Introducción

Análisis textual: Entender

Análisis textual: Estandarizar

Análisis textual: Describir

Análisis textual: Modelar

Referencias

Análisis Avanzado de Datos.

Nicolás López

1. Introducción
2. Análisis textual: Entender
3. Análisis textual: Estandarizar
4. Análisis textual: Describir
5. Análisis textual: Modelar
6. Referencias

Introducción

Introducción

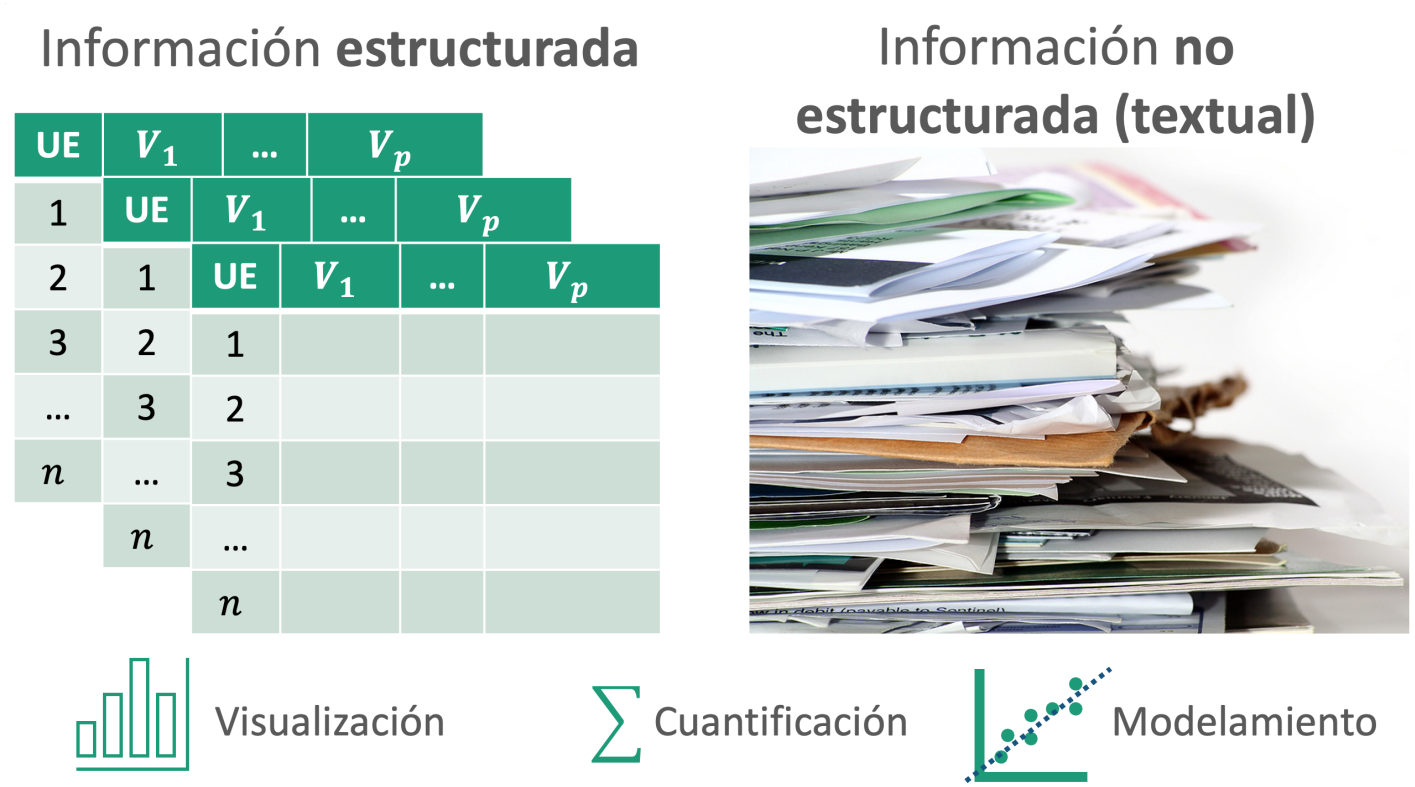


Figure 1: Información estructurada y no estructurada

En este caso, quisiéramos organizar esta información en una biblioteca (no supervisado). O quizá sería de interés usarla para recomendar nuevas lecturas de interés (supervisado).

