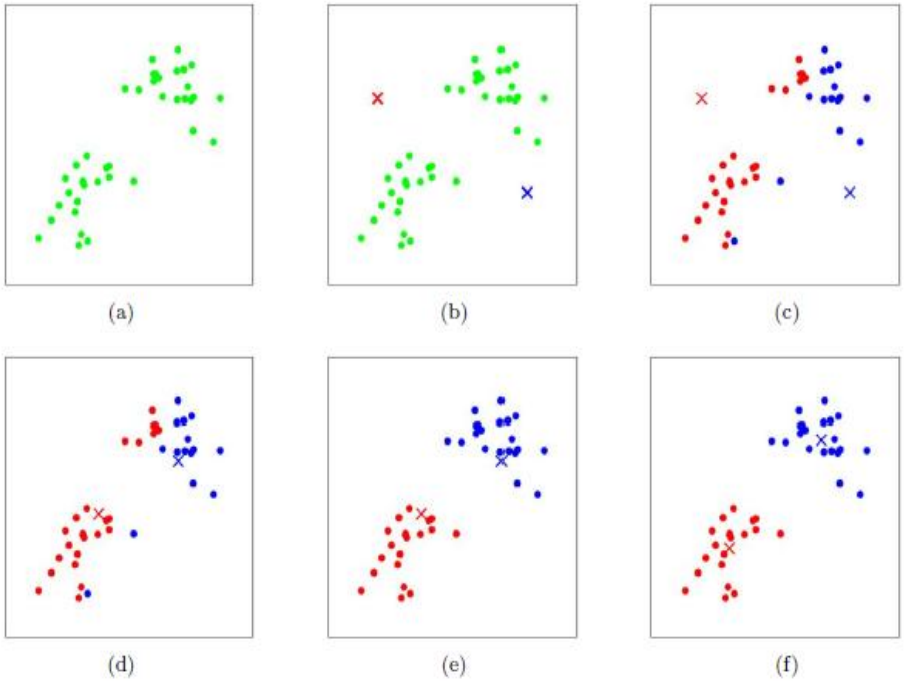
- Es un método de agrupamiento que se utiliza para dividir un conjunto de datos en grupos o clusters distintos no superpuestos
- Hay que especificar el número deseado de grupos y luego con el algoritmo se asigna cada observación a uno de ellos



Agrupamiento de acciones

K-means

1) Se inicializan los centros de los clusters aleatoriamente
2) Se asigna cada observación al centro más cercano -distancia euclidiana- y se recalculan los centros como un promedio de los puntos asignados al mismo. Se repite hasta que converge



Agrupamiento de acciones

- **Paquetes**

```
#caret (Classification And REgression Training)
library(caret)
#Graficos
library(ggplot2)
#Para obtener los datos de las acciones, quantmod: Quantitative Financial Modelling & Trading Framework
library(quantmod)
#Para trabajar con archivos de Excel
library(xlsx)
```

Agrupamiento de acciones

- **Datos de una acción**

```
#Obtencion de datos de una accion

getSymbols('AMZN', from = '2020-01-01', to = '2021-01-
01',warnings = FALSE, auto.assign = TRUE)

#Grafico

chartSeries(AMZN, type="line")

#Retornos de distinta periodicidad

dailyReturn(AMZN)

monthlyReturn(AMZN)
```

Agrupamiento de acciones

- **Datos de una acción**

```
#Volatilidad

m=length(AMZN$AMZN.Close)

AMZNVolatility <-volatility(AMZN,n=m)

AMZNVolatility[[m]]
```

- **Obtención de datos de varias acciones**

```
#Carga de los simbolos de las acciones

setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

source('acciones.R')
```

Agrupamiento de acciones

- **Obtención de datos de varias acciones**

```
i<-0

for(valor in simbolos) {
   i<-i+1
   print(i)
   simbolo <- getSymbols(valor, from = "2021-01-01", to =
   "2021-04-01", auto.assign = FALSE, src = 'yahoo')
   simbolo$Retorno <- dailyReturn(simbolo)

   #cantidad de dias del precio de cierre para calcular
   volatilidad
   m=length(simbolo[4])
   Volatilidad <-volatility(simbolo,n=m)
   Volatilidad <-Volatilidad[[m]]
```

Agrupamiento de acciones

- **Obtención de datos de varias acciones**

```
    if (i==1) {
      datasetAcciones <- data.frame(lapply(simbolo[, 5:7],
    mean, na.rm = TRUE))
      datasetAcciones <-cbind(datasetAcciones, Volatilidad)
      colnames(datasetAcciones)<-
    c("Volumen","Precio","Retorno", "Volatilidad")
    } else {
      colnames(simbolo)[5]  <- "Volumen"
      colnames(simbolo)[6]  <- "Precio"
      datasetAcciones <-
      rbind(datasetAcciones,c(lapply(simbolo[, 5:7], mean,
      na.rm = TRUE), Volatilidad = Volatilidad))
    }
row.names(datasetAcciones)[i]<-valor
}
closeAllConnections()
```

Agrupamiento de acciones

- **Almacenamiento de los datos**

```
#Almacenamiento y lectura

write.xlsx(datasetAcciones, "C:/acciones.xlsx",
row.names=TRUE)

datasetAcciones <- read.xlsx("C:/acciones.xlsx",
sheetIndex = 1, row.names=TRUE)
```

- **K-means**

```
#Modelo con 2 clusters, agrupo de acuerdo a volatilidad
y retorno

kmeansStocks<-kmeans(subset(datasetAcciones, select = -
c(Precio, Volumen)), centers=2)

kmeansStocks$centers
```

Agrupamiento de acciones

- **K-means**

```
#Asignacion del cluster a cada accion

datasetAcciones$cluster <-
as.factor(kmeansStocks$cluster)

table(datasetAcciones$cluster)

#Grafico del retorno y volatilidad, color por cluster

qplot(Retorno, Volatilidad,  colour=cluster,
data=datasetAcciones)
```