Statistical Language Models

Week 6

Feature Engineering

- Devising your own covariates
- Vocabulary (presence/absence of word)
- Frequency (count of words)
- Occurrence (distribution of words in text)

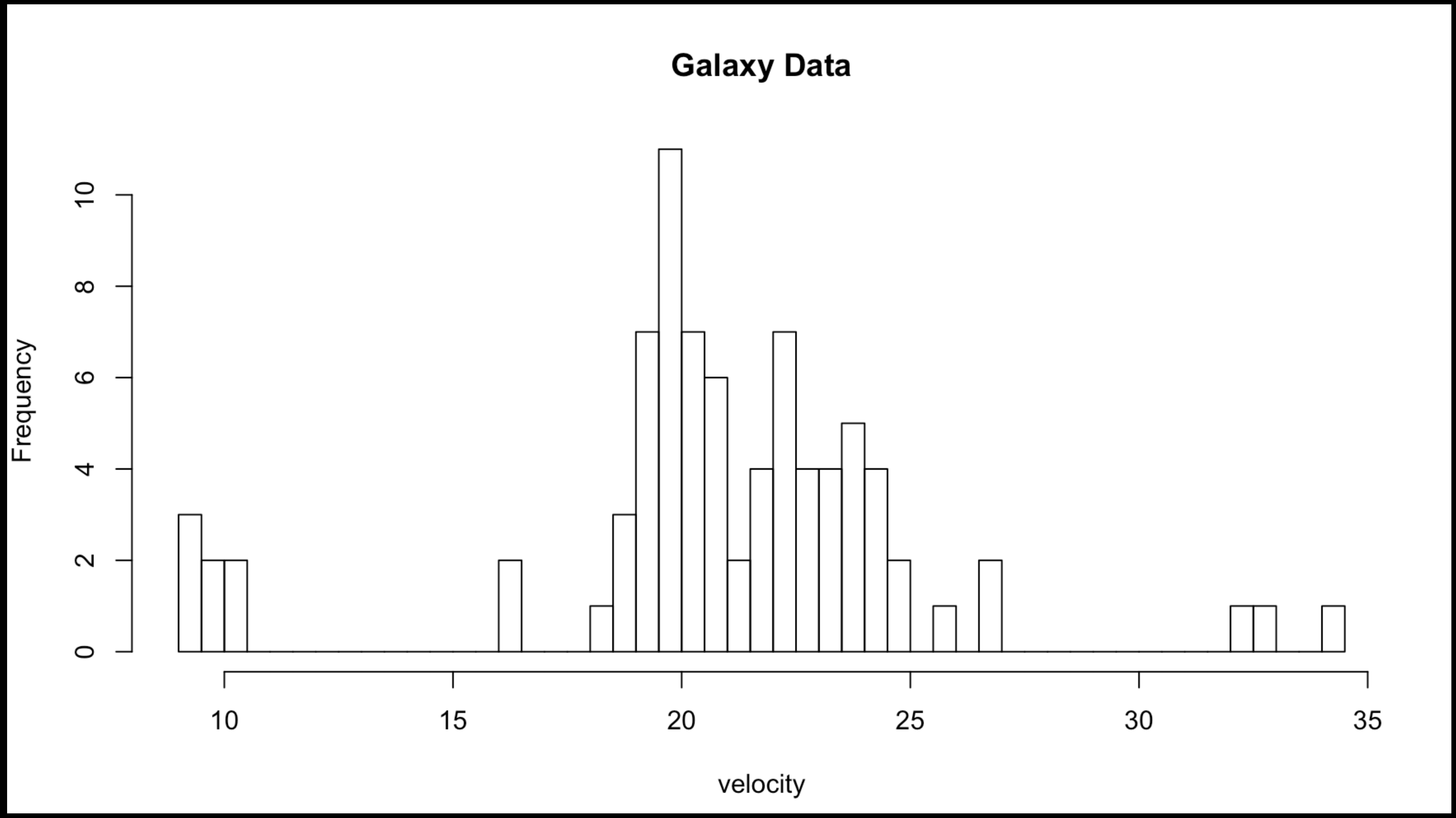
Clustering Algorithms

- How do they work?
- What makes them special?

Cluster these articles

Mixture Models

- $P(Y_i \mid \theta, \sigma, p) = \sum_{k=1}^{K} p_k N(Y_i \mid \theta_k, \sigma_k^2)$

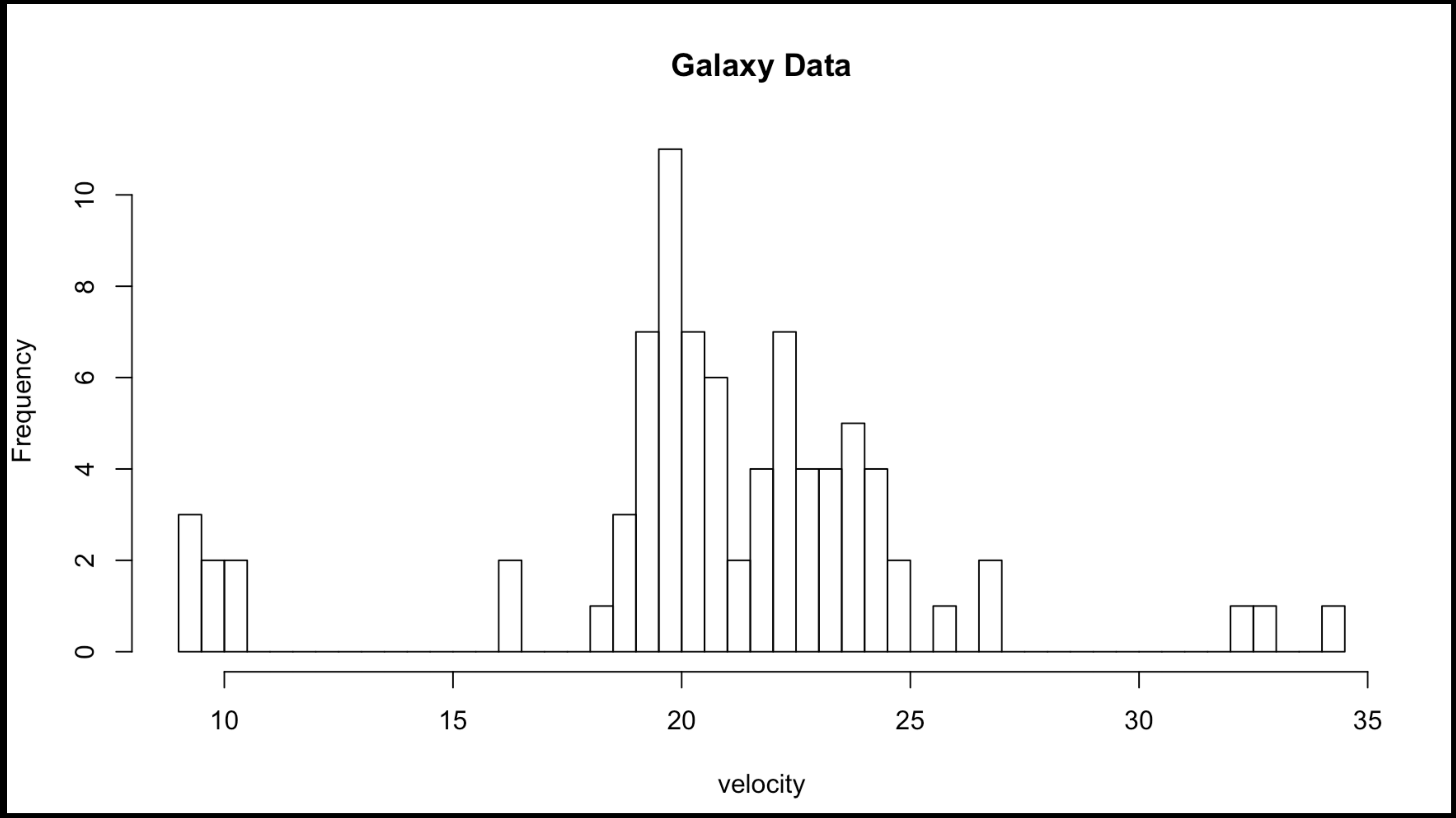


Mixture Models

- $P(Y_i \mid \theta, \sigma, p) = \sum_{k=1}^{K} p_k N(Y_i \mid \theta_k, \sigma_k^2)$