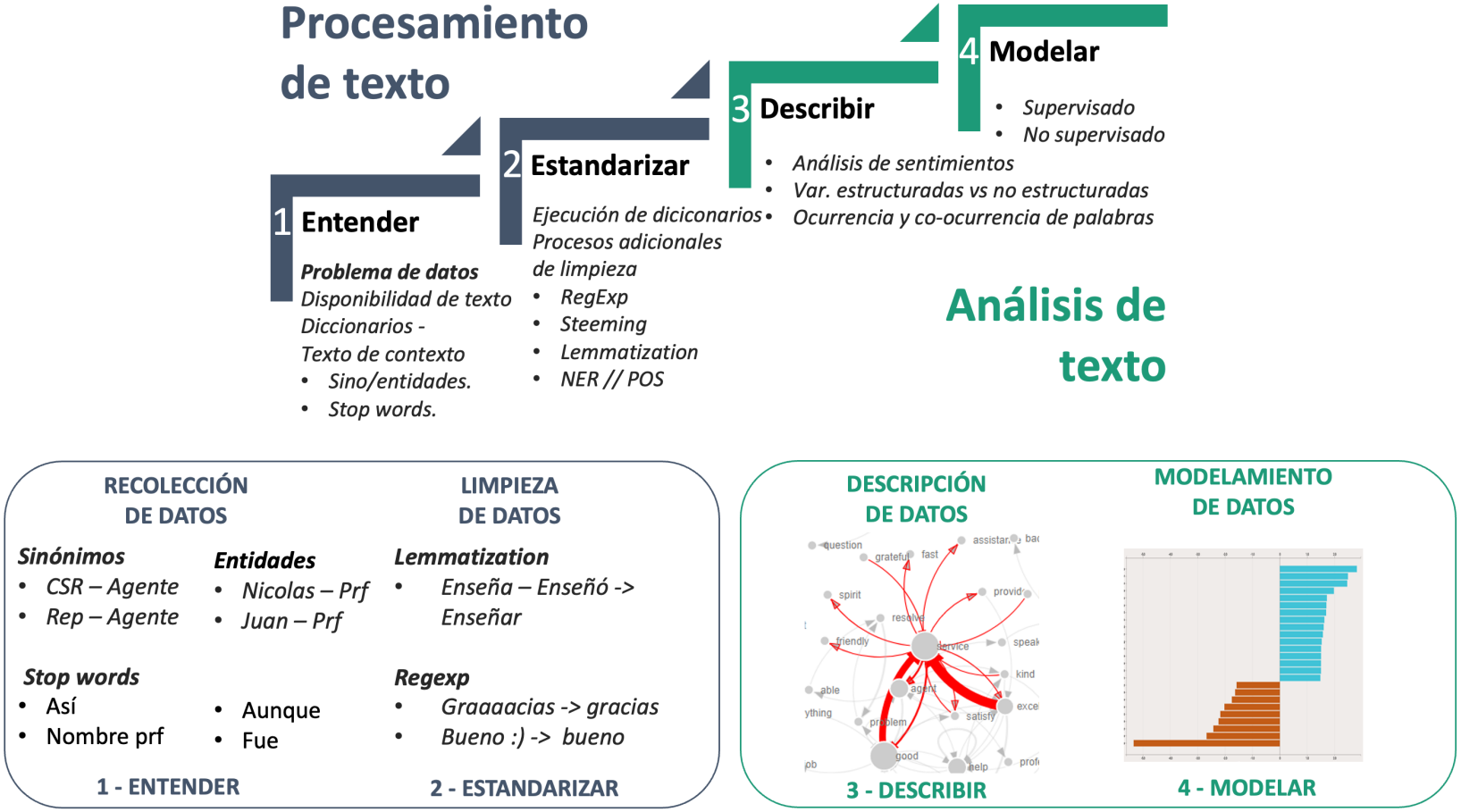
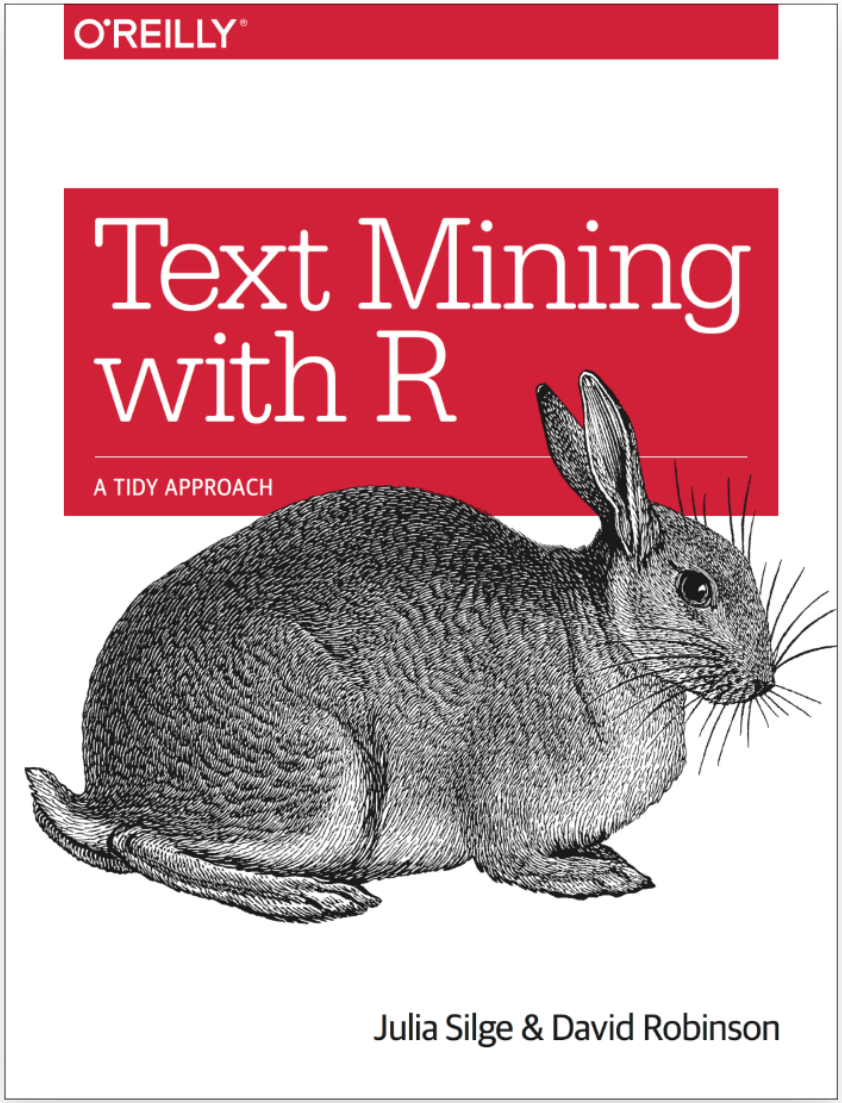


Figure 2: Pasos del análisis textual

Welcome to Text Mining with R

This is the website for *Text Mining with R*! Visit the GitHub repository for this site, find the book at O'Reilly, or buy it on Amazon.

This work by Julia Silge and David Robinson is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 United States License.



Preface »

Figure 3: Análisis textual desde R. disponible en https://www.tidytextmining.com/

Análisis textual: Entender

Análisis textual: Entender

- ¿De dónde se originan nuestros datos textuales?
- Criterios de inclusión y exclusión.
- ¿Representan estos una muestra de una colección o es una población de textos? ¿Es representativa?
  - Quizá tienden a responder únicamente clientes insatisfechos.
  - El texto considerado proviene de una muestra por conveniencia.
- ¿Qué información estructurada existe que pueda explicar el comportamiento visto marginalmente del texto?

El entendimiento permite acotar el problema e iniciar una depuración específica del contexto:

- Sinónimos:
  - Profe / profesor / teacher.
  - *Rep* / *Representative* / *Agent* / *CSR*.
- Entidades:
  - Universidad El Rosario.
  - Mercado libre.
- Stopwords
  - Profesor.
  - Reclamo.

