Dimensionality Reduction for Visualization

INFO 526 Data Analysis and Visualization

Adriana Picoral

Our sight perception system does not allow us to see in more dimensions than three, yet a lot of the data we produce in the world has more than three dimensions. We at this point should be convinced that visualization can produce powerful insights into data, so the solution for the problem of multidimensionality is to reduce the data dimensions so it can be visualized on a 2d or 3d plot.

Dimensionality reduction combines original variables into fewer variables, removing redudant information. Each method has its own advantages and disadvantages, and which method of dimension reduction you should use depends largely on your data and your question.

Types of words for different types of books

For this tutorial we will be working with part-of-speech of words in descriptions of two different types of books: science fiction books and programming books.

The words for each book description were labeled for part of speech, and then these labels were count for each description. Our hypothesis is that different types of books will be described differently, some types of books will require descriptions with more adjectives (e.g., helpful, important) or more verbs (e.g., lives, traveled). The frequencies (number of adjectives, number of verbs, and so on) are then normalized by the total number of words in the description. So all frequencies are between 0 and 1.

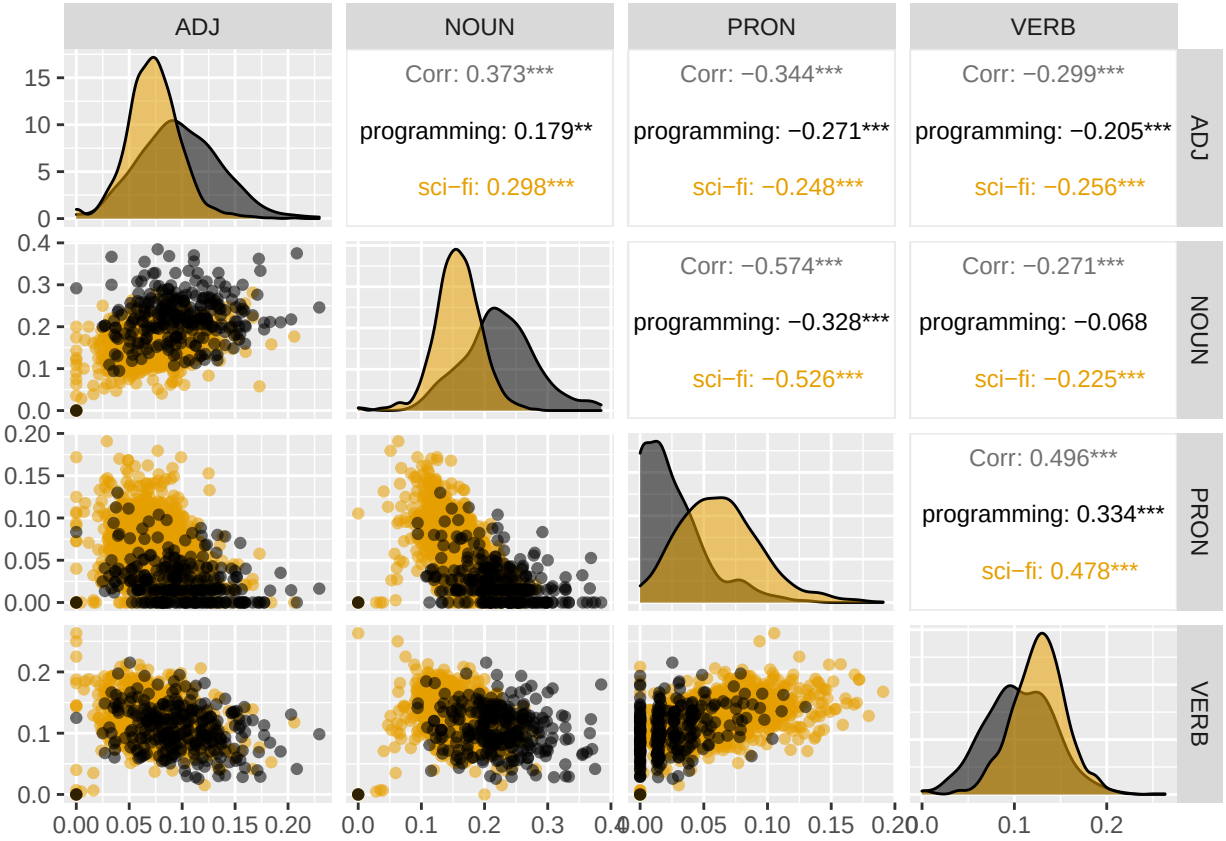
```
library(tidyverse)
book_descriptions <- read_csv("data/book_data.csv")
glimpse(book_descriptions)
```

```
## Rows: 1,521
## Columns: 16
## $ doc_id <chr> "text1", "text10", "text100", "text1000", "text1001", "text1002~
## $ type   <chr> "sci-fi", "sci-fi", "sci-fi", "sci-fi", "sci-fi", "sci-fi", "sc~
## $ ADJ    <dbl> 0.09090909, 0.06802721, 0.11718750, 0.05555556, 0.06122449, 0.1~
## $ ADP    <dbl> 0.09090909, 0.05442177, 0.09375000, 0.11111111, 0.02040816, 0.1~
## $ ADV    <dbl> 0.06951872, 0.04081633, 0.03906250, 0.05555556, 0.04081633, 0.0~
## $ AUX    <dbl> 0.04812834, 0.04761905, 0.04687500, 0.05555556, 0.07142857, 0.0~
## $ CCONJ  <dbl> 0.03743316, 0.04081633, 0.03125000, 0.00000000, 0.03061224, 0.0~
## $ DET    <dbl> 0.08556150, 0.09523810, 0.13281250, 0.11111111, 0.08163265, 0.1~
## $ INTJ   <dbl> 0.005347594, 0.013605442, 0.000000000, 0.000000000, 0.000000000~
## $ NOUN   <dbl> 0.13368984, 0.11564626, 0.20312500, 0.16666667, 0.09183673, 0.2~
## $ PART   <dbl> 0.02139037, 0.03401361, 0.03906250, 0.05555556, 0.08163265, 0.0~
## $ PRON   <dbl> 0.08021390, 0.06122449, 0.04687500, 0.00000000, 0.10204082, 0.0~
## $ PROPN  <dbl> 0.04812834, 0.06802721, 0.01562500, 0.11111111, 0.05102041, 0.0~
## $ PUNCT  <dbl> 0.14438503, 0.16326531, 0.11718750, 0.05555556, 0.13265306, 0.1~
## $ SCONJ  <dbl> 0.016042781, 0.027210884, 0.000000000, 0.000000000, 0.020408163~
## $ VERB   <dbl> 0.12834225, 0.15646259, 0.10156250, 0.16666667, 0.20408163, 0.0~
```