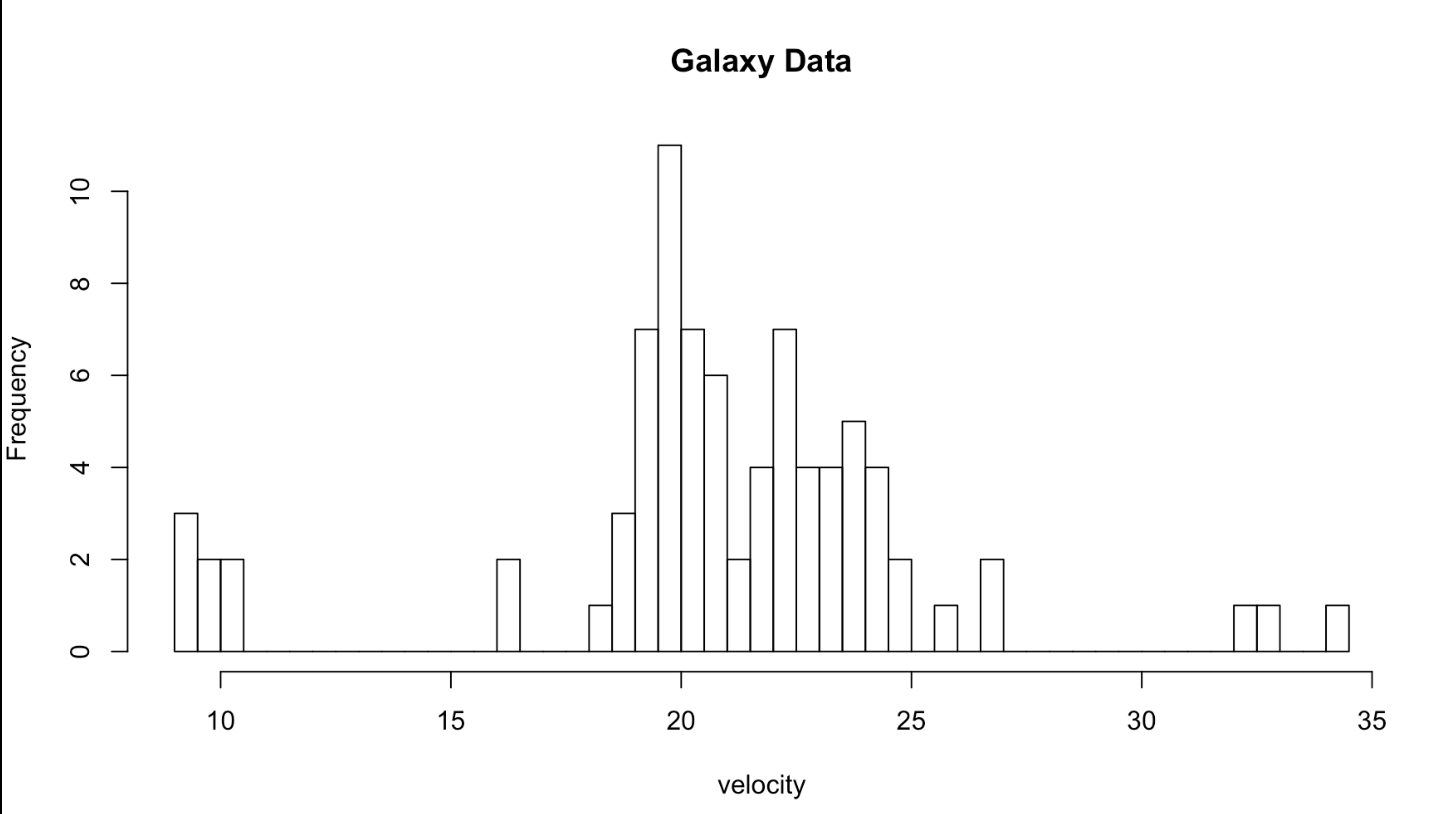
- $P(Z_i = k \mid p_k) = p_k$

- $P(Y_i \mid Z_i = k, \theta, \sigma, p) = N(Y_i \mid \theta_k, \sigma_k^2)$



Mixture Models

- $P(Y_i \mid \theta, \sigma, p) = \sum_{k=1}^{K} p_k N(Y_i \mid \theta_k, \sigma_k^2)$
- $P(Z_i = k \mid p_k) = p_k$
- $P(Y_i \mid Z_i = k, \theta, \sigma, p) = N(Y_i \mid \theta_k, \sigma_k^2)$
- $P(Y_i, Z_i = k \mid \theta, \sigma, p) = p_k N(Y_i \mid \theta_k, \sigma_k^2)$
- $P(Z_i = k \mid Y_i, \theta, \sigma, p) = \dfrac{p_k N(Y_i \mid \theta_k, \sigma_k^2)}{\sum_{j=1}^{K} p_j N(Y_i \mid \theta_j, \sigma_j^2)}$



Z defines group membership

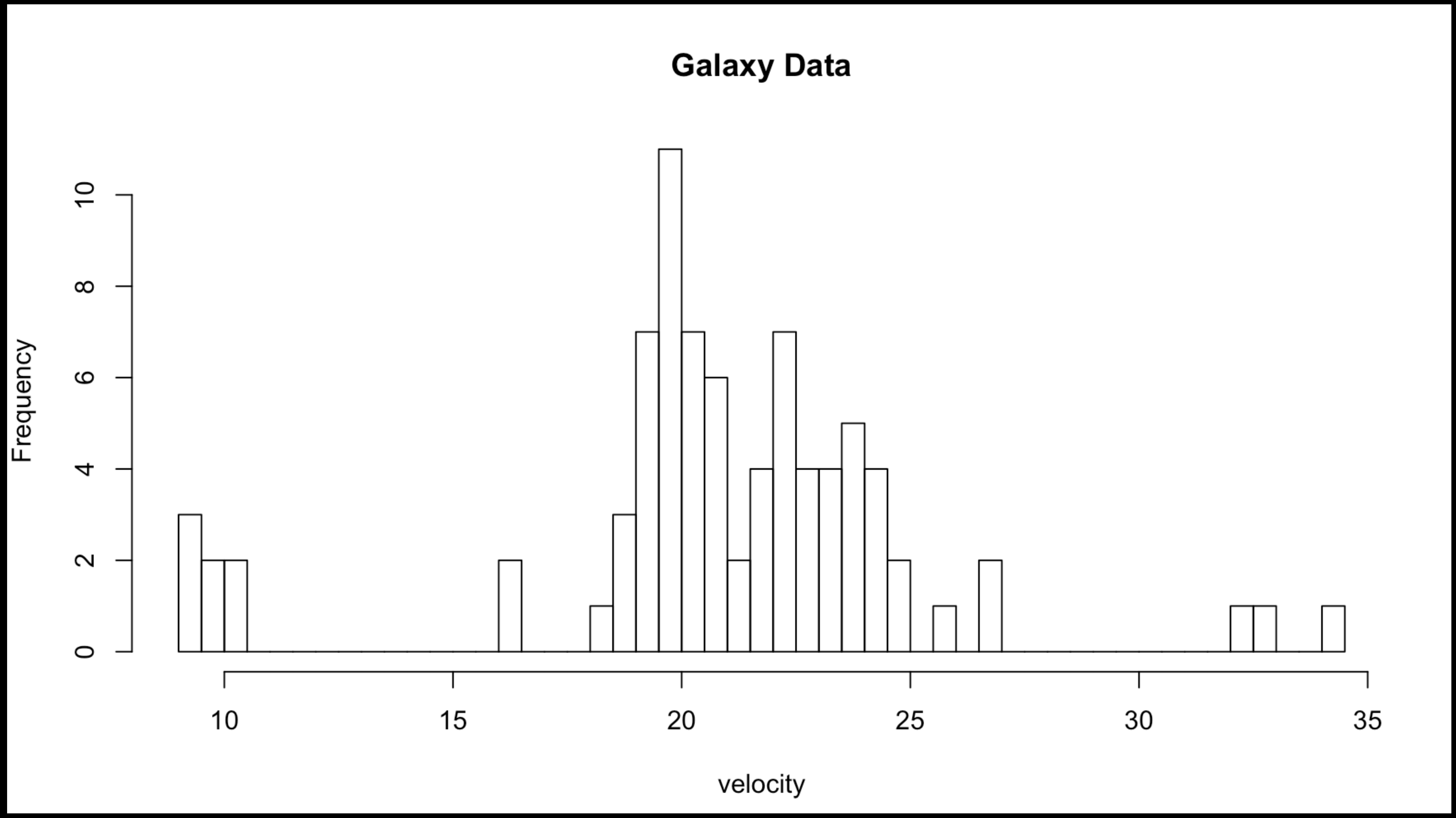
- Single group for an observational unit: Hard clustering
- Multiple groups for an observational unit: Soft clustering