Se recolectan expresiones y palabras **en contexto** determinantes en nuestro conjunto de datos.

1. Diccionarios de Sinónimos: Conocimiento de sinónimos para una palabra relevante en el contexto de interés: puede requerir un primer acercamiento con la persona conocedora del contexto en el cual se generan los datos:

- *Rep* / *Representative* / *Agent* / *CSR*.

2. Diccionarios de Entidades: Para el caso de la retroalimentación a los profesores, seguramente cada UE puede ser cruzada con el nombre del profesor en una tabla estructurada. Este *match* permite identificar los nombres de los profesores con facilidad.
3. Diccionarios de Stopwords: En este mismo caso, los nombres de los profesores probablemente se presenten com una **stopword** a ser retirada. También existen listas genéricas de stopwords pre-establecidas (usar con cuidado).

Análisis textual: Estandarizar

Análisis textual: Estandarizar

Los diccionarios son aplicados sobre versiones tokenizadas de los documentos:

```
library(tidytext)
library(tidyverse)
data0 = c('La Proferosa maría es buena',
          'Miprofesor pedro es regular',
          'elteacher camilo me encanta :)',
          'el prf. Ignacio es 1a')

datal = tibble(fback=data0) %>%
        mutate(ID=row_number()) %>%
        unnest_tokens(word,fback)
head(datal,7)
```

```
## # A tibble: 7 x 2
##      ID word
##   <int> <chr>
## 1     1 la
## 2     1 proferosa
## 3     1 maría
## 4     1 es
## 5     1 buena
## 6     2 miprofesor
## 7     2 pedro
```

R lo maneja mediante tibbles, desde python serán *comprehension lists* optimizados para este análisis. Note que reduce el texto a minúsculas y retira

A través de un left join se hace el match con uno de los diccionarios:

```
dict_syn = tibble(word=c('profe','teacher','prf'),
                  Palabra1=c('profesor','profesor','profesor'))
datal %>% left_join(dict_syn,by='word')
```

```
## # A tibble: 18 x 3
##       ID word       Palabra1
##    <int> <chr>      <chr>
##  1     1 la         <NA>
##  2     1 proferosa  <NA>
##  3     1 maría      <NA>
##  4     1 es         <NA>
##  5     1 buena      <NA>
##  6     2 miprofesor <NA>
##  7     2 pedro      <NA>
##  8     2 es         <NA>
##  9     2 regular    <NA>
## 10     3 elteacher  <NA>
## 11     3 camilo     <NA>
## 12     3 me         <NA>
## 13     3 encanta    <NA>
## 14     4 el         <NA>
## 15     4 prf        profesor
## 16     4 ignacio    <NA>
## 17     4 es         <NA>
## 18     4 1a         <NA>
```

```
datal    = datal %>%
           left_join(dict_syn,by='word') %>%
           mutate(word0 = paste0(ifelse(is.na(Palabra1),
                                 word,
                                 Palabra1),
                         sep=" ")) %>%
           group_by(ID) %>%
           summarize(fback0 = paste(word0,collapse=""))
datal
```

```
## # A tibble: 4 x 2
##      ID fback0
##   <int> <chr>
## 1     1 "la proferosa maría es buena "
## 2     2 "miprofesor pedro es regular "
## 3     3 "elteacher camilo me encanta "
## 4     4 "el profesor ignacio es 1a "
```

El procedimiento funcionó, pero es normal que hayan errores tipográficos que dificulten el proceso. En este paso probablemente sea necesario hacer uso de **expresiones regulares**:

```
data1 = sub('[ ]?[Pp]ro?fe?s?o?r?a?\\.?|teacher',
            ' profesor ',
            data0)
tibble(fback0 = data1)
```

```
## # A tibble: 4 x 1
##   fback0
##   <chr>
## 1 La profesor osa maría es buena
## 2 Mi profesor  pedro es regular
## 3 el profesor  camilo me encanta :)
## 4 el profesor  Ignacio es 1a
```

Note que en este caso, el diccionario es 'colapsado' a una sola expresión, y además, la regexp generó un error en la estandarización.

- El proceso de **estandarización** a través de diccionarios continúa similarmente para las EN y stopwords.
- También pueden encontrarse EN dadas por el texto (y no por conocimiento experto). Para esto existen modelos pre entrenados, o también se puede hacer uso de *transfer learning* con datos de contexto pre etiquetados.
- Se continúa con la **lematización y/o steeming**. Este también puede basarse en diccionarios o en modelos de lenguaje.
- Algunos procesos se benefician de POS (asignación de palabras de un texto a su categoría gramatical) para una depuración informativa del texto a analizar.

Análisis textual: Describir

Datos de ejemplo - Libro Metamorfósis (inglés)

Descarga de datos del proyecto gutenbergr

```
library(gutenbergr)
meta_kafka <- gutenberg_download(5200)
dim(meta_kafka)
```

```
## [1] 1872    2
```

Encabezado del libro

```
head(meta_kafka)
```

```
## # A tibble: 6 x 2
##   gutenberg_id text
##          <int> <chr>
## 1         5200 "Metamorphosis"
## 2         5200 ""
## 3         5200 "by Franz Kafka"
## 4         5200 ""
## 5         5200 "Translated by David Wyllie"
## 6         5200 ""
```

