Topic Models AO:Chapter 4.4:LDA

@anemptyarchive

2019/12/07-2019/12/07

Contents

4.4 LDA

・コード全体

```
# 利用パッケージ
library(RMeCab)
library(tidyverse)
```

・テキスト処理

```
## 抽出する単語の指定
# 品詞（大分類）を指定
PoS_1 <- "名詞|^動詞|形容詞"

# 品詞（細分類）を指定
PoS_2 <- "一般|^自立"

# 最低出現頻度を指定
Freq <- 5

# 抽出しない単語を指定
stop_words <- "[a-z]"

# 形態素解析
mecab_df <- docDF("フォルダ名", type = 1) # テキストファイルの保存先を指定する

# 文書dの語彙vの出現回数（N_dv）の集合
N_dv <- mecab_df %>%
        filter(grepl(PoS_1, POS1)) %>%         # 指定した品詞（大分類）を取り出す
        filter(grepl(PoS_2, POS2)) %>%         # 指定した品詞（細分類）を取り出す
        filter(!grepl(stop_words, TERM)) %>%  # ストップワードを除く
        select(-c(TERM, POS1, POS2)) %>%       # 数値列のみを残す
        filter(apply(., 1, sum) >= Freq) %>%  # 指定した頻度以上の語彙を取り出す
        t()                                    # 転置

# 確認用の行列名
dimnames(N_dv) <- list(paste0("d=", 1:nrow(N_dv)), # 行名
                       paste0("v=", 1:ncol(N_dv))) # 列名

# 文書dの単語数（N_d）のベクトル
N_d <- apply(N_dv, 1, sum) # 行方向に和をとる
```

```
# 文書数 (D)
D <- nrow(N_dv)

# 総語彙数 (V)
V <- ncol(N_dv)
```

・パラメータの初期設定

```
# トピック数 (K)
K <- 4   # 任意の値を指定する

# トピックの変分事後分布 (q_dvk) の集合
q_dvk <- array(0, dim = c(D, V, K),
               dimnames = list(paste0("d=", 1:D),
                               paste0("v=", 1:V),
                               paste0("k=", 1:K))) # 確認用

# 事前分布のパラメータ (alpha, beta)
alpha <- 2   # 任意の値を指定する
beta  <- 2   # 任意の値を指定する

# トピック分布の変分事後分布のパラメータ (alpha_dk) の集合
alpha_dk <- seq(1, 3, 0.01) %>%                # 任意の範囲を指定する
            sample(D * K, replace = TRUE) %>%  # 値をランダムに生成
            matrix(nrow = D, ncol = K,         # 次元の設定
                   dimnames = list(paste0("d=", 1:D), paste0("k=", 1:K))) # 行名, 列名

# 単語分布の変分事後分布のパラメータ (beta_kv) の集合
beta_kv <- seq(1, 3, 0.01) %>%                 # 任意の範囲を指定する
           sample(K * V, replace = TRUE) %>%   # 値をランダムに生成
           matrix(nrow = K, ncol = V,          # 次元の設定
                  dimnames = list(paste0("k=", 1:K), paste0("v=", 1:V))) # 行名, 列名
```

