Class 19: Merging Datasets, Managing Factors, and Word Sentiments

MGSC 310

Prof. Jonathan Hersh

Class 19 Announcements

1. Quiz 7 due at midnight
2. Pset 5 due Tuesday, November 17 (posted)

Class 19: Outline

1. Merging Datasets
2. Wrangling Factor Levels
3. Word Sentiments

Merging Datasets – Primary Keys

Artist Table

Primary Key

Artist	Artist ID
Justin Bieber	1
Drake	2
The Smiths	3

- Real world datasets are stored in separate tables to save storage space
- The unique (row) identifier for every observation in a table is called the **primary key** (or primary keys)

Merging Datasets – Foreign Keys

Artist Table

Primary Keys

Artist Name	Artist ID
Justin Bieber	1
Drake	2
The Smiths	3

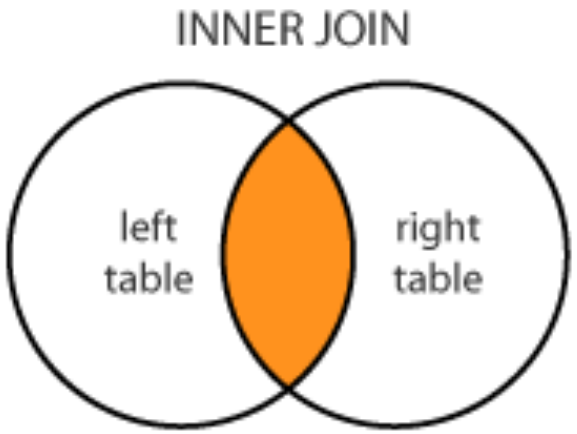
- When we want to merge in data from another table, we need to associate the primary key (artist ID in artist table) with the foreign key in the separate table (Artist ID in album table)
- These may have different names in different tables!

- Album Table

Foreign Key

Album Name	Album ID	Artist ID
Thank Me Later	1	2
Take Care	2	2
Nothing Was The Same	3	2
Views	4	2
Scorpion	5	2
Louder Than Bombs	6	3
Thank U Next	7	5

Inner Join: Only Keeps Data Matching Keys



- Inner join keeps only data that matches on primary and foreign key

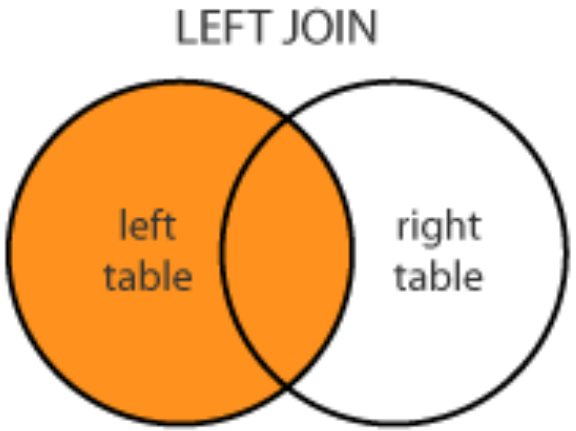
Artist Name	Artist ID
Justin Bieber	1
Drake	2
The Smiths	3

Album Name	Album ID	Artist ID
Thank Me Later	1	2
Take Care	2	2
Nothing Was The Same	3	2
Views	4	2
Scorpion	5	2
Louder Than Bombs	6	3
Thank U Next	7	5

Album Name	Album ID	Artist ID	Artist Name
Take Care	2	2	Drake
Nothing Was The Same	3	2	Drake
Views	4	2	Drake
Scorpion	5	2	Drake
Louder Than Bombs	6	3	The Smiths

- Since Justin Bieber has no albums in the album table, and Thank U Next has no associated artist in the artist table, these rows are not kept using an inner join