Tokenización

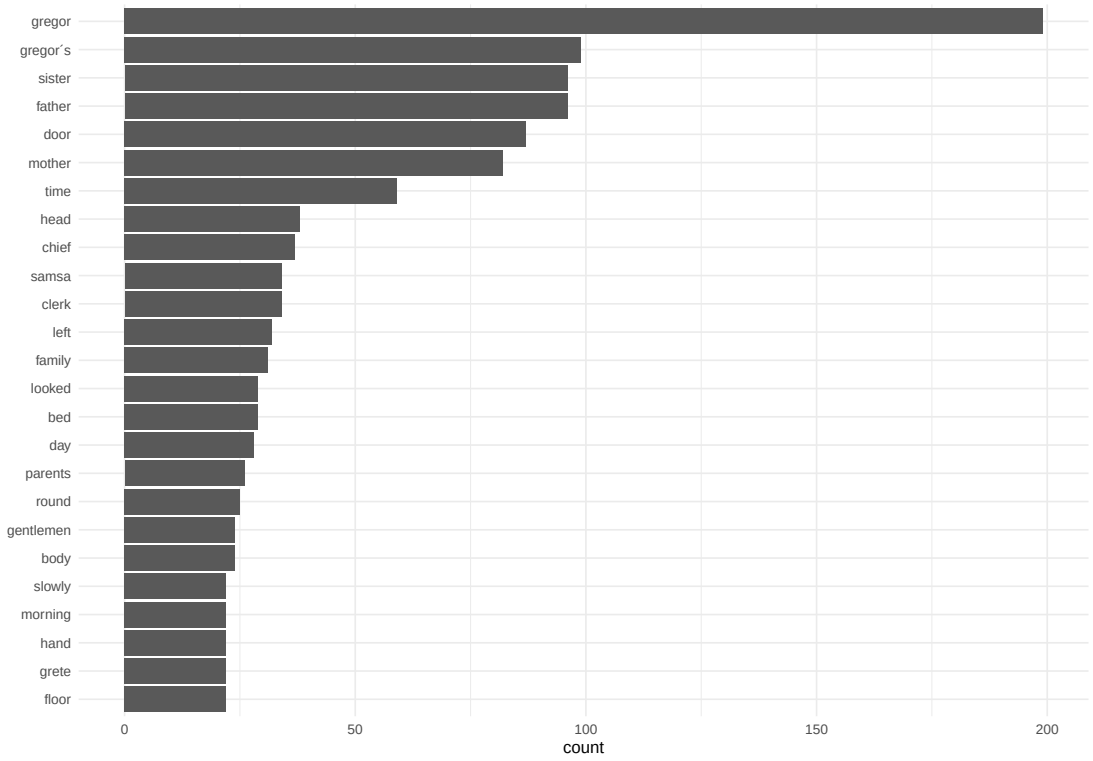
Tokenización a nivel palabra para limpieza rápida

```
tidy_kafka <- meta_kafka %>%
              filter(text!='') %>%
              filter(row_number()>=5) %>%
              mutate(n_line=row_number()) %>%
              unnest_tokens(word, text) %>%
              anti_join(stop_words)
```

Visualización de unigramas

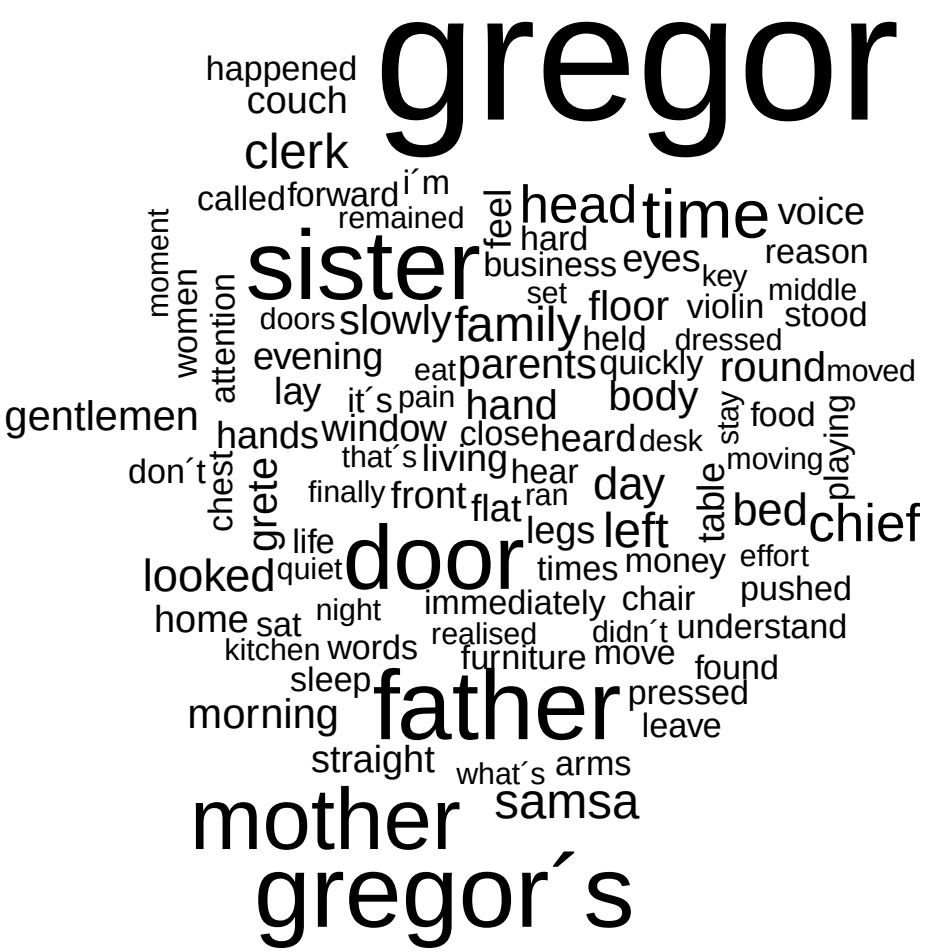
Visualización de palabras más frecuentes (unigramas)

```
tidy_kafka %>% group_by(word) %>%
               summarize(count=n()) %>%
               filter(count > 20) %>%
               mutate(word = reorder(word, count)) %>%
               ggplot(aes(count, word)) +
               geom_col() +
               theme_minimal() +
               labs(y = NULL)
```



Visualización de palabras más frecuentes (unigramas)