・**LDA**

```
# 推定回数を指定
Iter <- 50

# 推移の確認用の受け皿を用意
trace_alpha <- array(0, dim = c(D, K, Iter + 1),
                     dimnames = list(paste0("d=", 1:D),
                                     paste0("k=", 1:K),
                                     paste0("Est", 1:(Iter + 1))))
trace_beta  <- array(0, dim = c(K, V, Iter + 1),
                     dimnames = list(paste0("k=", 1:K),
                                     paste0("v=", 1:V),
                                     paste0("Est", 1:(Iter + 1))))

# 初期値を代入
trace_alpha[, , 1] <- alpha_dk
trace_beta[, , 1]  <- beta_kv

# パラメータ推定
```

```
for(i in 1:Iter) {

  # パラメータを初期化
  new_alpha_dk <- matrix(data = alpha, nrow = D, ncol = K,
                         dimnames = list(paste0("d=", 1:D), paste0("k=", 1:K)))

  new_beta_kv <- matrix(data = beta, nrow = K, ncol = V,
                        dimnames = list(paste0("k=", 1:K), paste0("v=", 1:V)))

  for(d in 1:D) { ## (各文書)

    # 負担率を計算
    tmp_q_alpha_k <- digamma(alpha_dk[d, ]) - digamma(sum(alpha_dk[d, ]))              ## KdimVec <- Kd
    tmp_q_beta_vk <- N_dv[d, ] * t(digamma(beta_kv) - digamma(apply(beta_kv, 1, sum))) ## V*KdimMat <-
    q_dvk[d, , ]  <- t(exp(tmp_q_alpha_k + t(tmp_q_beta_vk))) ## V*KdimVec <- t(KdimVec + t(V*KdimMat))

    # 負担率を正規化
    q_dvk[d, , ] <- q_dvk[d, , ] / apply(q_dvk[d, , ], 1, sum) ## V*KdimMat <- V*KdimMat / VdimVec

    # alpha_dk を更新
    new_alpha_dk[d, ] <- new_alpha_dk[d, ] + apply(q_dvk[d, , ], 2, sum) ## KdimVec <- KdimVec + KdimVe

    # beta_kv を更新
    new_beta_kv <- new_beta_kv + t(q_dvk[d, , ] * N_dv[d, ])   ## K*VdimMat <- K*VdimMat + t(V*KdimMat

  } ## (/各文書)

  # パラメータを更新
  alpha_dk <- new_alpha_dk
  beta_kv  <- new_beta_kv

  # 推移の確認用
  trace_alpha[, , i + 1] <- alpha_dk
  trace_beta[, , i + 1]  <- beta_kv
}
```

・コードの解説

文書ごとにパラメータ推定を行っていきます。

R ではベクトルとマトリクスとを計算するとき、ベクトルの各要素をマトリクスの 1 行 1 列目の要素から列方向に順番に対応させて計算していきます。つまり、ベクトルの要素の数とマトリクスの列の要素の数 (行数) を一致させると、ベクトルの 1 つ目の要素をマトリクスの 1 行目の各要素に対応させることができます。なので、適宜転置して計算していきます。

・負担率の更新

```
# 負担率を計算
tmp_q_alpha_k <- digamma(alpha_dk[d, ]) - digamma(sum(alpha_dk[d, ]))              ## KdimVec <- KdimVe
tmp_q_beta_vk <- N_dv[d, ] * t(digamma(beta_kv) - digamma(apply(beta_kv, 1, sum))) ## V*KdimMat <- Vdim
q_dvk[d, , ]  <- t(exp(tmp_q_alpha_k + t(tmp_q_beta_vk))) ## V*KdimVec <- t(KdimVec + t(V*KdimMat))

# 負担率を正規化
```

```
q_dvk[d, , ] <- q_dvk[d, , ] / apply(q_dvk[d, , ], 1, sum) ## V*KdimMat <- V*KdimMat / VdimVec
```

負担率 q_{dvk} は

$$q_{dvk} \propto \exp\left\{\Psi(\alpha_{dk}) - \Psi\Big(\sum_{k=1}^{K}\alpha_{dk}\Big) + N_{dv}\left(\Psi(\beta_{kv}) - \Psi\Big(\sum_{v=1}^{V}\beta_{kv}\Big)\right)\right\} \tag{4.8}$$

この計算を行い正規化したものです。ディガンマ関数 $\Psi()$ の計算は `digamma()` で行えます。

前の $\boldsymbol{\alpha}$ の計算は、K 次元ベクトルとスカラーなのでそのまま行います。
後の $\boldsymbol{\beta}$ の計算は、括弧の中は K 行 V 列のマトリクスと K 次元ベクトルなのでそのまま行います。ただし `N_dv[d, ]` は V 次元ベクトルなので、転置してから計算します。

`tmp_q_alpha_k` は K 次元ベクトル、`tmp_q_beta_vk` は V 行 K 列のマトリクスなので、$\boldsymbol{\beta}$ の方を転置してから計算します。
ただし、V 行 K 列のマトリクスとして `q_dvk[d, , ]` に代入するため、もう一度転置してから代入します。

最後に、v について 1 から V まで和をとったもので割って正規化します。

・トピック分布のパラメータの更新

```
# alpha_dk を更新
new_alpha_dk[d, ] <- new_alpha_dk[d, ] + apply(q_dvk[d, , ], 2, sum) ## KdimVec <- KdimVec + KdimVec
```