Left Join: Keeps Matching Keys and Non-Matching in Left Dataset



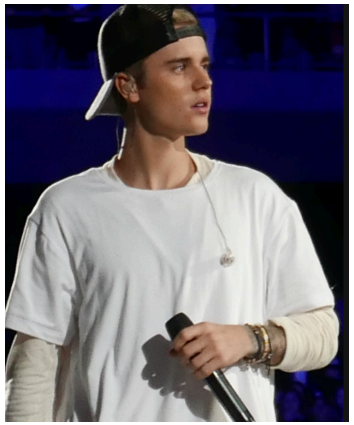
- Suppose artist is the left dataset and album is the right dataset in our merge statement

Artist Name	Artist ID
Justin Bieber	1
Drake	2
The Smiths	3

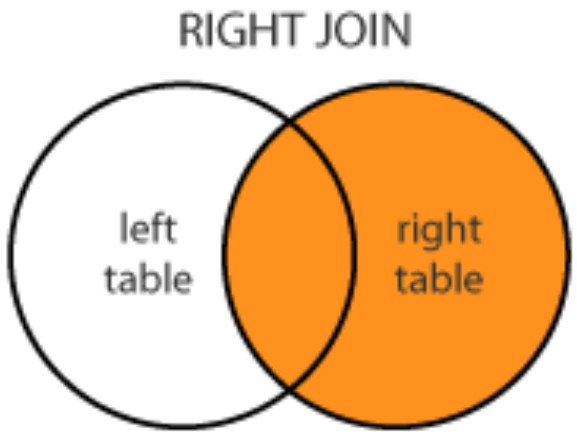
Album Name	Album ID	Artist ID
Thank Me Later	1	2
Take Care	2	2
Nothing Was The Same	3	2
Views	4	2
Scorpion	5	2
Louder Than Bombs	6	3
Thank U Next	7	5

Album Name	Album ID	Artist ID	Artist Name
NA	NA	1	Justin Bieber
Take Care	2	2	Drake
Nothing Was The Same	3	2	Drake
Views	4	2	Drake
Scorpion	5	2	Drake
Louder Than Bombs	6	3	The Smiths

- Left join keeps Bieber, but any of his album information is stored as missing – we don't know his albums?



Right Join: Keeps Matching Keys and Non-Matching in Right Dataset



- Suppose artist is the left dataset, and album is right dataset in our merge statement

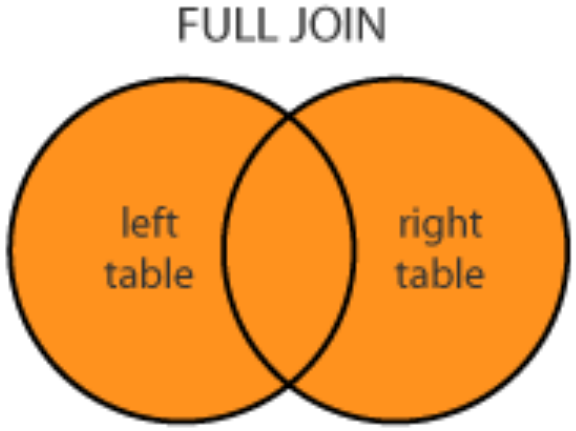
Artist Name	Artist ID
Justin Bieber	1
Drake	2
The Smiths	3

Album Name	Album ID	Artist ID
Thank Me Later	1	2
Take Care	2	2
Nothing Was The Same	3	2
Views	4	2
Scorpion	5	2
Louder Than Bombs	6	3
Thank U Next	7	5

Album Name	Album ID	Artist ID	Artist Name
Take Care	2	2	Drake
Nothing Was The Same	3	2	Drake
Views	4	2	Drake
Scorpion	5	2	Drake
Louder Than Bombs	6	3	The Smiths
Thank U Next	7	5	NA

- Right join keeps any non-matching in the right dataset (Thank U Next), but any of the artist information information is stored as missing – we don't know who made these albums

Full Join: Keeps Matching Keys and All Non-Matching Data



- Suppose artist is the left dataset, and album is right dataset in our merge statement