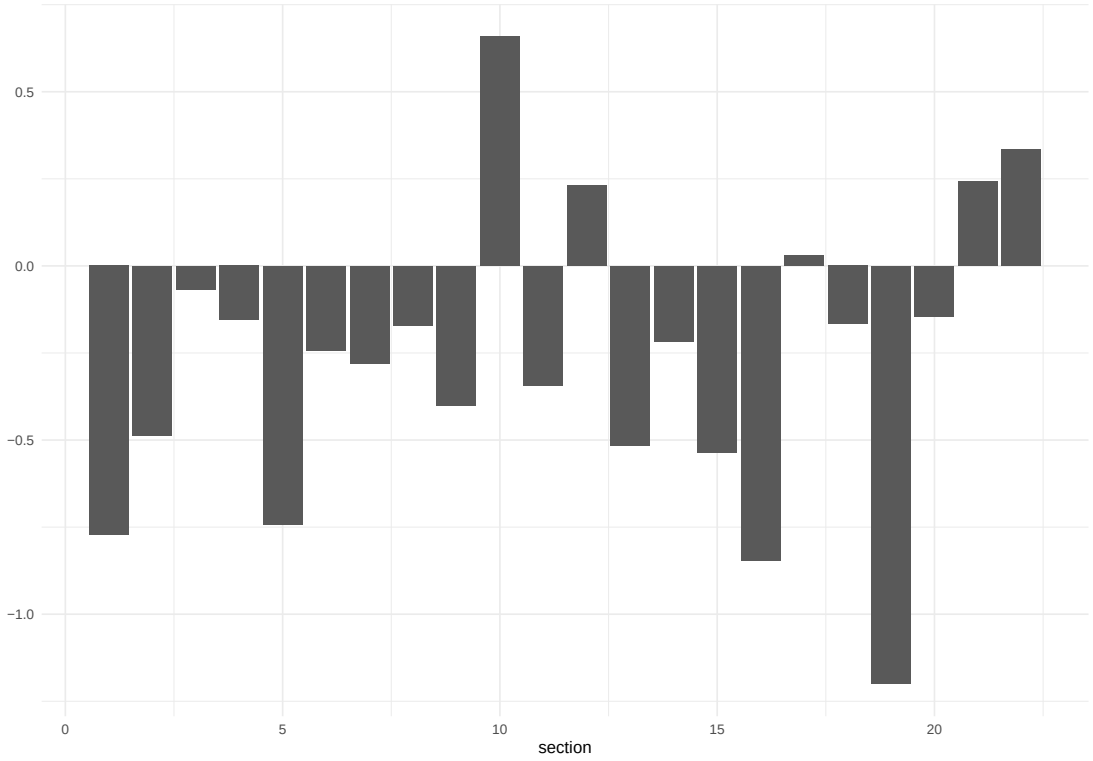
```
library(wordcloud)
tidy_kafka %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100))
```



Análisis unigrama de sentimientos

Sentimientos marginales (no condicionales ni conjuntos) a través de diccionario

```
library(textdata)
affin_sent_dict = get_sentiments("afinn")
kafka_sent = tidy_kafka %>% left_join(affin_sent_dict) %>%
                mutate(section = 1 + round(n_line/85)) %>%
                group_by(section) %>%
                summarize(avg_sent = mean(value,na.rm=TRUE))
ggplot(kafka_sent,aes(x=section,y=avg_sent))+
  geom_col() + theme_minimal() + labs(y = NULL)
```



Palabras más frecuentes por emoción

```
tidy_kafka %>% left_join(affin_sent_dict) %>%
  mutate(sign=ifelse(value > 0,'Positive',
                     ifelse((value) < 0 , 'Negative',
                            'Neutral'))) %>%
  filter(sign%in%c('Positive','Negative')) %>%
  group_by(sign,word) %>%
  summarize(count=n()) %>% filter(count>8)
```

```
## # A tibble: 8 x 3
## # Groups:   sign [2]
##   sign     word      count
##   <chr>    <chr>     <int>
## 1 Negative hard         12
## 2 Negative leave        12
## 3 Negative pain         11
## 4 Negative stopped       9
## 5 Negative tears         9
## 6 Positive carefully     9
## 7 Positive cleaner       9
## 8 Positive straight     18
```

Note la posible ambigüedad en los significados **en contexto**.

Bigramas: texto en contexto

Una manera simple de añadir la realización conjunta de palabras es mediante **bigramas**

```
kafka_big = tidy_kafka %>% group_by(n_line) %>%
            summarize(frase_limpia = paste0(word,collapse=' ')) %>%
            unnest_tokens(bigram, frase_limpia,
                          token = "ngrams", n = 2) %>%
            filter(!is.na(bigram))

head(kafka_big,10)
```

```
## # A tibble: 10 x 2
##    n_line bigram
##     <int> <chr>
##  1      1 morning gregor
##  2      1 gregor samsa
##  3      1 samsa woke
##  4      1 woke troubled
##  5      1 troubled dreams
##  6      1 dreams found
##  7      2 transformed bed
##  8      2 bed horrible
##  9      2 horrible vermin
## 10      2 vermin lay
```

Estos bigramas son las parejas ordenadas de palabras presentes en el texto.

Esta ordenación permite ver conjuntamente la relación entre las palabras, además, permite analizar los sentimientos de las palabras en el contexto de la palabra inmediatamente anterior

```
kafka_topbi = kafka_big %>%
              count(bigram, sort = TRUE)

head(kafka_topbi,15)
```

```
## # A tibble: 15 x 2
##    bigram              n
##    <chr>           <int>
##  1 chief clerk        29
##  2 gregor's father    24
##  3 gregor's mother    17
##  4 gregor's sister    14
##  5 chest drawers       9
##  6 mother sister       9
##  7 father mother       6
##  8 entrance hall       4
##  9 gregor door         4
## 10 gregor father       4
## 11 key lock            4
## 12 mother gregor       4
## 13 time gregor         4
## 14 violin playing      4
## 15 clerk gregor        3
```

No parece frecuente un cambio de polaridad en los sentimientos de manera bivariada.

