Ulta Text Analysis

Andrew Mashhadi

04 June, 2023

Load Libraries

```
library(dplyr)
library(tidytext)
library(ggplot2)
library(ggpubr)
library(tidyr)
library(stringr)
```

Analysis

Reading in the the full dataframe of webscraped data `full_df`.

```
## set data path
PATH_ULTA_TEXT_DIR <- Sys.getenv("ULTA_TEXT_DATA")
PATH_ULTA_PRODUCT_NAMES <-paste0(PATH_ULTA_TEXT_DIR, '/products/product_dict.csv')

## read in product names
product_names <- read.csv(PATH_ULTA_PRODUCT_NAMES)[, 2:3]

## get full paths to all data files
csv_files <- list.files(PATH_ULTA_TEXT_DIR, pattern = "csv$", full.names = T)
brand_names <- list.files(PATH_ULTA_TEXT_DIR, pattern = "csv$", full.names = F)

# initiallize df
full_df <- data.frame(matrix(ncol = 7, nrow = 0))
colnames(full_df) <- c('id', 'class', 'text', 'reviews',
                       'rating', 'price', 'brand')

## read in product names
for (i in 1:length(csv_files)) {

  tmp <- read.csv(csv_files[i])[, 2:7]

  tmp <- tmp %>%
    mutate(brand=str_replace(brand_names[i], ".csv", ""))

  full_df <- rbind(full_df, tmp)

}
```

```r
# remove duplicates (sometimes same product page woudl appear a few times)
full_df <- full_df %>%
  distinct()

full_df <- full_df %>%
  group_by(id) %>%
  mutate(rev_num=1:n()) %>%
  ungroup()


# join with product names found seperately
full_df <- full_df %>%
   inner_join(product_names %>%
                distinct())
```

```
## Joining, by = "id"
```

Create dataframe for each word in each review.

```r
ona_df_og <- full_df %>%
   na.omit() %>%
   filter(reviews != "")

tidy_revs <- ona_df_og %>%
   unnest_tokens(word, reviews) %>%
   filter(!grepl('[0-9]', word)) %>%
   mutate(word = str_remove_all(word, "[:punct:]")) %>%
   anti_join(stop_words)
```

```
## Joining, by = "word"
```

```r
tidy_revs %>%
  count(word, sort=T) %>%
  head(n=10)
```

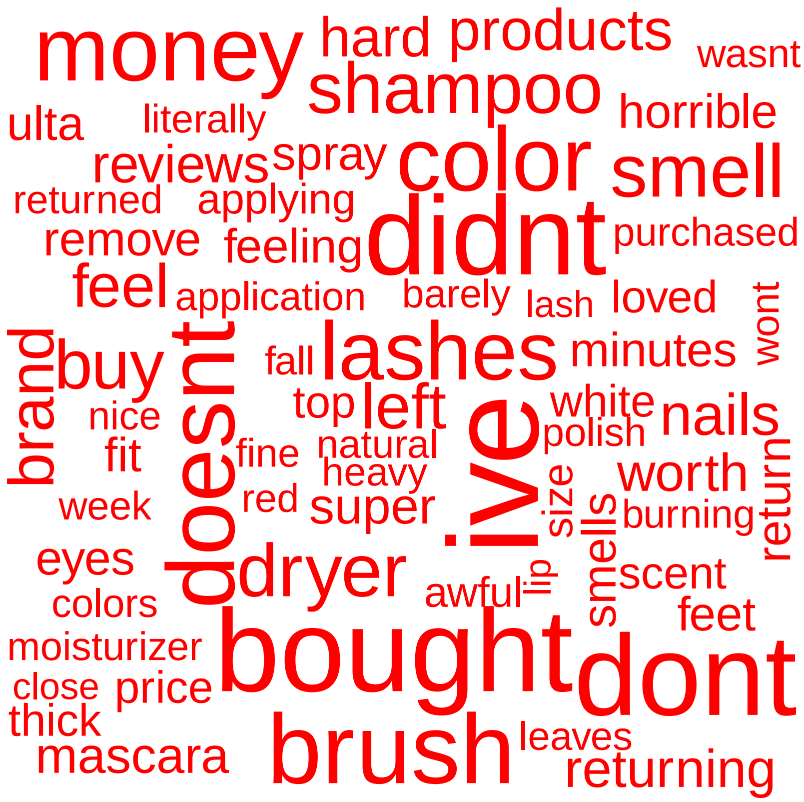
```
## # A tibble: 10 x 2
##    word        n
##    <chr>   <int>
##  1 hair    26666
##  2 product 19839
##  3 skin    18969
##  4 love    17996
##  5 ive      8449
##  6 im       7560
##  7 dry      7465
##  8 dont     6566
##  9 doesnt   5993
## 10 day      5885
```

We now build word cloud of words for positive reviews.

```r
# load library
library(wordcloud)

# set seed
set.seed(1128)

tidy_revs %>%
  filter(rating > 4.5) %>%
  filter(!(word %in% c("hair", "product"))) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100, colors = "blue"))
```

We now build word cloud of words for negative reviews.

```r
# set seed
set.seed(1128)

tidy_revs %>%
  filter(rating < 1.5) %>%
  filter(!(word %in% c("hair", "product"))) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100, colors = "red"))
```



Get sentiment per product. Join sentiment to main dataframe.

```
# get sentiments for each review
rev_sents <- tidy_revs %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(id, rev_num) %>%
  summarise(sentiment = sum(value))
```

```
## Joining, by = "word"
## `summarise()` has grouped output by 'id'. You can override using the `.groups`
## argument.
```

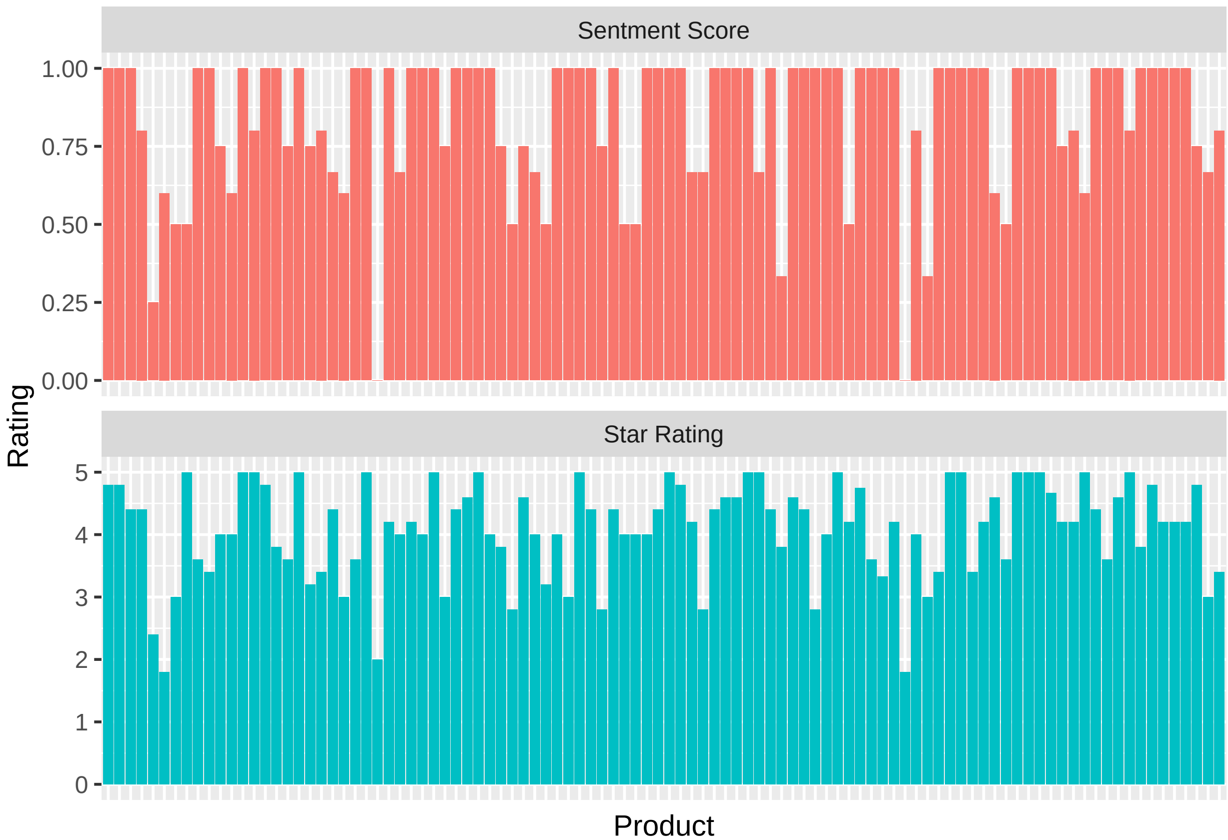
```
# treat each review equally (with weight) then get prop. positive
rev_sents <- rev_sents %>%
  filter(sentiment != 0) %>%
  mutate(sentiment=as.integer(sentiment > 0)) %>%
  group_by(id) %>%
  summarise(pos_sentiment=mean(sentiment))


# merge sentiments with reviews
ona_df <- ona_df_og %>%
  inner_join(rev_sents, by="id") %>%
  group_by(id) %>%
  slice(1) %>%
  ungroup() %>%
  select(-c(reviews, rev_num))
```

Correlation test between sentiment scores and star ratings. Sample 100 products and demonstrate the effectiveness of the sentiment analysis with a plot.

```
# correlation test
cor.test(ona_df$rating, ona_df$pos_sentiment)
```

```
##
##  Pearson's product-moment correlation
##
## data:  ona_df$rating and ona_df$pos_sentiment
## t = 55.625, df = 12969, p-value < 0.00000000000000022
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4248867 0.4526775
## sample estimates:
##       cor
## 0.4388871
```

```
# sentiment vs star rating evaluation figure
set.seed(19)
samp_df <- slice_sample(ona_df, n = 100)

bind_rows(samp_df %>%
            mutate(rating_type="Sentment Score",
                   rev_rating=pos_sentiment) %>%
            select(-c(rating, pos_sentiment, text)),
          samp_df %>%
            mutate(rating_type="Star Rating",
                   rev_rating=rating) %>%
            select(-c(rating, pos_sentiment, text))) %>%
  ggplot(aes(name, rev_rating, fill = rating_type)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~rating_type, ncol = 1, scales = "free_y") +
  theme(axis.text.x=element_blank(),
      axis.ticks.x=element_blank()) +
  xlab("Product") +
  ylab("Rating")
```



Using Sentiment and Rating with Description

All description words by sentiment and rating.

```
tidy_desc <- ona_df %>%
  filter(pos_sentiment != 0.5) %>%
  mutate(sentiment = case_when(
                          pos_sentiment < 0.5 ~ "negative",
                          pos_sentiment > 0.5 ~ "positive")) %>%
  select(-c(pos_sentiment)) %>%
  unnest_tokens(word, text) %>%
  filter(!grepl('[0-9]', word)) %>%
  mutate(word = str_remove_all(word, "[:punct:]")) %>%
  anti_join(stop_words)
```

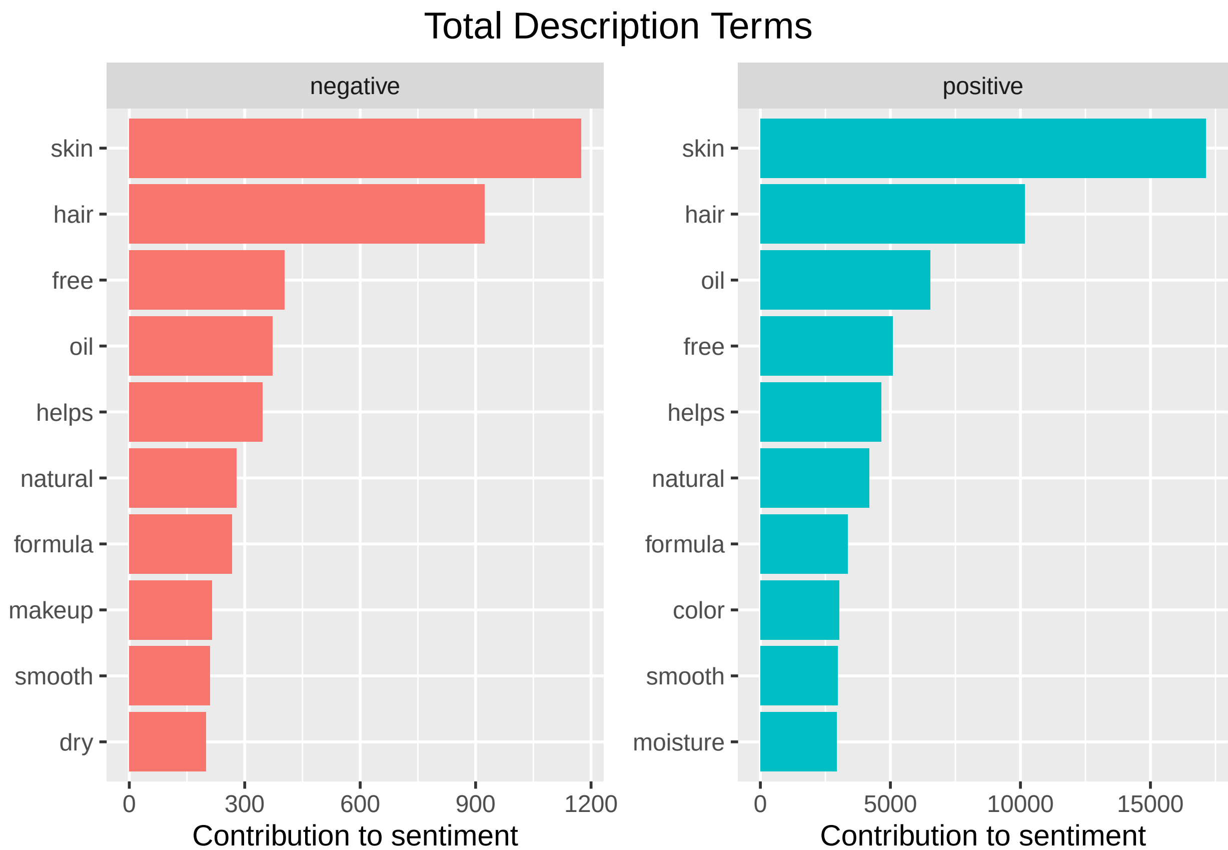
```
## Joining, by = "word"
```

```
tidy_desc %>%
  count(sentiment, word, sort = TRUE) %>%
  group_by(sentiment) %>%
  slice_max(n, n = 10) %>%
  mutate(word = reorder(word, n))
```

```
## # A tibble: 20 x 3
## # Groups:   sentiment [2]
##    sentiment word          n
##    <chr>     <fct>     <int>
```

```
##  1 negative  skin      1174
##  2 negative  hair       923
##  3 negative  free       403
##  4 negative  oil        372
##  5 negative  helps      347
##  6 negative  natural    279
##  7 negative  formula    267
##  8 negative  makeup     215
##  9 negative  smooth     210
## 10 negative  dry        200
## 11 positive  skin     17124
## 12 positive  hair     10165
## 13 positive  oil       6535
## 14 positive  free      5094
## 15 positive  helps     4645
## 16 positive  natural   4193
## 17 positive  formula   3355
## 18 positive  color     3036
## 19 positive  smooth    2975
## 20 positive  moisture  2936
```

```
A <- tidy_desc %>%
     count(sentiment, word, sort = TRUE) %>%
     ungroup() %>%
     filter(sentiment=="negative") %>%
     slice_max(n, n = 10) %>%
     mutate(word = reorder(word, n)) %>%
     ggplot(aes(n, word)) +
     geom_col(show.legend = FALSE, fill="#F8766D") +
     facet_wrap(~sentiment, scales = "free_y") +
     labs(x = "Contribution to sentiment",
          y = NULL)

B <- tidy_desc %>%
     count(sentiment, word, sort = TRUE) %>%
     ungroup() %>%
     filter(sentiment=="positive") %>%
     slice_max(n, n = 10) %>%
     mutate(word = reorder(word, n)) %>%
     ggplot(aes(n, word)) +
     geom_col(show.legend = FALSE, fill="#00BFC4") +
     facet_wrap(~sentiment, scales = "free_y") +
     labs(x = "Contribution to sentiment",
          y = NULL)

f <- ggarrange(A, B,
               labels = NULL,
               ncol = 2, nrow = 1)
annotate_figure(f,
               top = text_grob("Total Description Terms",
                                color = "Black",
                                size = 14))
```




```
#######################################################

rtidy_desc <- ona_df %>%
  mutate(rating = case_when(
                          rating >= 3.5 ~ "good",
                          rating < 3.5 ~  "bad")) %>%
  select(-c(pos_sentiment)) %>%
  unnest_tokens(word, text) %>%
  filter(!grepl('[0-9]', word)) %>%
  mutate(word = str_remove_all(word, "[:punct:]")) %>%
  anti_join(stop_words)
```