Artist Name	Artist ID
Justin Bieber	1
Drake	2
The Smiths	3

Album Name	Album ID	Artist ID
Thank Me Later	1	2
Take Care	2	2
Nothing Was The Same	3	2
Views	4	2
Scorpion	5	2
Louder Than Bombs	6	3
Thank U Next	7	5

Album Name	Album ID	Artist ID	Artist Name
NA	NA	1	Justin Bieber
Take Care	2	2	Drake
Nothing Was The Same	3	2	Drake
Views	4	2	Drake
Scorpion	5	2	Drake
Louder Than Bombs	6	3	The Smiths
Thank U Next	7	5	NA

- Full join keeps all matching and non-matching data.
- Lots of NAs are introduced!

Let’s get some dummy data for merging

```r
library(tidyverse)

artist_DF <- data.frame(
  artist_name = c("Justin Bieber",
                  "Drake",
                  "The Smiths"),
  artist_ID = 1:3
)
```

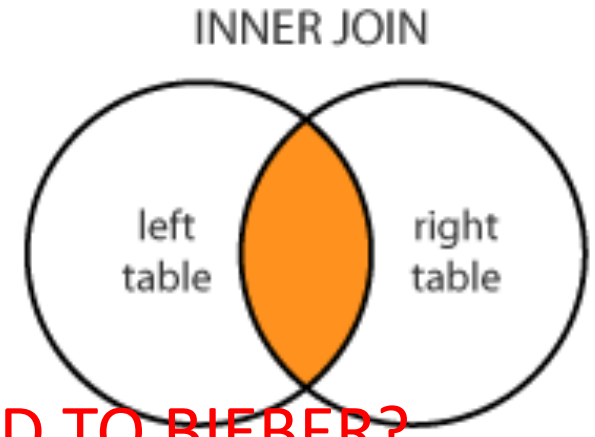
```
> artist_DF
          artist artist_ID
1 Justin Bieber         1
2          Drake         2
3     The Smiths         3
```

```r
album_DF <- data.frame(
  album_name = c("Thank Me Later", "Take Care",
                 "Nothing Was The Same", "Views",
                 "Scorpion",
                 "Strangeways Here We Come",
                 "Thank U Next"),
  album_ID = 1:7,
  artist_ID = c(rep(2,5),3,4)
)
```

```
> album_DF
                album_name album_ID artist_ID
1           Thank Me Later        1         2
2                Take Care        2         2
3     Nothing Was The Same        3         2
4                    Views        4         2
5                  Scorpion        5         2
6 Strangeways Here We Come        6         3
7           Thank U Next        7         4
```

Inner Join: Returns All Albums That Match on Artist ID

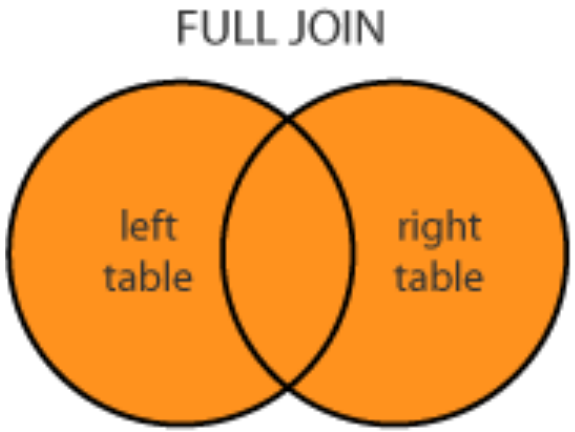
```
# inner join
DF <- inner_join(x = artist_DF,
                 y = album_DF,
                 by = "artist_ID")
```



WHAT HAPPENED TO BIEBER?

```
> merged_DF
      artist artist_ID                album_name album_ID
1      Drake         2              Thank Me Later        1
2      Drake         2                  Take Care         2
3      Drake         2       Nothing Was The Same         3
4      Drake         2                       Views        4
5      Drake         2                    Scorpion        5
6 The Smiths         3 Strangeways Here We Come         6
```

