PCA_example

Fran Camacho

2025-04-22

Using PCA to reduce highly dimensional social media data

[Chapter 13 from the book "Machine Learning with R", by Brett Lantz].

PCA is a feature extraction technique that reduces the dimensionality of a dataset by synthesizing a smaller set of features from the complete set.

```
if (!require(tidyverse)) install.packages('tidyverse', dependencies = T)
library(tidyverse)

library(glue)

if (!require(irlba)) install.packages('irlba', dependencies = T)
# irbla: "implicitly restarted Lanczos bidiagonalization algorithm"
library(irlba)
```

Load the data:

```
sns_data <- read_csv("./CSVs/snsdata.csv")
```

```
## Rows: 30000 Columns: 40
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): gender
## dbl (39): gradyear, age, friends, basketball, football, soccer, softball, vo...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
sns_data
```

```
## # A tibble: 30,000 x 40
##    gradyear gender   age friends basketball football soccer softball volleyball
##       <dbl> <chr>  <dbl>   <dbl>      <dbl>    <dbl>  <dbl>    <dbl>      <dbl>
##  1     2006 M       19.0       7          0        0      0        0          0
##  2     2006 F       18.8       0          0        1      0        0          0
##  3     2006 M       18.3      69          0        1      0        0          0
##  4     2006 F       18.9       0          0        0      0        0          0
##  5     2006 <NA>    19.0      10          0        0      0        0          0
##  6     2006 F       NA       142          0        0      0        0          0
```

```
##  7     2006 F       18.9      72          0        0      0        0          0
##  8     2006 M       18.3      17          0        0      0        1          0
##  9     2006 F       19.1      52          0        0      0        0          0
## 10     2006 F       18.7      39          0        0      0        0          0
## # i 29,990 more rows
## # i 31 more variables: swimming <dbl>, cheerleading <dbl>, baseball <dbl>,
## #   tennis <dbl>, sports <dbl>, cute <dbl>, sex <dbl>, sexy <dbl>, hot <dbl>,
## #   kissed <dbl>, dance <dbl>, band <dbl>, marching <dbl>, music <dbl>,
## #   rock <dbl>, god <dbl>, church <dbl>, jesus <dbl>, bible <dbl>, hair <dbl>,
## #   dress <dbl>, blonde <dbl>, mall <dbl>, shopping <dbl>, clothes <dbl>,
## #   hollister <dbl>, abercrombie <dbl>, die <dbl>, death <dbl>, ...
```

we will select() only the columns corresponding to the features:

```
sns_terms <- sns_data |> select(basketball:drugs)
sns_terms
```

```
## # A tibble: 30,000 x 36
##    basketball football soccer softball volleyball swimming cheerleading baseball
##         <dbl>    <dbl>  <dbl>    <dbl>      <dbl>    <dbl>        <dbl>    <dbl>
##  1          0        0      0        0          0        0            0        0
##  2          0        1      0        0          0        0            0        0
##  3          0        1      0        0          0        0            0        0
##  4          0        0      0        0          0        0            0        0
##  5          0        0      0        0          0        0            0        0
##  6          0        0      0        0          0        0            0        0
##  7          0        0      0        0          0        0            0        0
##  8          0        0      0        1          0        0            0        0
##  9          0        0      0        0          0        0            0        0
## 10          0        0      0        0          0        0            0        0
## # i 29,990 more rows
## # i 28 more variables: tennis <dbl>, sports <dbl>, cute <dbl>, sex <dbl>,
## #   sexy <dbl>, hot <dbl>, kissed <dbl>, dance <dbl>, band <dbl>,
## #   marching <dbl>, music <dbl>, rock <dbl>, god <dbl>, church <dbl>,
## #   jesus <dbl>, bible <dbl>, hair <dbl>, dress <dbl>, blonde <dbl>,
## #   mall <dbl>, shopping <dbl>, clothes <dbl>, hollister <dbl>,
## #   abercrombie <dbl>, die <dbl>, death <dbl>, drunk <dbl>, drugs <dbl>
```

The **PCA technique will only work with a matrix of numeric data**. Because each of the resulting 36 columns is a count, no more data preparation is needed. (If the dataset included categorical features, it would be necessary to convert these to numeric before proceeding).

Base R includes a built-in PCA function called prcomp(), which becomes slow to run as datasets get larger. We'll use a drop-in substitute from the irlba package by Bryan W. Lewis, which can be stopped early to return only a subset of the full set of potential principal components.

