Lec8: Unsupervised Learning II

Isidoro Garcia Urquieta

2021

Agenda

- Reducción de Dimensionalidad
- Modelos de Factores
- Principal Components Analysis
- Principal Components Regression
- Partial Least Squares
- Principios de Natural Language Processing
- Topic Modelling

Reducción de Dimensionalidad

La clase de hoy se trata de **reducción de dimensionalidad**. En el contexto de Big data esto es muy importante.

- Buscamos los 'factores' más importantes de la base de datos $X_{n,p}$.
- Para lograr esto, veremos algunos algoritmos muy interesantes que reducen el espacio de X.
- Los modelos de Factores son muy importantes en el mundo de Data Science. Tienen aplicaciones muy variadas.

Principal Component Analysis

Supongamos que seguimos con una base de datos $X_{n,p}$. Queremos hacer una visualización de 2x2. Sin embargo, tendríamos que hacer $\binom{p}{2}$ gráficas! Para $p = 10$ son 45! para p grandes se vuelve un problema imposible.

Veamos como se estima:

El primer componente principal es:

$$z_1 = \phi_{11}X_1 + \phi_{21}X_2 + ... + \phi_{p1}X_p$$

Donde $\Sigma_{j=1}^{p}\phi_{j1}^2 = 1$. Que tiene la **mayor varianza**. Esto es, es la combinación lineal de las $X's$ que captura la mayor varianza.

Noten como cambié el orden de la regresión! Porqué?

Principal Component Analysis

Como calculamos el primer componente?

1. Centramos $X_{n,p}$ a que todas las variables tengan media cero.
2. Resolvemos:

$$argmax_{\phi_{11},\ldots,\phi_{p1}}\{\frac{1}{N}\Sigma_{i=1}^{N}(\Sigma_{j=1}^{p}\phi_{j1}x_{ij})^2\}$$

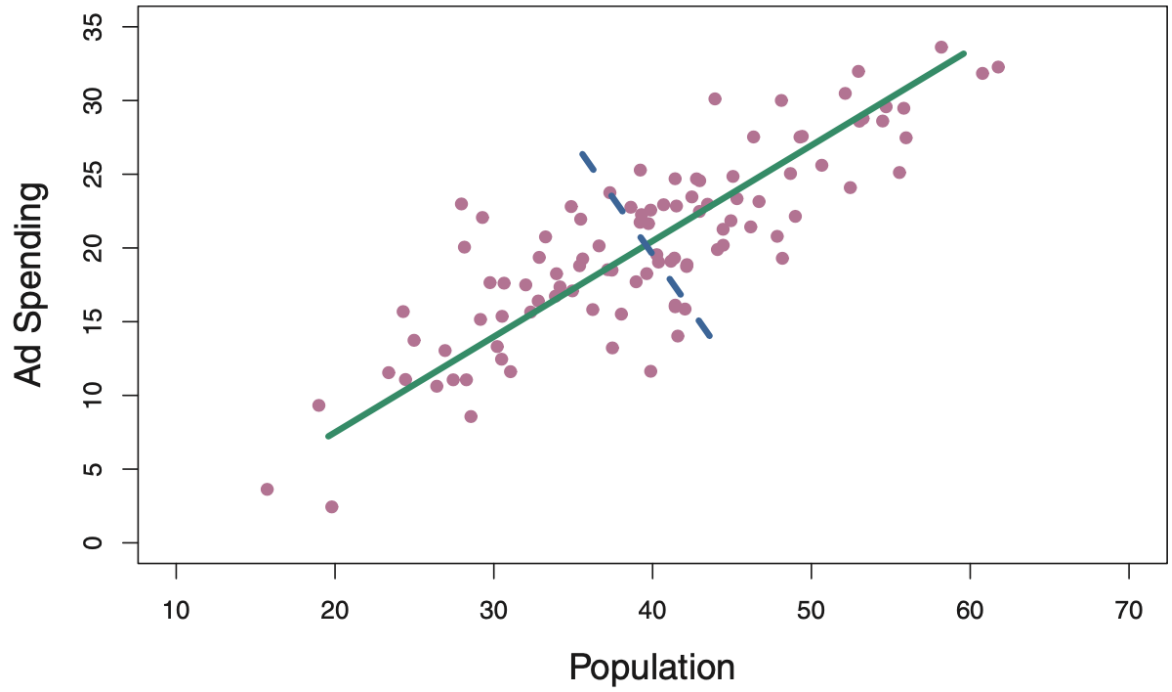
$$subject\ to\ \ \Sigma_{j=1}^{p}\phi_{j1}^2 = 1$$

Este problema se resuelve con eigen decomposición.