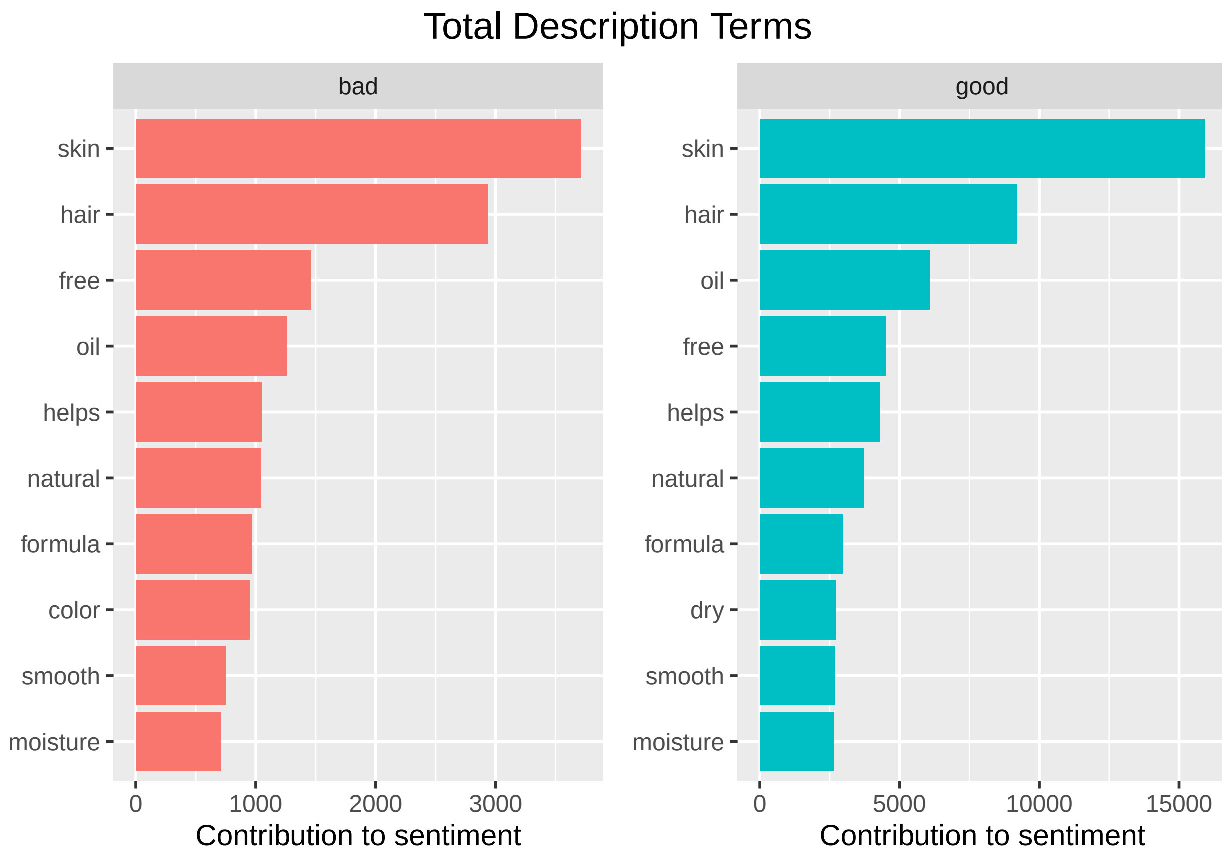
```
## Joining, by = "word"
```

```
rtidy_desc %>%
  count(rating, word, sort = TRUE) %>%
  group_by(rating) %>%
  slice_max(n, n = 10) %>%
  mutate(word = reorder(word, n))
```

```
## # A tibble: 20 x 3
## # Groups:   rating [2]
##    rating word       n
##    <chr>  <fct>  <int>
##  1 bad    skin    3711
##  2 bad    hair    2938
##  3 bad    free    1462
##  4 bad    oil     1255
```

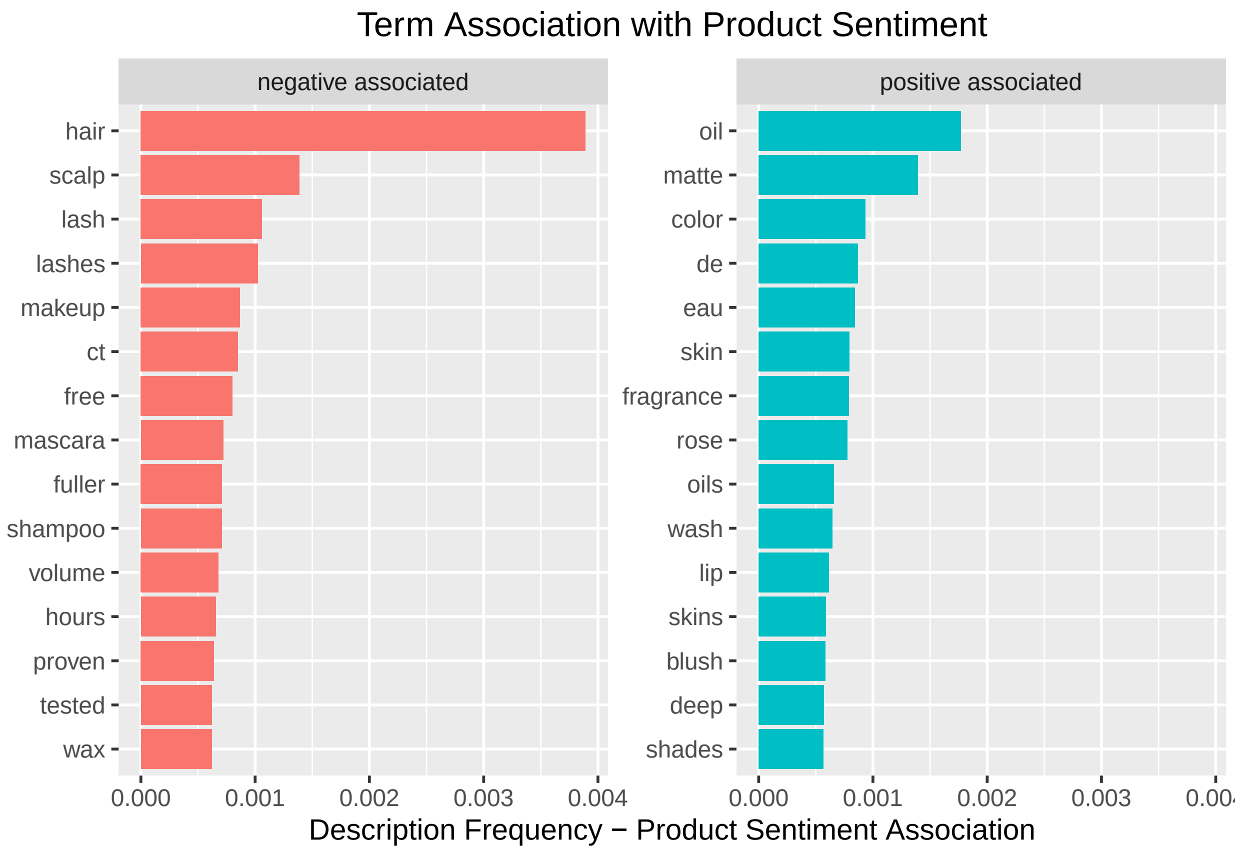
```
##   5 bad    helps     1047
##   6 bad    natural   1046
##   7 bad    formula    965
##   8 bad    color      950
##   9 bad    smooth     747
##  10 bad    moisture   707
##  11 good   skin     15928
##  12 good   hair      9190
##  13 good   oil       6076
##  14 good   free      4497
##  15 good   helps     4297
##  16 good   natural   3733
##  17 good   formula   2955
##  18 good   dry       2726
##  19 good   smooth    2701
##  20 good   moisture  2653
```

```r
A <- rtidy_desc %>%
     count(rating, word, sort = TRUE) %>%
     ungroup() %>%
     filter(rating=="bad") %>%
     slice_max(n, n = 10) %>%
     mutate(word = reorder(word, n)) %>%
     ggplot(aes(n, word)) +
     geom_col(show.legend = FALSE, fill="#F8766D") +
     facet_wrap(~rating, scales = "free_y") +
     labs(x = "Contribution to sentiment",
          y = NULL)

B <- rtidy_desc %>%
     count(rating, word, sort = TRUE) %>%
     ungroup() %>%
     filter(rating=="good") %>%
     slice_max(n, n = 10) %>%
     mutate(word = reorder(word, n)) %>%
     ggplot(aes(n, word)) +
     geom_col(show.legend = FALSE, fill="#00BFC4") +
     facet_wrap(~rating, scales = "free_y") +
     labs(x = "Contribution to sentiment",
          y = NULL)

f <- ggarrange(A, B,
               labels = NULL,
               ncol = 2, nrow = 1)
annotate_figure(f,
               top = text_grob("Total Description Terms",
                               color = "Black",
                               size = 14))
```

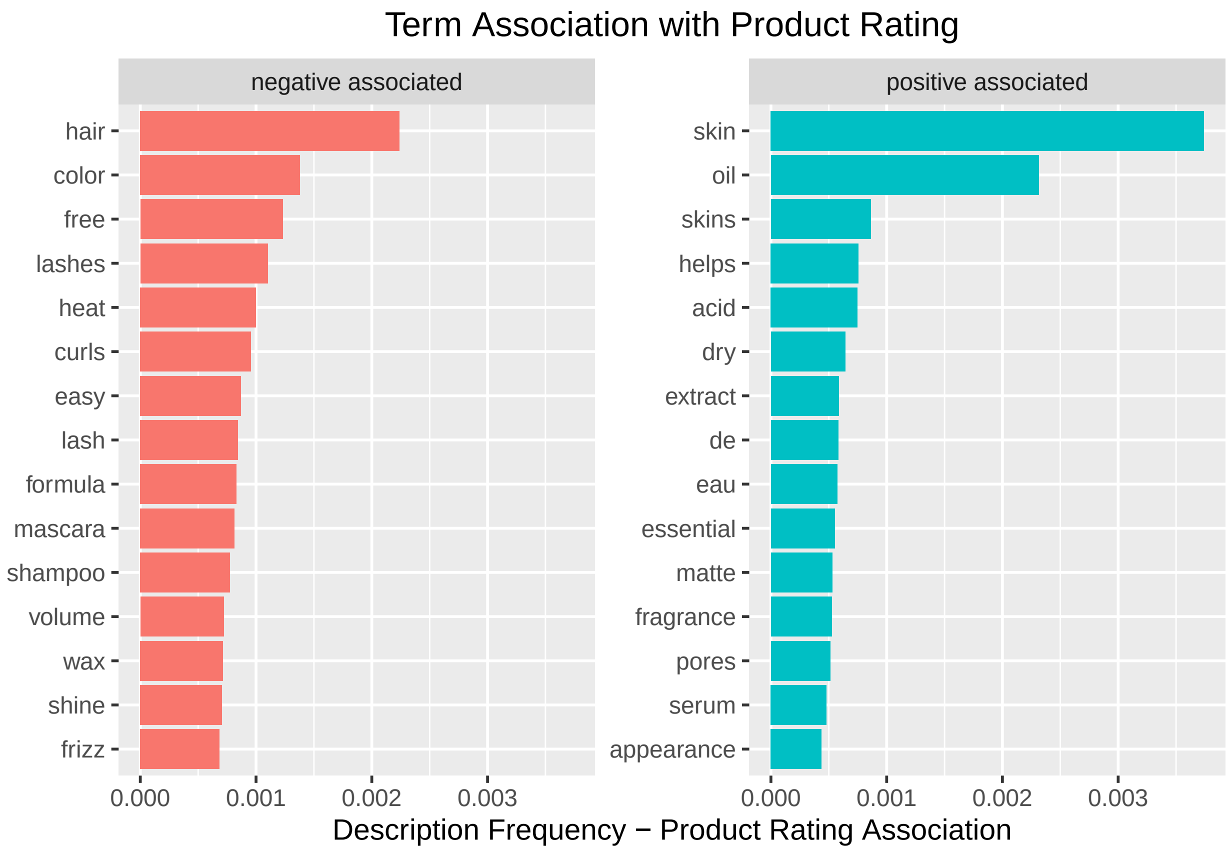


Obtain descriptive term associations with product sentiment (or rating).

```
tidy_desc %>%
  count(sentiment, word, sort = TRUE) %>%
  ungroup() %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(positive=positive/sum(positive), negative=negative/sum(negative)) %>%
  mutate(diff = positive-negative) %>%
  arrange(desc(diff)) %>%
  filter(diff!=0) %>%
  mutate(emphasis = case_when(
                        diff > 0 ~ "positive associated",
                        diff < 0 ~ "negative associated"),
         diff=abs(diff)) %>%
  group_by(emphasis) %>%
  slice_max(diff, n=15) %>%
  select(-c(positive, negative)) %>%
  ungroup() %>%
  mutate(word = reorder(word, diff)) %>%
  ggplot(aes(diff, word, fill = emphasis)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~emphasis, scales = "free_y") +
  labs(x = "Description Frequency - Product Sentiment Association",
       y = NULL) +
  ggtitle("Term Association with Product Sentiment") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
###################################################


rtidy_desc %>%
  count(rating, word, sort = TRUE) %>%
  ungroup() %>%
  pivot_wider(names_from = rating, values_from = n, values_fill = 0) %>%
  mutate(good=good/sum(good), bad=bad/sum(bad)) %>%
  mutate(diff = good-bad) %>%
  arrange(desc(diff)) %>%
  filter(diff!=0) %>%
  mutate(emphasis = case_when(
                        diff > 0 ~ "positive associated",
                        diff < 0 ~ "negative associated"),
         diff=abs(diff)) %>%
  group_by(emphasis) %>%
  slice_max(diff, n=15) %>%
  select(-c(good, bad)) %>%
  ungroup() %>%
  mutate(word = reorder(word, diff)) %>%
  ggplot(aes(diff, word, fill = emphasis)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~emphasis, scales = "free_y") +
  labs(x = "Description Frequency - Product Rating Association",
       y = NULL) +
  ggtitle("Term Association with Product Rating") +
  theme(plot.title = element_text(hjust = 0.5))
```



Looks like hair, scalp, eye make-up lean more negative, while face, skin, lips, and fragrance lean more positive.