Full Join: Returns All Albums and All Artists, Regardless of Match



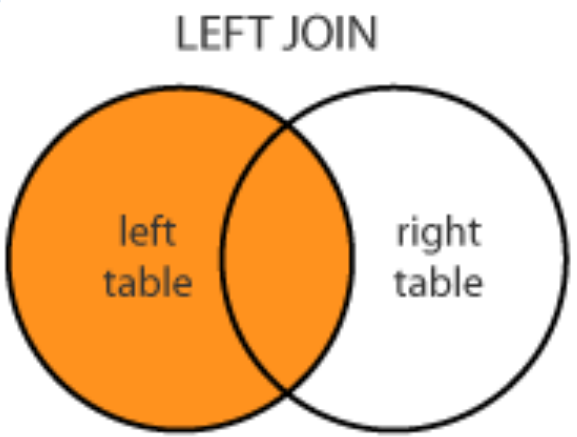
"NA"s introduced because information unknown across tables

```
# full join
merged_DF <-
  full_join(x = artist_DF,
            y = album_DF,
            by = "artist_ID")

merged_DF
```

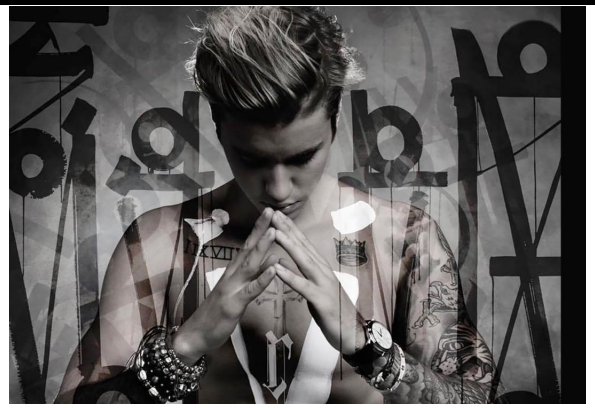
```
> merged_DF
         artist artist_ID                album_name album_ID
1 Justin Bieber         1                      <NA>       NA
2         Drake         2         Thank Me Later         1
3         Drake         2              Take Care         2
4         Drake         2  Nothing Was The Same         3
5         Drake         2                 Views         4
6         Drake         2               Scorpion         5
7    The Smiths         3 Strangeways Here We Come         6
8          <NA>         4          Thank U Next         7
```

Left Join: Returns All Artists, and Matching Album ID Information



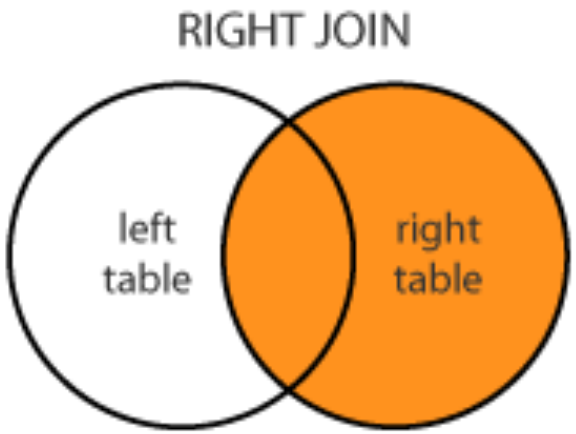
"NA"s for Justin's album because no information in album table on his masterpieces

```
# left join
merged_DF <-
  left_join(x = artist_DF,
            y = album_DF,
            by = "artist_ID")
```



```
> merged_DF
         artist artist_ID                album_name album_ID
1 Justin Bieber         1                      <NA>       NA
2         Drake         2         Thank Me Later          1
3         Drake         2             Take Care           2
4         Drake         2  Nothing Was The Same           3
5         Drake         2                 Views           4
6         Drake         2               Scorpion          5
7    The Smiths         3 Strangeways Here We Come         6
```