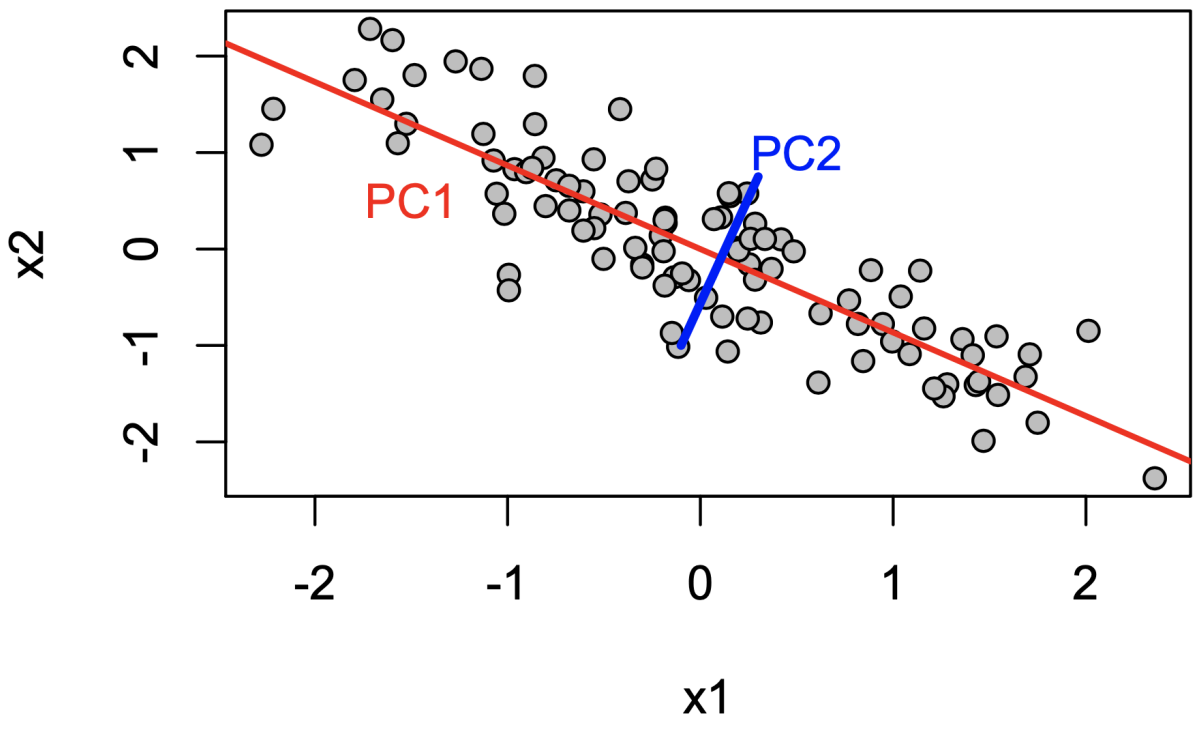
3. Los componentes subsecuentes son los que maximizan la varianza condicional además en que sean ortogonales al componente anterior.

La interpretación es genial. Los loadings del primer componente ϕ_1 define la dirección en la que todas las variables tienen mayor varianza.

Ejemplo con 2 variables



Ejemplo 2



Varianza explicada

Con el PCA, podemos calcular cuanta varianza estamos explicando con los k componentes?

La varianza total de la base es:

$$\Sigma_{j=1}^{p} Var(X_j) = \Sigma_{j=1}^{p} \frac{1}{N} \Sigma_{i=1}^{N} x_{ij}^2$$

La varianza explicada de cada componente es:

$$\frac{1}{n} \Sigma_{i=1}^{n} z_{ik}^2$$

PCA en R

La librería más común para correr PCA en R es `prcomp`.

```
pcfood<-prcomp(food,
               scale = TRUE,
               center = TRUE)
```

- `food` es la base de datos
- `scale` asegura que la varianza sea 1
- `center` asegura que la media sea cero.

PCA en R

prcomp {stats} R Documentation

Principal Components Analysis

Description

Performs a principal components analysis on the given data matrix and returns the results as an object of class `prcomp`.

Usage

```
prcomp(x, ...)

## S3 method for class 'formula'
prcomp(formula, data = NULL, subset, na.action, ...)

## Default S3 method:
prcomp(x, retx = TRUE, center = TRUE, scale. = FALSE,
       tol = NULL, rank. = NULL, ...)

## S3 method for class 'prcomp'
predict(object, newdata, ...)
```

Arguments

`formula`	a formula with no response variable, referring only to numeric variables.
`data`	an optional data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`subset`	an optional vector used to select rows (observations) of the data matrix `x`.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The 'factory-fresh' default is `na.omit`.
`...`	arguments passed to or from other methods. If `x` is a formula one might specify `scale.` or `tol`.
`x`	a numeric or complex matrix (or data frame) which provides the data for the principal components analysis.
`retx`	a logical value indicating whether the rotated variables should be returned.
`center`	a logical value indicating whether the variables should be shifted to be zero centered. Alternately, a vector of length equal the number of columns of `x` can be supplied. The value is passed to `scale`.
`scale.`	a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is `FALSE` for consistency with S, but in general scaling is advisable. Alternatively, a vector of length equal the number of columns of `x` can be supplied. The value is passed to `scale`.
`tol`	a value indicating the magnitude below which components should be omitted. (Components are omitted if their standard deviations are less than or equal to `tol` times the standard deviation of the first component.) With the default null setting, no components are omitted (unless `rank.` is specified less than `min(dim(x))`.). Other settings for tol could be `tol = 0` or `tol = sqrt(.Machine$double.eps)`, which would omit essentially constant components.
`rank.`	optionally, a number specifying the maximal rank, i.e., maximal number of principal components to be used. Can be set as alternative or in addition to `tol`, useful notably when the desired rank is considerably smaller than the dimensions of the matrix.
`object`	object of class inheriting from `"prcomp"`
`newdata`	An optional data frame or matrix in which to look for variables with which to predict. If omitted, the scores are used. If the original fit used a formula or a data frame or a matrix with column names, `newdata` must contain columns with the same names. Otherwise it must contain the same number of columns, to be used in the same order.