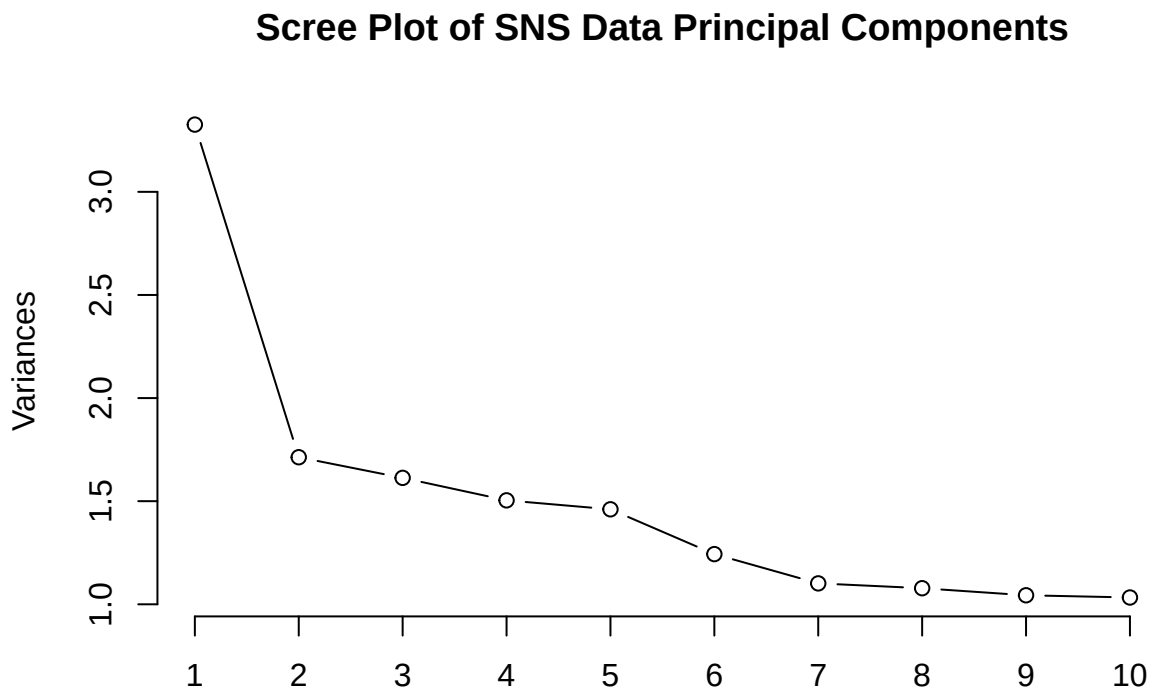
```
set.seed(2023)

sns_pca <- sns_terms |>
           prcomp_irlba(n = 10, center = TRUE, scale = TRUE)
#                       limit the result to the first 10 principal components,
#                               standardizing the data by centering each feature around zero,
#                                                 and scaling them to have a variance of one.
```

Recall that each component in the PCA captures a decreasing amount of the dataset's variance and that we requested 10 of the possible 36 components.

A scree plot, named after the "scree" landslide patterns that form at the bottom of cliffs, helps visualize the amount of variance captured by each component and may thus help to determine the optimal number of components to use.

```
screeplot(sns_pca, npcs = 10, type = "lines",
          main = "Scree Plot of SNS Data Principal Components")
```



The scree plot shows that there is a substantial drop in the variance captured between the first and second components. The second through fifth components capture approximately the same amount of variance, and then there are additional substantial drops between the fifth and sixth components and between the sixth and seventh components. The seventh through tenth compo- nents capture approximately the same amount of variance.

Based on this result, we might decide to use one, five, or six principal components as our reduced-dimensionality dataset.

```
summary(sns_pca)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     1.82375 1.30885 1.27008 1.22642 1.20854 1.11506 1.04948
## Proportion of Variance 0.09239 0.04759 0.04481 0.04178 0.04057 0.03454 0.03059
## Cumulative Proportion  0.09239 0.13998 0.18478 0.22657 0.26714 0.30167 0.33227
```

```
##                             PC8     PC9    PC10
## Standard deviation     1.03828 1.02163 1.01638
## Proportion of Variance 0.02995 0.02899 0.02869
## Cumulative Proportion  0.36221 0.39121 0.41990
```

A component's proportion of variance is its variance out of the total for all components — not only the 10 shown here, but also the remaining 26 that we could have created. Therefore, the cumulative proportion of variance maxes out at 41.99% rather than the 100% that would be explained by all 36 components.

Using PCA as a dimensionality reduction technique requires the user to determine how many components to keep. In this case, if we choose five components, we will capture 26.7% of the vari- ance, or one-fourth of the total information in the original data. Whether or not this is sufficient depends on how much of the remaining 73.3% of variance is signal or noise—something that we can only determine by attempting to build a useful learning algorithm.