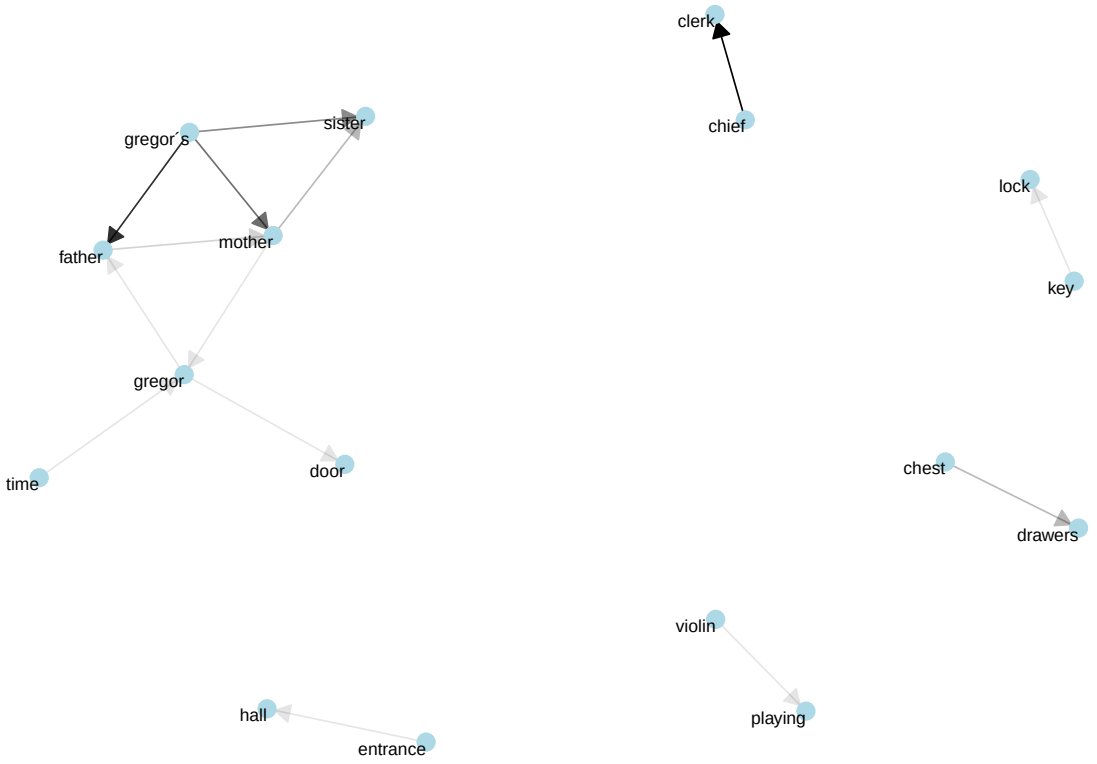
Bigramas como grafo

Es interesante ver a los bigramas como un grafo direccionado ponderado $G = (V, E)$.

```
library(igraph)
bigrams_separ <- kafka_big %>%
                 separate(bigram, c("word1", "word2"), sep = " ")
bigram_counts <- bigrams_separ %>%
                 count(word1, word2, sort = TRUE)
bigram_graph  <- bigram_counts %>%
                 filter(n >= 4) %>%
                 graph_from_data_frame()
```

```
library(ggraph)
a <- grid::arrow(type = "closed", length = unit(.15, "inches"))

ggraph(bigram_graph, layout = "fr") +
  geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                 arrow = a, end_cap = circle(.07, 'inches')) +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
  theme_void()
```



Esto nos permite hablar de características del grafo en el contexto textual:

- Conexión del grafo.
- Grado del grafo.
- Distancia máxima.
- Clustering de palabras.
- Características de las palabras como nodos (por ejemplo centralidad).

Análisis textual: Modelar

Análisis textual: Modelar

Una de las familias de modelos para el entendimiento de un corpus son los **modelos de lenguaje**

Figure 4: Información no estructurada y modelos de lenguaje

Buscan descubrir los temas que trata el texto, anotarlos (etiquetarlos de acuerdo a la estructura) y usarlo para nuestras necesidades (predicción, estructuración de datos,. . .).

LDA

La idea es simple:

- Los documentos están compuestos por múltiples temas: algunas temas se presentan más frecuentemente que otros para el texto dado.
- Los temas son una colección de palabras asociadas: algunas más prevalentes que otras para el tema dado.

Podemos imaginar el proceso generativo del texto a través de un modelo estadístico:

- Los documentos están compuestos por una distribución sobre un número determinado de temas.
- Los temas están compuestos por una distribución sobre un vocabulario.

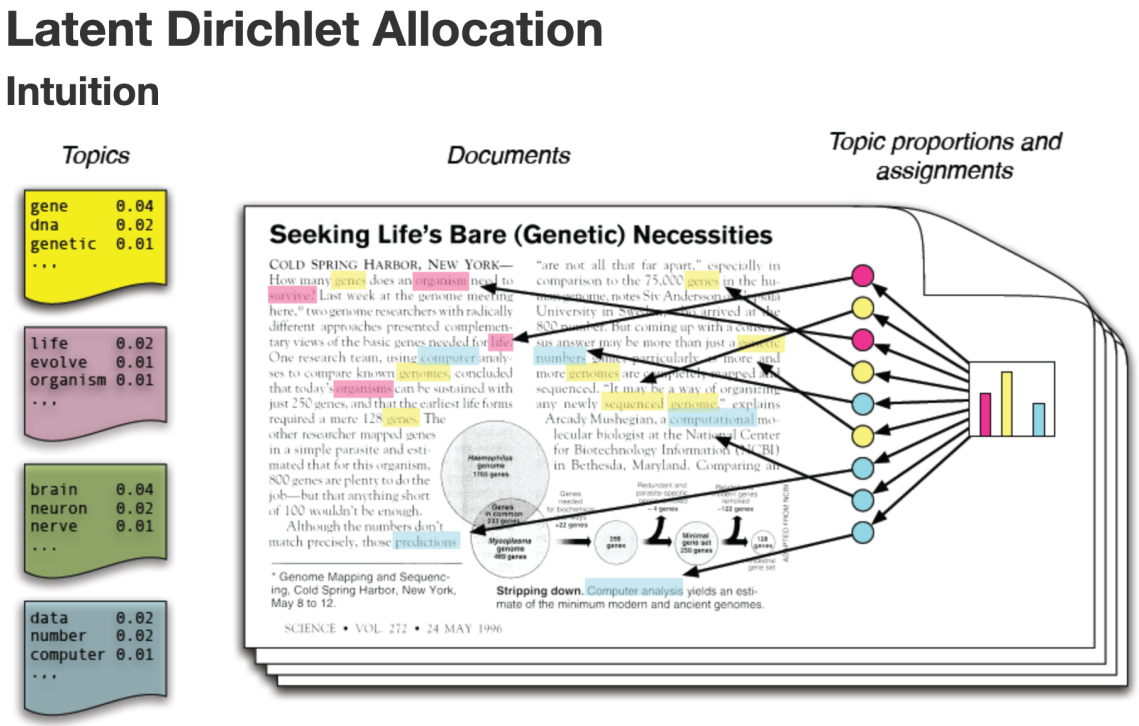


Figure 5: Información no estructurada y modelos de lenguaje (Blei)

El problema estadístico es encontrar dicha estructura:

- El histograma por documento.
- Los temas de cada palabra.
- La composición de los temas.