Details

The calculation is done by a singular value decomposition of the (centered and possibly scaled) data matrix, not by using `eigen` on the covariance matrix. This is generally the preferred method for numerical accuracy. The `print` method for these objects prints the results in a nice format and the `plot` method produces a scree plot.

PCA en `R`

`predict(pcfood)` te arroja una matriz de n, k. A partir de ahí puede uno extraer cada componente principal.

Interpretación inversa

- Hasta ahora, vimos que los loadings ϕ indican la dirección en la que la base varía más.
- Los scores z indican la magnitud (proyección) de esa dirección.
- Sin embargo, hay otra interpretación más general:

Los componentes principales proporcionan espacios lineales que **se acercan** mucho a las observaciones reales. Es decir, el primer componente es una línea que pasa los más cerca posible a todos los puntos de la base.

Por ende, provee un buen resumen de los datos.

Por ende, se puede expresar el modelo de factores así:

$$x_i j = \Sigma_{k=1}^{K} z_{ik} \phi_{jk}$$

En otras palabras, los K componentes principales pueden aproximar muy bien los datos reales.

Modelos de Factores

El modelo de factores estima la siguiente ecuación:

$$E[x_i k] = \phi_{j1} z_{i1} + ... + \phi_{jK} z_{iK}$$

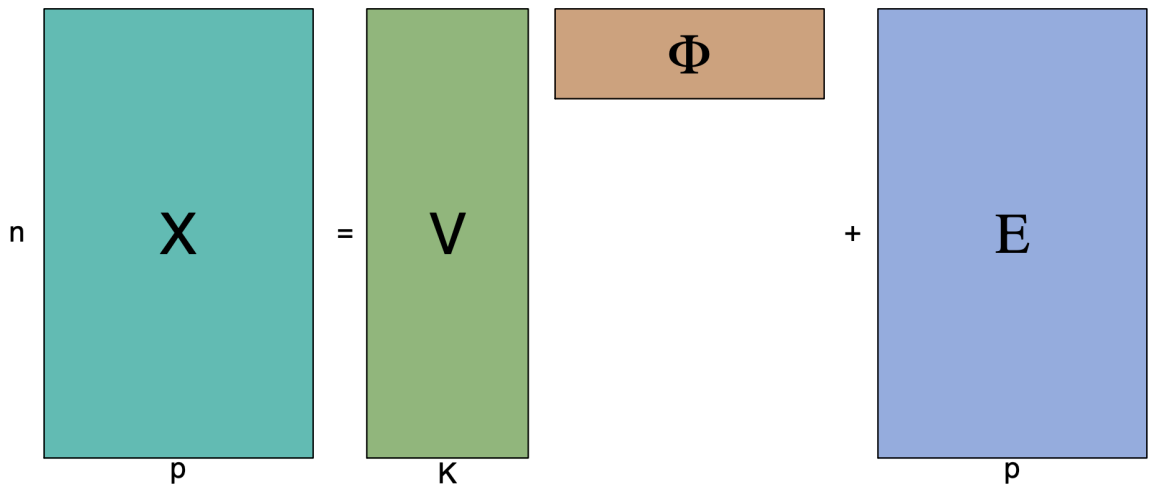
Donde $i \in \{1, ..n\}$, $j \in \{1, ..p\}$ X y ϕ son vectores de tamaño p (uno para cada variable de la base de datos). v_{iK} son scores que indican como afecta (carga) esa observación al factor k.

$Z = [z_{i1}, ..., z_{ik}]$ son los **factores** no observados de la base. Esto capturan mucho de la variazna de $X_{n,p}$.

ϕ_{jk} son los loadings de los factores $j \in \{1, ..., p\}$. Son simplemente los coeficientes de la regresion de cada x vs los factores z.

Modelos de Factores

$$\mathbb{E}[x_{ij}] = \varphi_{j1} v_{i1} + \ldots + \varphi_{jK} v_{iK}, \quad i = 1..n, \; j = 1..p$$



La base de datos (todas las columnas y observaciones) se corre contra K factores V, que tienen unos loadings ϕ que relacionan cada variable p vs cada factor k. E es ruido blanco.

Podemos tratar a los factores como no observados (PCA) o usar y para estimarlos (Partial Least Squares).

Principal Components Regression

Una vez calculados los factores con PCA, podemos usarlos para correr una regresión con una y en lugar de x. Esto es, $y \sim Z$.

Con esto, estaríamos reduciendo la dimensionalidad sin perder tanta info (LASSO like!).

- Los componentes principales tienen mucha varianza, eso es bueno para los estimadores de OLS.
- Los componentes son ortogonales. Esto ayuda que no haya multicolinealidad.
- Cuando es buena idea correr PCR? Elaboren

Principal Components Regression

Pros:

- Los componentes van a llevar las fuerzas dominantes de la base.

Cons:

- Si estas interesado en una variable que no es dominante (pero igualmente importante). Entonces PCR no te sirve.
- No hay garantía de que las fuerzas principales sean relevantes a y

LASSO vs PCR

Ambos son buenas herramientas. Típicamente la mejor depende del caso de uso.