Right Join: Returns All Albums and Matching Artist Information




```
# right join
merged_DF <-
  right_join(x = artist_DF,
             y = album_DF,
             by = "artist_ID")
```

```
> merged_DF
      artist artist_ID                album_name album_ID
1      Drake         2          Thank Me Later        1
2      Drake         2               Take Care        2
3      Drake         2   Nothing Was The Same        3
4      Drake         2                  Views        4
5      Drake         2               Scorpion        5
6 The Smiths         3 Strangeways Here We Come        6
7       <NA>         4             Thank U Next        7
```

"NA"s for albums that don't have an artist match

Dplyr Joins

dplyr part of the tidyverse 0.8.3

Search... Intro Reference Articles News

Join two tbls together

Source: `R/join.r`

- Arguments
- Join types
- Grouping
- Examples

These are generic functions that dispatch to individual tbl methods - see the method documentation for details of individual data sources. `x` and `y` should usually be from the same data source, but if `copy` is `TRUE`, `y` will automatically be copied to the same source as `x`.

```
inner_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"),
  ...)

left_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)

right_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"),
  ...)

full_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)

semi_join(x, y, by = NULL, copy = FALSE, ...)
```

Class 19: Outline

1. Merging Datasets
2. **Wrangling Factor Levels**
3. Word Sentiments

Wine Labels Example

```r
#-----------------------------------------------
# Factor Levels - Wine Labels Example
#-----------------------------------------------
# see more on forcats https://forcats.tidyverse.org/
# recoding factor labels by hand
library('tidyverse')
WineDF <- read_csv(here::here("datasets","WineExample.csv"))
glimpse(WineDF)
summary(WineDF)

# View(WineDF)
table(WineDF$Variety, WineDF$Color)
```

```
> table(WineDF$Variety, WineDF$Color)

     RED WHITE
  V1  10     0
  V2   5     0
  V3   4     0
  V4   1     0
  V5   0     9
  V6   0     3
  V7   0     2
  V8   0     1
```

Relabeling Factor Levels Manually Using fct_recode()

```
# relabel by hand but only have to relabel changes
library('forcats')
WineDF <- WineDF %>%
  mutate(Variety_lbl1 = fct_recode(Variety,
                                   "OtherRed" = "V3",
                                   "OtherRed" = "V4",
                                   "OtherWhite" = "V7",
                                   "OtherWhite" = "V8"))

table(WineDF$Variety, WineDF$Variety_lbl1)
```

```
> table(WineDF$Variety, WineDF$Variety_lbl1)

    V1 V2 OtherRed V5 V6 OtherWhite
  V1 10  0        0  0  0          0
  V2  0  5        0  0  0          0
  V3  0  0        4  0  0          0
  V4  0  0        1  0  0          0
  V5  0  0        0  9  0          0
  V6  0  0        0  0  3          0
  V7  0  0        0  0  0          2
  V8  0  0        0  0  0          1
```

Reordering Factor Levels By Frequency

```r
# reorder factors by frequency of values
WineDF <- WineDF %>%
  mutate(Variety_lbl2 = fct_infreq(Variety))

table(WineDF$Variety, WineDF$Variety_lbl2)

levels(WineDF$Variety_lbl2)
```

```
> table(WineDF$Variety, WineDF$Variety_lbl2)

     V1 V5 V2 V3 V6 V7 V4 V8
  V1 10  0  0  0  0  0  0  0
  V2  0  0  5  0  0  0  0  0
  V3  0  0  0  4  0  0  0  0
  V4  0  0  0  0  0  0  1  0
  V5  0  9  0  0  0  0  0  0
  V6  0  0  0  0  3  0  0  0
  V7  0  0  0  0  0  2  0  0
  V8  0  0  0  0  0  0  0  1
> levels(WineDF$Variety_lbl2)
[1] "V1" "V5" "V2" "V3" "V6" "V7" "V4" "V8"
```