For simplicity here, we'll reduce the original 36-dimension dataset to five principal components. By default, the irlba_prcomp() function automatically saves a version of the original dataset that has been transformed into the lower-dimension space. This is found in the resulting sns_pca list object with the name x, which we can examine with the str() command:

```
str(sns_pca$x)
```

```
##  num [1:30000, 1:10] -1.448 3.492 -0.646 -1.041 4.322 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:10] "PC1" "PC2" "PC3" "PC4" ...
```

The transformed dataset is a numeric matrix with 30,000 rows like the original dataset but 10 rather than 36 columns with names from PC1 to PC10. We can see this more clearly by using the head() command output to see the first few rows:

```
head(sns_pca$x)
```

```
##             PC1         PC2       PC3        PC4         PC5         PC6
## [1,] -1.4477620  0.07976310 0.3357330 -0.3636082  0.03833596 -0.01559079
## [2,]  3.4922144  0.36554520 0.7966735 -0.1871626  0.57126163  3.02758235
## [3,] -0.6459385 -0.67798166 0.8000251  0.6243070  0.25122261 -0.40751994
## [4,] -1.0405145  0.08118501 0.4099638 -0.2555128 -0.02620989  0.27837411
## [5,]  4.3216304 -1.01754361 3.4112730 -1.9209916 -0.43409869 -1.11734548
## [6,]  0.2131225 -0.65882053 1.6215828  0.9372545  1.47217369  0.04614790
##               PC7          PC8         PC9        PC10
## [1,] -0.007278589 -0.004582346  0.19226144 -0.08086065
## [2,]  0.306304037 -1.142422251  0.72992534 -0.11203923
## [3,] -0.454614417  0.704544996 -0.43734980  0.07735574
## [4,] -0.462898314 -0.175251793 -0.08843005 -0.26784326
## [5,]  2.122420077 -2.287638056  2.19992650  0.26536161
## [6,]  0.654207687  0.285263646  0.69439745  0.89649127
```

We can attempt to understand the components by visualizing the PCA loadings, or the weights that transform the original data into each of the principal components. Large loadings are more important to a particular component. These loadings are found in the sns_pca list object with the name rotation.

```
#sns_pca$rotation
```

output:

```
          PC1          PC2          PC3          PC4          PC5          PC6          PC7          PC8
```

[1,] 0.13469121 0.07904290 -0.157700393 0.2599239997 -0.311782958 -0.008622618 0.077140271 0.129830646 0.037084130 -0.04632527 [2,] 0.14425913 0.07164152 -0.117417952 0.2636356296 -0.249524923 -0.022148815 -0.287150510 -0.002471849 0.139662537 0.16291274 [3,] 0.06532029 0.05506134 -0.077551174 0.1138728270 -0.123696040 -0.013879379 -0.061172863 -0.167780047 -0.361469663 0.17388500 [4,] 0.07599929 0.05432792 -0.110824905 0.1670762149 -0.186477936 -0.020473521 0.349419135 0.367581104 0.090638924 -0.19063099 . . . [33,] 0.19611697 -0.15475838 0.138846785 -0.1729653896 -0.022982392 -0.035008129 0.027181174 0.155090786 -0.177264862 -0.13551168 [34,] 0.14941048 -0.15341393 0.047957830 -0.1344638474 0.006451136 0.012439587 -0.025977047 0.336966709 -0.337197410 -0.04847647 [35,] 0.18311990 -0.06777140 0.183474140 -0.1833871002 -0.084374627 -0.059833729 0.032729810 0.081964744 0.111177319 0.19094600 [36,] 0.22682609 -0.08018234 0.184824611 -0.1730842575 -0.078483698 -0.162273896 0.158536409 -0.099372610 0.151178223 -0.05874809

This is a numeric matrix with 36 rows corresponding to each of the original columns in the dataset and 10 columns that provide the loadings for the principal components.

To construct our visualization, we will need to pivot this data such that it has one row per social media term per principal component; that is, we will have 36 * 10 = 360 rows in the longer version of the dataset.