Associated Wordcloud

```
set.seed(7)
library(wordcloud)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
rtidy_desc  %>%
  count(word, rating, sort = TRUE) %>%
  acast(word ~ rating, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("red", "blue"),
                   max.words = 100)
```

Best rating and sentiment brand analysis. Reporting average price of top 10.

```r
# set hyper-parameter
alpha <- 0.6

# merge sentiments with reviews, keep number of reviews first
ona_df <- ona_df_og %>%
  inner_join(rev_sents, by="id") %>%
  group_by(id) %>%
  slice_max(rev_num) %>%
  ungroup() %>%
  select(-c(reviews))

# by brand
brand_desc <- ona_df %>%
  select(c(text, rating, price, brand, pos_sentiment, rev_num)) %>%
  group_by(brand) %>%
  summarise(text=paste0(text, collapse = " "),
            rating=mean(rating),
            price=mean(price),
            pos_sentiment=mean(pos_sentiment),
            rev_num=sum(rev_num))

# top 10 sentiment & rated brands
top10 <- brand_desc %>%
```

```r
  filter(rev_num > 25) %>%
  select(-c(text)) %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  arrange(desc(mixed_rating), rev_num) %>%
  head(n=10)

cat("Mean price of brands with top 10 highest rating / sentiment: $",
    mean(top10$price), "\n")
```

```
## Mean price of brands with top 10 highest rating / sentiment: $ 61.15955
```

```r
cat("Mean price of all products: $", mean(ona_df$price), "\n")
```

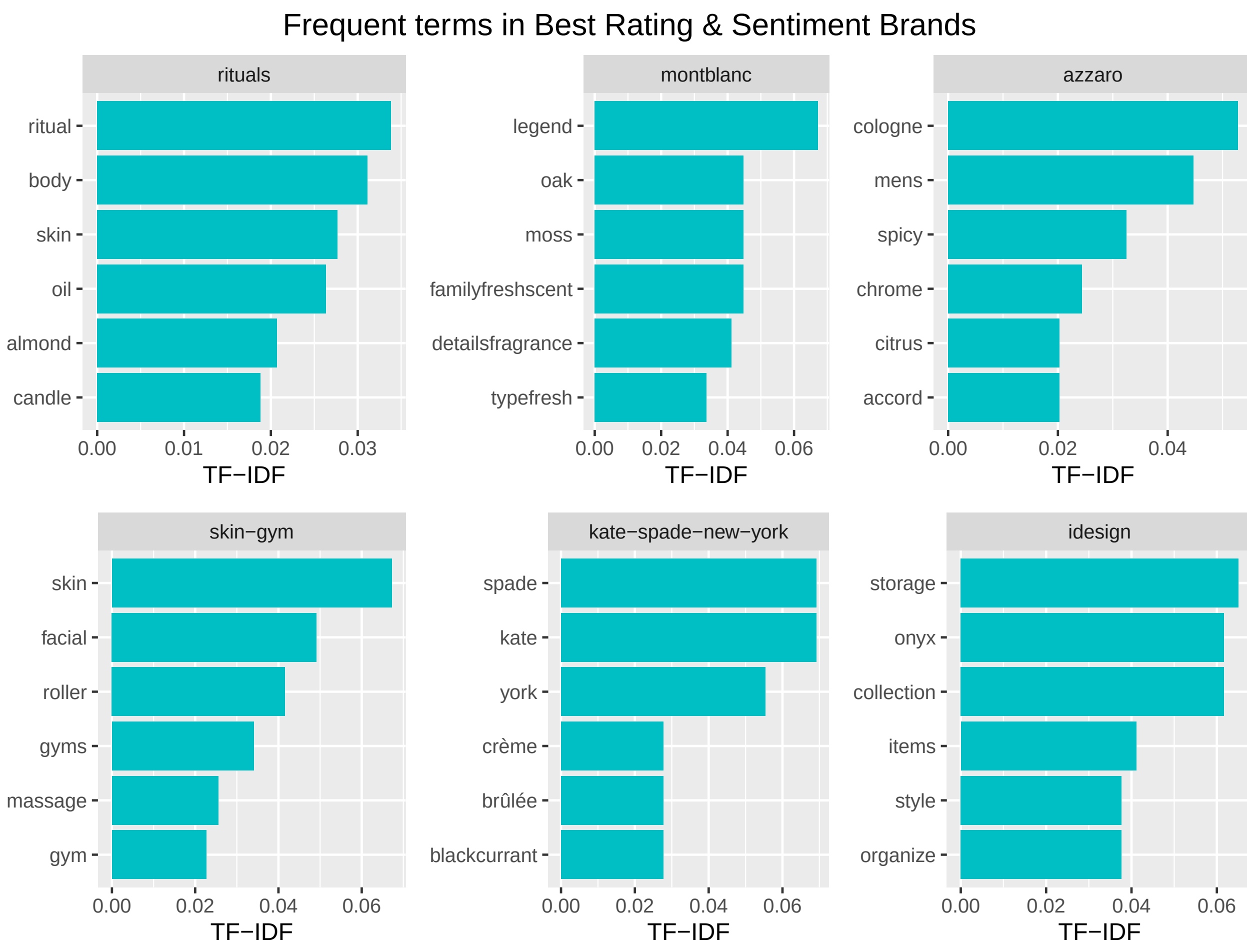
```
## Mean price of all products: $ 29.64213
```

```r
# by word
tidy_brand <- brand_desc %>%
  filter(rev_num > 25) %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  arrange(desc(mixed_rating), rev_num) %>%
  head(n=6) %>%
  unnest_tokens(word, text) %>%
  filter(!grepl('[0-9]', word)) %>%
  mutate(word = str_remove_all(word, "[:punct:]")) %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

```r
L <- list()
brands <- c("rituals", "montblanc", "azzaro", "skin-gym",
            "kate-spade-new-york", "idesign")
for (i in 1:length(brands)) {

  L[[i]] <- tidy_brand %>%
            count(brand, word, sort = TRUE) %>%
            bind_tf_idf(word, brand, n) %>%
            filter(brand==brands[i]) %>%
            filter(!(word %in% brands)) %>%
            group_by(brand) %>%
            slice_max(tf_idf, n=6, with_ties=FALSE) %>%
            mutate(word=reorder(word, tf_idf)) %>%
            ggplot(aes(tf_idf, word)) +
            geom_col(show.legend = FALSE, fill="#00BFC4") +
            facet_wrap(~brand, scales = "free_y") +
            labs(x = "TF-IDF",
                 y = NULL)
}

f <- ggarrange(plotlist=L,
               labels = NULL,
               ncol = 3, nrow = 2)

annotate_figure(f,
               top = text_grob(paste("Frequent terms in Best Rating &",
                                     "Sentiment Brands"),
                               color = "Black",
```

```
                  size = 14))
```



Lowest rating and sentiment brand analysis. Reporting average price of worst 10.

```
# top 10 sentiment & rated brands
top10 <- brand_desc %>%
  filter(rev_num > 25) %>%
  select(-c(text)) %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  arrange(mixed_rating, rev_num) %>%
  head(n=10)

cat("Mean price of brands with lowest 10 worst rating / sentiment: $",
    mean(top10$price), "\n")
```

```
## Mean price of brands with lowest 10 worst rating / sentiment: $ 15.60673
```

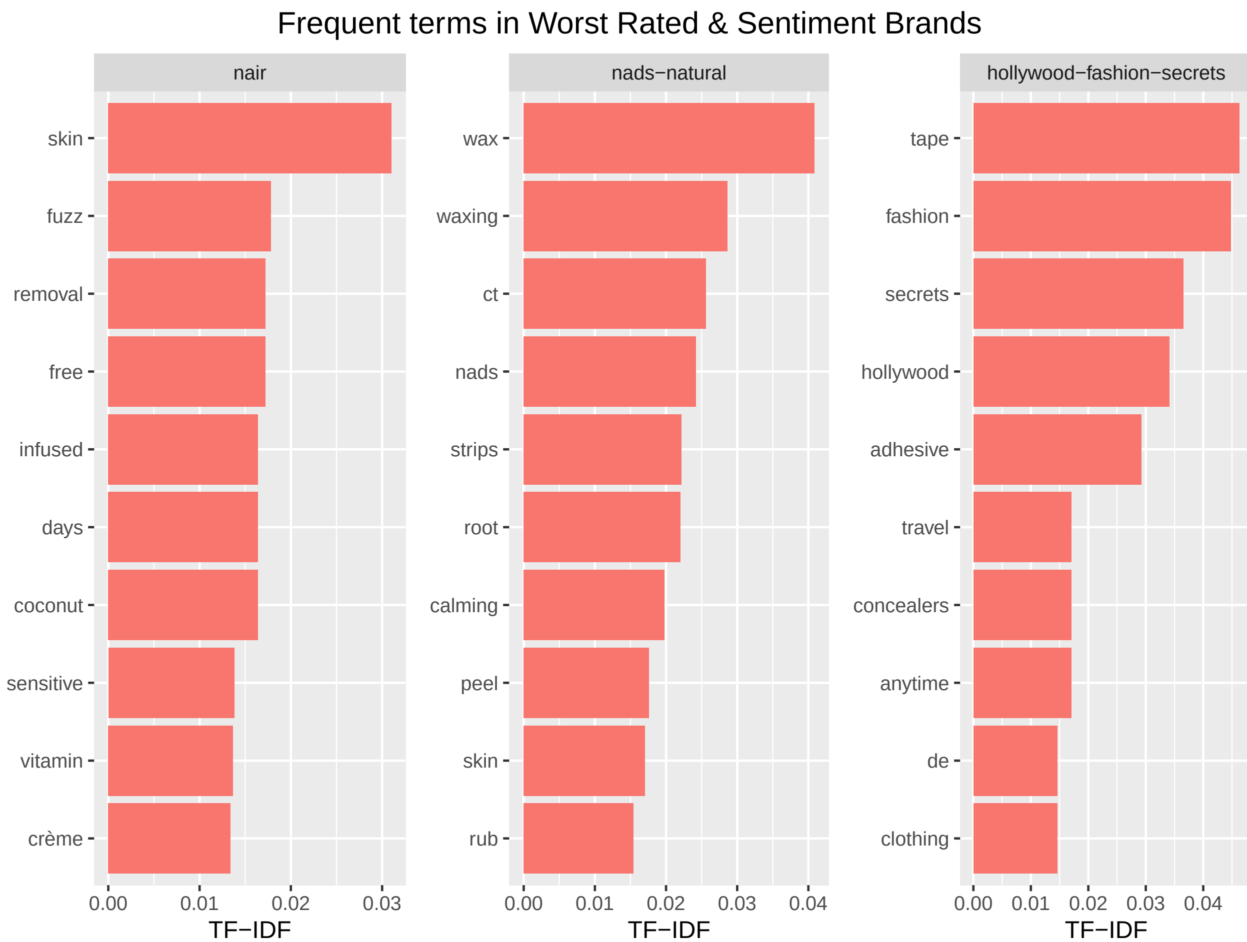
```
cat("Mean price of all products: $", mean(ona_df$price), "\n")
```

```
## Mean price of all products: $ 29.64213
```

```r
# by word
tidy_brand <- brand_desc %>%
  filter(rev_num > 25) %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  arrange(mixed_rating, rev_num) %>%
  head(n=6) %>%
  unnest_tokens(word, text) %>%
  filter(!grepl('[0-9]', word)) %>%
  mutate(word = str_remove_all(word, "[:punct:]")) %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

```r
L <- list()
brands <- c("nair", "nads-natural", "hollywood-fashion-secrets") #,
            #"punky-colour", "pravana", "invisibobble")
for (i in 1:length(brands)) {

  L[[i]] <- tidy_brand %>%
            count(brand, word, sort = TRUE) %>%
            bind_tf_idf(word, brand, n) %>%
            filter(brand==brands[i]) %>%
            filter(!(word %in% brands)) %>%
            group_by(brand) %>%
            slice_max(tf_idf, n=10, with_ties=FALSE) %>%
            mutate(word=reorder(word, tf_idf)) %>%
            ggplot(aes(tf_idf, word)) +
            geom_col(show.legend = FALSE, fill="#F8766D") +
            facet_wrap(~brand, scales = "free_y") +
            labs(x = "TF-IDF",
                 y = NULL)
}

f <- ggarrange(plotlist=L,
               labels = NULL,
               ncol = 3, nrow = 1)

annotate_figure(f,
               top = text_grob(paste("Frequent terms in Worst Rated &",
                                     "Sentiment Brands"),
                               color = "Black",
                               size = 14))
```



Best rating and sentiment product category analysis. Reporting average price of best 10.

```
# set hyper-parameter
alpha <- 0.6

# by product category
product_cats <- ona_df %>%
  select(c(text, rating, price, class, pos_sentiment, rev_num)) %>%
  group_by(class) %>%
  summarise(text=paste0(text, collapse = " "),
            rating=mean(rating),
            price=mean(price),
            pos_sentiment=mean(pos_sentiment),
            rev_num=sum(rev_num))

# top 10 sentiment & rated product classes
top10 <- product_cats %>%
  filter(rev_num > 30) %>%
  select(-c(text)) %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  arrange(desc(mixed_rating), rev_num) %>%
  head(n=10)
```

```r
cat("Mean price of product categories with top 10 highest rating / sentiment: $",
    mean(top10$price), "\n")
```

```
## Mean price of product categories with top 10 highest rating / sentiment: $ 52.41063
```

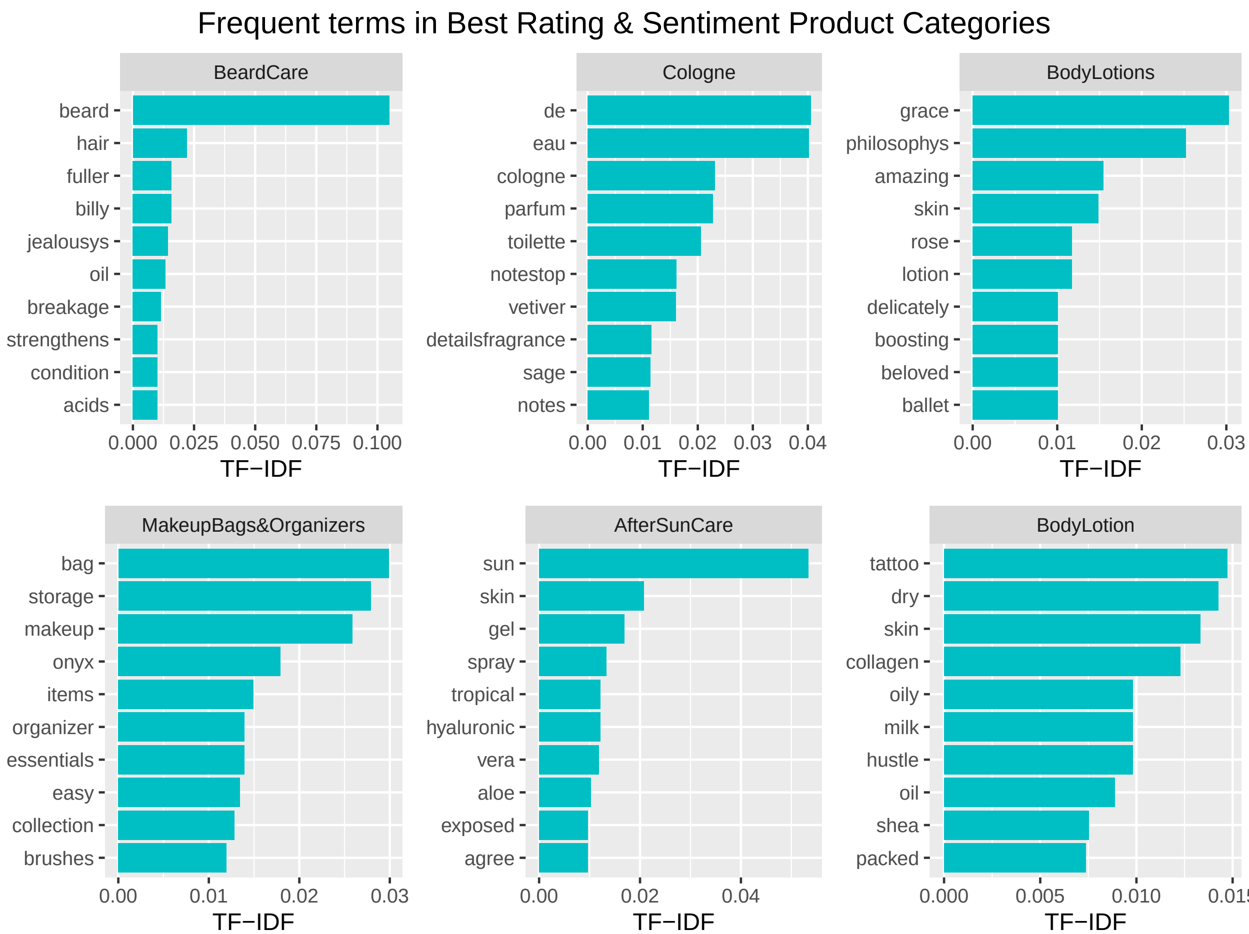
```r
cat("Mean price of all products: $", mean(ona_df$price), "\n")
```

```
## Mean price of all products: $ 29.64213
```

```r
# by word
tidy_product_cat <- product_cats %>%
  filter(rev_num > 30) %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  arrange(desc(mixed_rating), rev_num) %>%
  head(n=6) %>%
  unnest_tokens(word, text) %>%
  filter(!grepl('[0-9]', word)) %>%
  mutate(word = str_remove_all(word, "[:punct:]")) %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

```r
L <- list()
pclasses <- c("BeardCare", "Cologne", "BodyLotions", "MakeupBags&Organizers",
            "AfterSunCare", "BodyLotion")
for (i in 1:length(pclasses)) {

  L[[i]] <- tidy_product_cat %>%
            count(class, word, sort = TRUE) %>%
            bind_tf_idf(word, class, n) %>%
            filter(class==pclasses[i]) %>%
            group_by(class) %>%
            slice_max(tf_idf, n=10, with_ties=FALSE) %>%
            mutate(word=reorder(word, tf_idf)) %>%
            ggplot(aes(tf_idf, word)) +
            geom_col(show.legend = FALSE, fill="#00BFC4") +
            facet_wrap(~class, scales = "free_y") +
            labs(x = "TF-IDF",
                 y = NULL)

}

f <- ggarrange(plotlist=L,
               labels = NULL,
               ncol = 3, nrow = 2)

annotate_figure(f,
               top = text_grob(paste("Frequent terms in Best Rating & Sentiment",
                                     "Product Categories"),
                               color = "Black",
                               size = 14))
```



Lowest rating and sentiment product category analysis. Reporting average price of worst 10.