Overview of variables

We can quickly visualize the distribution and correlation of variables and all the pairs that are made among the variables using `ggpairs()` function from the `GGally` library.

```
library(GGally)
library(ggthemes)
ggpairs(book_descriptions,
        columns = c("ADJ", "NOUN", "PRON", "VERB"),
        aes(color = type, alpha = .3),
        upper = list(continuous = wrap("cor", size = 3))) +
  scale_color_colorblind() +
  scale_fill_colorblind() +
  theme()
```



At this point you should be familiar with scatterplots and density plots. In addition, the visualization above shows the correlation between variables. The plot above shows that sci-fi book descriptions (in yellow) seem to have fewer adjectives than programming book descriptions, fewer nouns, but more pronouns and verbs. Nouns (e.g., hero, book, language) usually have complementary distribution to pronouns (e.g., it, they, him, she) since these two word types act as subject and object in sentences (so the more of one word type, the fewer the other type). A total of four variables is not too much, so a plot like the one above is perfect fine to visualize all variables. However, it's not uncommon for data sets to have hundreds or even thousands of variables. Dimensionality reduction is useful not only for visualization, but other applications like machine learning.

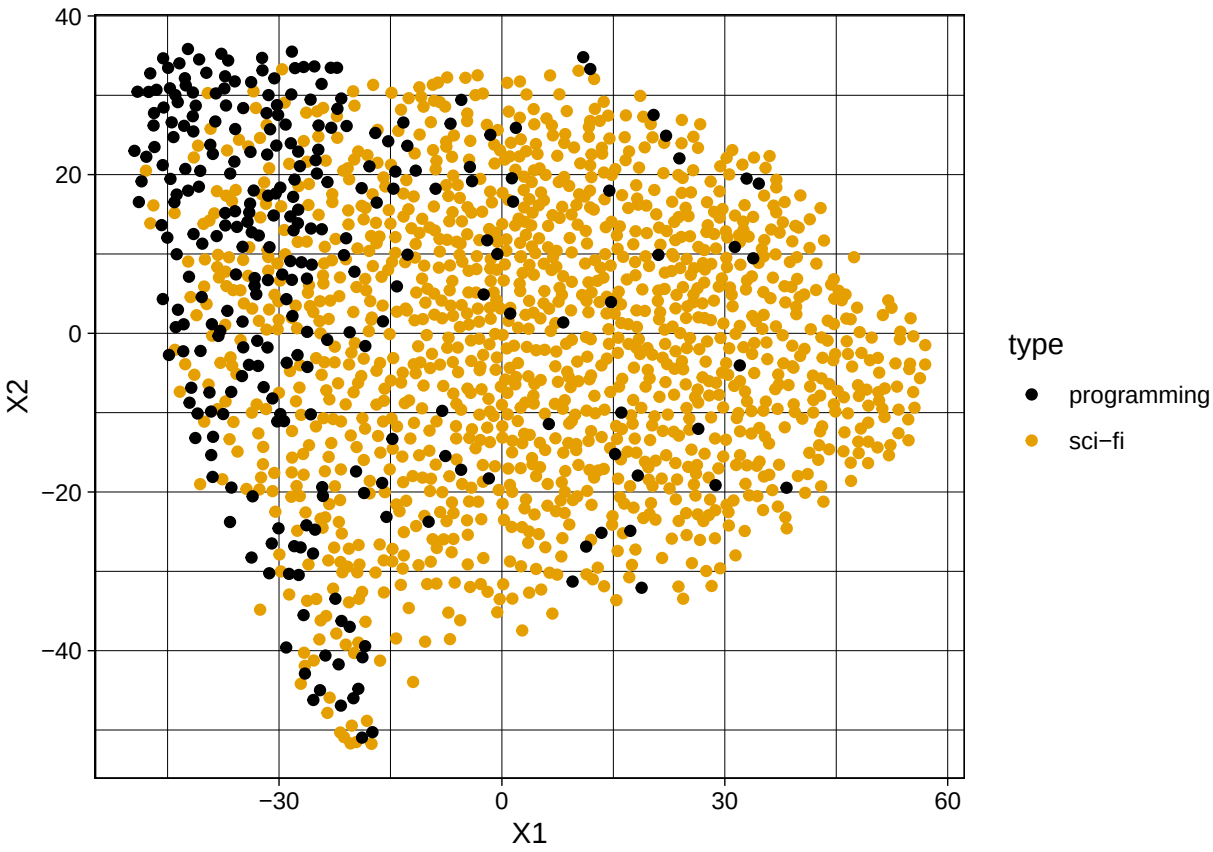
Uniform manifold approximation and projection (UMAP)

UMAP is a general-purpose dimension reduction algorithm. This projection method is computationally efficient and it best used with a large number of variables. Its main advantage is that it recovers well-separated clusters, using a relatively fast algorithm.

The disadvantage is that there's a number of hyperparameters that need to be adjusted to optimize clustering, such as number of neighbors, minimum distance, and spread. In adition to optimizing these hyperparameters, the data analyst would also play with which variables to include in the algorithm to produce better separated clusters.

Here's an example of how to run UMAP using the `uwot` library.

```
# load library
library(uwot)

# calculate dimensions based on all 16 numeric variables
umap_books <- umap(book_descriptions[, 1:16],
                   n_neighbors = 20,
                   min_dist = 5,
                   spread = 10)

# transform output (originally a vector) to a dataframe
umap_books_df <- umap_books %>%
  data.frame()

# add description type to dataframe
umap_books_df$type <- book_descriptions %>%
  pull(type)

# build visualization
umap_books_df %>%
  ggplot(aes(x = X1,
             y = X2,
             color = type)) +
  geom_point() +
  theme_linedraw() +
  scale_color_colorblind()
```