The combined tibble, with 11 columns and 36 rows, is piped into the pivot_longer() function, which pivots the table from wide to long format. The three parameters tell the function to pivot the 10 columns from PC1 to PC10, with the former column names now becoming the rows for a column named PC and the former column values now becoming row values of a column named Contribution.

```
sns_pca_long <- tibble(SNS_Term = colnames(sns_terms),
                       as_tibble(sns_pca$rotation)) |>
                pivot_longer(PC1:PC10, names_to = "PC", values_to = "Contribution")
```

The full command has created a tibble with 3 columns and 360 rows:

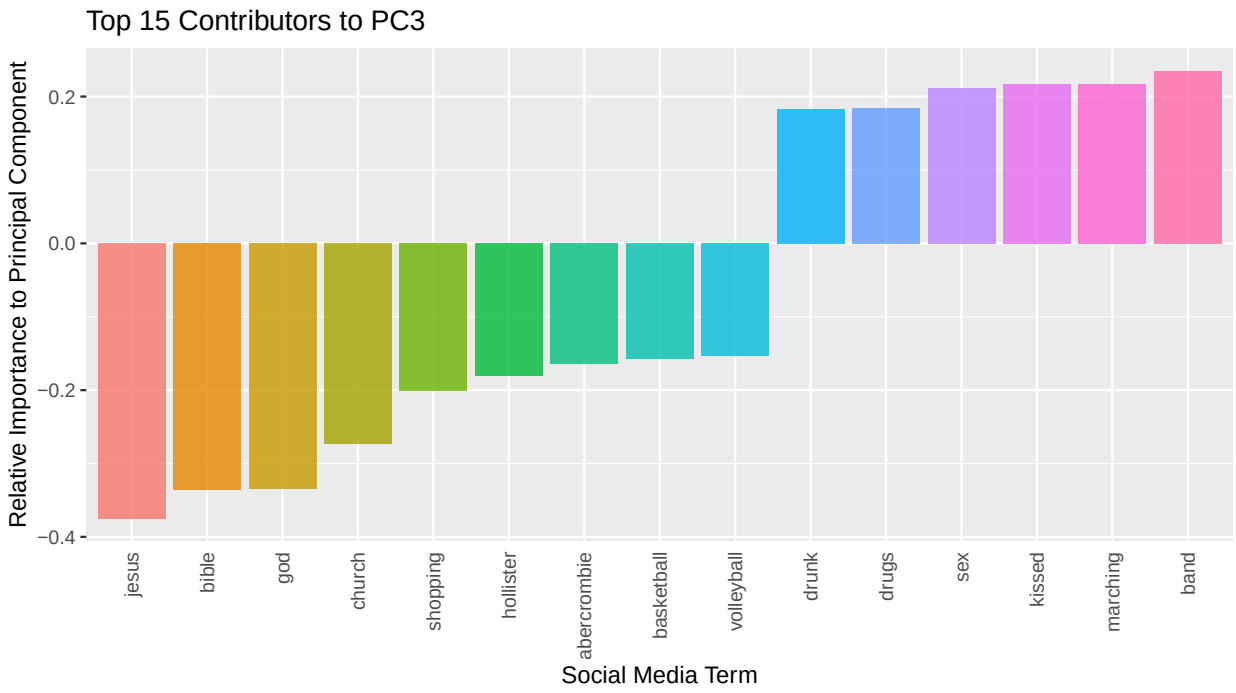
```
sns_pca_long
```

```
## # A tibble: 360 x 3
##    SNS_Term   PC    Contribution
##    <chr>      <chr>        <dbl>
##  1 basketball PC1          0.135
##  2 basketball PC2          0.0790
##  3 basketball PC3         -0.158
##  4 basketball PC4          0.260
##  5 basketball PC5         -0.312
##  6 basketball PC6         -0.00862
##  7 basketball PC7          0.0771
##  8 basketball PC8          0.130
##  9 basketball PC9          0.0371
## 10 basketball PC10        -0.0463
## # i 350 more rows
```

The ggplot() function can now be used to plot the most important contributing terms for a given principal component. For example, to look at the third principal component, we'll filter() the rows to limit to only PC3,

select the top 15 largest contribution values—considering both posi- tive and negative values using the abs() absolute value function—and mutate the SNS_Term to reorder by the contribution amount. Ultimately, this is piped into ggplot() with a number of adjustments to the formatting:

```
sns_pca_long |>
          filter(PC == "PC3") |>
          top_n(15, abs(Contribution)) |>
          mutate(SNS_Term = reorder(SNS_Term, Contribution)) |>
          ggplot(aes(SNS_Term, Contribution, fill = SNS_Term)) +
          geom_col(show.legend = FALSE, alpha = 0.8) +
          theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
                axis.ticks.x = element_blank()) +
          labs(x = "Social Media Term",
               y = "Relative Importance to Principal Component",
               title = "Top 15 Contributors to PC3")
```



By repeating the above ggplot code for the four other principal components among the first five:

(Using functions to avoid repetition of code)