Note que:

- Los temas se mantienen a través de los documentos.
- La cantidad que exhibe cada documento de los temas es lo que varía (el histograma del gráfico anterior).
- El orden de las palabras no importa: EL PROCESO NO GENERARÍA TEXTO COHERENTE.

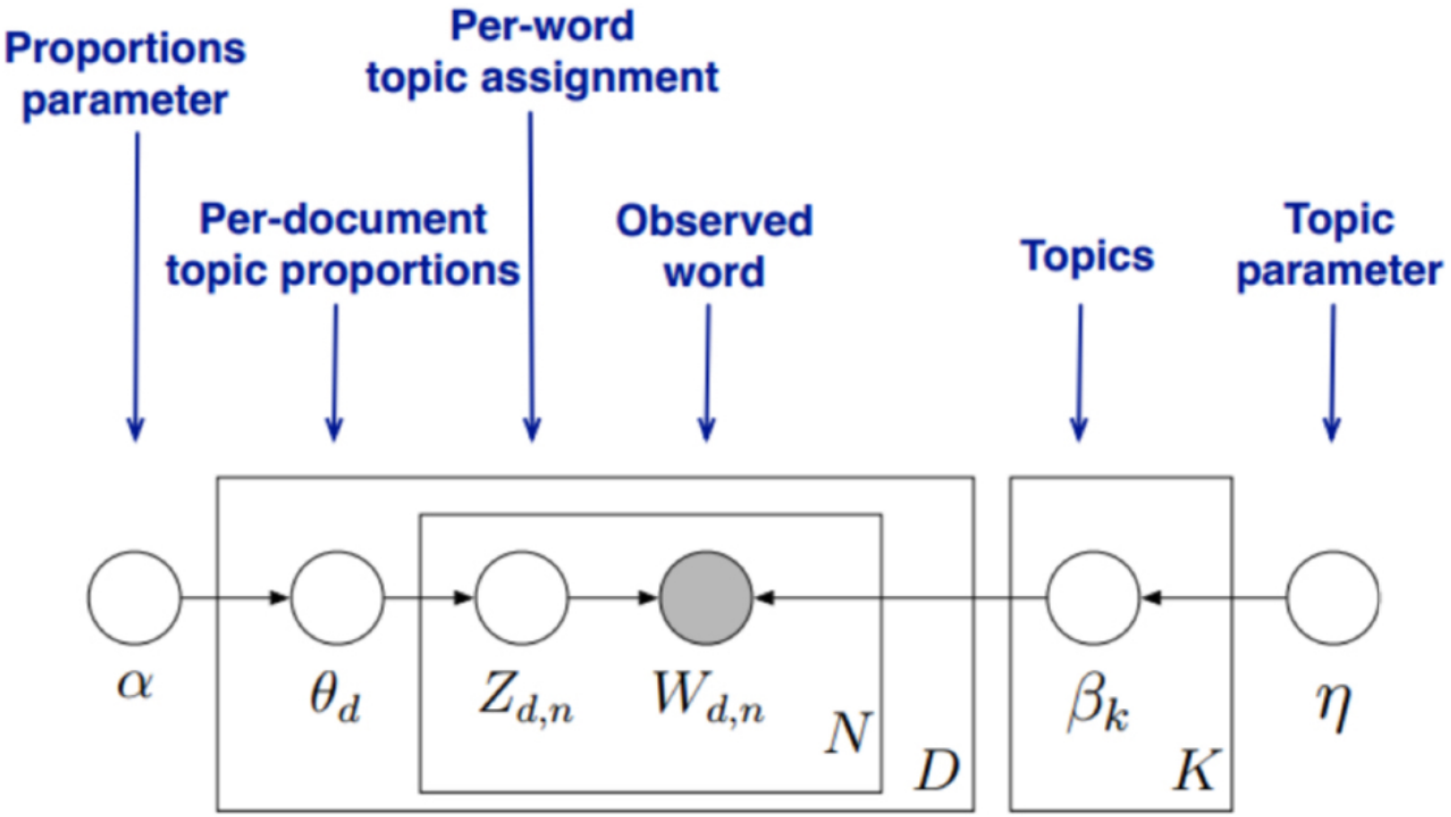


Figure 6: LDA como modelo gráfico

- Los nodos son variables aleatorias y los *edges* indican dependencia.
- Se asumen que hay bastantes variables escondidas en el corpus - **EM algorithm**.

Volviendo al libro objeto de nuestro análisis, los bigramas frecuentes permiten encontrar las palabras que en contexto tienen significado relevante:

```
kafka_clean = tidy_kafka %>%
              mutate(section = 1 + round(n_line/85),next_word = lead(word,1)) %>%
              left_join(kafka_topbi %>% filter(n >= 4) %>% separate(bigram, c("word", "next_word"), sep = " ")) %>%
              mutate(lag_n = lag(n,1)) %>%
              mutate(text0 = ifelse(is.na(n),
                                    word,
                                    paste(word,next_word,sep="_"))) %>%
              filter(is.na(lag_n))

kafka_clean %>% filter(!is.na(n)) %>% head(10) %>%
            select(word,next_word,text0)
```

```
## # A tibble: 10 x 3
##    word     next_word text0
##    <chr>    <chr>     <chr>
##  1 chest    drawers   chest_drawers
##  2 gregor   door      gregor_door
##  3 gregor   door      gregor_door
##  4 gregor   door      gregor_door
##  5 chief    clerk     chief_clerk
##  6 chief    clerk     chief_clerk
##  7 chief    clerk     chief_clerk
##  8 chief    clerk     chief_clerk
##  9 chief    clerk     chief_clerk
## 10 gregor's sister    gregor's_sister
```

Múltiples alternativas pueden usarse en este escenario para ir más allá de un modelo **BOW**.