Para LASSO

- Nos interesa la interpretación de las variables 'crudas'
- Asume que algunas variables no importan

Para PCR

- Asume que todas las variables importan, pero rankea su importancia

Hay alguna manera de hacer componentes de manera **supervisada**? si! Partial Least squares

Partial Least Squares

Partial Least Squares encuentra de manera supervisada los componentes que explican mucho de la X pero también de la Y.

Pasos:

1. Estandarizar las variables X
2. Corre la regresión de y vs x y guarda los coeficientes β
3. Fija los loadings del primer componentes con los coeficientes de la regresión de 1

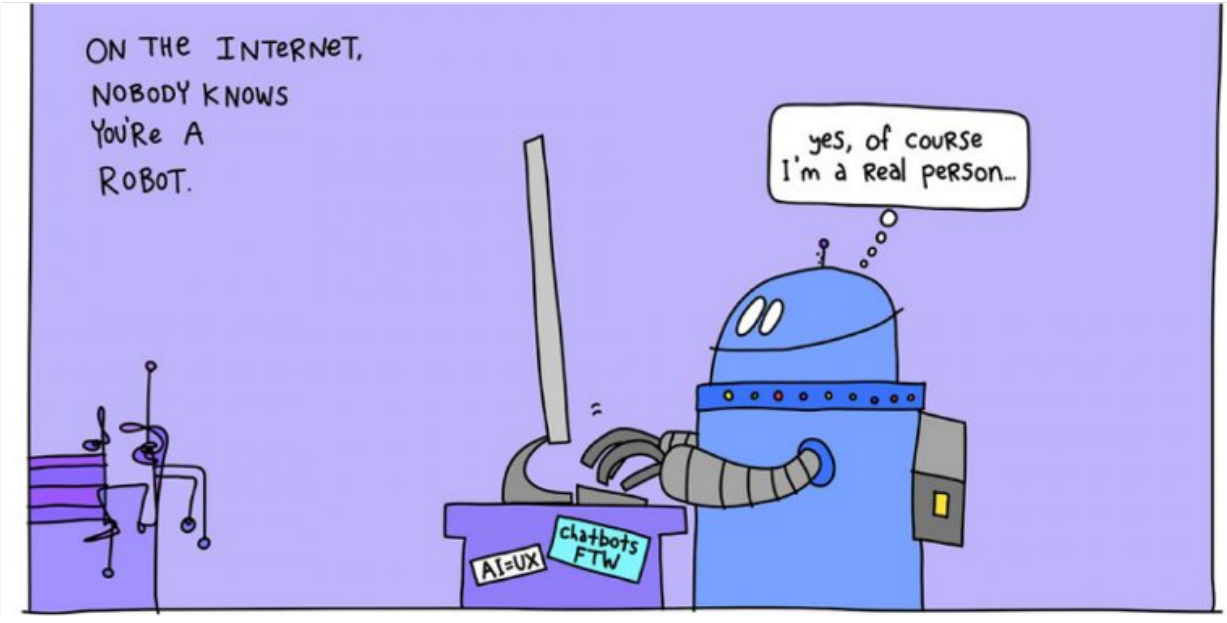
$$Z_1 = x'\beta$$

4. Captura los residuales del primer componente Z_1 vs X.
5. Repite 1 y 2 con los residuales

Natural Language Processing (NLP)

El NLP estudia el texto de lenguajes naturales (inglés, español, etc) en la computadora para aplicar técnicas estadísticas.

Con el mundo del internet, ahora guardamos conversaciones, reviews, descripciones, etc. Para finanzas, las noticias o redes sociales son un input con un poder predictivo altísimo.



Natural Language Processing

Lograr que la computadora lea y entienda el texto es una tarea muy compleja. El lenguaje humano es muy rico

- Muchas palabras tienen muchos significados. El contexto importa
- Esto hace muchas frases ambiguas.
- El Slang importa también.
- SIEMPRE ESTAS EN BIG DATA. Mientras más grande el texto, más palabras tendremos (más filas y columnas)

Esto hace que la mayoría de los algoritmos se basen en conteos de palabras o análisis de sentimiento. Estos son análisis muy sencillos. No obstante siguen siendo muy reveladores.

Pipeline de NLP clásico

1. Tokenization: Pasamos un string a una unidad de texto que no sea útil (n-gramas). Esto puede ser palabra, pares, tripletas, etc.
2. Stop words: Quitar palabras comunes o conectores que no agregar información relevante.
3. Stemming: identificar las raíces de las palabras para juntarlas

4 Word embeddings: Representar las palabras en forma de matrices númericas (Document Term Matrices)

Con estos pasos, podemos usar los algoritmos que hemos aprendido hasta ahora pero con el texto.