```r
# set hyper-parameter
alpha <- 0.6

# top 10 sentiment & rated product classes
top10 <- product_cats %>%
  filter(rev_num > 30) %>%
  select(-c(text)) %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  arrange(mixed_rating, rev_num) %>%
  head(n=10)

cat("Mean price of product categories with lowest 10 worst rating / sentiment: $",
    mean(top10$price), "\n")
```

```
## Mean price of product categories with lowest 10 worst rating / sentiment: $ 24.78093
```

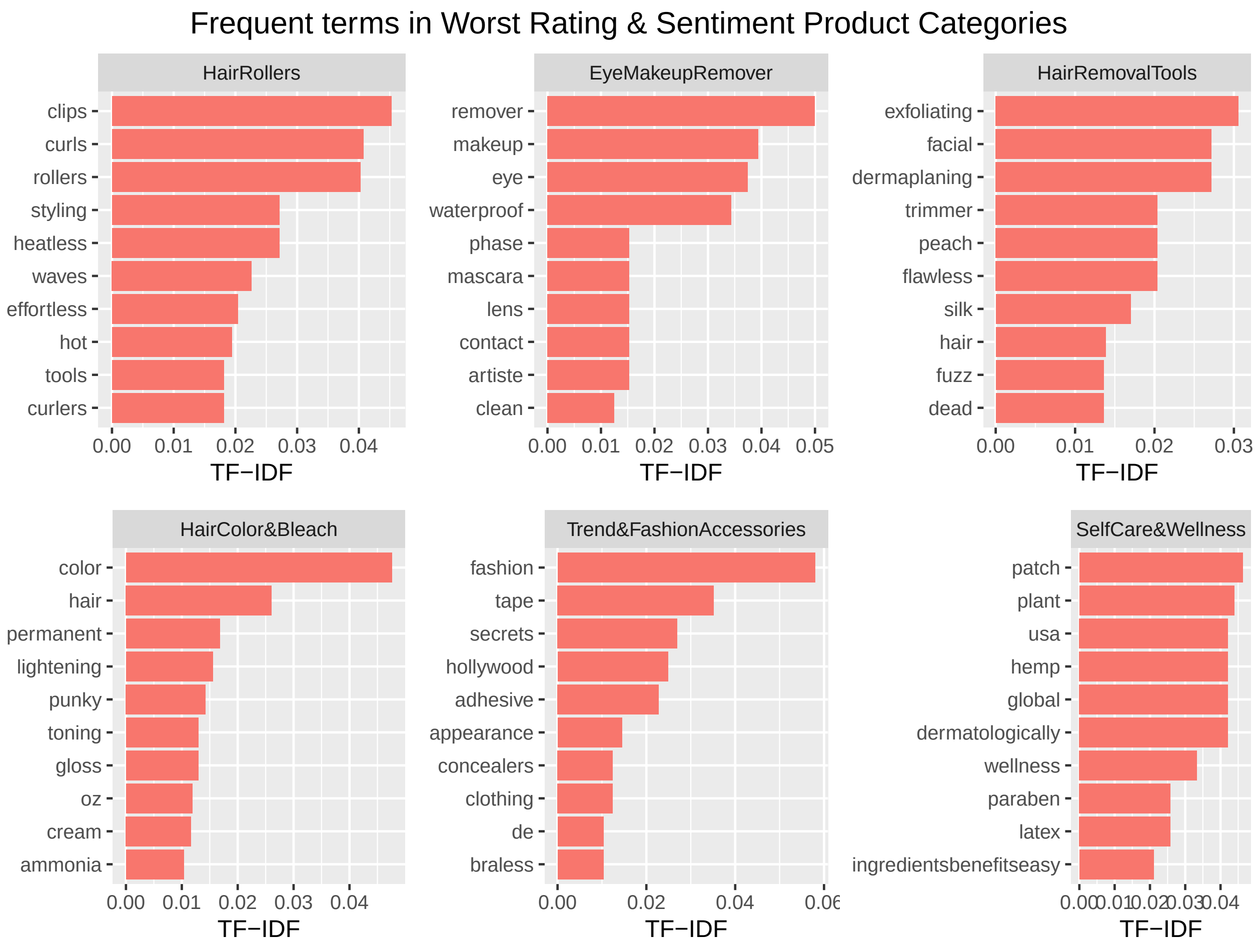
```r
cat("Mean price of all products: $", mean(ona_df$price), "\n")
```

```
## Mean price of all products: $ 29.64213
```

```r
# by word
tidy_product_cat <- product_cats %>%
  filter(rev_num > 30) %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  arrange(mixed_rating, rev_num) %>%
  head(n=6) %>%
  unnest_tokens(word, text) %>%
  filter(!grepl('[0-9]', word)) %>%
  mutate(word = str_remove_all(word, "[:punct:]")) %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

```r
L <- list()
pclasses <- c("HairRollers", "EyeMakeupRemover", "HairRemovalTools",
              "HairColor&Bleach", "Trend&FashionAccessories", "SelfCare&Wellness")
for (i in 1:length(pclasses)) {


  L[[i]] <- tidy_product_cat %>%
            count(class, word, sort = TRUE) %>%
            bind_tf_idf(word, class, n) %>%
            filter(class==pclasses[i]) %>%
            group_by(class) %>%
            slice_max(tf_idf, n=10, with_ties=FALSE) %>%
            mutate(word=reorder(word, tf_idf)) %>%
            ggplot(aes(tf_idf, word)) +
            geom_col(show.legend = FALSE, fill="#F8766D") +
            facet_wrap(~class, scales = "free_y") +
            labs(x = "TF-IDF",
                 y = NULL)
}

f <- ggarrange(plotlist=L,
               labels = NULL,
               ncol = 3, nrow = 2)

annotate_figure(f,
               top = text_grob(paste("Frequent terms in Worst Rating &",
                                     "Sentiment Product Categories"),
                               color = "Black",
                               size = 14))
```



Assess best brand descriptions within specific product classes. Use categories, shampoo, face-moisturizer, and mascara.

```
product_brands <- ona_df %>%
  filter(class=="Shampoo" | class=="FaceMoisturizer" | class=="Mascara") %>%
  select(c(text, rating, price, class, brand, pos_sentiment, rev_num)) %>%
  group_by(class, brand) %>%
  summarise(text=paste0(text, collapse = " "),
            rating=mean(rating),
            price=mean(price),
            pos_sentiment=mean(pos_sentiment),
            rev_num=sum(rev_num)) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'class'. You can override using the
## `.groups` argument.
```

```
# print list of top brands for each group
product_brands %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  group_by(class) %>%
  slice_max(mixed_rating, n=4, with_ties = FALSE) %>%
```

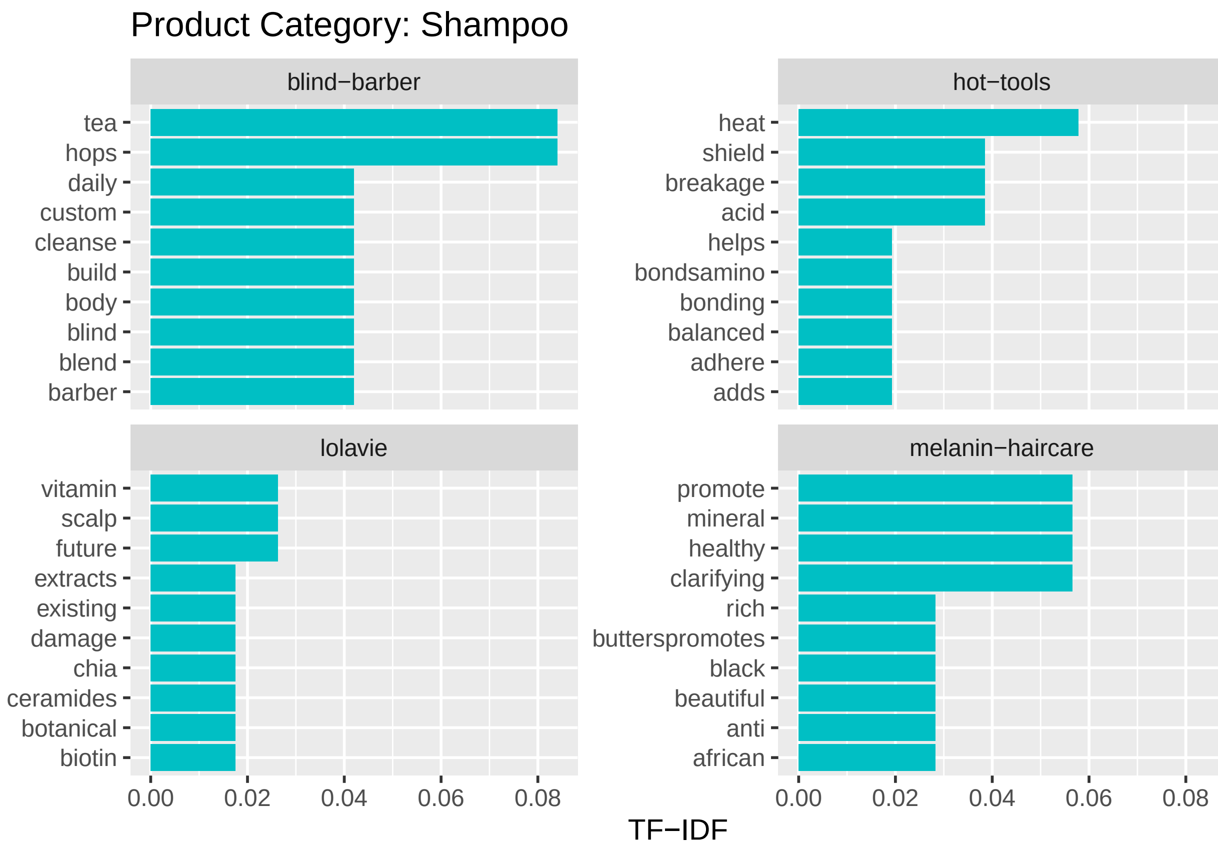
```r
  select(class, brand) %>%
  print(n=12)
```

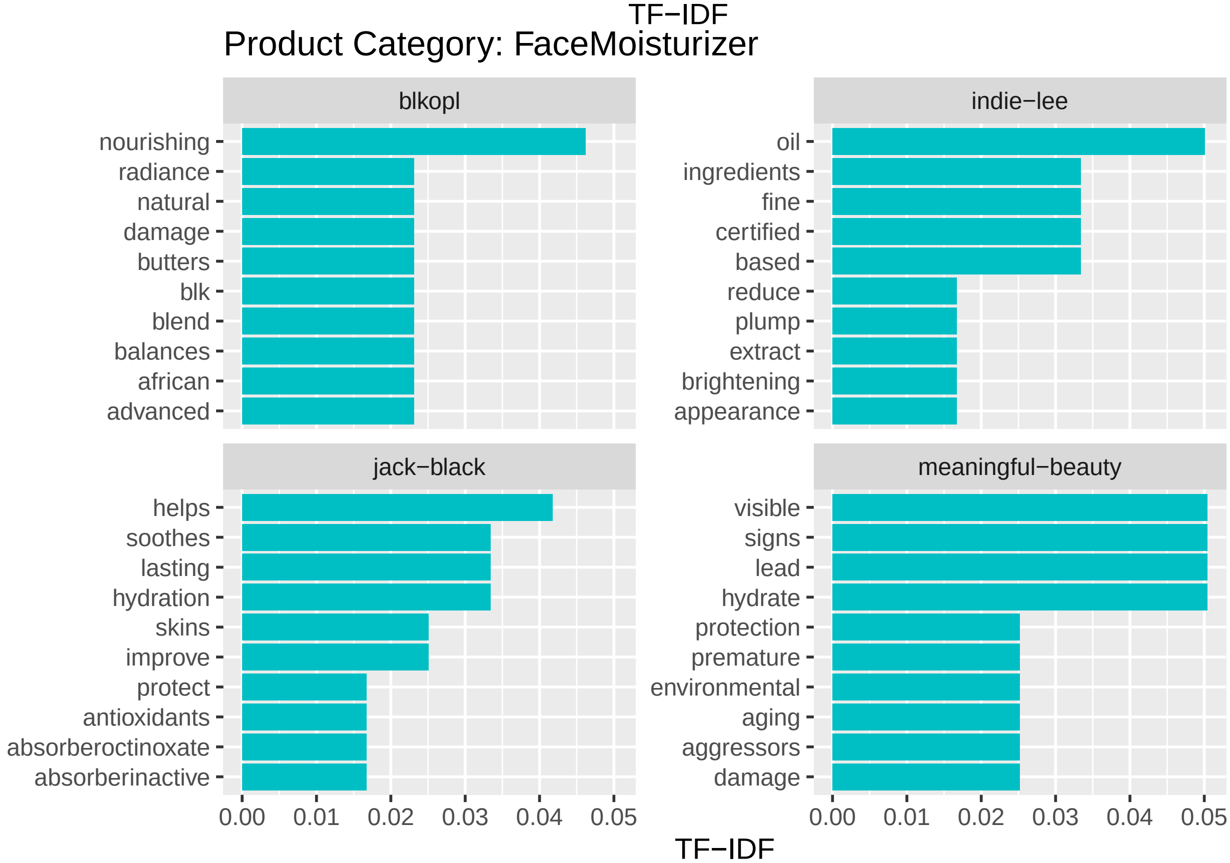
```
## # A tibble: 12 x 2
## # Groups:   class [3]
##    class           brand
##    <chr>           <chr>
##  1 FaceMoisturizer blkopl
##  2 FaceMoisturizer indie-lee
##  3 FaceMoisturizer jack-black
##  4 FaceMoisturizer meaningful-beauty
##  5 Mascara         elf-cosmetics
##  6 Mascara         pixi
##  7 Mascara         uoma-beauty
##  8 Mascara         winky-lux
##  9 Shampoo         blind-barber
## 10 Shampoo         hot-tools
## 11 Shampoo         lolavie
## 12 Shampoo         melanin-haircare
```

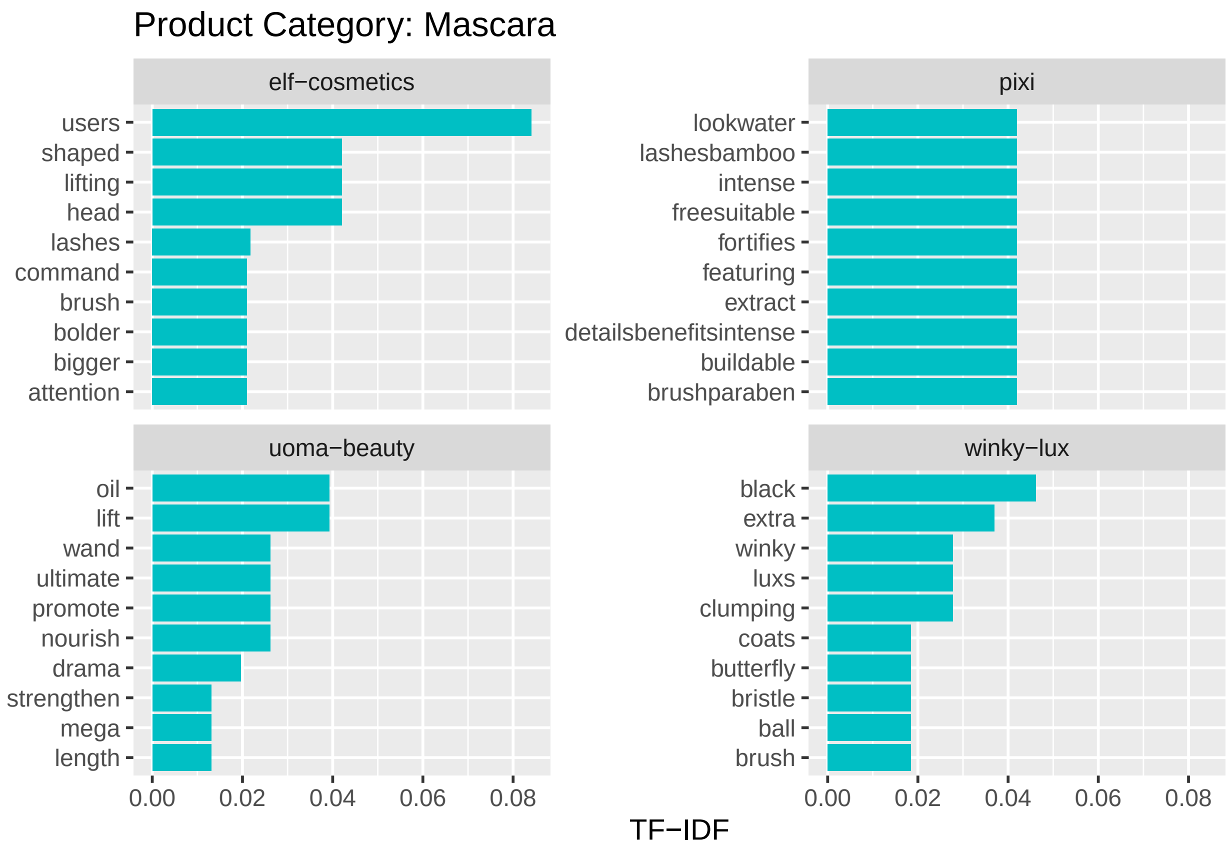
```r
tidy_brands <- product_brands %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  group_by(class) %>%
  slice_max(mixed_rating, n=4, with_ties = FALSE) %>%
  group_by(class, brand) %>%
  unnest_tokens(word, text) %>%
  filter(!grepl('[0-9]', word)) %>%
  mutate(word = str_remove_all(word, "[:punct:]")) %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

```r
pclasses <- c("Shampoo", "FaceMoisturizer", "Mascara")

for (i in 1:length(pclasses)) {

  print(tidy_brands %>%
    filter(class==pclasses[i]) %>%
    count(brand, word, sort = TRUE) %>%
    bind_tf_idf(word, brand, n) %>%
    group_by(brand) %>%
    slice_max(tf_idf, n=10, with_ties=FALSE) %>%
    mutate(word=reorder(word, tf_idf)) %>%
    ggplot(aes(tf_idf, word)) +
    geom_col(show.legend = FALSE, fill="#00BFC4") +
    facet_wrap(~brand, scales = "free_y") +
    labs(x = "TF-IDF",
         y = NULL,
         title = paste("Product Category:", pclasses[i])))


}
```


Product Category: Shampoo
blind−barber
tea
hops
daily
custom
cleanse
build
body
blind
blend
barber
hot−tools
heat
shield
breakage
acid
helps
bondsamino
bonding
balanced
adhere
adds
lolavie
vitamin
scalp
future
extracts
existing
damage
chia
ceramides
botanical
biotin
melanin−haircare
promote
mineral
healthy
clarifying
rich
butterspromotes
black
beautiful
anti
african
0.00
0.02
0.04
0.06
0.08
TF−IDF


Product Category: FaceMoisturizer
blkopl
nourishing
radiance
natural
damage
butters
blk
blend
balances
african
advanced
indie−lee
oil
ingredients
fine
certified
based
reduce
plump
extract
brightening
appearance
jack−black
helps
soothes
lasting
hydration
skins
improve
protect
antioxidants
absorberoctinoxate
absorberinactive
meaningful−beauty
visible
signs
lead
hydrate
protection
premature
environmental
aging
aggressors
damage
0.00
0.01
0.02
0.03
0.04
0.05
TF−IDF



Assess worst rated brand descriptions within specific product classes. Use categories, shampoo, face-moisturizer, and mascara.