Se busca caracterizar las 22 secciones del libro para detallar el sentimiento previamente encontrado en cada uno de ellos:

```
library(topicmodels)
kafka_clean_lda = kafka_clean %>% group_by(section) %>%
                  summarize(text_section = paste0(text0,collapse = ' '))

word_counts     = kafka_clean_lda %>% unnest_tokens(word,text_section) %>%
                  count(section,word, sort = TRUE) %>% cast_dtm(section, word, n)

word_counts
```

```
## <<DocumentTermMatrix (documents: 22, terms: 2193)>>
## Non-/sparse entries: 4691/43555
## Sparsity           : 90%
## Maximal term length: 17
## Weighting          : term frequency (tf)
```

Un modelo LDA es entrenado con k temas:

```
kafka_lda        = LDA(word_counts,
                       k = 50,
                       control = list(seed = 1234))

section_topics   = tidy(kafka_lda, matrix = "beta")

top_terms        = section_topics %>%
                   group_by(topic) %>%
                   slice_max(beta, n = 10) %>%
                   ungroup() %>%
                   arrange(topic, -beta)
```

Con este podemos determinar el 'histograma' de la representación intuitiva del LDA para cada sección:

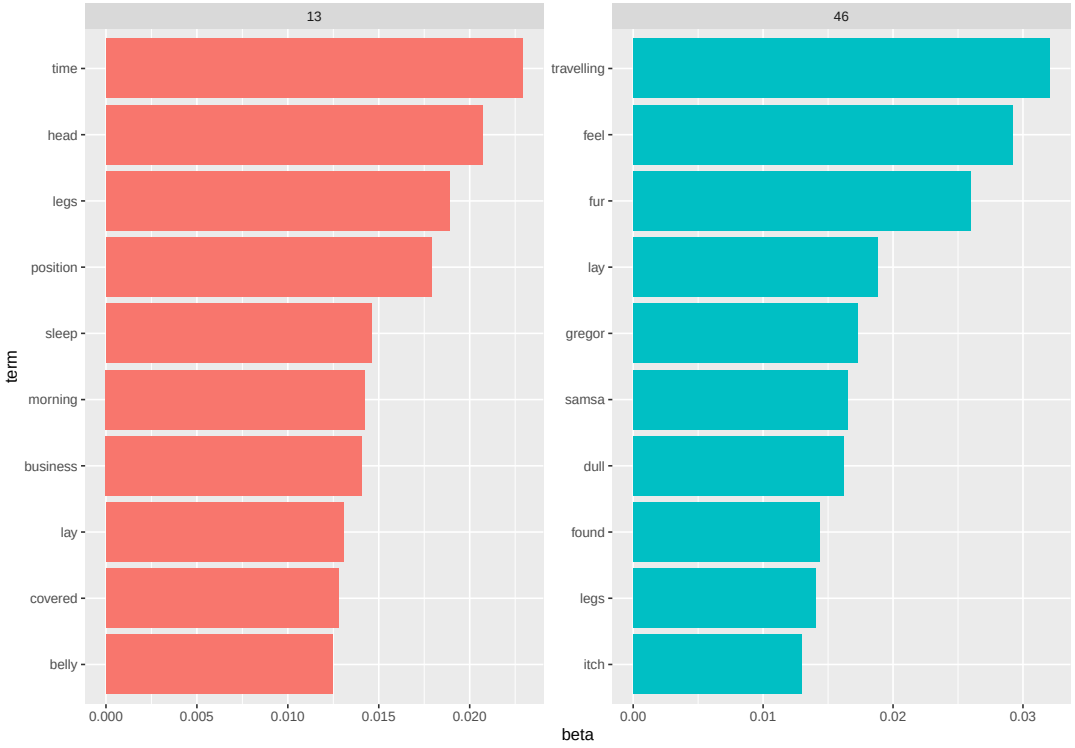
```
sections_gamma = tidy(kafka_lda, matrix = "gamma")

sections_gamma_example = sections_gamma %>%
                         filter(document==1) %>%
                         arrange(desc(gamma)) %>%
                         head(2)
sections_gamma_example
```

```
## # A tibble: 2 x 3
##   document topic gamma
##   <chr>    <int> <dbl>
## 1 1           46 0.550
## 2 1           13 0.448
```

Y a su vez podremos encontrar la distribución de los k temas mas relevantes para esta sección sobre las palabras:

```
top_terms %>% filter(topic%in%sections_gamma_example$topic) %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered()
```



Viendo el texto original, nos damos cuenta que recuperamos parte de la estructura del mismo

morning gregor samsa woke troubled dreams found transformed bed horrible vermin lay armour lifted head brown belly slightly domed divided arches stiff sections bedding cover ready slide moment legs pitifully thin compared size rest waved helplessly looked what's happened wasn't dream proper human lay peacefully familiar walls collection textile samples lay spread table samsa travelling salesman hung picture recently cut illustrated magazine housed nice gilded frame lady fitted fur hat fur boa sat upright raising heavy fur muff covered lower arm viewer gregor window dull weather drops rain heard hitting pane feel sad sleep bit forget nonsense unable sleeping couldn't position hard threw rolled hundred times shut eyes wouldn't floundering legs stopped feel mild dull pain god strenuous career i've chosen travelling day day business takes effort business home top there's curse travelling worries train connections bad irregular food contact people time friendly hell slight itch belly pushed slowly headboard lift head found itch covered lots white spots didn't feel legs drew quickly touched overcome cold shudder slid position time makes stupid you've sleep travelling salesmen live life luxury instance guest house morning copy contract

Referencias

Referencias

1. Hastie, T., Tibshirani, R., Friedman, J. The Elements of Statistical Learning. Springer.
2. Garet, Witten, Hastie, Tibshirani. Introduction to Statistical Learning with R.