Análisis comunes en NLP

Sencillos:

- Word counts (Word clouds, gráficas de barras, probabilidades)
- Detección de SPAM

Medio:

- Análisis de Sentimiento
- Topic Modelling

Alto:

- Chat bots
- Machine translation

Conteos y word clouds

En general, les recomiendo ir a www.tidytextmining.com para referencias sobre tidy texting. Las librerías más importantes son:

- `tidytext`: Te permite crear bases de one row per token. Escogiendo el tamaño del grama.
- `tm`: Te permite construir Document Term Matrix con mucha facilidad
- `wordcloud`: Visualizaciones cool de word clouds

Ejemplo

Veamos un ejemplo donde analicé los discursos de Calderón. Primero cargo una base ya pre-limpia donde tenemos una fila por discurso con algunos meta datos (fecha, lugar, audiencia).

```
# Cambiando el formato a fila por palabra
# Notas: quita puntuaciones y pasa todo a minusculas en automati
por_palabra<-
  discursos %>%
  unnest_tokens(output = palabra, input = contenido)

# Quitamos las stopwords en espanol
por_palabra<-
  por_palabra %>%
  anti_join(stop_words_esp)
```

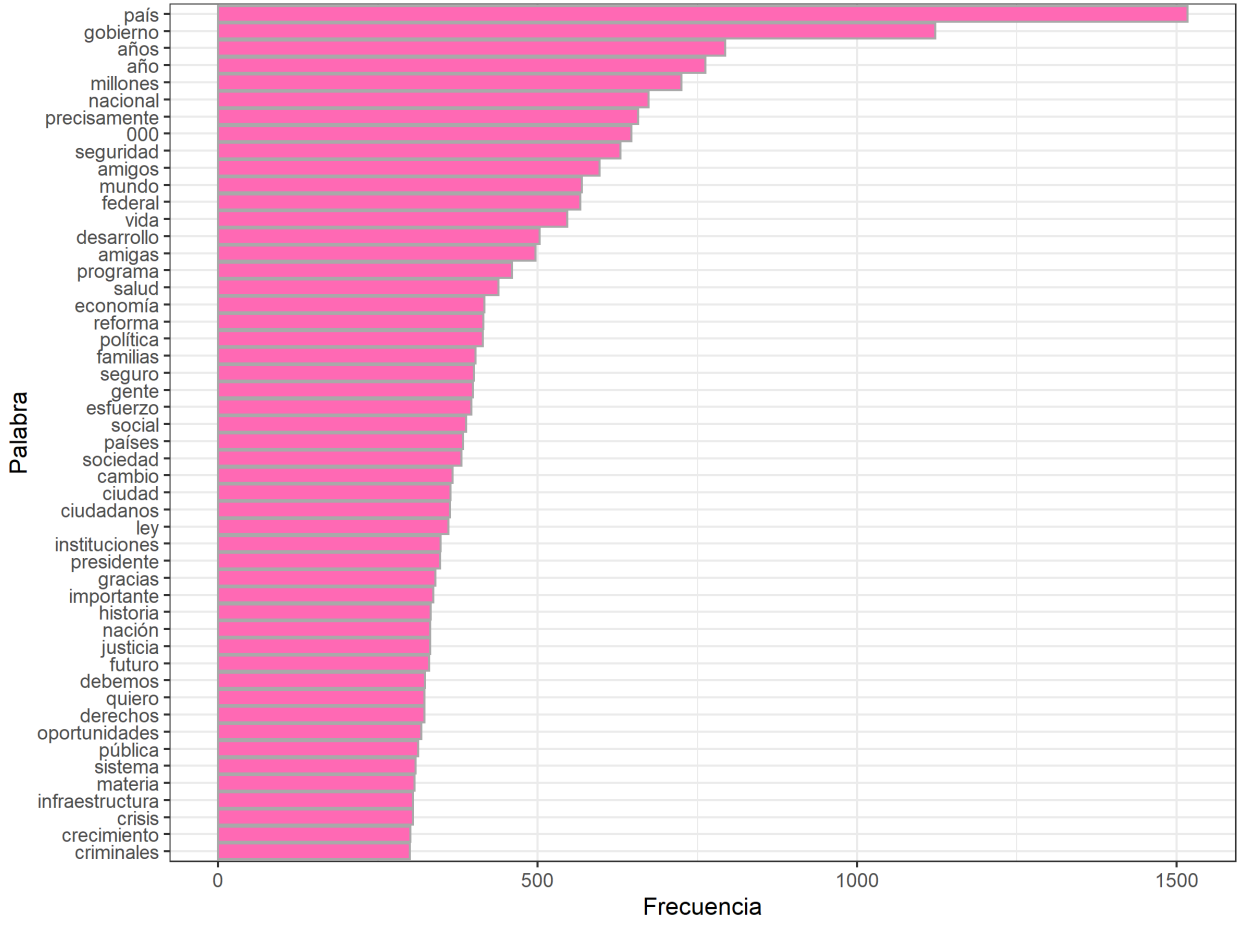
Ejemplo

Con esto ya puedes hacer wordclouds y graficas de barras

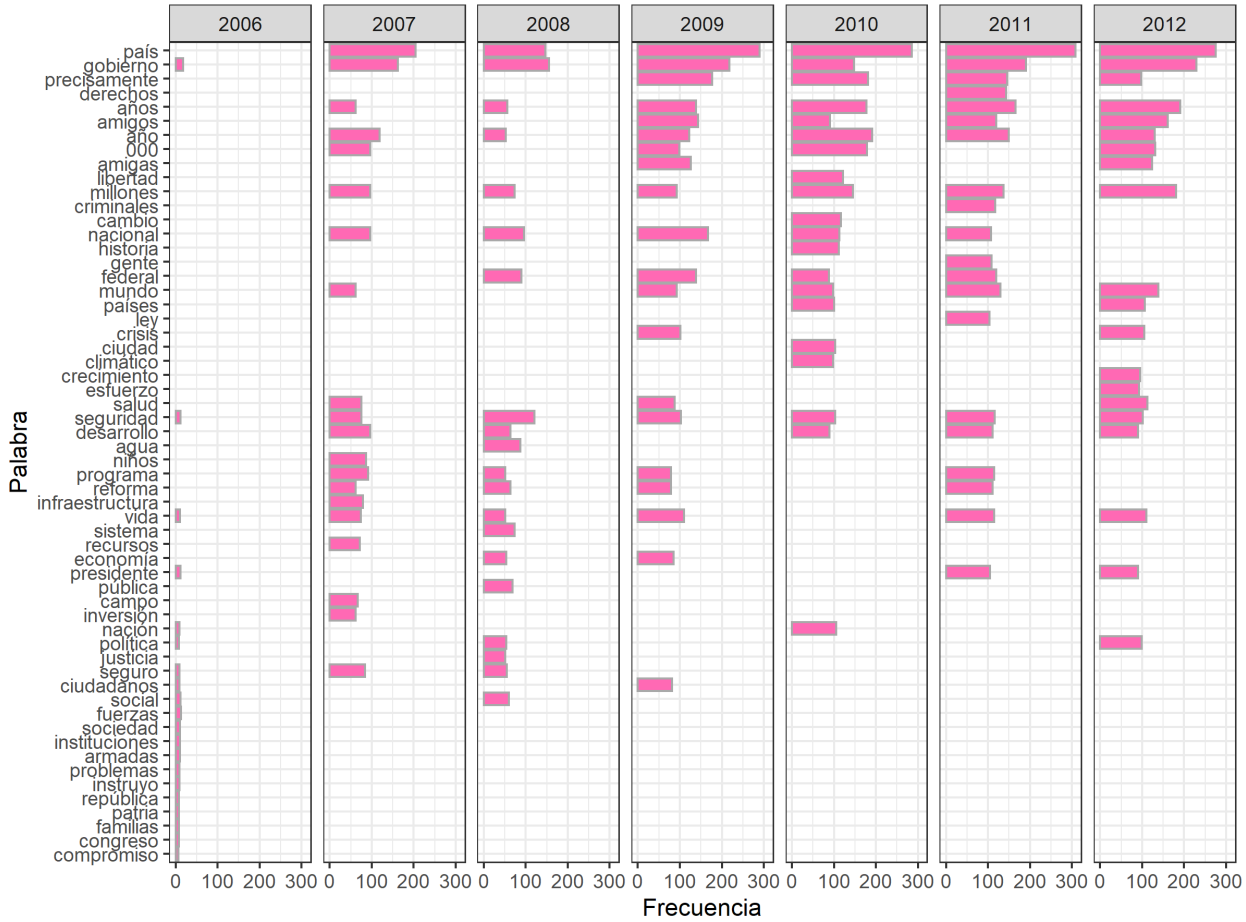
```
ggplot(conteos %>% filter(n>200), aes(reorder(palabra, n), n))+
  geom_col(fill = "hotpink", color = "darkgray")+
  coord_flip()+theme_bw()

wordcloud(conteos)
```