$\boldsymbol{\alpha}$ の各要素の計算式は

$$\alpha_{dk} = \alpha + \sum_{v=1}^{V} q_{dvk} \tag{4.6}$$

です。

推定の度に `new_alpha_dk` の初期値を全て `alpha` の値に設定します。なので、`new_alpha_dk` に `q_dvk` の値を加えていきます。

$\sum_{v=1}^{V}$ の計算を行うために、`q_dvk[d, , ]` に `apply()` を使い列方向に合計します。
この計算結果と `new_alpha_dk[d, ]` は、どちらも K 次元ベクトルなのでそのまま足します。

この処理を 1 行目から D 行目まで順番に行うことで、全ての要素の計算ができます。

・単語分布のパラメータの更新

```
# beta_kv を更新
new_beta_kv <- new_beta_kv + t(q_dvk[d, , ] * N_dv[d, ])   ## K*VdimMat <- K*VdimMat + t(V*KdimMat * Vd
```

$\boldsymbol{\beta}$ の各要素の計算式は

$$\beta_{kv} = \beta + \sum_{d=1}^{D} q_{dvk} N_{dv} \tag{4.7}$$

です。

推定の度に new_beta_kv の初期値を全て beta の値に設定します。なので、new_beta_kv に q_dvk の値を加えていきます。

q_dvk[d, ,] * N_dv[d,] の計算は、V 行 K 列のマトリクスと V 次元ベクトルなのでそのまま行います。
ただし、new_beta_kv は K 行 V 列のマトリクスなので、計算結果を転置から足します。

new_beta_kv に D 回繰り返し足していくことで、$\sum_{d=1}^{D}$ の計算が行われます。

・推定結果の確認

・作図用関数の作成

```
### トピック分布のパラメータ
fn_plotAlpha <- function(alpha_dk) {

  # データフレームを作成
  alpha_WideDF <- cbind(as.data.frame(alpha_dk),
                        doc = as.factor(1:D))

  # データフレームを long 型に変換
  alpha_LongDF <- pivot_longer(
    alpha_WideDF,
    cols = -doc,          # 変換せずにそのまま残す現列の名前
    names_to = "topic",   # 現列名をセルの値とする新しい列の名前
    names_prefix = "k=",  # 現列名から取り除く頭の文字列
    names_ptypes = list(topic = factor()),  # 現列名を値とする際の型
    values_to = "alpha"   # 現セルの値を格納する新しい列の名前
  )

  # 作図
  ggplot(data = alpha_LongDF, mapping = aes(x = topic, y = alpha, fill = topic)) + # データ
    geom_bar(stat = "identity", position = "dodge") +  # 棒グラフ
    facet_wrap(~ doc, labeller = label_both) +         # グラフの分割
    labs(title = "LDA:alpha")                          # タイトル
}


### 単語分布のパラメータ
fn_plotBeta <- function(beta_kv) {

  # データフレームを作成
  beta_WideDF <- cbind(as.data.frame(beta_kv),
                       topic = as.factor(1:K))

  # データフレームを long 型に変換
  beta_LongDF <- pivot_longer(
    beta_WideDF,
    cols = -topic,        # 変換せずにそのまま残す現列の名前
    names_to = "word",    # 現列名をセルの値とする新しい列の名前
```