```
# print list of top brands for each group
product_brands %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  group_by(class) %>%
  slice_min(mixed_rating, n=4, with_ties = FALSE) %>%
  select(class, brand, mixed_rating) %>%
  print(n=12)
```

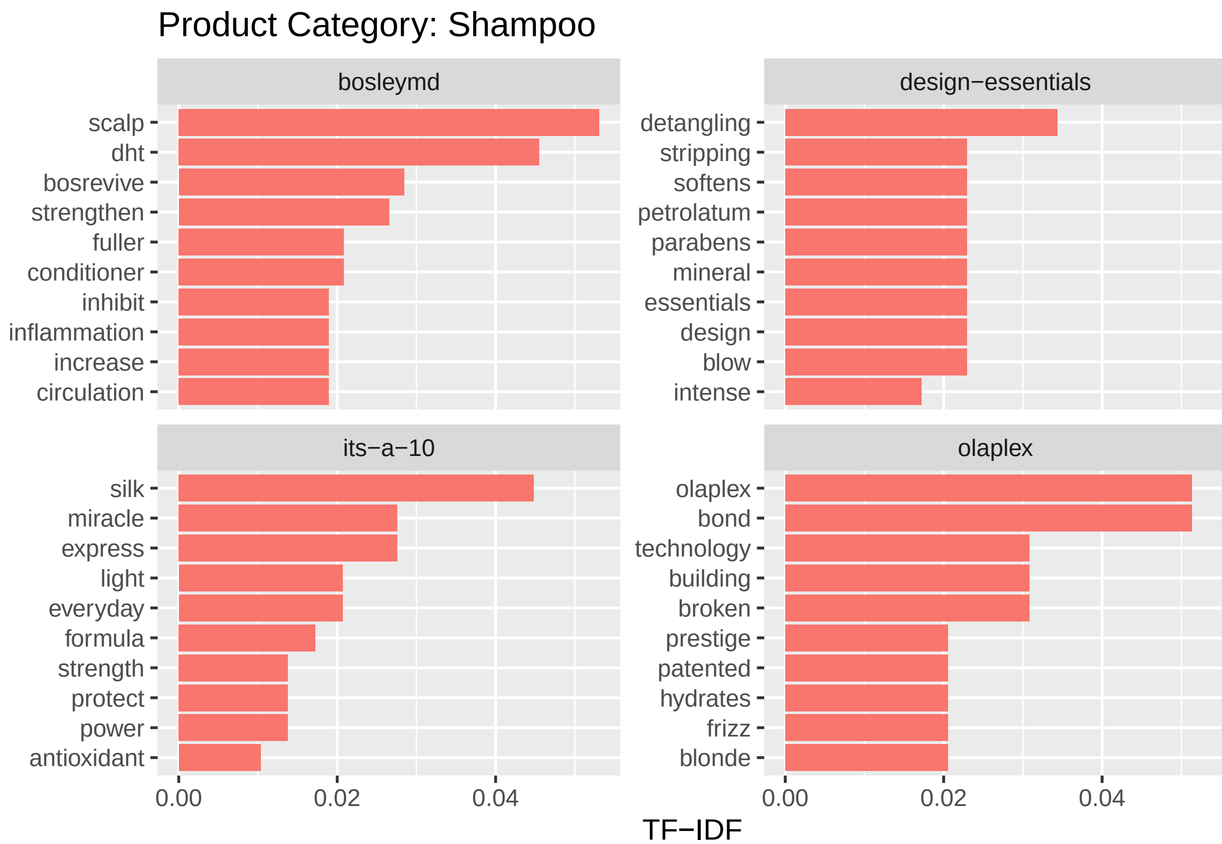
```
## # A tibble: 12 x 3
## # Groups:   class [3]
##    class          brand              mixed_rating
##    <chr>          <chr>                     <dbl>
##  1 FaceMoisturizer byoma                    0.481
##  2 FaceMoisturizer vitamins-sea-beauty      0.488
##  3 FaceMoisturizer buttah-skin              0.496
##  4 FaceMoisturizer nars                     0.5
##  5 Mascara        morphe                    0.16
##  6 Mascara        revlon                    0.366
##  7 Mascara        lottie-london             0.438
##  8 Mascara        hynt-beauty               0.464
##  9 Shampoo        its-a-10                  0.441
## 10 Shampoo        olaplex                   0.455
## 11 Shampoo        design-essentials         0.5
```

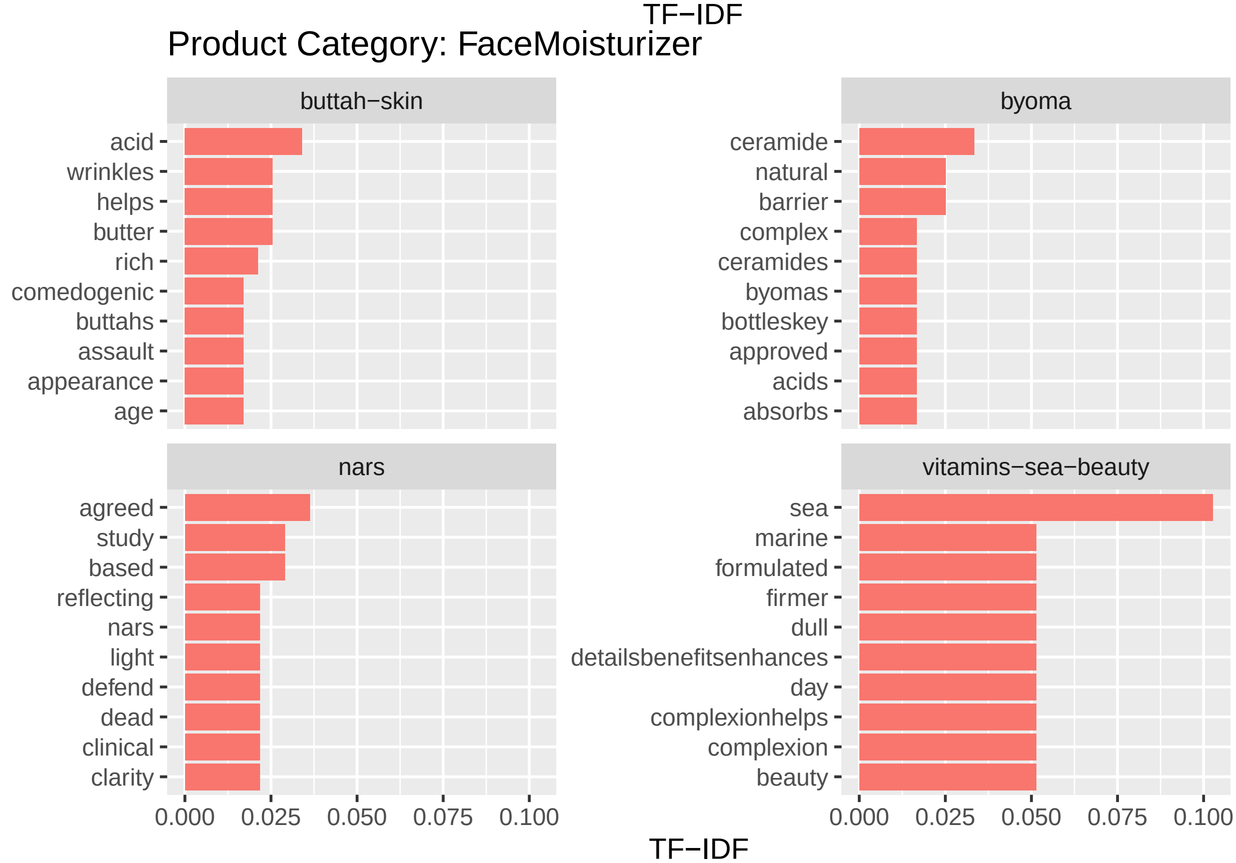
```
## 12 Shampoo         bosleymd                   0.531
```

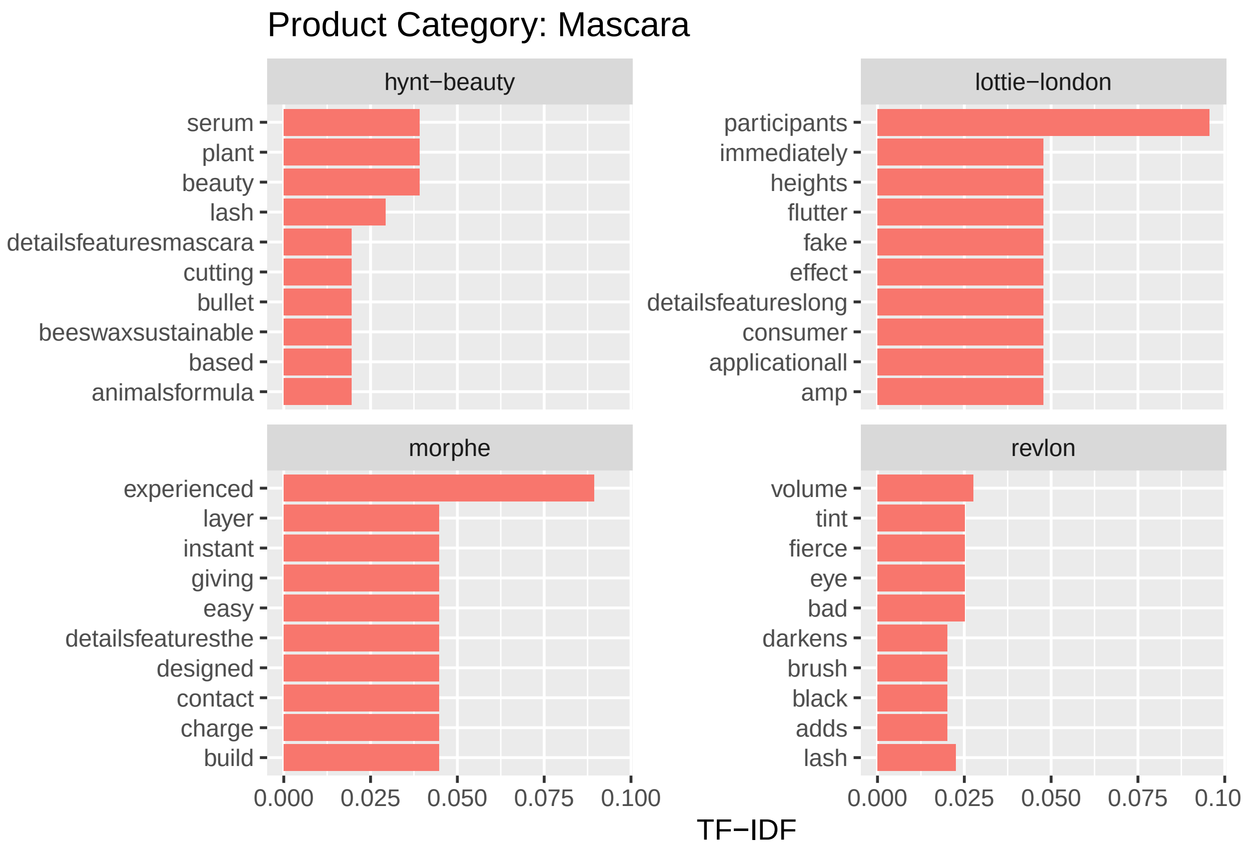
```
tidy_brands <- product_brands %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  group_by(class) %>%
  slice_min(mixed_rating, n=4, with_ties = FALSE) %>%
  group_by(class, brand) %>%
  unnest_tokens(word, text) %>%
  filter(!grepl('[0-9]', word)) %>%
  mutate(word = str_remove_all(word, "[:punct:]")) %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

```
pclasses <- c("Shampoo", "FaceMoisturizer", "Mascara")

for (i in 1:length(pclasses)) {

  print(tidy_brands %>%
    filter(class==pclasses[i]) %>%
    count(brand, word, sort = TRUE) %>%
    bind_tf_idf(word, brand, n) %>%
    group_by(brand) %>%
    slice_max(tf_idf, n=10, with_ties=FALSE) %>%
    mutate(word=reorder(word, tf_idf)) %>%
    ggplot(aes(tf_idf, word)) +
    geom_col(show.legend = FALSE, fill="#F8766D") +
    facet_wrap(~brand, scales = "free_y") +
    labs(x = "TF-IDF",
         y = NULL,
         title = paste("Product Category:", pclasses[i])))


}
```


Product Category: Shampoo
bosleymd
scalp
dht
bosrevive
strengthen
fuller
conditioner
inhibit
inflammation
increase
circulation
design−essentials
detangling
stripping
softens
petrolatum
parabens
mineral
essentials
design
blow
intense
its−a−10
silk
miracle
express
light
everyday
formula
strength
protect
power
antioxidant
olaplex
olaplex
bond
technology
building
broken
prestige
patented
hydrates
frizz
blonde
0.00
0.02
0.04
TF−IDF


Product Category: FaceMoisturizer
buttah−skin
acid
wrinkles
helps
butter
rich
comedogenic
buttahs
assault
appearance
age
byoma
ceramide
natural
barrier
complex
ceramides
byomas
bottleskey
approved
acids
absorbs
nars
agreed
study
based
reflecting
nars
light
defend
dead
clinical
clarity
vitamins−sea−beauty
sea
marine
formulated
firmer
dull
detailsbenefitsenhances
day
complexionhelps
complexion
beauty
0.000
0.025
0.050
0.075
0.100
TF−IDF



Assess best product class descriptions within specific brand. Using brands peach-lily and colourpop.

```
product_cats <- ona_df %>%
  filter(brand =="peach-lily" | brand == "colourpop") %>%
  select(c(text, rating, price, class, brand, pos_sentiment, rev_num)) %>%
  group_by(brand, class) %>%
  summarise(text=paste0(text, collapse = " "),
            rating=mean(rating),
            price=mean(price),
            pos_sentiment=mean(pos_sentiment),
            rev_num=sum(rev_num)) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'brand'. You can override using the
## `.groups` argument.
```

```
# print list of top cats for each brand
product_cats %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  group_by(brand) %>%
  slice_max(mixed_rating, n=4, with_ties = FALSE) %>%
  select(class, brand) %>%
  print(n=8)
```