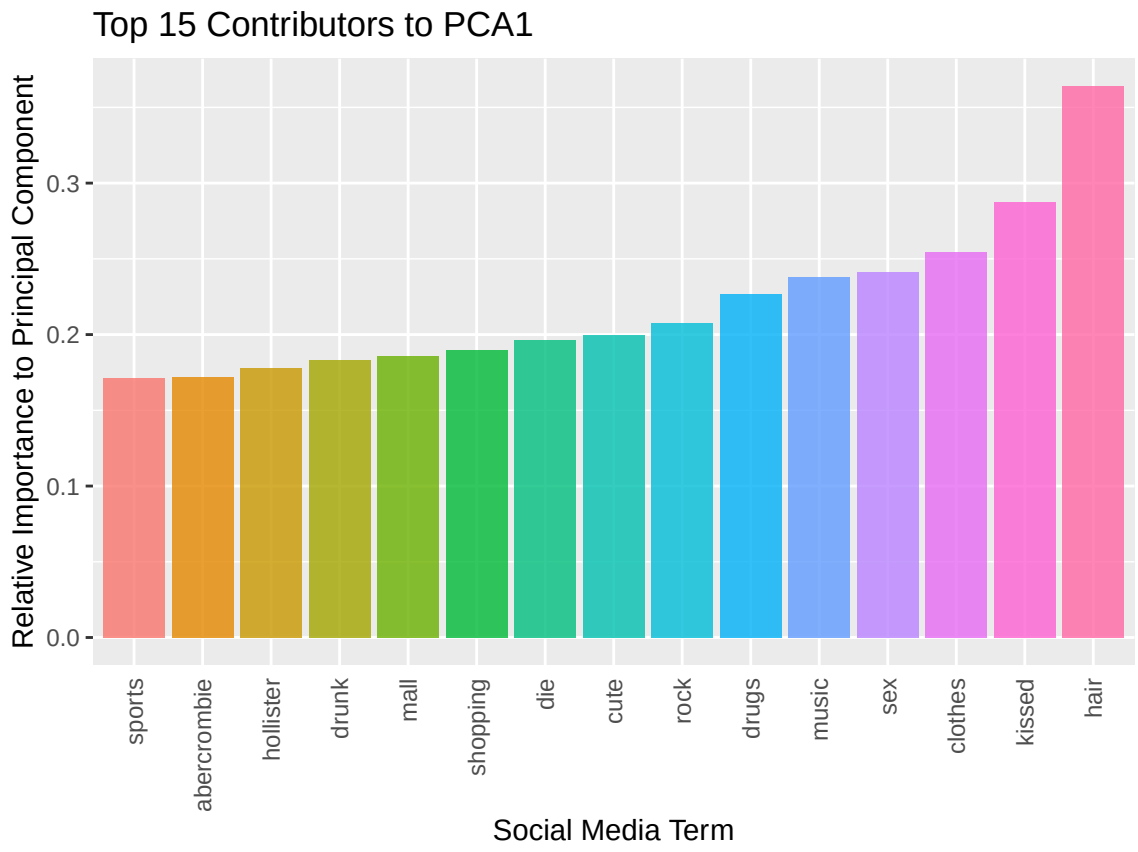
```
#generate data
get_data_from_pca <- function(pca) {
  print(pca)
  datos_pca <- sns_pca_long |>
          filter(PC == pca) |>
          top_n(15, abs(Contribution)) |>
          mutate(SNS_Term = reorder(SNS_Term, Contribution))
  return(datos_pca)
}

#plot data
plot_pca <- function(pca_data, pca) {
```

```
  pca_data |>
    ggplot() +
    aes(SNS_Term, Contribution, fill = SNS_Term) +
    geom_col(show.legend = FALSE, alpha = 0.8) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5),
          axis.ticks.x = element_blank()) +
    xlab("Social Media Term") +
    ylab("Relative Importance to Principal Component") +
    ggtitle(glue("Top 15 Contributors to {pca}"))  # string interpolation using glue
}
```

```
plot_pca(get_data_from_pca("PC1"), "PCA1")
```