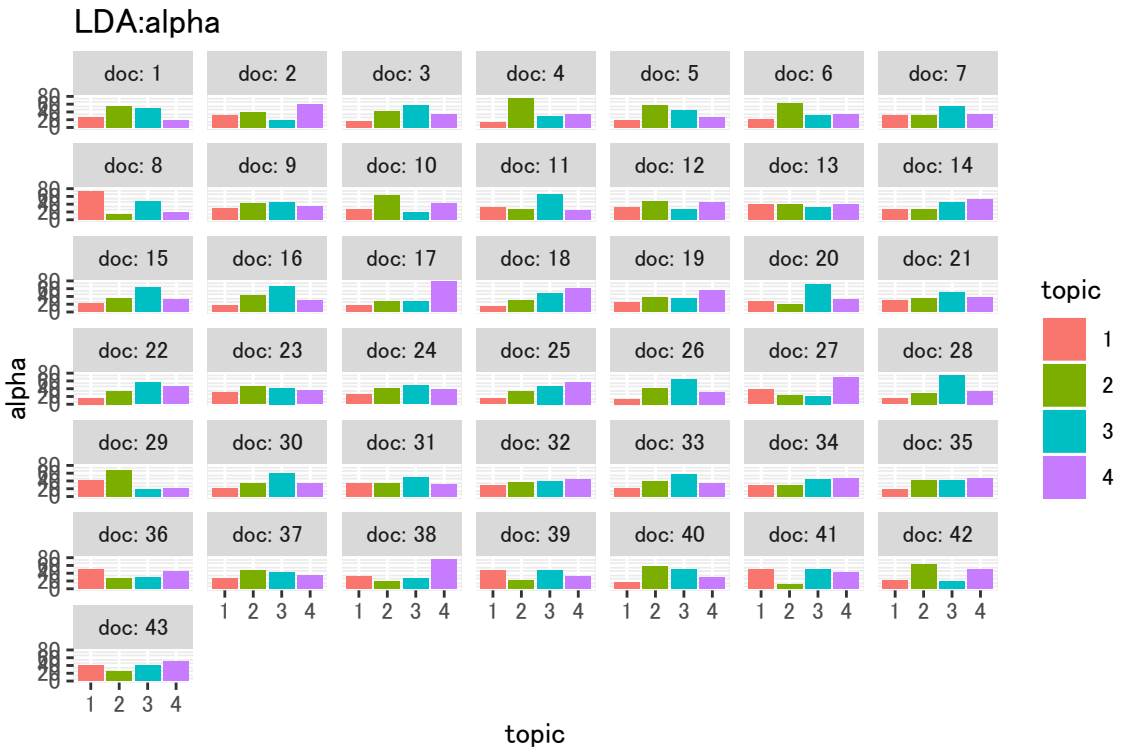
```
    names_prefix = "v=",  # 現列名から取り除く頭の文字列
    names_ptypes = list(word = factor()),  # 現列名を値とする際の型
    values_to = "beta"    # 現セルの値を格納する新しい列の名前
  )

  # 作図
  ggplot(data = beta_LongDF, mapping = aes(x = word, y = beta, fill = word)) +
    geom_bar(stat = "identity", position = "dodge") +  # 棒グラフ
    facet_wrap(~ topic, labeller = label_both) +       # グラフを分割
    scale_x_discrete(breaks = seq(1, V, by = 10)) +    # x軸目盛
    theme(legend.position = "none") +                  # 凡例
    labs(title = "LDA")                                # タイトル
}


### トピック分布（平均値）
fn_plotTheta <- function(alpha_dk) {

  #パラメータの平均値を算出
  theta_dk <- alpha_dk / apply(alpha_dk, 1, sum)

  # データフレームを作成
  theta_WideDF <- cbind(as.data.frame(theta_dk),
                        doc = as.factor(1:D))

  # データフレームを long 型に変換
  theta_LongDF <- pivot_longer(
    theta_WideDF,
    cols = -doc,          # 変換せずにそのまま残す現列の名前
    names_to = "topic",   # 現列名をセルの値とする新しい列の名前
    names_prefix = "k=",  # 現列名から取り除く頭の文字列
    names_ptypes = list(topic = factor()),  # 現列名を値とする際の型
    values_to = "prob"    # 現セルの値を格納する新しい列の名前
  )

  # 作図
  ggplot(data = theta_LongDF, mapping = aes(x = topic, y = prob, fill = topic)) +
    geom_bar(stat = "identity", position = "dodge") +  # 棒グラフ
    facet_wrap(~ doc, labeller = label_both) +         # グラフの分割
    labs(title = "LDA:Theta")                          # タイトル
}


### 単語分布（平均値）
fn_plotPhi <- function(beta_kv) {

  #パラメータの平均値を算出
  phi_kv <- beta_kv / apply(beta_kv, 1, sum)

  # データフレームを作成
  phi_WideDF <- cbind(as.data.frame(phi_kv),
                      topic = as.factor(1:K))
```

```r
  # データフレームを long 型に変換
  phi_LongDF <- pivot_longer(
    phi_WideDF,
    cols = -topic,         # 変換せずにそのまま残す現列の名前
    names_to = "word",     # 現列名をセルの値とする新しい列の名前
    names_prefix = "v=",   # 現列名から取り除く頭の文字列
    names_ptypes = list(word = factor()),  # 現列名を値とする際の型
    values_to = "prob"     # 現セルの値を格納する新しい列の名前
  )

  # 作図
  ggplot(data = phi_LongDF, mapping = aes(x = word, y = prob, fill = word)) +
    geom_bar(stat = "identity", position = "dodge") +  # 棒グラフ
    facet_wrap(~ topic, labeller = label_both) +       # グラフを分割
    scale_x_discrete(breaks = seq(1, V, by = 10)) +    # x 軸目盛
    theme(legend.position = "none") +                  # 凡例
    labs(title = "LDA:Phi")                            # タイトル
}


### トピック分布のパラメータの推移を作図
fn_plotTraceAlpha <- function(trace_alpha, DocNum = 1) {

  if(DocNum > D){
    return("ERROR:DocNum > D")
  }

  # 文書番号を指定
  DocNum <- DocNum

  # データフレームを作成
  trace_alpha_WideDF <- cbind(as.data.frame(trace_alpha[DocNum, ,]),
                              topic = as.factor(1:K))

  # データフレームを long 型に変換
  trace_alpha_LongDF <- pivot_longer(
    trace_alpha_WideDF,
    cols = -topic,           # 変換せずにそのまま残す現列の名前
    names_to = "Iteration",  # 現列名をセルの値とする新しい列の名前
    names_prefix = "Est",    # 現列名から取り除く頭の文字列
    names_ptypes = list(Iteration = numeric()),  # 現列名を値とする際の型
    values_to = "alpha"      # 現セルの値を格納する新しい列の名前
  )

  # 描画
  ggplot(data = trace_alpha_LongDF, mapping = aes(x = Iteration, y = alpha, color = topic)) +
    geom_line() +  # 棒グラフ
    labs(title = "LDA:alpha",
         subtitle = paste0("d=", DocNum)) # タイトル
}


### 単語分布のパラメータの推移を作図
```