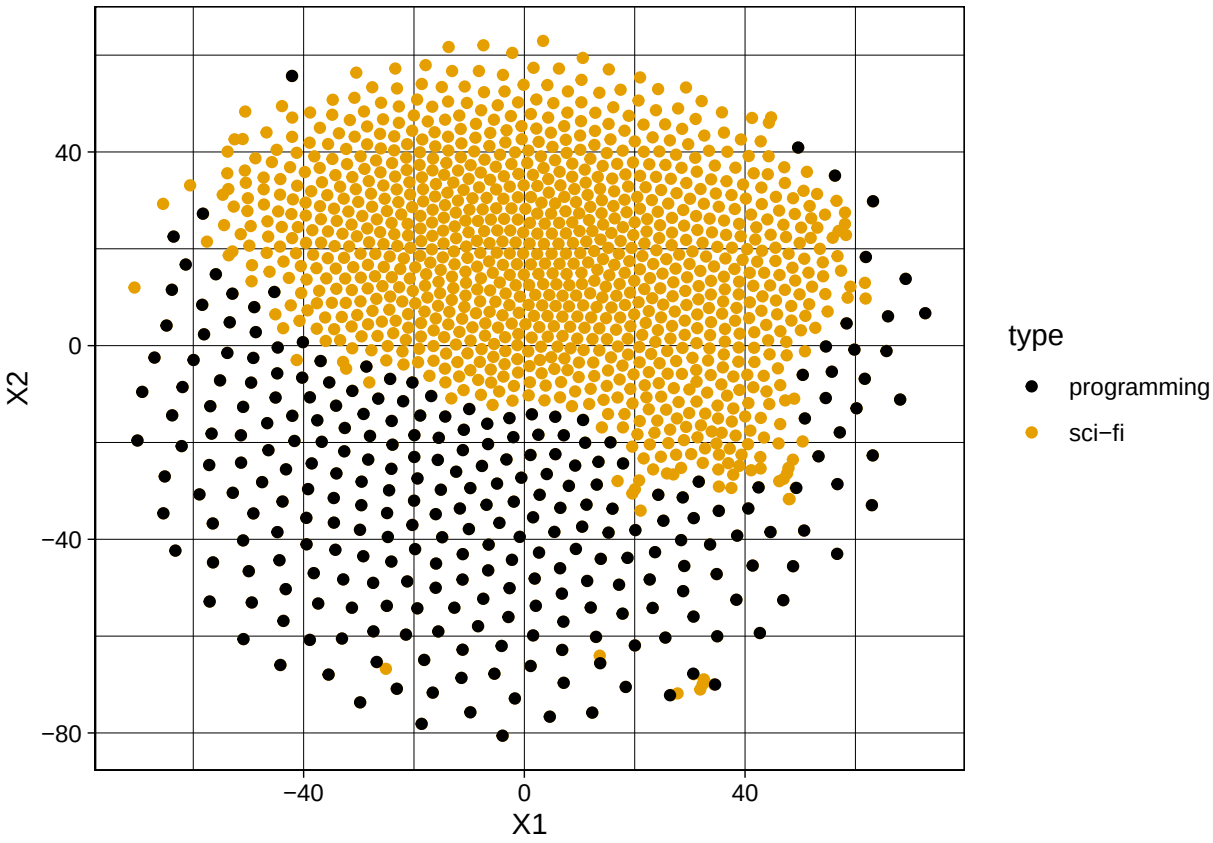


T-distributed stochastic neighbor embedding (t-SNE)

T-distributed Stochastic Neighbor Embedding (t-SNE) is similar to UMAP in the sense that its main goal is to split the data into clusters. While it usually does a better job in clustering than UMAP, it's computationally expensive. It also has a number of hyperparameters that need to be optimized to get better clustering.

```
# load library
library(Rtsne)

# set seed because there's a random element to this algorithm
set.seed(11)

# run algorithm with all 16 variables.
tsne_books <- Rtsne(book_descriptions[, 1:16],
                    pca = FALSE,
                    perplexity = 10,
                    theta = 0.0,
                    check_duplicates = FALSE)

# transform output in a data frame for visualization
tsne_books_df <- tsne_books$Y %>%
  data.frame()

# add description type to dataframe
tsne_books_df$type <- book_descriptions %>%
  pull(type)
```

```
# plot it
tsne_books_df %>%
  ggplot(aes(x = X1,
             y = X2,
             color = type)) +
  geom_point() +
  theme_linedraw() +
  scale_color_colorblind()
```



Principal Component Analysis (PCA)

PCA is one of the most used dimensionality reduction algorithm. It's a rotation methods, used to combine correlated variables into a set of linearly uncorrelated variables or dimensions (a.k.a. principal components). Its advantages is that the dimensions can be interpreted. For that reason, interpretability, we will limit our variables to four variables that we know have an impact on different types of texts.

```
# select a few variables
book_desc_variables <- book_descriptions %>%
  select(ADJ, NOUN, PRON, VERB)
```