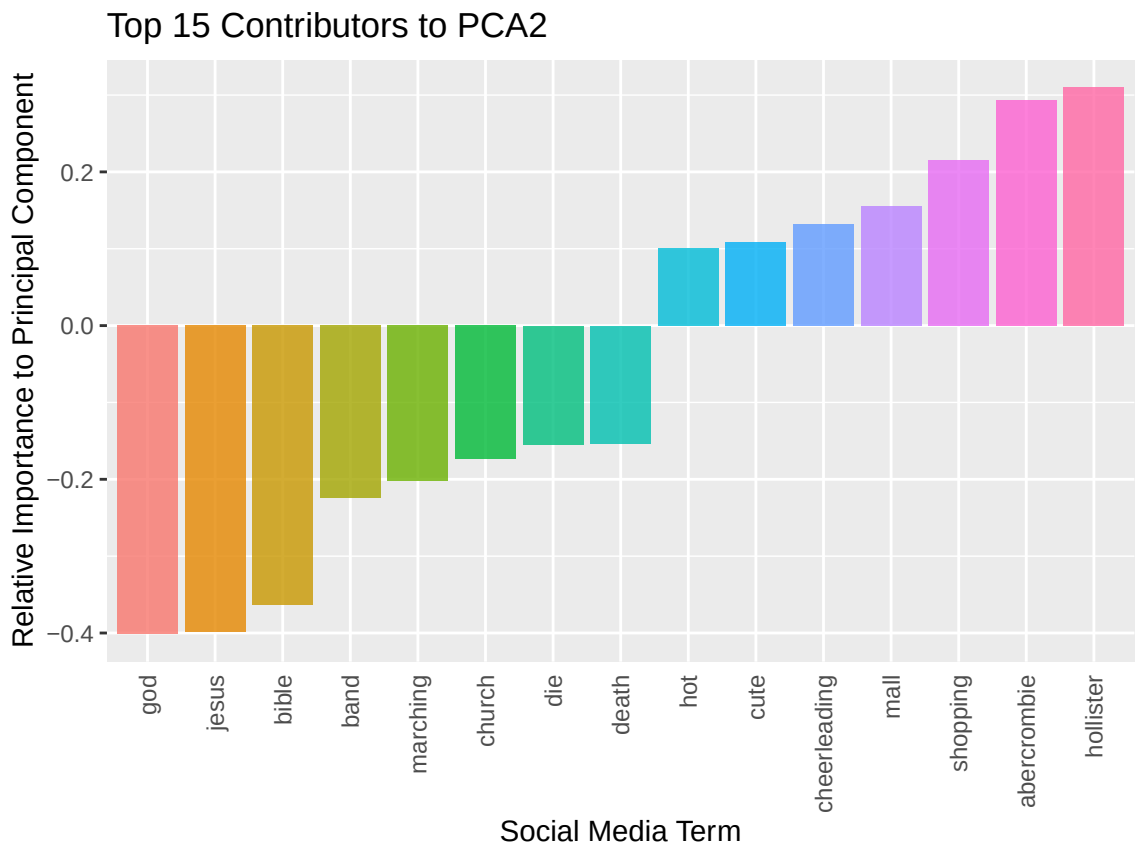
```
## [1] "PC1"
```




```
pca2 <- get_data_from_pca("PC2")
```

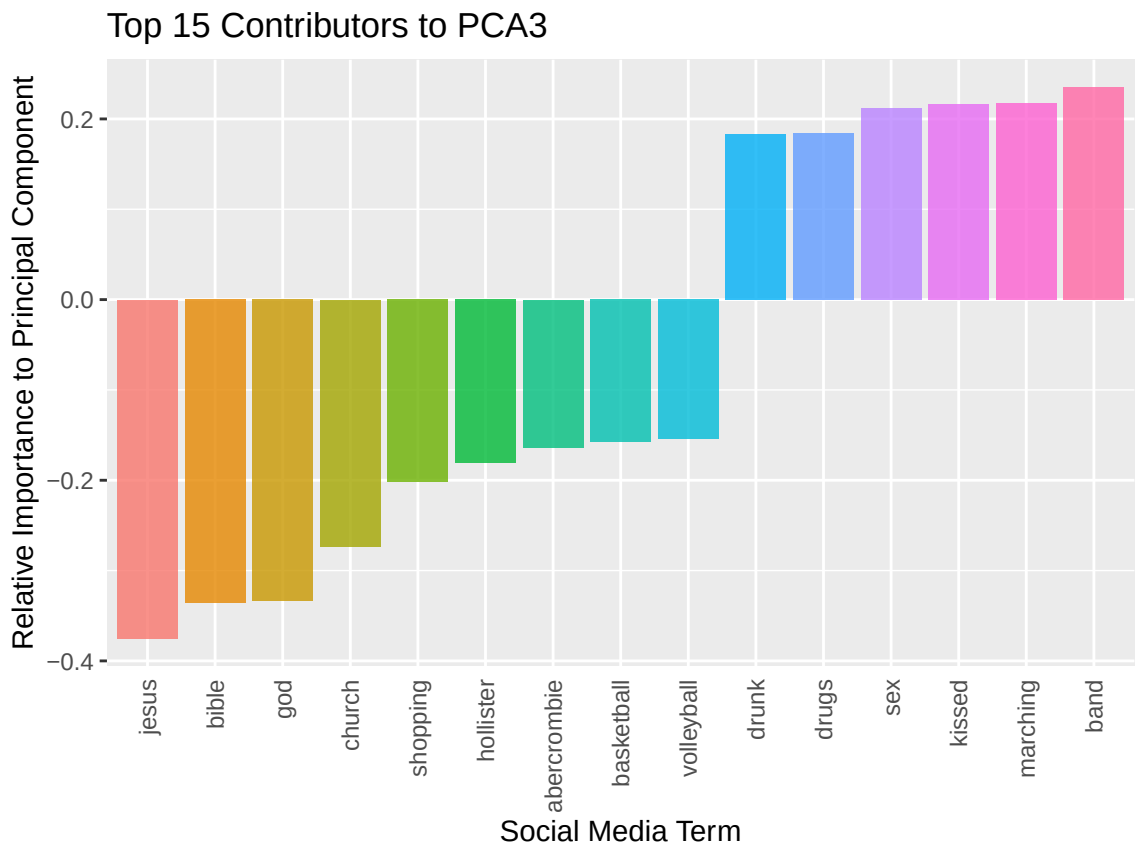
```
## [1] "PC2"
```

```
plot_pca(pca2, "PCA2")
```