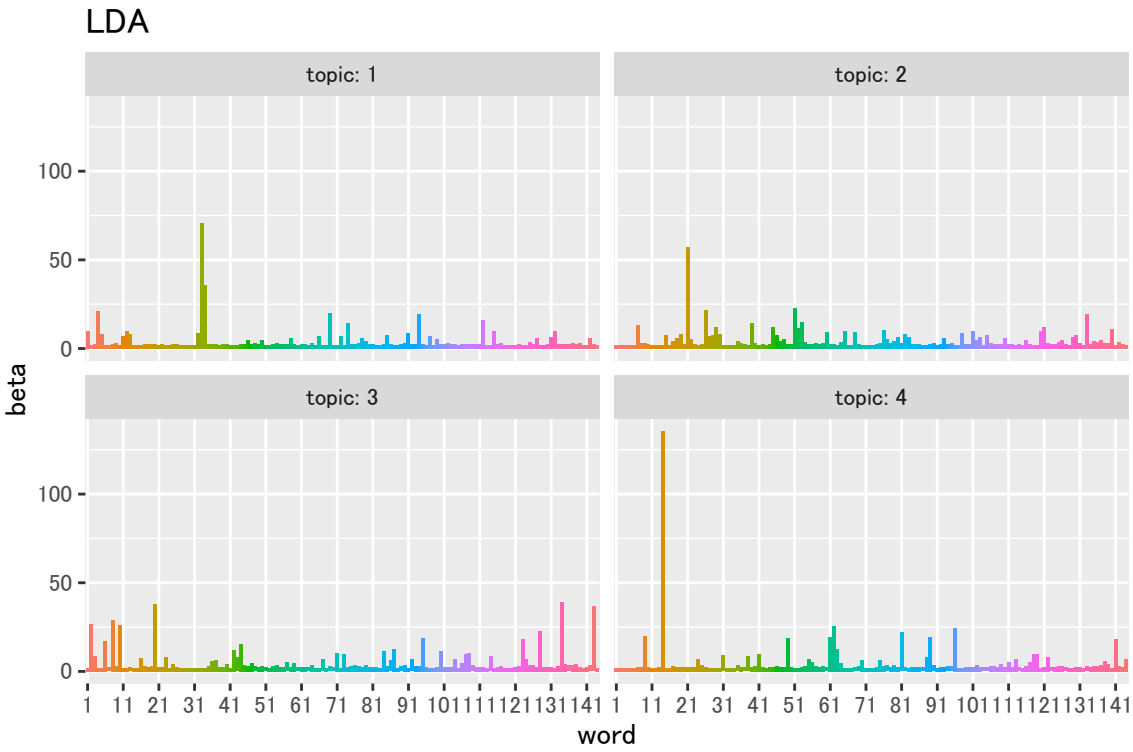
```r
fn_plotTraceBeta <- function(trace_beta, TopicNum = 1) {

  if(TopicNum > K){
    return("ERROR:TopicNum > K")
  }

  # トピック番号を指定
  TopicNum <- TopicNum

  # データフレームを作成
  trace_beta_WideDF <- cbind(as.data.frame(trace_beta[TopicNum, , ]),
                             word = as.factor(1:V))

  # データフレームを long 型に変換
  trace_beta_LongDF <- pivot_longer(
    trace_beta_WideDF,
    cols = -word,            # 変換せずにそのまま残す現列の名前
    names_to = "Iteration",  # 現列名をセルの値とする新しい列の名前
    names_prefix = "Est",    # 現列名から取り除く頭の文字列
    names_ptypes = list(Iteration = numeric()),  # 現列名を値とする際の型
    values_to = "beta"       # 現セルの値を格納する新しい列の名前
  )

  # 描画
  ggplot(data = trace_beta_LongDF, mapping = aes(x = Iteration, y = beta, color = word)) +
    geom_line(alpha = 0.5) +           # 棒グラフ
    theme(legend.position = "none") +  # 凡例
    labs(title = "LDA:beta",
         subtitle = paste0("k=", TopicNum)) # タイトル
}
```