We will use the function `prcomp` from base R to run pca. We don't need to scale and center our variables because they are all the same scale. We can run the function with its default parameters.

```
# run pca
prcomp_books <- prcomp(book_desc_variables)
```

The other advantage of PCA is that we can get the variance explained of original variable in our data.

```
# variance explained for all dimensions
data.frame(variable = c("ADJ", "NOUN", "PRON", "VERB"),
           variance_explained = (prcomp_books$sdev / sum(prcomp_books$sdev))) %>%
  kable(digits = 2)
```

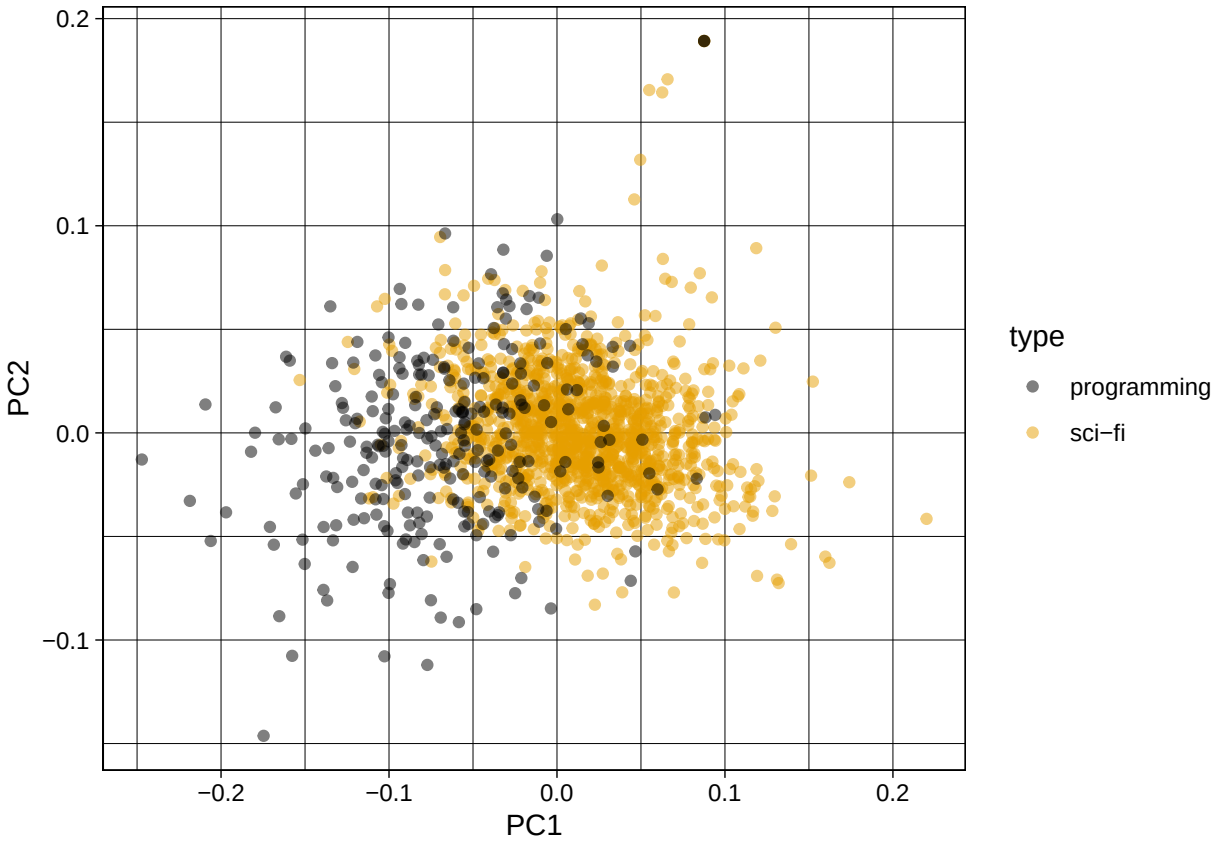
variable	variance_explained
ADJ	0.42
NOUN	0.24
PRON	0.19
VERB	0.16

Let's prepare our data for plotting.

```
# put output in a dataframe
pca_books_df <- prcomp_books$x %>%
  data.frame()

pca_books_df$type <- book_descriptions %>%
  pull(type)
```