The geometry

- Changing cluster shape

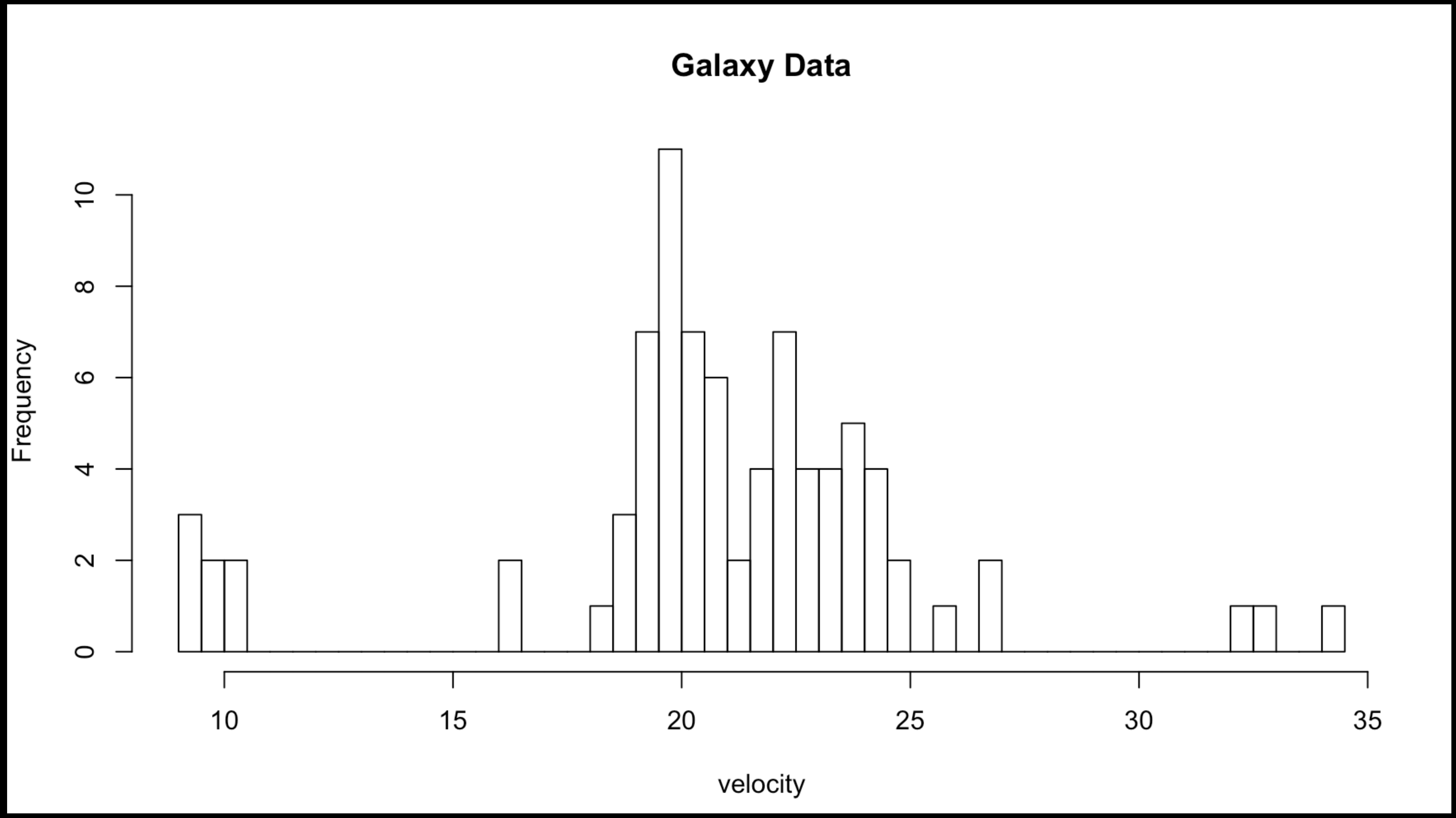
Repeated measures within an observation

- $P(Y_{ij} \mid \theta, \sigma, p) = \sum_{k=1}^{K} p_{ik} N(Y_{ij} \mid \theta_k, \sigma_k^2)$