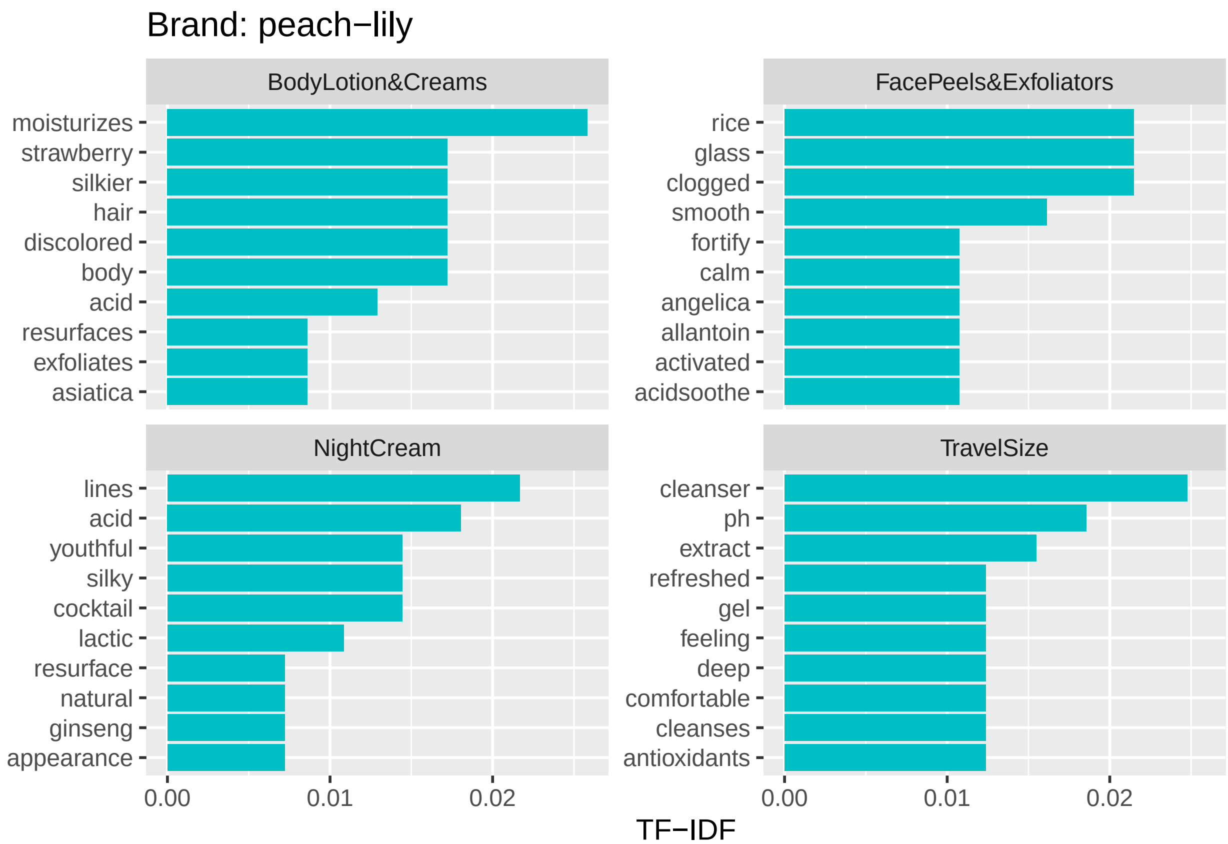
```
## # A tibble: 8 x 2
## # Groups:   brand [2]
```

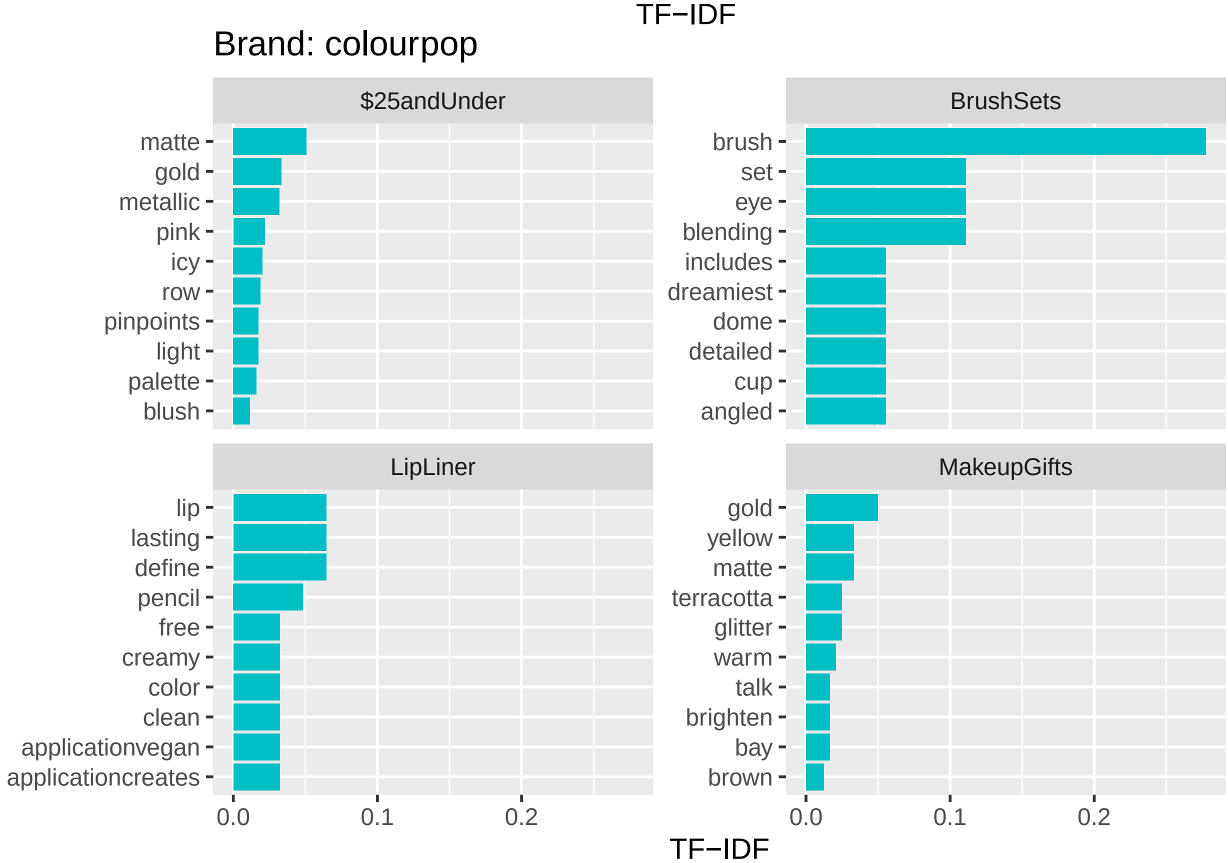
```
##   class                 brand
##   <chr>                 <chr>
## 1 MakeupGifts           colourpop
## 2 LipLiner              colourpop
## 3 $25andUnder           colourpop
## 4 BrushSets             colourpop
## 5 FacePeels&Exfoliators peach-lily
## 6 NightCream            peach-lily
## 7 TravelSize            peach-lily
## 8 BodyLotion&Creams     peach-lily
```

```
tidy_cats <- product_cats %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  group_by(brand) %>%
  slice_max(mixed_rating, n=4, with_ties = FALSE) %>%
  group_by(class, brand) %>%
  unnest_tokens(word, text) %>%
  filter(!grepl('[0-9]', word)) %>%
  mutate(word = str_remove_all(word, "[:punct:]")) %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

```
brands <- c("peach-lily", "colourpop")

for (i in 1:length(brands)) {

  print(tidy_cats %>%
    filter(brand==brands[i]) %>%
    count(class, word, sort = TRUE) %>%
    bind_tf_idf(word, class, n) %>%
    group_by(class) %>%
    slice_max(tf_idf, n=10, with_ties=FALSE) %>%
    mutate(word=reorder_within(word, tf_idf, class)) %>%
    ggplot(aes(tf_idf, word)) +
    geom_col(show.legend = FALSE, fill="#00BFC4") +
    facet_wrap(~class, scales = "free_y") +
    scale_y_reordered() +
    labs(x = "TF-IDF",
         y = NULL,
         title = paste("Brand:", brands[i])))


}
```





Assess worst rated brand descriptions within specific product classes. Using brands peach-lily and colourpop.

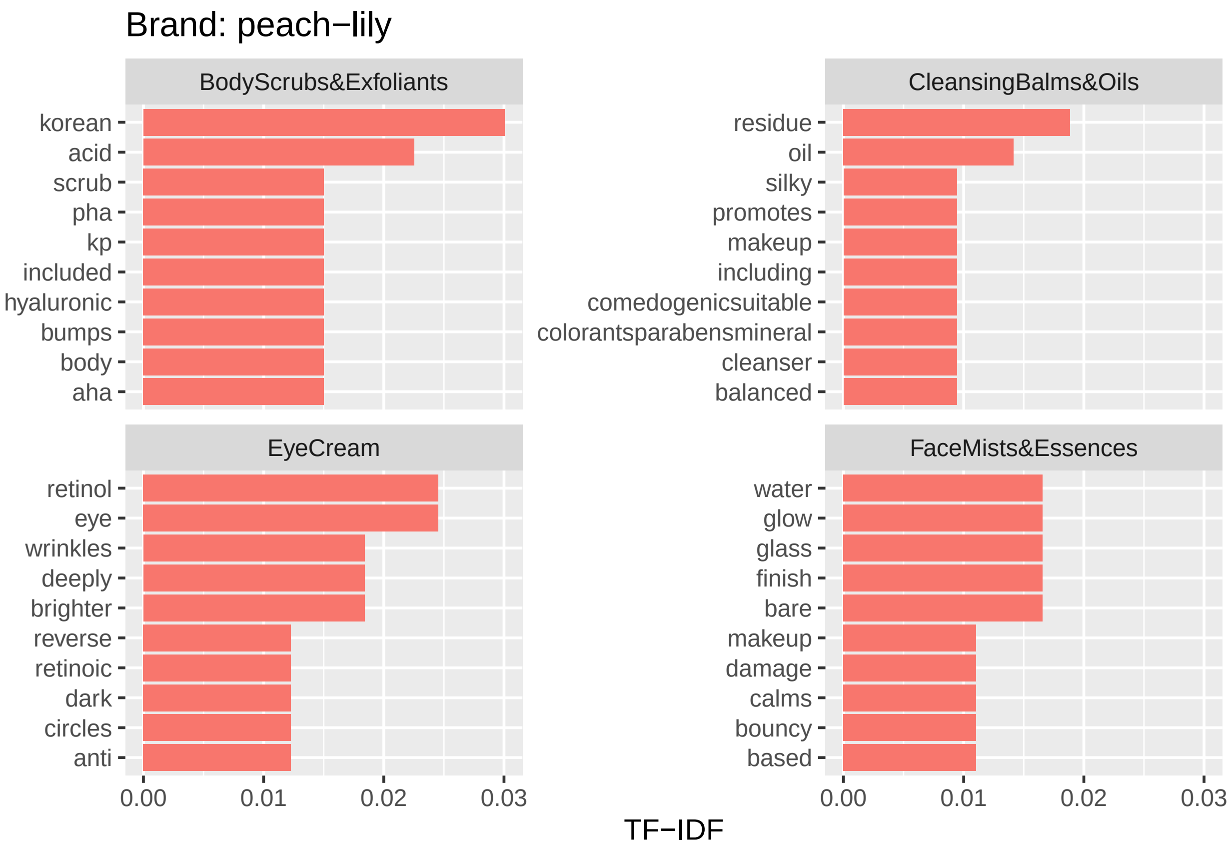
```r
# print list of top cats for each brand
product_cats %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  group_by(brand) %>%
  slice_min(mixed_rating, n=4, with_ties = FALSE) %>%
  select(class, brand) %>%
  print(n=8)
```

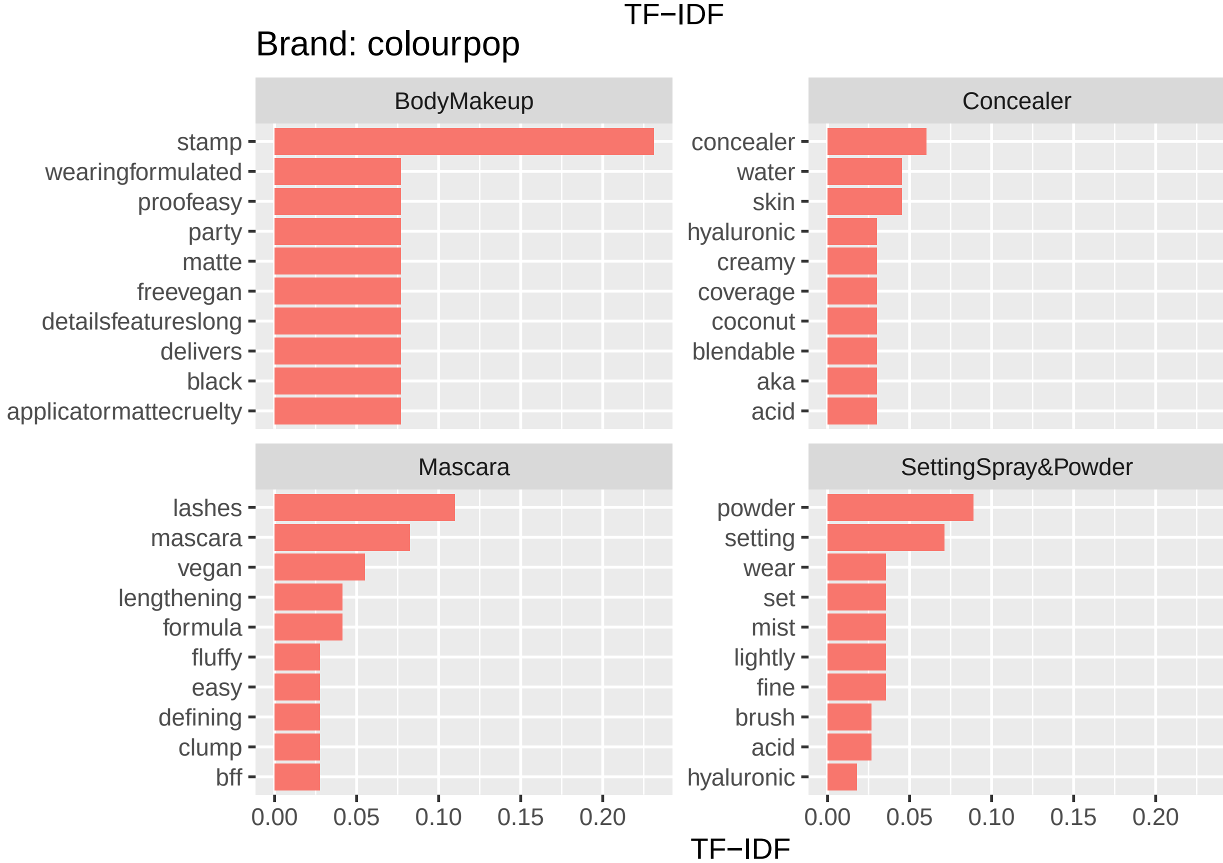
```
## # A tibble: 8 x 2
## # Groups:   brand [2]
##   class                  brand
##   <chr>                  <chr>
## 1 Mascara                colourpop
## 2 Concealer              colourpop
## 3 SettingSpray&Powder    colourpop
## 4 BodyMakeup             colourpop
## 5 CleansingBalms&Oils    peach-lily
## 6 BodyScrubs&Exfoliants peach-lily
## 7 EyeCream               peach-lily
## 8 FaceMists&Essences     peach-lily
```

```r
tidy_cats <- product_cats %>%
  mutate(mixed_rating=alpha*pos_sentiment+(1-alpha)*(rating/5)) %>%
  group_by(brand) %>%
  slice_min(mixed_rating, n=4, with_ties = FALSE) %>%
  group_by(class, brand) %>%
  unnest_tokens(word, text) %>%
  filter(!grepl('[0-9]', word)) %>%
  mutate(word = str_remove_all(word, "[:punct:]")) %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

```r
brands <- c("peach-lily", "colourpop")

for (i in 1:length(brands)) {

  print(tidy_cats %>%
    filter(brand==brands[i]) %>%
    count(class, word, sort = TRUE) %>%
    bind_tf_idf(word, class, n) %>%
    group_by(class) %>%
    slice_max(tf_idf, n=10, with_ties=FALSE) %>%
    mutate(word=reorder_within(word, tf_idf, class)) %>%
    ggplot(aes(tf_idf, word)) +
    geom_col(show.legend = FALSE, fill="#F8766D") +
    facet_wrap(~class, scales = "free_y") +
    scale_y_reordered() +
    labs(x = "TF-IDF",
         y = NULL,
         title = paste("Brand:", brands[i])))


}
```