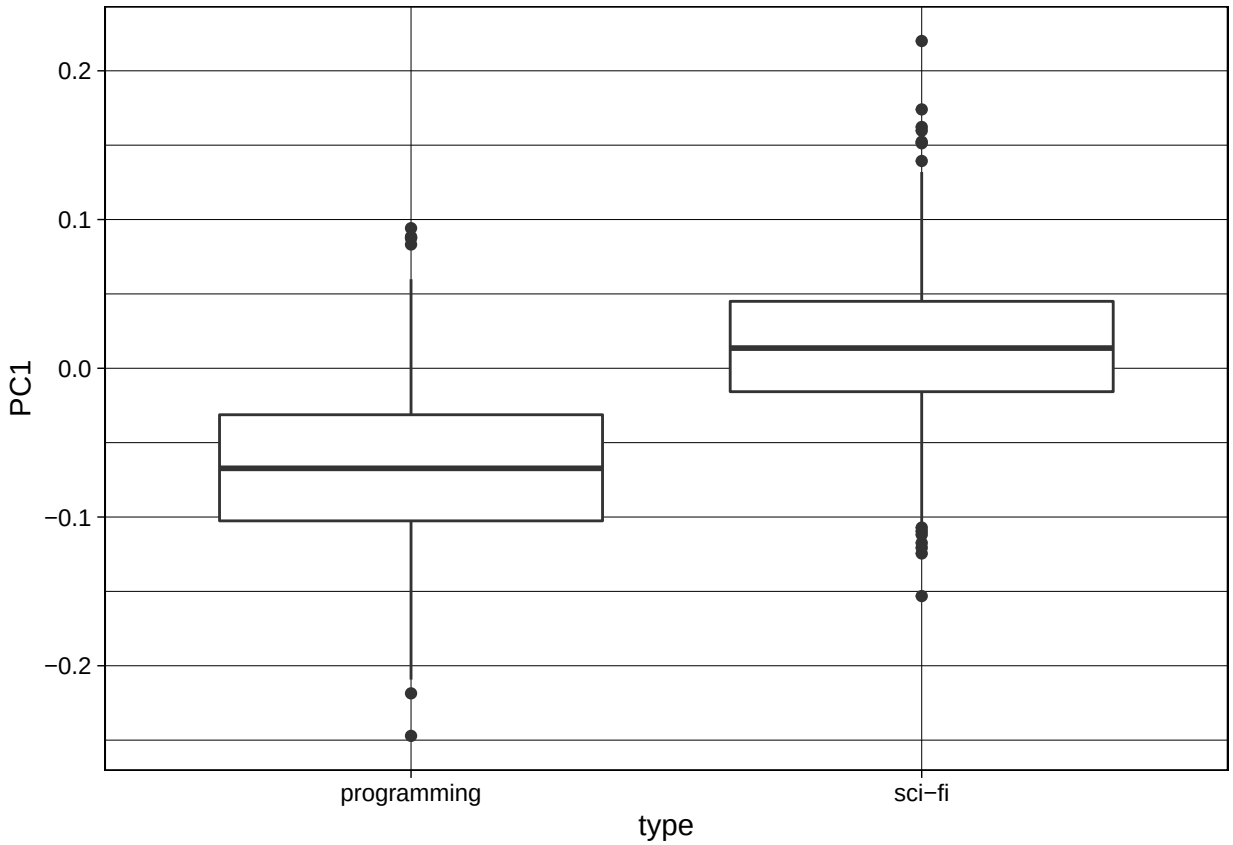
We can start with a scatterplot of the two principal components that explain the variance in the data more than the other two.

```
# plot it
pca_books_df %>%
  ggplot(aes(x = PC1,
             y = PC2,
             color = type)) +
  geom_point(alpha = .5) +
  theme_linedraw() +
  scale_color_colorblind()
```



It seems we can focus on the first dimension only.

```
# do a boxplot instead
pca_books_df %>%
  ggplot(aes(y = PC1, x = type)) +
  geom_boxplot() +
  theme_linedraw()
```



We can interpret what that dimension means by looking at how the original variables load on each dimension.

```
# how original variables load on each dimension
prcomp_books$rotation %>%
  data.frame() %>%
  kable(digits = 2)
```

	PC1	PC2	PC3	PC4
ADJ	-0.28	0.18	0.94	-0.12
NOUN	-0.77	-0.55	-0.09	0.30
PRON	0.48	-0.29	0.30	0.77
VERB	0.31	-0.76	0.17	-0.54