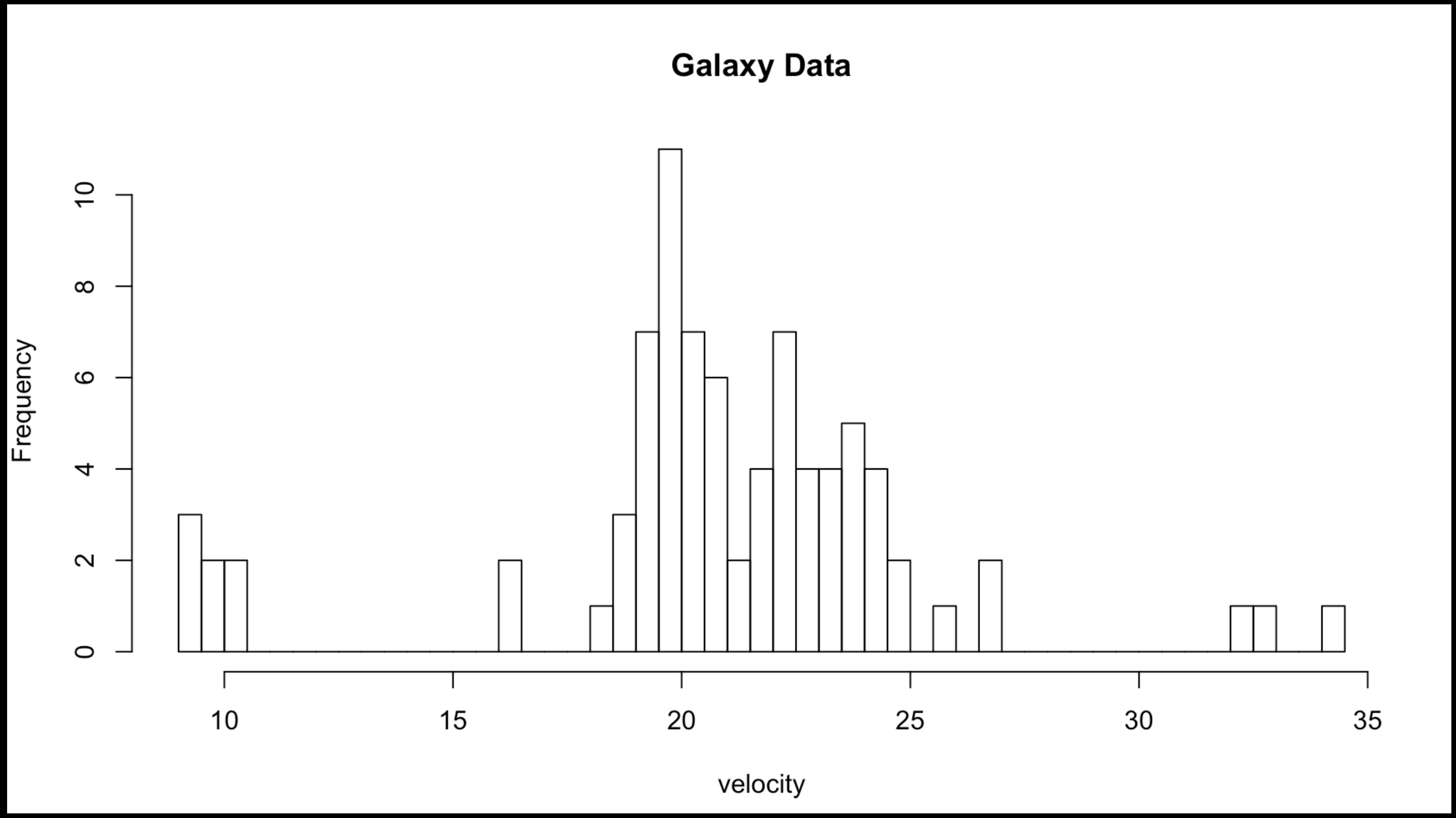
Mixture Models

- $P(Y_{ij} \mid \theta, \sigma, p) = \sum_{k=1}^{K} p_{ik} N(Y_{ij} \mid \theta_k, \sigma_k^2)$
- $P(Z_{ij} = k \mid p_{ik}) = p_{ik}$
- $P(Y_{ij} \mid Z_{ij} = k, \theta, \sigma, p) = N(Y_i \mid \theta_k, \sigma_k^2)$



Mixture Models

- $P(Y_{ij} \mid \theta, \sigma, p) = \sum_{k=1}^{K} p_{ik} N(Y_{ij} \mid \theta_k, \sigma_k^2)$
- $P(Z_{ij} = k \mid p_{ik}) = p_{ik}$
- $P(Y_{ij} \mid Z_{ij} = k, \theta, \sigma, p) = N(Y_i \mid \theta_k, \sigma_k^2)$
- $P(Y_{ij}, Z_{ij} = k \mid \theta, \sigma, p) = p_{ik} N(Y_i \mid \theta_k, \sigma_k^2)$
- $P(Z_{ij} = k \mid Y_{ij}, \theta, \sigma, p) = \dfrac{p_{ik} N(Y_{ij} \mid \theta_k, \sigma_k^2)}{\sum_{\ell=1}^{K} p_{i\ell} N(Y_{ij} \mid \theta_\ell, \sigma_\ell^2)}$



Bag of words model

- Observational unit: Document
- Within document observe words,
    - order does not matter
    - Words are considered iid observations within a document

- Goal is to cluster documents into topics. One document has many topics.

Data generating mechanism

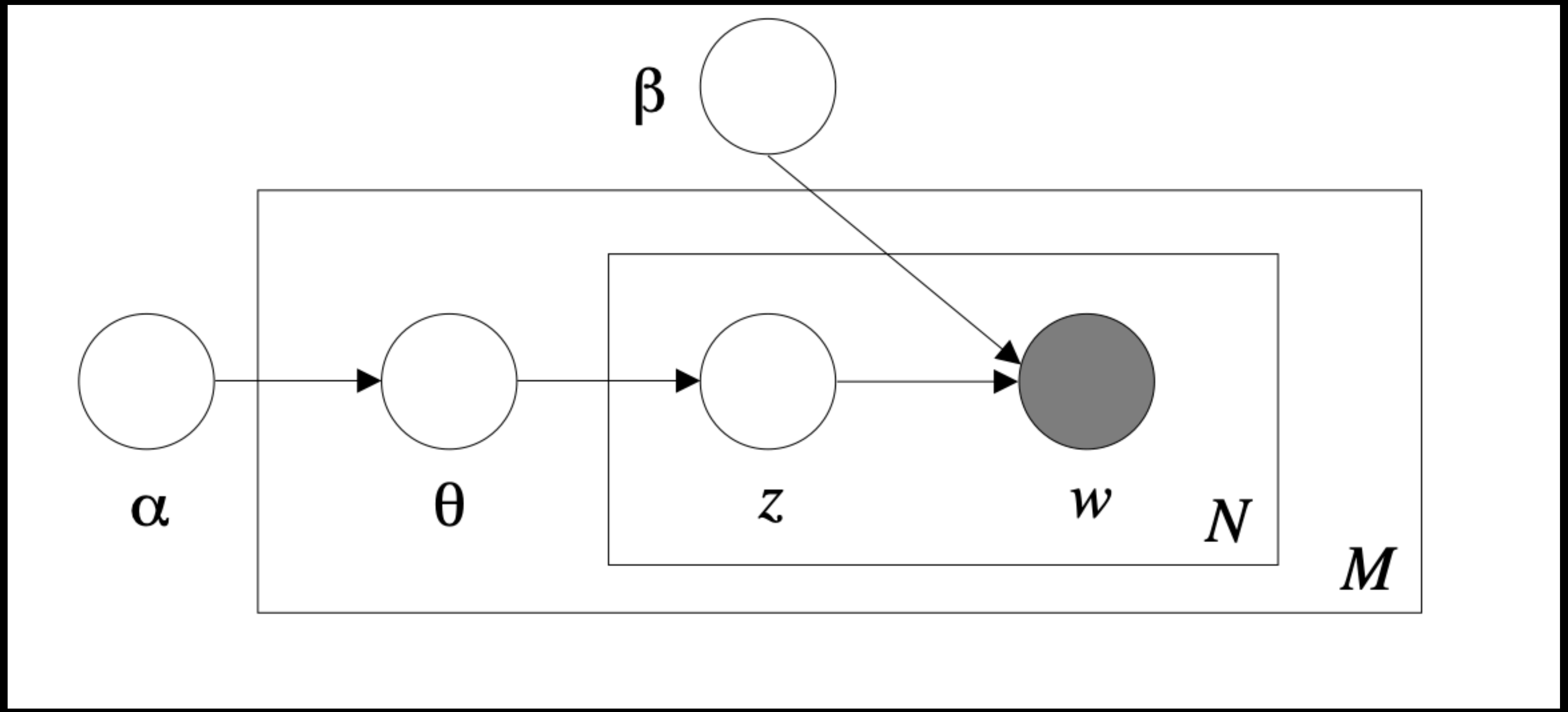
- Make a song containing N words (possibly sample N from Poisson)
- Song has a Dirichlet allocation of topics describing what it is about
- For word i in 1:N
    - Randomly sample one of the topics
    - Topic has a Dirichlet allocation of words within the topic
    - Randomly sample a word within topic

D.Blei, A. NG, M. Jordan (2003) "Latent Dirichlet Allocation", JMLR
http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

- LDA assumes the following generative process for each document **w** in a corpus **D**:

- 1. Choose N ~ Poisson(ξ), there are N word in the document
  2. Choose a topic allocation θ~Dir(α)
  3. For each of the **N** words $\mathbf{w}_n$:

- (a) Choose a topic $\mathbf{z_n}$ ~ Multinomial(**θ**). Each position in the doc has a latent topic
  (b) Choose a word $\mathbf{w_n}$ from $\mathbf{p(w_n | z_n, \beta)}$, a multinomial probability conditioned on the topic $\mathbf{z_n}$.