・描画

```r
# トピック分布のパラメータ
fn_plotAlpha(alpha_dk)
```

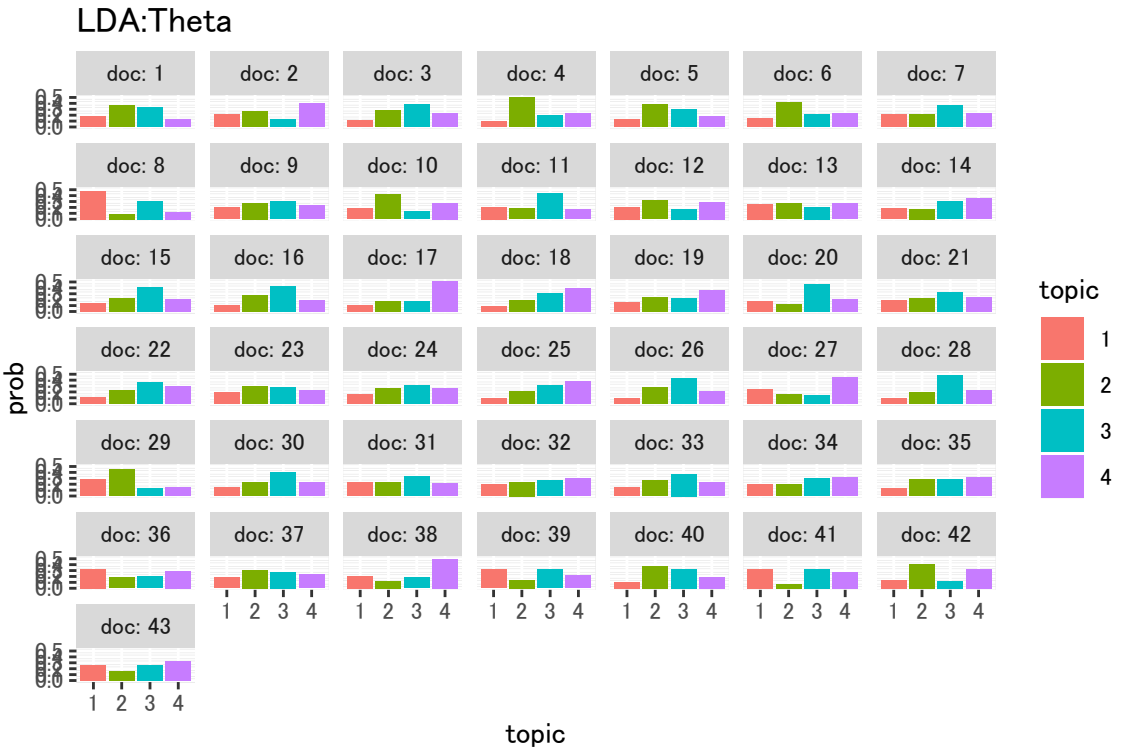



```
# 単語分布のパラメータ
fn_plotBeta(beta_kv)
```