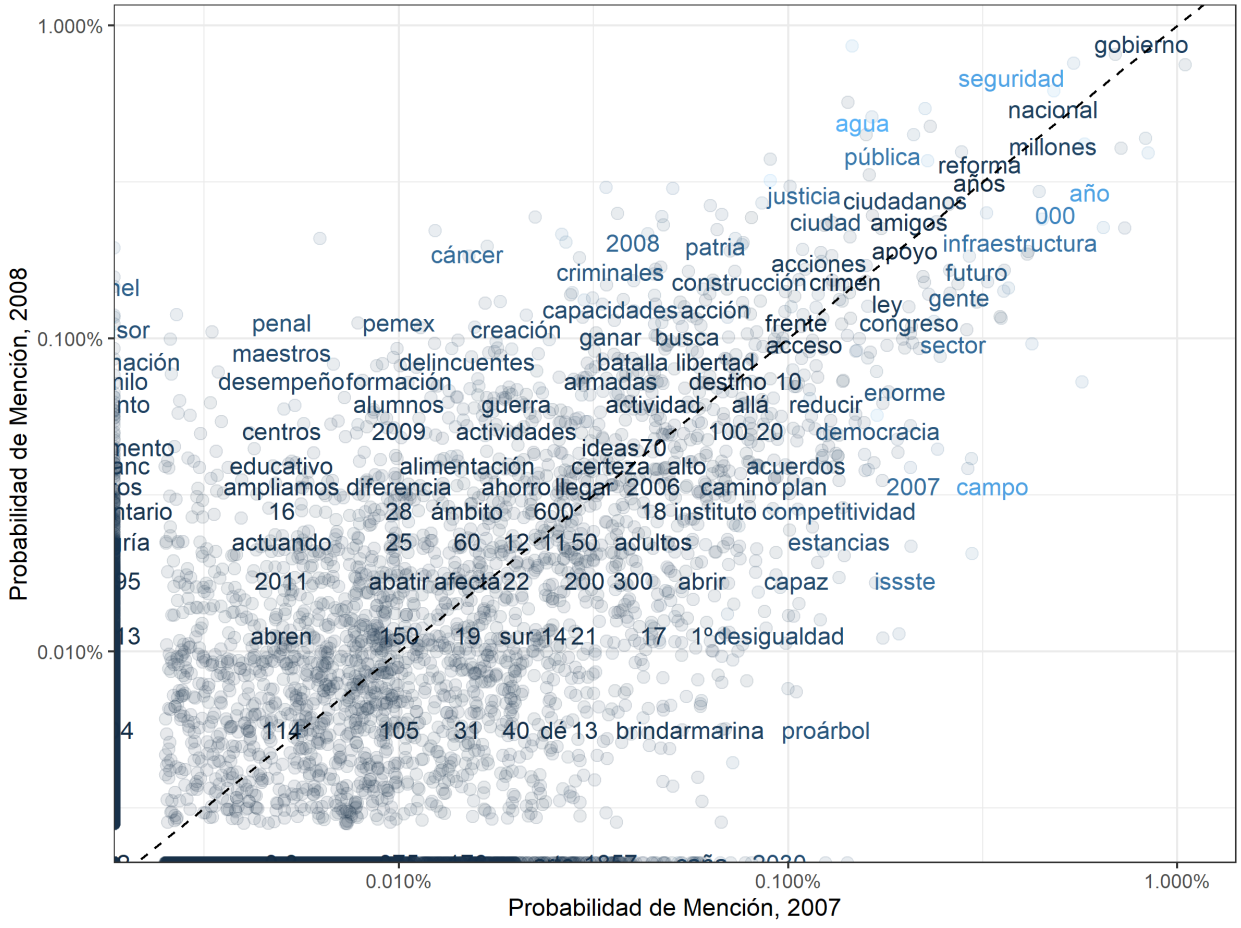
Ejemplo


país
gobierno
años
año
millones
nacional
precisamente
000
seguridad
amigos
mundo
federal
vida
desarrollo
amigas
programa
salud
economía
reforma
política
familias
seguro
gente
esfuerzo
social
países
sociedad
cambio
ciudad
ciudadanos
ley
instituciones
presidente
gracias
importante
historia
nación
justicia
futuro
debemos
quiero
derechos
oportunidades
pública
sistema
materia
infraestructura
crisis
crecimiento
criminales
Palabra
0
500
1000
1500
Frecuencia

Ejemplo


2006
2007
2008
2009
2010
2011
2012
Palabra
país
gobierno
precisamente
derechos
años
amigos
año
000
amigas
libertad
millones
criminales
cambio
nacional
historia
gente
federal
mundo
países
ley
crisis
ciudad
climático
crecimiento
esfuerzo
salud
seguridad
desarrollo
agua
niños
programa
reforma
infraestructura
vida
sistema
recursos
economía
presidente
pública
campo
inversión
nación
política
justicia
seguro
ciudadanos
social
fuerzas
sociedad
instituciones
armadas
problemas
instruyo
república
patria
familias
congreso
compromiso
0 100 200 300
Frecuencia

Ejemplo


1.000%
0.100%
0.010%
Probabilidad de Mención, 2008
0.010%
0.100%
1.000%
Probabilidad de Mención, 2007
gobierno
seguridad
nacional
agua
millones
pública
reforma
años
año
justicia
ciudadanos
000
ciudad
amigos
cáncer
2008
patria
apoyo
infraestructura
criminales
acciones
futuro
construcción
crimen
gente
capacidades
acción
ley
penal
pemex
creación
ganar
busca
frente
congreso
maestros
batalla
libertad
acceso
sector
delincuentes
desempeño
formación
armadas
destino
10
enorme
alumnos
guerra
actividad
allá
reducir
centros
2009
actividades
100
20
democracia
ideas
70
educativo
alimentación
certeza
alto
acuerdos
ampliamos
diferencia
ahorro
llegar
2006
camino
plan
2007
campo
16
28
ámbito
600
18
instituto
competitividad
actuando
25
60
12
11
50
adultos
estancias
2011
abatir
afectá
22
200
300
abrir
capaz
issste
abren
150
19
sur
14
21
17
1ºdesigualdad
114
105
31
40
dé
13
brindarmarina
proárbol

Análisis de Sentimiento

El análisis de sentimiento se basa en diccionarios. Esto es, a partir de una base tokenizada, puedo asignar un score de sentmiento a cada palabra?

Existen algunos diccionarios muy comunes:

- Afinn: Asigna scores de -5 a 5 por felicidad.
- Bing: Asigna emociones (enojo, felicidad, etc)
- Loughran: Asigna emociones complejas.