- LDA assumes the following generative process for each document **w** in a corpus **D**:

- 1. Choose N ~ Poisson(ξ), there are N word in the document
  2. Choose a topic allocation θ~Dir(α), the Dirichlet has a prior vector α.
  3. For each of the **N** words $\mathbf{w}_n$:

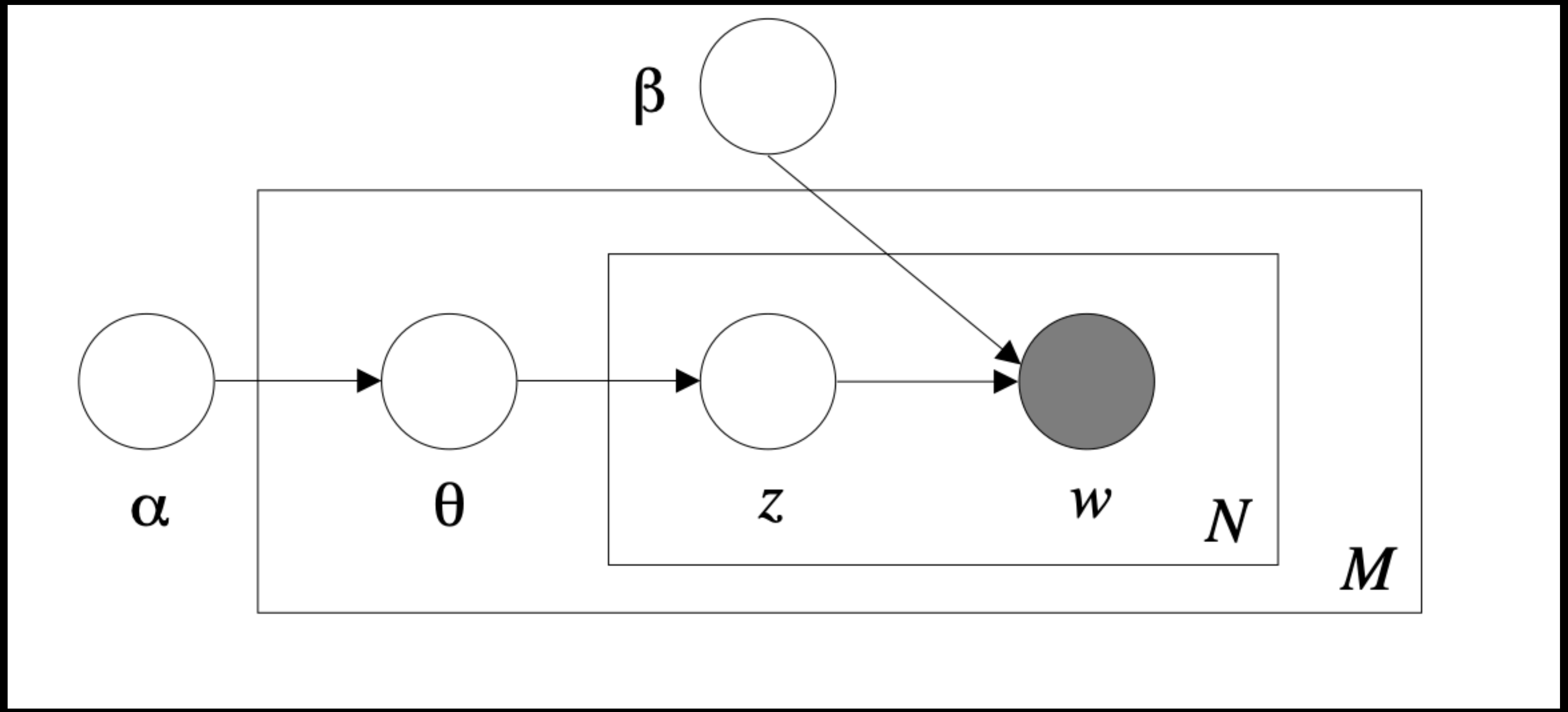
- (a) Choose a topic $\mathbf{z_n}$ ~ Multinomial(**θ**). Each position in the doc has a latent topic
  (b) Choose a word $\mathbf{w_n}$ from $\mathbf{p(w_n | z_n, \beta)}$, a multinomial probability conditioned on the topic $\mathbf{z_n}$. Each topic has it's own pdf over words, the word Dirichlet has prior vector **β**

Getting stared

- Convert corpus into a document term matrix.
- library(topicmodels)

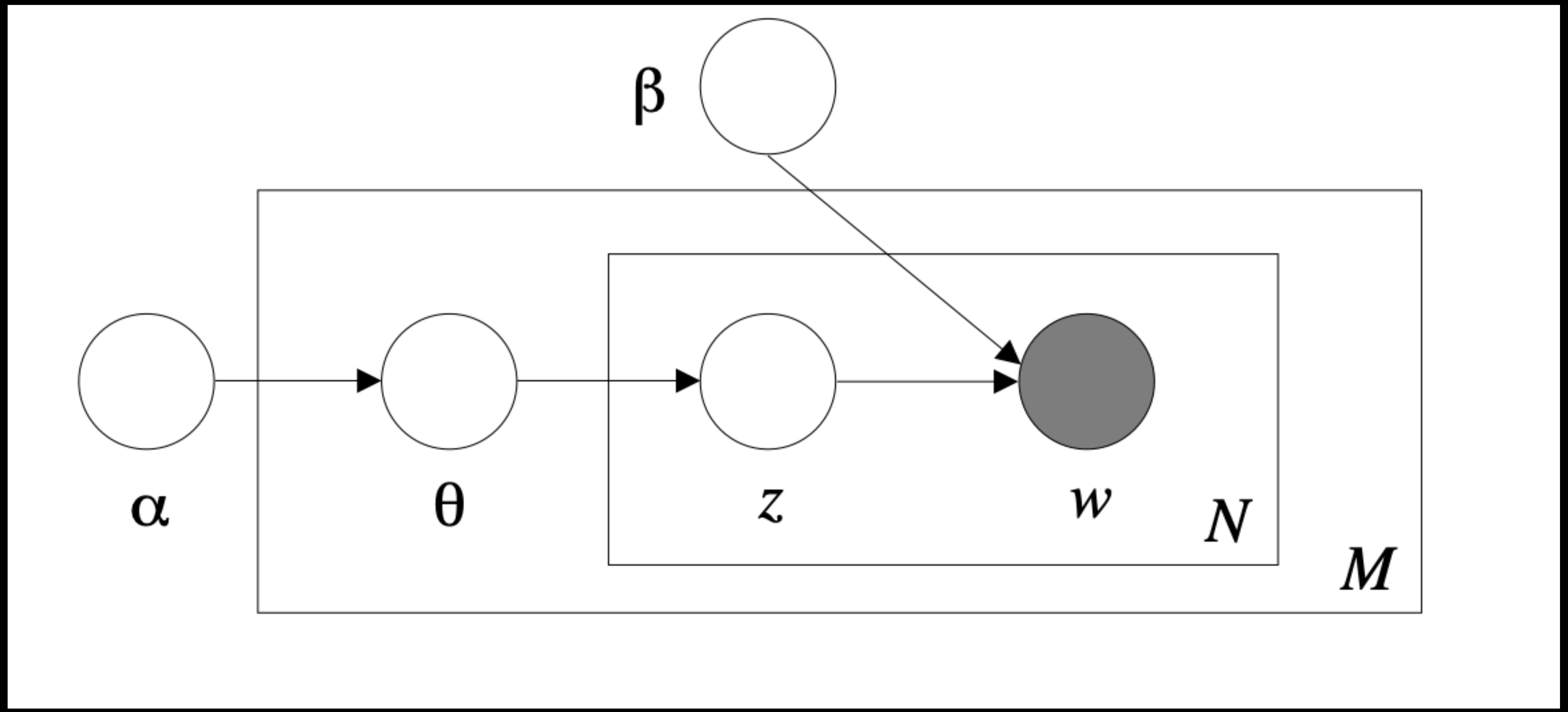
Example from Associated Press Articles

- library(topicmodels)
- data("AssociatedPress")
- AssociatedPress
- APLDA = LDA(AssociatedPress,k=5)   # < 1 minute to run
-