“Lumping” Factors Into Most Common and Creating an “Other” Category for the Rest

```r
# lump into top 5 factors
WineDF <- WineDF %>%
  mutate(Variety_lbl3 = fct_lump(Variety, 5))

table(WineDF$Variety, WineDF$Variety_lbl3)
```

```
> table(WineDF$Variety, WineDF$Variety_lbl3)

    V1 V2 V3 V5 V6 Other
  V1 10  0  0  0  0     0
  V2  0  5  0  0  0     0
  V3  0  0  4  0  0     0
  V4  0  0  0  0  0     1
  V5  0  0  0  9  0     0
  V6  0  0  0  0  3     0
  V7  0  0  0  0  0     2
  V8  0  0  0  0  0     1
```

For More See forcats Package

forcats part of the tidyverse 0.5.0

Get started Reference News

Overview

R uses **factors** to handle categorical variables, variables that have a fixed and known set of possible values. Factors are also helpful for reordering character vectors to improve display. The goal of the **forcats** package is to provide a suite of tools that solve common problems with factors, including changing the order of levels or the values. Some examples include:

- `fct_reorder()`: Reordering a factor by another variable.
- `fct_infreq()`: Reordering a factor by the frequency of values.
- `fct_relevel()`: Changing the order of a factor by hand.
- `fct_lump()`: Collapsing the least/most frequent values of a factor into “other”.

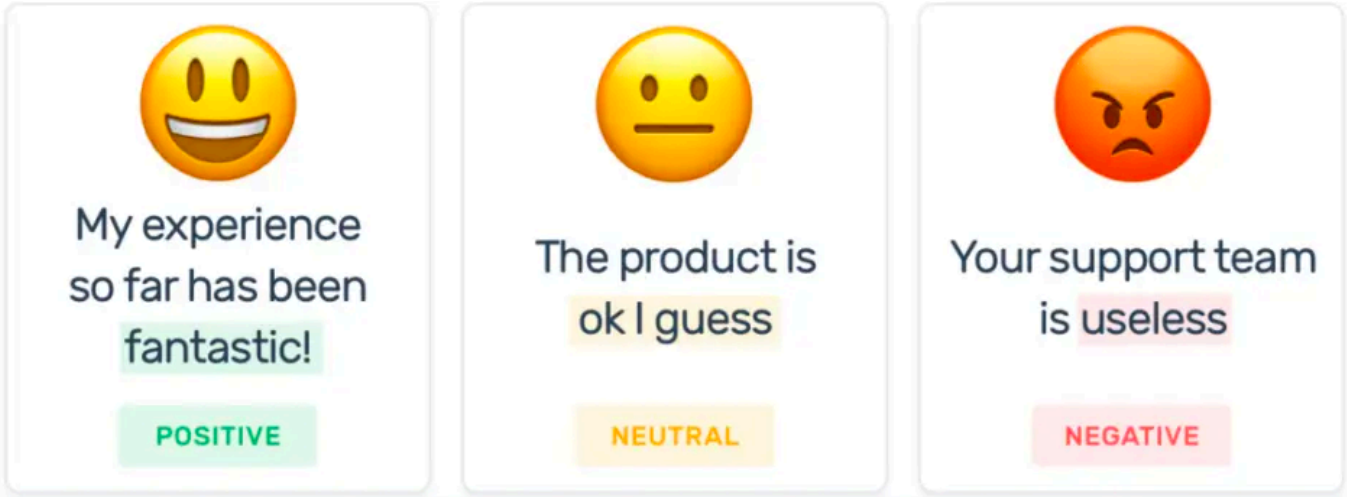
You can learn more about each of these in `vignette("forcats")`. If you’re new to factors, the best place to start is the chapter on factors in R for Data Science.