Brand: peach−lily
BodyScrubs&Exfoliants
korean
acid
scrub
pha
kp
included
hyaluronic
bumps
body
aha
CleansingBalms&Oils
residue
oil
silky
promotes
makeup
including
comedogenicsuitable
colorantsparabensmineral
cleanser
balanced
EyeCream
retinol
eye
wrinkles
deeply
brighter
reverse
retinoic
dark
circles
anti
FaceMists&Essences
water
glow
glass
finish
bare
makeup
damage
calms
bouncy
based
0.00
0.01
0.02
0.03
TF−IDF


Brand: colourpop
BodyMakeup
stamp
wearingformulated
proofeasy
party
matte
freevegan
detailsfeatureslong
delivers
black
applicatormattecruelty
Concealer
concealer
water
skin
hyaluronic
creamy
coverage
coconut
blendable
aka
acid
Mascara
lashes
mascara
vegan
lengthening
formula
fluffy
easy
defining
clump
bff
SettingSpray&Powder
powder
setting
wear
set
mist
lightly
fine
brush
acid
hyaluronic
0.00
0.05
0.10
0.15
0.20
TF−IDF

Topic Modeling

Create corpus.

```r
library(tm)
```

```
## Loading required package: NLP
##
## Attaching package: 'NLP'
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```r
brands_df <- ona_df %>%
  select(c(text, rating, price, class, brand, pos_sentiment, rev_num)) %>%
  group_by(brand) %>%
  summarise(text=paste0(text, collapse = " "),
            rating=mean(rating),
            price=mean(price),
            pos_sentiment=mean(pos_sentiment),
            rev_num=sum(rev_num)) %>%
  ungroup()

# tidy desc data by brand
tidy_brands <- brands_df %>%
  unnest_tokens(word, text) %>%
  filter(!grepl('[0-9]', word)) %>%
  mutate(word = str_remove_all(word, "[:punct:]")) %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

```r
# cast tidy data into dtm
dtm <- tidy_brands %>%
  count(brand, word, sort=TRUE) %>%
  cast_dtm(brand, word, n)

# inspect dtm
dtm %>% inspect()
```

```
## <<DocumentTermMatrix (documents: 576, terms: 59966)>>
## Non-/sparse entries: 262774/34277642
## Sparsity           : 99%
## Maximal term length: 121
## Weighting          : term frequency (tf)
## Sample             :
##                      Terms
## Docs                   color dry formula free hair helps natural oil skin
##   bareminerals            45  10      30  139    1    46      45  95  210
##   clarins                  7  19      14   34    0    36      64  26  333
##   dermalogica              0  29      23   31    0   107      37  48  418
##   drunk-elephant           8  15       9   27   53    21       7  58  267
##   joico                   98  48       9   15  465    75      18 136    5
##   kiehls-since-1851        0  30      49   29   11    90      18 120  345
```

```
##   lancome                   24  10      70  37   3    14       29  22  261
##   living-proof              79  39       4  67  372   26       38  29    0
##   nyx-professional-makeup   58  10      94  63   4     4       24  36   87
##   tarte                     25  14      55  42   5    88       63  42  121
##                          Terms
## Docs                      smooth
##   bareminerals                43
##   clarins                     27
##   dermalogica                 26
##   drunk-elephant               9
##   joico                       11
##   kiehls-since-1851           38
##   lancome                     28
##   living-proof                24
##   nyx-professional-makeup     54
##   tarte                       36
```

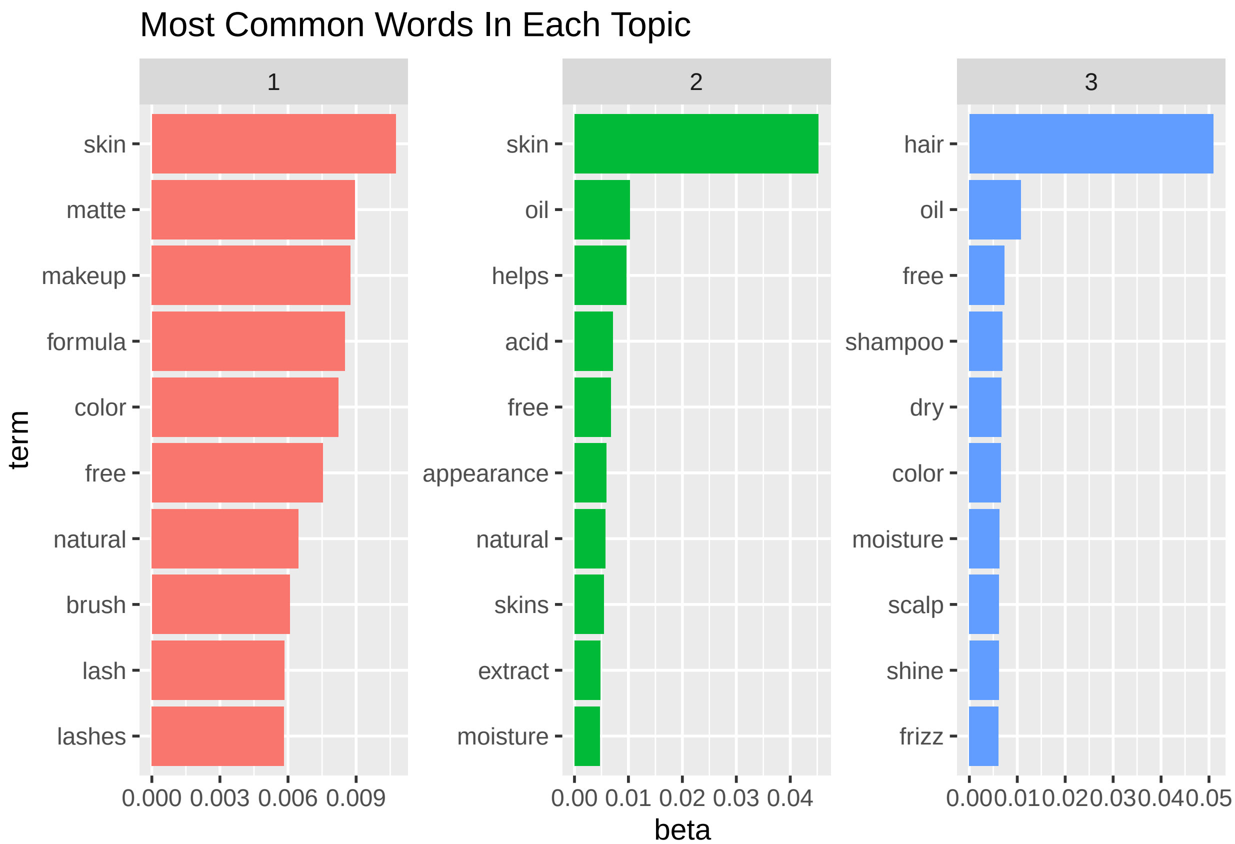
Build topic models.

```
library(topicmodels)
options(scipen=2)

brands_lda <- LDA(dtm, k = 3, control = list(seed = 1234))
brands_lda
```

```
## A LDA_VEM topic model with 3 topics.
```

Terms that are most common within each topic.

```
brands_topics <- tidy(brands_lda, matrix = "beta")

brands_top_terms <- brands_topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, -beta)

brands_top_terms %>%
  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() +
  ggtitle("Most Common Words In Each Topic")
```



Topic 3 appears to be hair related, topic 2 looks like facial and body cleansers/moisturizers, and topic 1 looks like makeup and lashes.

Now we look at the each brands probability of being in each topic.

```
brand_documents <- tidy(brands_lda, matrix = "gamma")
brand_documents
```

```
## # A tibble: 1,728 x 3
##    document          topic      gamma
##    <chr>             <int>      <dbl>
##  1 joico                 1 0.00000845
##  2 dermalogica           1 0.00000938
##  3 nioxin                1 0.0000212
##  4 living-proof          1 0.0000108
##  5 redken                1 0.0000134
##  6 la-roche-posay        1 0.0000127
##  7 kiehls-since-1851     1 0.00000987
##  8 clarins               1 0.00847
##  9 dashing-diva          1 1.00
## 10 tree-hut              1 0.0000119
## # ... with 1,718 more rows
```

```
brand_documents %>%
  group_by(topic) %>%
  slice_max(gamma, n = 10) %>%
  ungroup() %>%
```

```r
  arrange(topic, -gamma) %>%
  select(c(document, topic)) %>%
  print(n=30)
```

```
## # A tibble: 30 x 2
##    document                topic
##    <chr>                   <int>
##  1 nyx-professional-makeup     1
##  2 anastasia-beverly-hills     1
##  3 dashing-diva                1
##  4 kiss                        1
##  5 real-techniques             1
##  6 maybelline                  1
##  7 ardell                      1
##  8 fenty-beauty-by-rihanna     1
##  9 makeup-revolution           1
## 10 juvias-place                1
## 11 dermalogica                 2
## 12 neutrogena                  2
## 13 la-roche-posay              2
## 14 cerave                      2
## 15 peter-thomas-roth           2
## 16 fresh                       2
## 17 hempz                       2
## 18 vichy                       2
## 19 murad                       2
## 20 boscia                      2
## 21 joico                       3
## 22 living-proof                3
## 23 redken                      3
## 24 paul-mitchell               3
## 25 drybar                      3
## 26 bumble-bumble               3
## 27 devacurl                    3
## 28 lanza                       3
## 29 fekkai                      3
## 30 not-your-mothers            3
```

Brands sufficiently match topics as described above.

Price differences, ratings, and sentiment between clusters.

```r
brand_topics <- brand_documents %>%
  group_by(document) %>%
  slice_max(gamma) %>%
  mutate(brand=document) %>%
  select(-c(document, gamma))
```

```
## Adding missing grouping variables: `document`
```

```
brands_df <- brands_df %>%
  left_join(brand_topics)
```

```
## Joining, by = "brand"
```

```
brands_df %>%
  group_by(topic) %>%
  summarise(Topic.Price = mean(price),
            Topic.Rating = mean(rating),
            Topic.Sentiment = mean(pos_sentiment))
```

```
## # A tibble: 3 x 4
##   topic Topic.Price Topic.Rating Topic.Sentiment
##   <int>       <dbl>        <dbl>           <dbl>
## 1     1        25.8         3.95           0.816
## 2     2        39.7         4.17           0.837
## 3     3        33.7         4.04           0.812
```

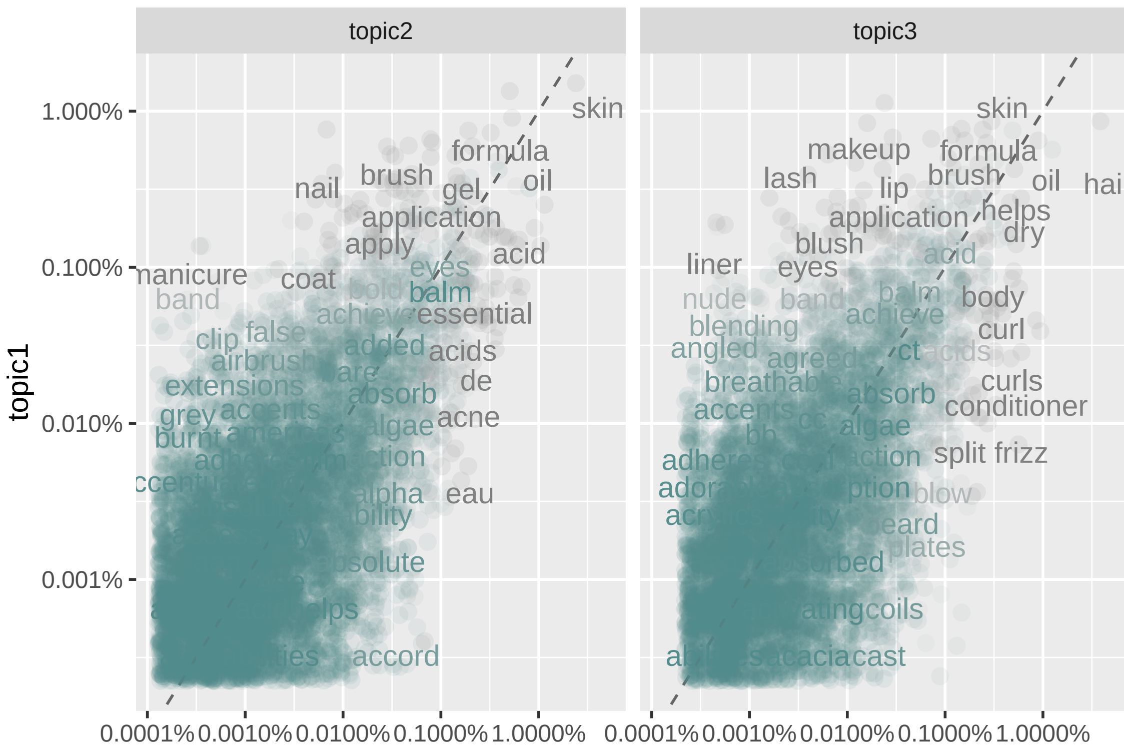
Nothing substantially different.

Correlation Plots

Using the clusters 2 and 3, plots.

```
library(scales)


# tidy desc data by brand
tidy_brands <- brands_df %>%
  unnest_tokens(word, text) %>%
  filter(!grepl('[0-9]', word)) %>%
  mutate(word = str_remove_all(word, "[:punct:]")) %>%
  anti_join(stop_words) %>%
  mutate(topic = paste0("topic", topic))

frequency <- tidy_brands %>%
  count(topic, word) %>%
  group_by(topic) %>%
  mutate(proportion = n / sum(n)) %>%
  select(-n) %>%
  pivot_wider(names_from = topic, values_from = proportion) %>%
  pivot_longer(topic2:topic3,
               names_to = "topic", values_to = "proportion")

# expect a warning about rows with missing values being removed
ggplot(frequency, aes(x = proportion, y = topic1,
                      color = abs(topic1 - proportion))) +
  geom_abline(color = "gray40", lty = 2) +
  geom_jitter(alpha = 0.1, size = 2.5, width = 0.3, height = 0.3) +
  geom_text(aes(label = word), check_overlap = TRUE, vjust = 1.5) +
  scale_x_log10(labels = percent_format()) +
  scale_y_log10(labels = percent_format()) +
```

```
  scale_color_gradient(limits = c(0, 0.001),
                       low = "darkslategray4", high = "gray75") +
  facet_wrap(~topic, ncol = 2) +
  theme(legend.position="none") +
  labs(y = "topic1", x = NULL)
```



Using the clusters 2 and 3, estimates.

```
# t3 vs t1
cor.test(data = frequency[frequency$topic == "topic3",],
         ~ proportion + topic1)
```

```
##
##  Pearson's product-moment correlation
##
## data:  proportion and topic1
## t = 33.965, df = 5939, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3817894 0.4243801
## sample estimates:
##       cor
## 0.4033031
```

```
# t2 vs t1
cor.test(data = frequency[frequency$topic == "topic2",],
         ~ proportion + topic1)
```

```
##
##  Pearson's product-moment correlation
```

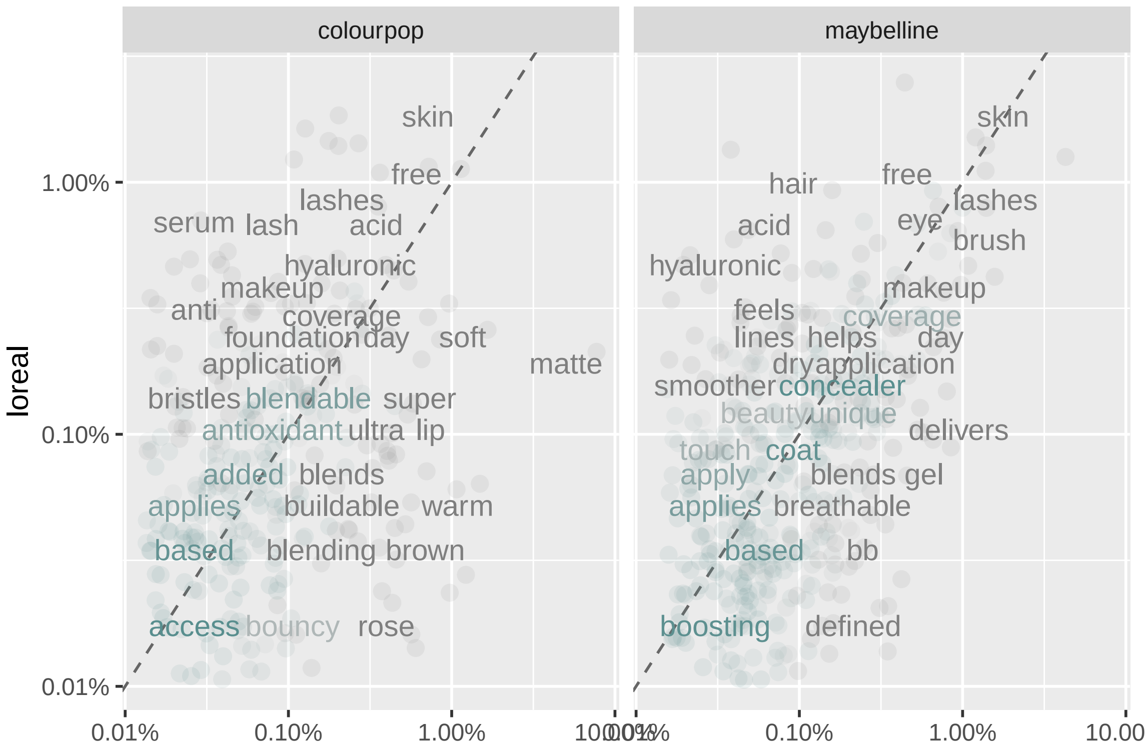
```
##
## data:  proportion and topic1
## t = 72.171, df = 7600, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6241617 0.6508441
## sample estimates:
##       cor
## 0.6376942
```

```
# t2 vs t3
cor.test(frequency$proportion[frequency$topic == "topic2"],
         frequency$proportion[frequency$topic == "topic3"])
```

```
##
##  Pearson's product-moment correlation
##
## data:  frequency$proportion[frequency$topic == "topic2"] and frequency$proportion[frequency$topic ==
## t = 25.72, df = 6651, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2787597 0.3224742
## sample estimates:
##       cor
## 0.3007749
```

Using two different brands, plots.