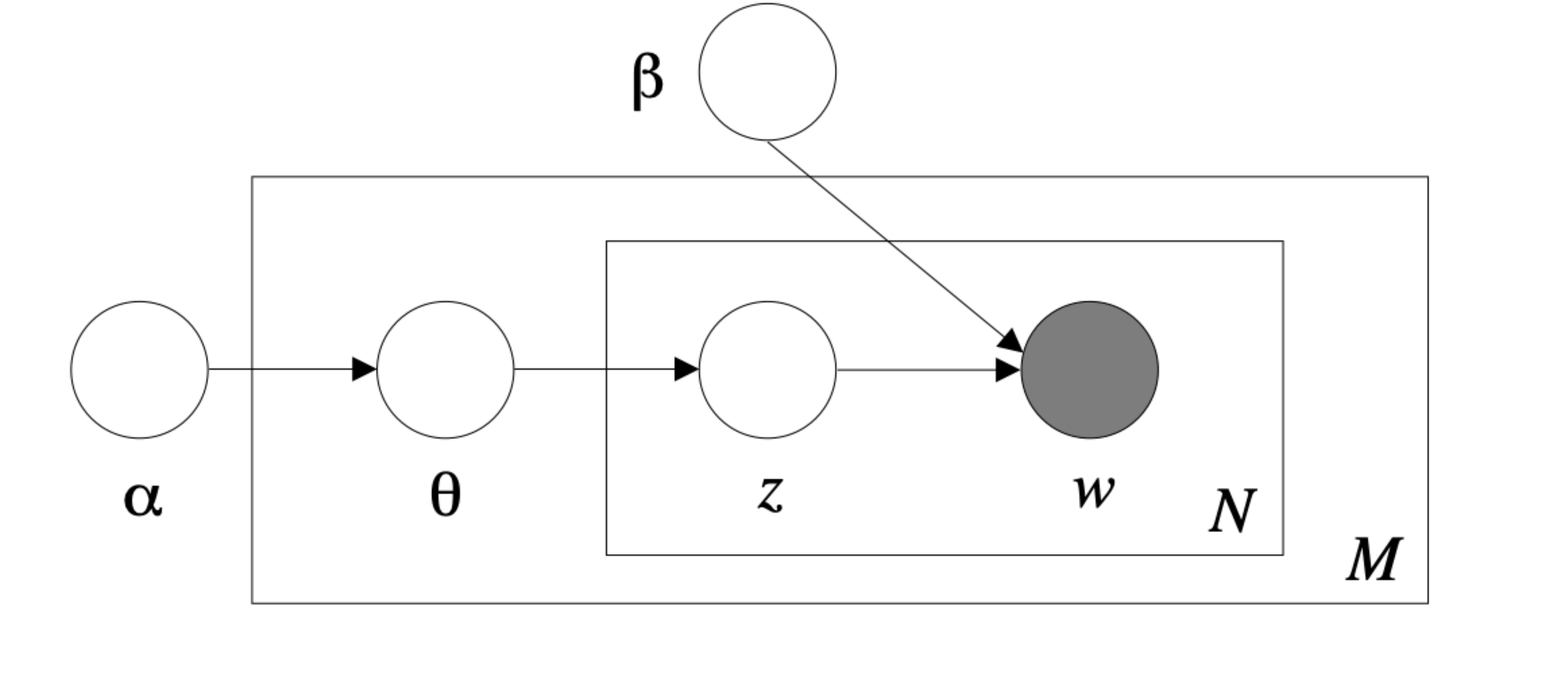


- Within topic word probabilities: their beta == Blei's $\mathbf{p(w_n \mid z_n, \beta)}$

- AP_topics = tidy(APLDA, matrix = "beta")



- library(ggplot2)
- library(dplyr)
- AP_TopWords = AP_topics %>%
- group_by(topic) %>% # take an action within topic values
- top_n(10, beta) %>% # find the largest 10 values based on the 'beta' column
- ungroup() %>% # stop acting within a topic
- arrange(topic, -beta) # sort the



- AP_TopWords %>%
- mutate(term = reorder_within(term, beta, topic)) %>% # Used for faceting (glue topic to term) basically make sure that topic 1 is my topic #1
- ggplot(aes(term, beta, fill = factor(topic))) +
- geom_col(show.legend = FALSE) +
- facet_wrap(~ topic, scales = "free") +
- coord_flip() +
- scale_x_reordered()

Also consider bigger differences differentiators between topics

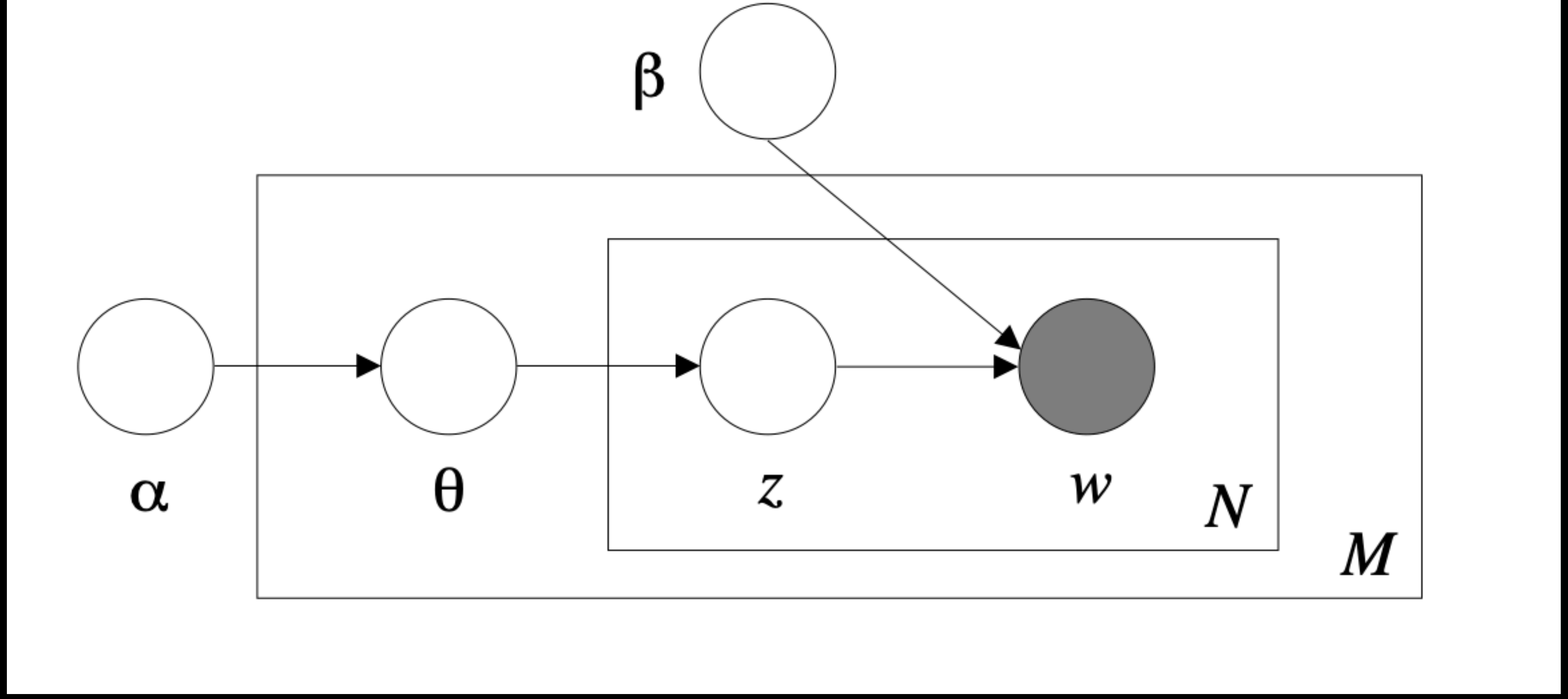
- library(tidyr)
- beta_spread = AP_topics %>%
- mutate(topic = paste0("topic", topic)) %>% # set the names so I remember what the numbers are
- spread(topic, beta) %>% #Make topics into columns, rows are then words
- filter(topic1 > .001 | topic2 > .001| topic3 > .001| topic4 > .001| topic5 > .001) %>% #keep only major words
- mutate(log_ratio21 = log2(topic2 / topic1)) %>% mutate(log_ratio31 = log2(topic3 / topic1)) %>%
- mutate(log_ratio41 = log2(topic4 / topic1)) %>% mutate(log_ratio51 = log2(topic5 / topic1))

Biggest differences in words

- beta_spread %>%
- filter(log_ratio21 > 50 | log_ratio21 < -50) %>%   #keep only major words
- mutate(log_ratio21 = sort(log_ratio21)) %>%
- ggplot(aes(term, log_ratio21)) +
- geom_col(show.legend = FALSE) +
- coord_flip() +
- scale_x_reordered()