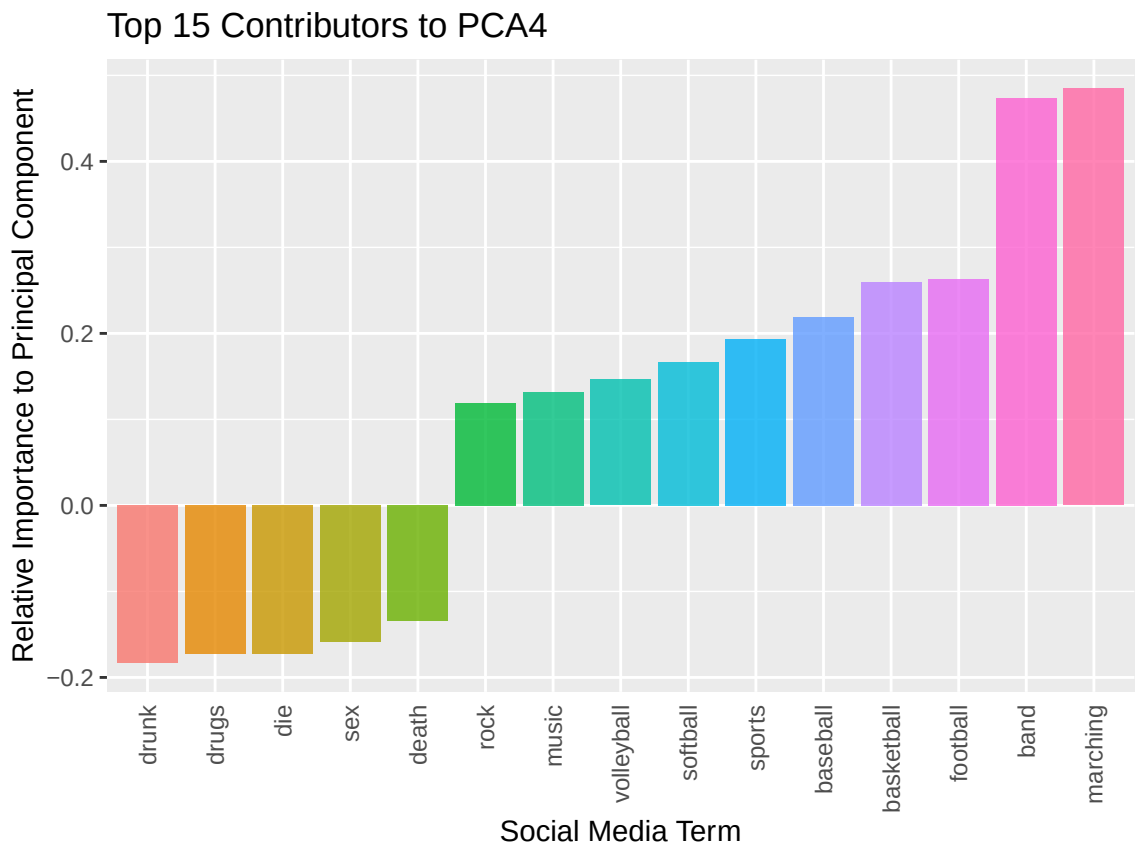

```
plot_pca(get_data_from_pca("PC3"), "PCA3")
```

```
## [1] "PC3"
```