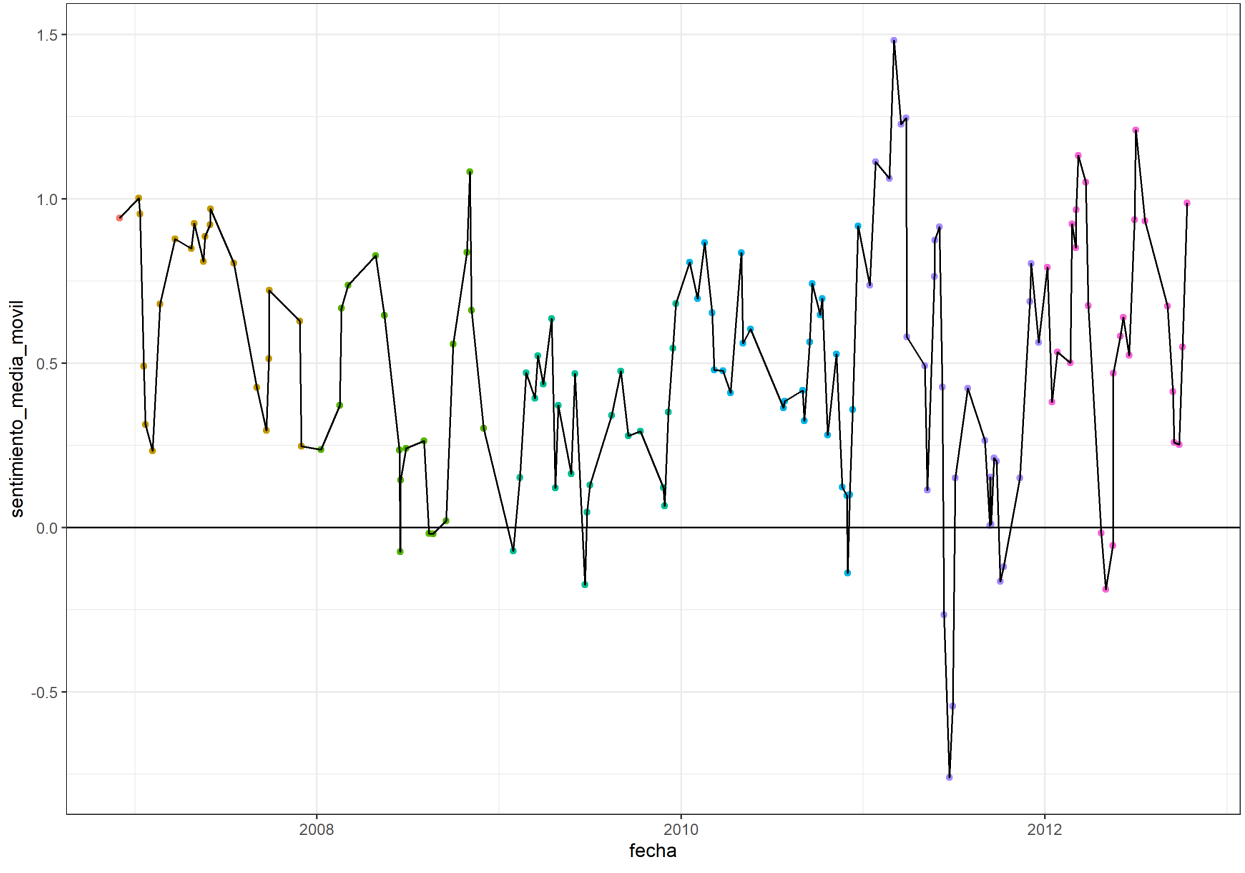
En general se van a enfrentar al reto de traducir todo esto si tienen textos que no sean en inglés. Hay maneras de darle la vuelva con la API de google translate.

Análisis de Sentimiento

```
##########################################
# Sentiment Analysis
##########################################
load("./Bases output/por_palabra.Rdata")
# Primero cargo las palabras del diccionario
diccionario<-list.files("./Bases input/", pattern = ".csv")
afin<-fread(str_c("./Bases input/", diccionario))
bing<-get_sentiments("bing")
loughran<-get_sentiments("loughran")
nrc<-get_sentiments("nrc")

sentimiento_afin<-
  inner_join(por_palabra, afin %>% select(Palabra, Puntuacion),
```

Ejemplo



Topic Modeling

Es un método muy parecido a PCA pero para texto. La diferencia es que, dado que son matrices (DTM) muy sparse 1) Te quedas sin memoria y 2) Los errores no pueden ser gausianos.

Lo que propuso Dirichlet fue modelar los conteos de los token como realizaciones de una distribución multinomial

- Para cada palabra en el texto, escoges un tópico k.
- Esta palabra tiene una probabilidad θ_{kj} para cada palabra j
- Escoges una palabra aleatoriamente de acuerdo a la probabilidad

Esto genera un modelo:

$$x_i \sim MN(w_{i1}\theta_1 + ... + w_{iK}\theta_K, m_i)$$

donde m_i es la longitud de la DTM. La probabilidad de cada palabra es $\sum_k w_{iK}\theta_K$.

Topic Modeling

Así, estamos modelando:

$$E[\frac{x_i}{m_i}] = w_{i1}\theta_1 + ... + w_{iK}\theta_K$$

θ: La probabilidad de cada palabra dentro del tópico. $\Sigma_{j=1}^{p}\theta_{kj} = 1$ Esto es, si sumamos todas las palbras y su probabilidad de pertenecer al tópico k, nos da 1.

w_i: La probablidad de cada tópico K.

Topic Modeling en `R`

La librería que yo conozco es `maptpx`. A partir de ahí la funcion es:

- `tpc<-topics(x,K=10, tol=10)`

Despues, usamos las palabras más comunes de cada tópico para interpretarlos.