Most similar words

- beta_spread %>%
- filter(log_ratio21 < .5 & log_ratio21 > -.5) %>%   #heavy trimming required
- mutate(log_ratio21 = sort(log_ratio21)) %>%
- ggplot(aes(term, log_ratio21)) +
- geom_col(show.legend = FALSE) +
- coord_flip() +
- scale_x_reordered()



- Their gamma == Blei's $\mathbf{P(z_n|\theta,w_n)}$

- ap_documents = tidy(APLDA, matrix = "gamma")

- Estimate of the proportion of words from a document that are generated from a specific topic

Common words within a topic

- Count the most common words for a particular document; before class document 2 was mainly pulling from topic # 1
- tidy(AssociatedPress) %>%
- filter(document == 9) %>%
- arrange(desc(count))

GETTING STARTED

- wordcount = uniquesongs %>%   #matrix with cols for document and full text
-   unnest_tokens(output = word, input = songlyric) %>%
- # anti_join(stop_words) %>%   # here it’s better to not exclude these 1149 words
-   group_by(track_title) %>%   # count within a document
-   count(word,sort=TRUE)%>%
-   ungroup()

- DTM = wordcount %>% cast_dtm(term=word,document=track_title,value=n)

Some words shouldn't be included

- Jude, Octopus, Walrus, eggman,…
- DTM95 = removeSparseTerms(DTM,.95) # remove terms that are are least this proportion sparse

Text Mining with R
Chapter 3: tf-idf
Chapter 6 Topic Modelling

- k = 10
- # trim out empty documents
- DTMatrix = as.matrix(DTM95)
- sumDTMatrix = apply(DTMatrix,1,sum)
- BeatlesLDA = LDA(DTM95, k)

- library(wordcloud) ## Loading required package: RColorBrewer
- par(mfrow = c(1, 1))
- v = sort(colSums(as.matrix(DTM95)), decreasing = TRUE)
- MainWords = names(v)
- d = data.frame(word = MainWords, freq = v)
- wordcloud(d$word, colors = rainbow(4), random.color = T, d$freq, min.freq = 10,scale=c(8,2))   # scale argument is so that you can see it projected

- summed = matrix(colSums(as.matrix(DTM95)), ncol = 1)
- rownames(summed) = colnames(DTM95)
- summedsorted = summed[sort(summed, decreasing = T, index.return = T)$ix,]
- d = data.frame(word = names(summedsorted), freq = summedsorted)
- barplot(d$freq[1:50], las = 2, names.arg = d$word[1:50], col = "lightblue", main = "Top 50 most frequent words", ylab = "Word frequencies")

Most frequent words

- #Find the least sparse words of the 349 songs:
- sort(apply(as.matrix(DTM)>0,2,sum),decreasing=TRUE)[1:10]
- #If a word appears in all documents it shouldn’t matter in defining the topic

Inverse Document Frequency: measure of commonality of words

- $idf(term) = ln\left(\frac{n_{docs}}{n_{docswithword}}\right)$
- Ndocs = dim(as.matrix(DTM))[1]
- book_words = wordcount %>% mutate(Ndocs=Ndocs)
- #count docs with the word
- Ndocswithword = wordcount %>% group_by(word) %>% count() %>% ungroup()
- Ndocswithword = Ndocswithword %>% rename(NdocsWithWord =n) #rename col
- wordcountidf = left_join(book_words, Ndocswithword) # extend the tibble

Inverse Document Frequency

- $$idf(term) = ln\left(\frac{n_{docs}}{n_{docswithword}}\right)$$

- wordcountidf = wordcountidf %>% mutate(idf = log(Ndocs/NdocsWithWord)) # extend the tibble with the idf column

- #Most common words within songs
- freq_by_rank= wordcountidf %>%
-     group_by(track_title) %>%
-     mutate(rank = row_number())
- rank1words = freq_by_rank%>% filter(rank ==1)
- head(sort(table(rank1words$word),decreasing=TRUE),25)

tf_idf

- $$tfidf(term) = \left(\frac{N_{occur}}{N_{\text{words in doc}}}\right) * ln\left(\frac{n_{docs}}{n_{docswithword}}\right)$$

- #Term frequency weighted by the commonality of the word ≈ importance of the word

- WordsInDoc = wordcountidf %>% group_by(track_title)%>% summarize(NwordsinDoc = sum(n))

- wordcountidf = left_join(wordcountidf ,WordsInDoc)

tf_idf

- $$tfidf(term) = \left(\frac{N_{occur}}{N_{\text{words in doc}}}\right) * ln\left(\frac{n_{docs}}{n_{docswithword}}\right)$$

- #Term frequency weighted by the commonality of the word ≈ importance of the word

- wordcountTFIDF = wordcountidf %>% mutate(termfreq = n/NwordsinDoc, tfidf = n/NwordsinDoc*idf)

- #alternatively: obtain tf_idf directly in one move:

- wordcountTFIDF = wordcountTFIDF %>% bind_tf_idf(word,track_title,n)

Most/Least important words

- $tfidf(term) = \left(\frac{N_{occur}}{N_{\text{words in doc}}}\right) * ln\left(\frac{n_{docs}}{n_{docswithword}}\right)$
- wordcountTFIDF %>%
- select(-idf,-termfreq,-Ndocs,-NdocsWithWord,-tfidf) %>%
- arrange(desc(tf_idf))
- wordcountTFIDF %>%
- select(-idf,-termfreq,-Ndocs,-NdocsWithWord,-tfidf) %>%
- arrange(tf_idf)

Beatles LDA and using tf_idf for stop words

- k = 10
- wordcount = uniquesongs %>%
- unnest_tokens(output = word, input = songlyric) %>%
- group_by(track_title) %>%   count(word,sort=TRUE)%>%   ungroup()
- wordcount = wordcount %>% bind_tf_idf(word,track_title,n)
- wordcount = wordcount%>% filter(tf_idf>.001) #remove stop words

Beatles LDA

- k = 10
- wordcount = uniquesongs %>%
- unnest_tokens(output = word, input = songlyric) %>%
- group_by(track_title) %>%   count(word,sort=TRUE)%>%   ungroup()
- DTM = wordcount %>% cast_dtm(term=word,document=track_title,value=n)
- DTM95 = removeSparseTerms(DTM,.95)
- BeatlesLDA = LDA(DTM95, k) ########. FAILS