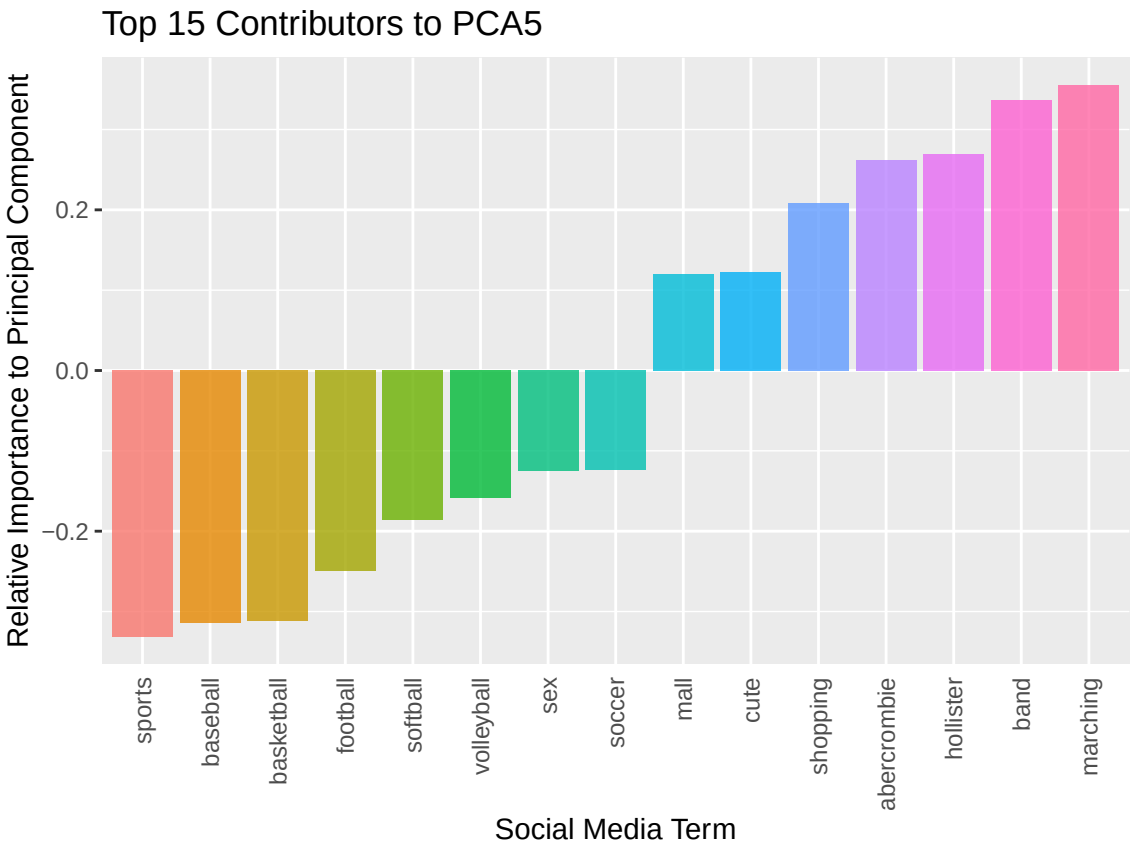

```
plot_pca(get_data_from_pca("PC4"), "PCA4")
```

```
## [1] "PC4"
```




```
plot_pca(get_data_from_pca("PC5"), "PCA5")
```

```
## [1] "PC5"
```



PC1 is particularly interesting, as every term has a positive impact; this may be distinguishing people who have anything versus nothing at all on their social media profiles. PC2 seems to favor shopping-related terms, while PC4 seems to be a combination of music and sports, without sex and drugs. Lastly, it seems that PC5 may be distinguishing between sports and non-sports-related terms. Examining the charts in this way will help to understand each component's impact on the predictive model.

An understanding of principal component analysis is of little value if the technique is not useful for building machine learning models.

By merging these components back into the original dataset, we can use them to make predictions about a profile's gender or number of friends. We'll begin by using the cbind() function to combine the first four columns of the original data frame with the transformed profile data from the PCA result:

```
sns_data_pca <- cbind(sns_data[1:4], sns_pca$x)
```

Next, we'll build a linear regression model predicting the number of social media friends as a function of the first five principal components.

```
m <- lm(friends ~ PC1 + PC2 + PC3 + PC4 + PC5, data = sns_data_pca)
m
```

```
##
## Call:
## lm(formula = friends ~ PC1 + PC2 + PC3 + PC4 + PC5, data = sns_data_pca)
##
## Coefficients:
```

```
## (Intercept)          PC1          PC2          PC3          PC4          PC5
##     30.1795       1.9857       0.9748      -2.5230       1.1160       0.8780
```

Because the value of the intercept is approximately 30.18, the average person in this dataset has about 30 friends. People with higher values of PC1, PC2, PC4, and PC5 are expected to have more friends, while higher values of PC3 are associated with fewer friends, assuming all else is equal.