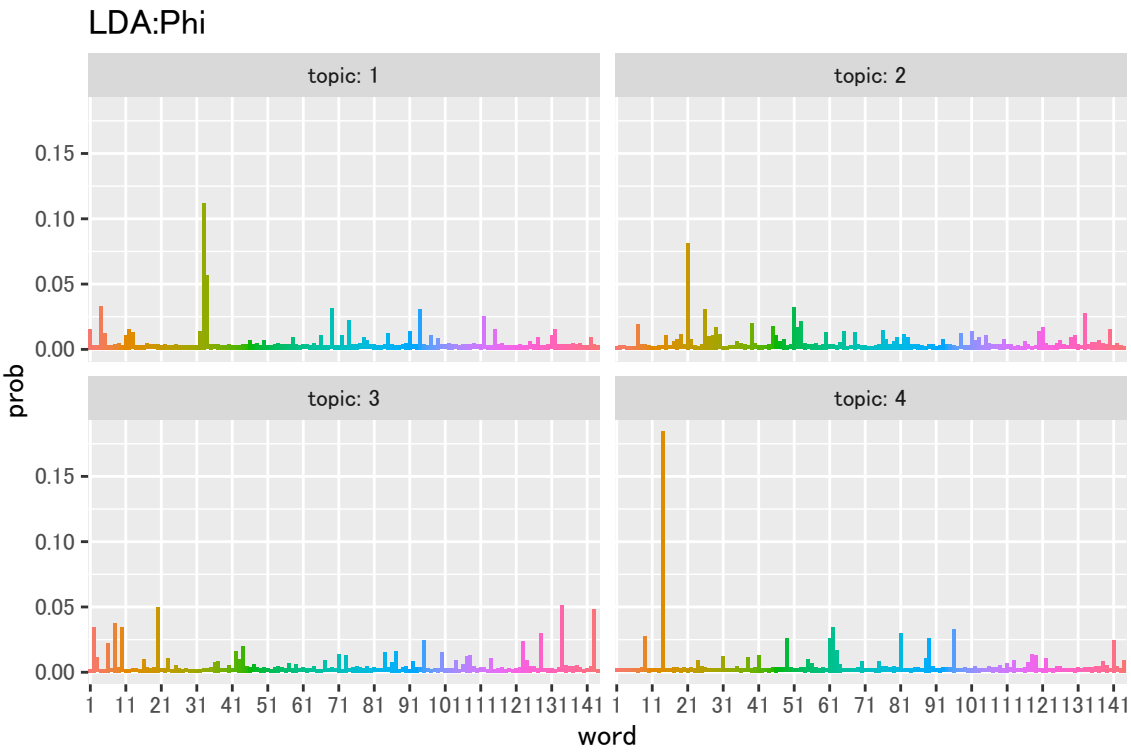



```
# トピック分布（平均値）
```

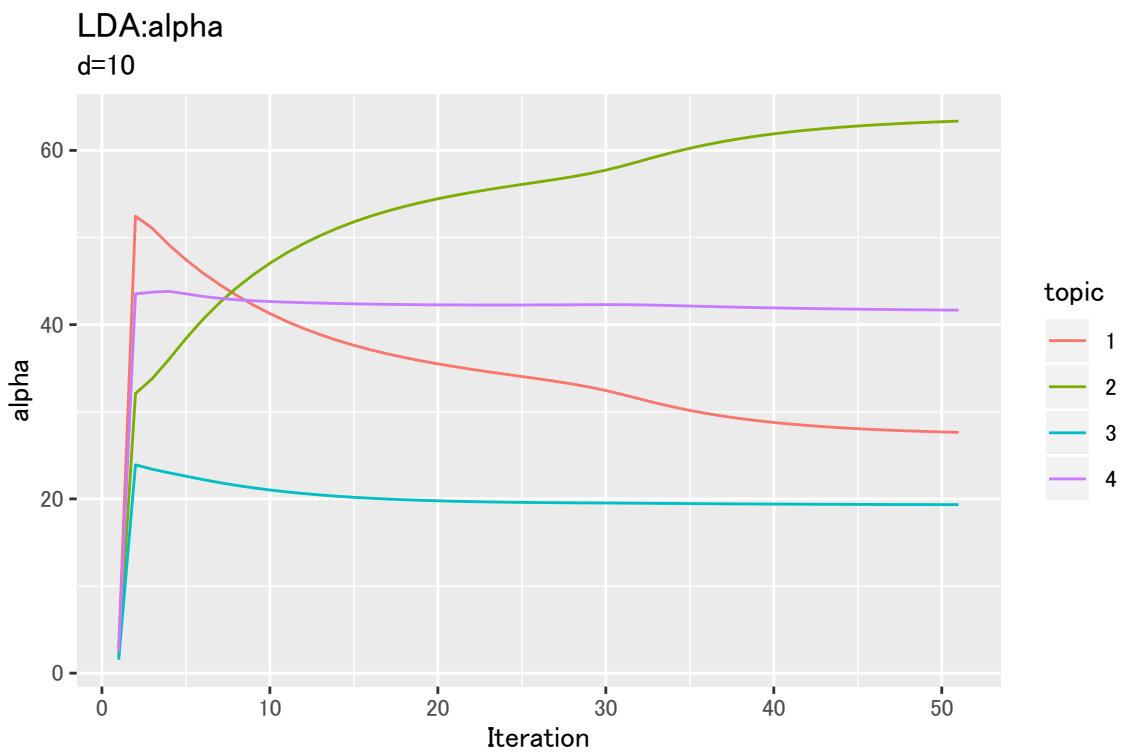
```
fn_plotTheta(alpha_dk)
```



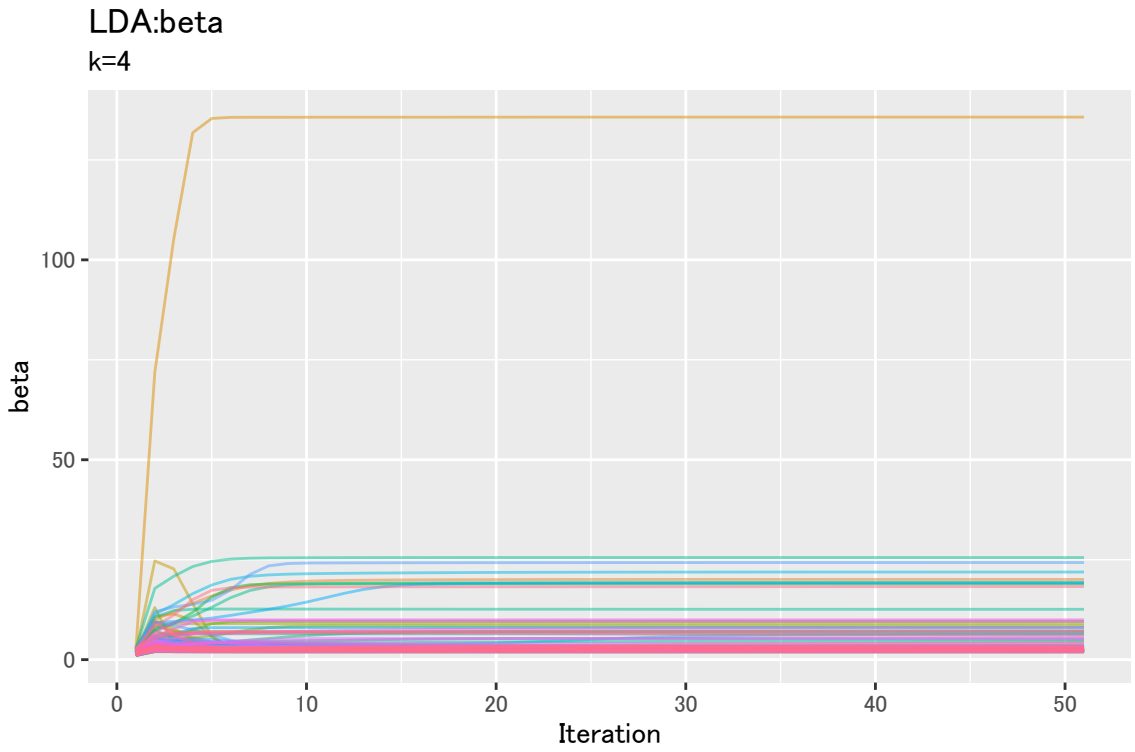

```
# 単語分布（平均値）
fn_plotPhi(beta_kv)
```



```
# トピック分布のパラメータの推移
fn_plotTraceAlpha(trace_alpha, DocNum = 10)
```




```
# 単語分布のパラメータの推移
fn_plotTraceBeta(trace_beta, TopicNum = 4)
```


LDA:beta
k=4
beta
100
50
0
0
10
20
30
40
50
Iteration