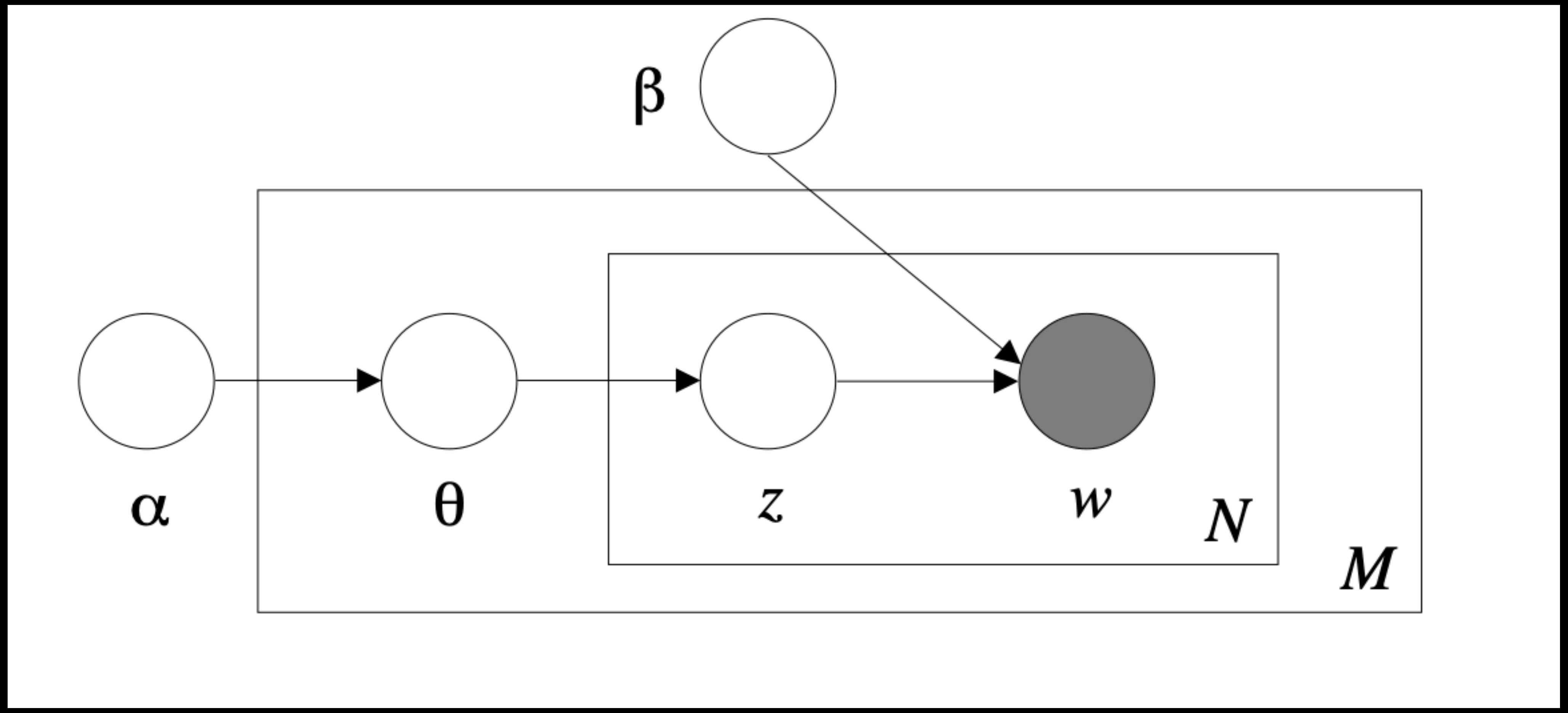
Beatles LDA

- #remove sparse documents
- DTM95matrix = as.matrix(DTM95)
- Removethese = names(which(apply(DTM95matrix,1,sum)<5)))

- DTM = wordcount %>% filter (!(wordcount$track_title %in% names(Removethese)))%>%
    - cast_dtm(term=word,document=track_title,value=n)
- DTM

Beatles LDA

- DTM95 = removeSparseTerms(DTM,.95)
- BeatlesLDA = LDA(DTM95, k) #########YAY!!!



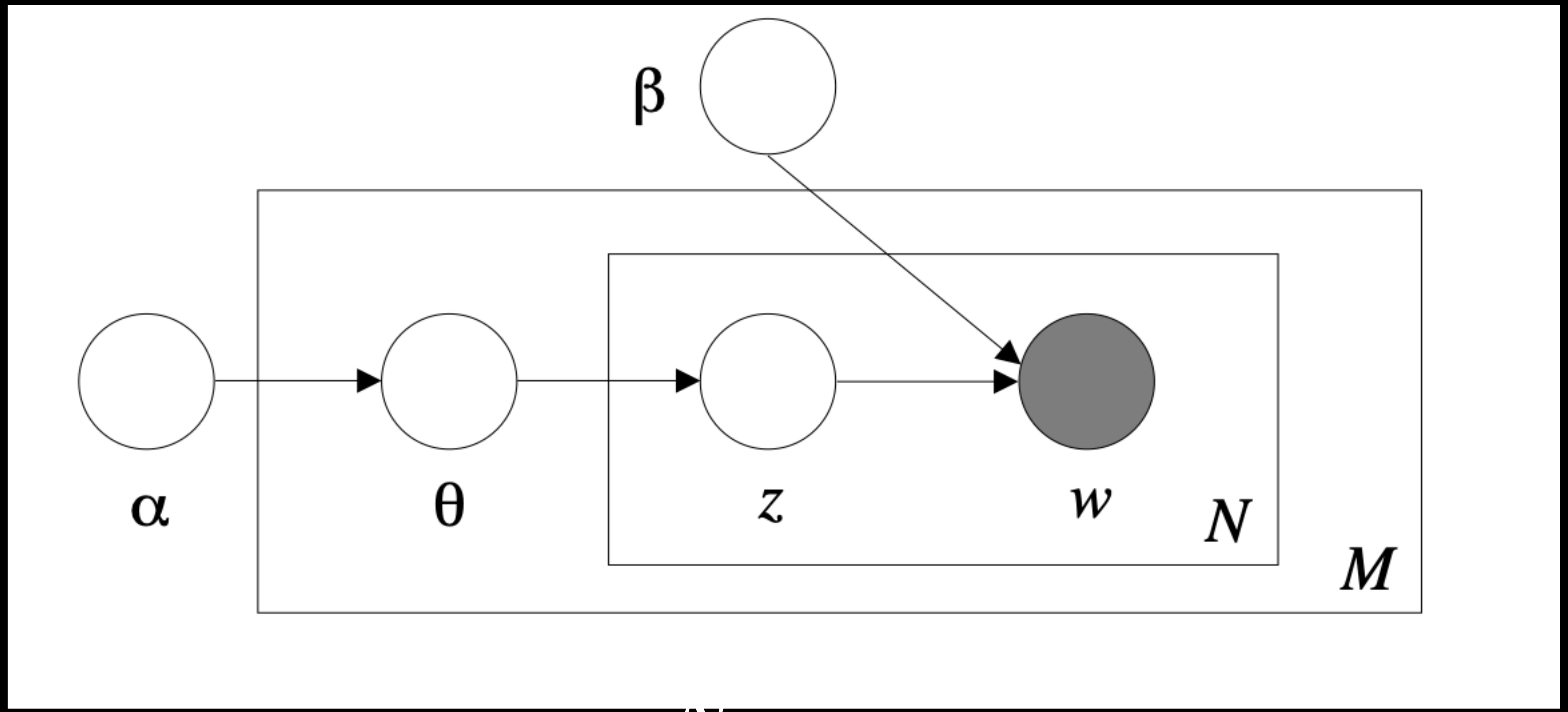
- LDA assumes the following generative process for each document **w** in a corpus **D**:

- 1. Choose N ~ Poisson(ξ), there are N word in the document
  2. Choose a topic allocation θ~Dir(α), the Dirichlet has a prior vector α.
  3. For each of the **N** words $\mathbf{w}_n$:

- (a) Choose a topic $\mathbf{z_n}$ ~ Multinomial(**θ**). Each position in the doc has a latent topic
  (b) Choose a word $\mathbf{w_n}$ from $\mathbf{p(w_n | z_n, \beta)}$, a multinomial probability conditioned on the topic $\mathbf{z_n}$. Each topic has it's own pdf over words, the word Dirichlet has prior vector **β**

Estimation

- Gibbs & Variational EM Algorithm
- http://times.cs.uiuc.edu/course/598f16/notes/lda-survey.pdf
-



- $$P(W, Z, \theta \mid \alpha, \beta) = \prod_{j=1}^{K} P(\theta_j \mid \alpha) \prod_{t=1}^{N_j} P(z_{jt} \mid \theta_j) P(w_{jt} \mid \beta_{zt})$$

- Target distribution:

- $$P(Z, \theta \mid \alpha, \beta) \propto \prod_{j=1}^{K} P(\theta_j \mid \alpha) \prod_{t=1}^{N_j} P(z_{jt} \mid \theta_j) P(w_{jt} \mid \beta_{zt})$$

Variational Methods

- Approximate a distribution using an easy to use distribution. Typically fit minimizing Kullback-Leibler (KL) divergence between the variational distributions **q** and the true posteriors **p**

- Select a variational distribution q with parameters γ, π
- $P(Z, \theta \mid \alpha, \beta) \approx q(z, \theta \mid \gamma, \pi)$
- KL divergence of p to q is
- $D(q \mid\mid p) = \int_\theta \sum_z q(z_j, \theta_j \mid \gamma_j, \pi_j) log \left(\frac{q(z_j, \theta_j \mid \gamma_j, \pi_j)}{p(z_j, \theta_j \mid w_j, \alpha, \beta)} \right) d\theta$