```
frequency <- tidy_brands %>%
  filter(brand=="loreal" | brand=="colourpop" | brand=="maybelline") %>%
  count(brand, word) %>%
  group_by(brand) %>%
  mutate(proportion = n / sum(n)) %>%
  select(-n) %>%
  pivot_wider(names_from = brand, values_from = proportion) %>%
  pivot_longer(c(`colourpop`, `maybelline`),
               names_to = "brand", values_to = "proportion")

# expect a warning about rows with missing values being removed
ggplot(frequency, aes(x = proportion, y = loreal,
                      color = abs(loreal - proportion))) +
  geom_abline(color = "gray40", lty = 2) +
  geom_jitter(alpha = 0.1, size = 2.5, width = 0.3, height = 0.3) +
  geom_text(aes(label = word), check_overlap = TRUE, vjust = 1.5) +
  scale_x_log10(labels = percent_format()) +
  scale_y_log10(labels = percent_format()) +
  scale_color_gradient(limits = c(0, 0.001),
                       low = "darkslategray4", high = "gray75") +
  facet_wrap(~brand, ncol = 2) +
  theme(legend.position="none") +
  labs(y = "loreal", x = NULL)
```



Using the brands, estimates.

```
# loreal vs maybelline
cor.test(data = frequency[frequency$brand == "maybelline",],
         ~ proportion + loreal)
```

```
##
##  Pearson's product-moment correlation
##
## data:  proportion and loreal
## t = 16.911, df = 392, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5884294 0.7031478
## sample estimates:
##       cor
## 0.649469
```

```
# loreal vs colourpop
cor.test(data = frequency[frequency$brand == "colourpop",],
         ~ proportion + loreal)
```

```
##
##  Pearson's product-moment correlation
##
## data:  proportion and loreal
## t = 2.7815, df = 281, p-value = 0.005777
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.04800902 0.27503842
```

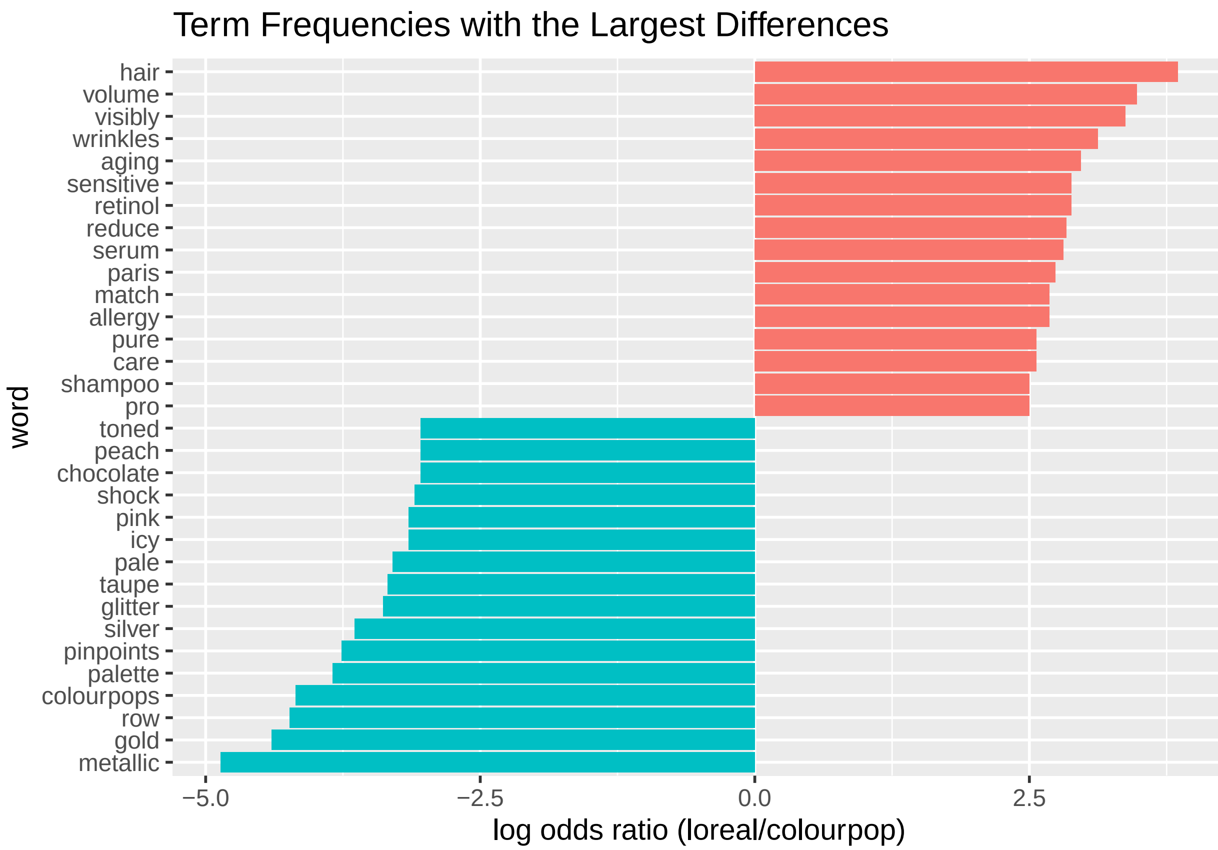
```
## sample estimates:
##       cor
## 0.1636902
```

```
# loreal vs colourpop
cor.test(frequency$proportion[frequency$brand == "colourpop"],
         frequency$proportion[frequency$brand == "maybelline"])
```

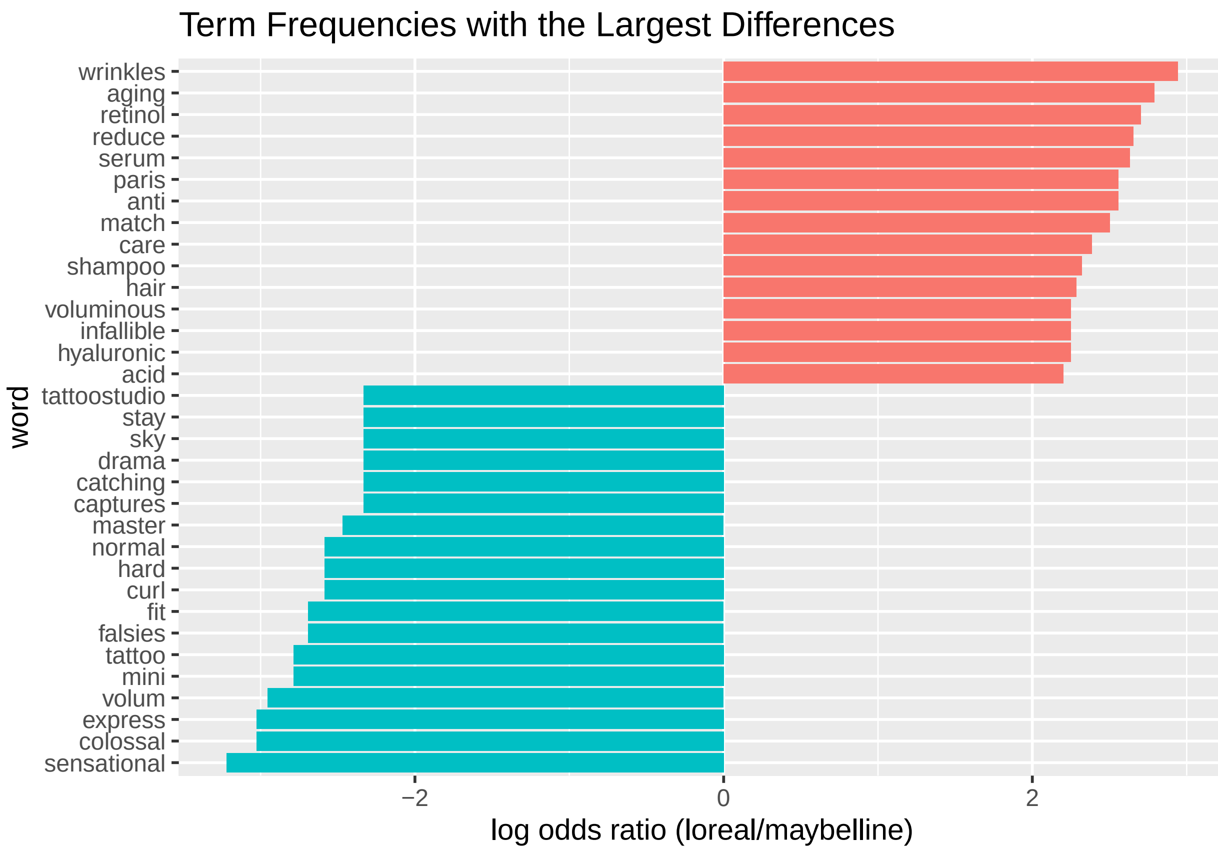
```
##
##  Pearson's product-moment correlation
##
## data:  frequency$proportion[frequency$brand == "colourpop"] and frequency$proportion[frequency$brand
## t = 3.9421, df = 260, p-value = 0.0001039
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1197402 0.3486358
## sample estimates:
##       cor
## 0.2374818
```

What else makes colourpop different from the other two?

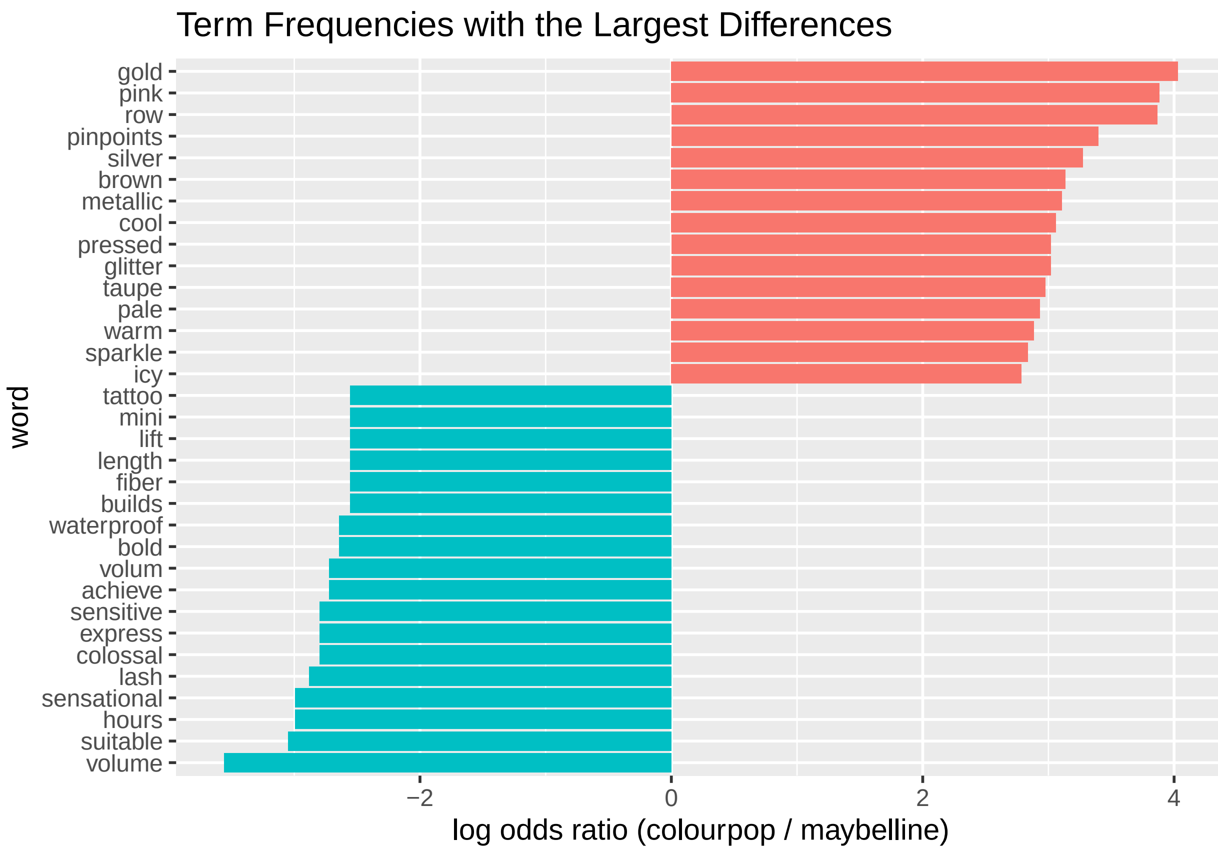
```
l_vs_c <- tidy_brands %>%
  filter(!word %in% c("loréal", "loreal", "loreals")) %>%
  filter(brand=="loreal" | brand=="colourpop") %>%
  count(brand, word) %>%
  group_by(brand) %>%
  pivot_wider(names_from = brand, values_from = n, values_fill = 0) %>%
  mutate_if(is.numeric, list(~(. + 1) / (sum(.) + 1))) %>%
  mutate(logratio = log(loreal / colourpop)) %>%
  arrange(desc(logratio))

l_vs_c %>%
  group_by(logratio < 0) %>%
  slice_max(abs(logratio), n = 15) %>%
  ungroup() %>%
  mutate(word = reorder(word, logratio)) %>%
  ggplot(aes(word, logratio, fill = logratio < 0)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  ylab("log odds ratio (loreal/colourpop)") +
  scale_fill_discrete(name = "", labels = c("loreal", "colourpop")) +
  ggtitle("Term Frequencies with the Largest Differences")
```




```
l_vs_m <- tidy_brands %>%
  filter(!word %in% c("loréal", "maybelline", "maybellines")) %>%
  filter(brand=="loreal" | brand=="maybelline") %>%
  count(brand, word) %>%
  group_by(brand) %>%
  pivot_wider(names_from = brand, values_from = n, values_fill = 0) %>%
  mutate_if(is.numeric, list(~(. + 1) / (sum(.) + 1))) %>%
  mutate(logratio = log(loreal / `maybelline`)) %>%
  arrange(desc(logratio))

l_vs_m %>%
  group_by(logratio < 0) %>%
  slice_max(abs(logratio), n = 15) %>%
  ungroup() %>%
  mutate(word = reorder(word, logratio)) %>%
  ggplot(aes(word, logratio, fill = logratio < 0)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  ylab("log odds ratio (loreal/maybelline)") +
  scale_fill_discrete(name = "", labels = c("loreal", "maybelline")) +
  ggtitle("Term Frequencies with the Largest Differences")
```




```
c_vs_m <- tidy_brands %>%
  filter(!word %in% c("maybelline", "maybellines", "colourpops")) %>%
  filter(brand=="colourpop" | brand=="maybelline") %>%
  count(brand, word) %>%
  group_by(brand) %>%
  pivot_wider(names_from = brand, values_from = n, values_fill = 0) %>%
  mutate_if(is.numeric, list(~(. + 1) / (sum(.) + 1))) %>%
  mutate(logratio = log(colourpop / maybelline)) %>%
  arrange(desc(logratio))

c_vs_m %>%
  group_by(logratio < 0) %>%
  slice_max(abs(logratio), n = 15) %>%
  ungroup() %>%
  mutate(word = reorder(word, logratio)) %>%
  ggplot(aes(word, logratio, fill = logratio < 0)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  ylab("log odds ratio (colourpop / maybelline)") +
  scale_fill_discrete(name = "", labels = c("colourpop", "maybelline")) +
  ggtitle("Term Frequencies with the Largest Differences")
```



Colourpop is a bit more Gen. Z (hip, colorful, cool) while maybelline and loreal are a bit more millennials.