Class 19: Outline

1. Merging Datasets
2. Wrangling Factor Levels
3. **Word Sentiments**

Word Sentiments

- Existing Libraries have mapped each word to a given “sentiment”
- Sentiment is the average feeling invoked by a word



Extracting Sentiments from Words Using 'Sentimentr'

```
#-------------------------------------------------
# Text Sentiment
#-------------------------------------------------
library('devtools')
devtools::install_github("trinker/sentimentr")


# average sentiment score
library('sentimentr')
library('tidyverse')

sentiment_by("I am very scared of the dark")

sentiment('I was very scared of the dark.
          Now I glow at night and life is amazing')
```

```
> sentiment_by("I am very scared of the dark")
   element_id word_count sd ave_sentiment
1:          1          7 NA    -0.7483697
> sentiment('I was very scared of the dark.
+           Now I glow at night and life is amazing')
   element_id sentence_id word_count  sentiment
1:          1           1          7 -0.7483697
2:          1           2          9  0.2500000
```

- Sentence 1 has negative sentiment
- Sentence 2 has positive sentiment

Which Words Affect Sentiment?

```
"I am very scared of the dark" %>%
  extract_sentiment_terms()

"Now I glow at night and life is amazing" %>%
  extract_sentiment_terms()

"and I was like baby, baby, baby oh" %>%
  extract_sentiment_terms()
```

```
> "I am very scared of the dark" %>%
+   extract_sentiment_terms()
   element_id sentence_id    negative
1:          1           1 scared,dark
>
> "Now I glow at night and life is amazing" %>%
+   extract_sentiment_terms()
   element_id sentence_id     positive
1:          1           1 glow,amazing
>
> "and I was like baby, baby, baby oh" %>%
+   extract_sentiment_terms()
   element_id sentence_id       positive
1:          1           1 baby,baby,baby
```

Visualizing Sentiment Scores: Red Negative Sentiment Green Positive

```
She woke me up daily.
Don't need no Starbucks.
She made my heart pound.
And skip a beat when I see her in the street and.
At school on the playground.
But I really wanna see her on the weekend.
She know she got me dazin'.
'Cause she was so amazin'.
And now my heart is breakin'.
But I just keep on sayin'.
Baby, baby, baby oh.
Like baby, baby, baby no.
Like baby, baby, baby oh.
I thought you'd always be mine (mine).
Baby, baby, baby oh.
Like baby, baby, baby no.
Like baby, baby, baby ooh.
I thought you'd always be mine.
Now I'm gone.
Yeah, yeah, yeah.
Yeah, yeah, yeah (now I'm all gone).
Yeah, yeah, yeah.
Yeah, yeah, yeah (now I'm all gone).
Yeah, yeah, yeah.
Yeah, yeah, yeah (now I'm all gone).
Gone, gone, gone, I'm gone." %>%
  sentiment_by(by = NULL) %>% highlight()
```

1: ***+.470***

Ooh whoa, ooh whoa, ooh whoa. You know you love me, I know you care. Just shout whenever and I'll be there. You are my love, you are my heart. And we will never, ever, ever be apart. Are we an item? Girl quit playin'. We're just friends, what are you sayin'. Said there's another, look right in my eyes. My first love, broke my heart for the first time. And I was like baby, baby, baby oh. Like baby, baby, baby no. Like baby, baby, baby oh. I thought you'd always be mine (mine). Baby, baby, baby oh. Like baby, baby, baby no. Like baby, baby, baby ooh. I thought you'd always be mine. Oh for you, I would have done whatever. And I just can't believe we ain't together. And I wanna play it cool. But I'm losin' you. I'll buy you anything. I'll buy you any ring. And I'm in pieces, baby fix me. And just shake me, 'til you wake me from this bad dream. I'm goin' down, down, down, down. And I can't just believe my first love won't be around. And I'm like baby, baby, baby oh. Like baby, baby, baby no. Like baby, baby, baby oh. I thought you'd always be mine (mine). Baby, baby, baby oh. Like baby, baby, baby no. Like baby, baby, baby ooh. I thought you'd always be mine. Luda, when I was thirteen, I had my first love. There was nobody that compared to my baby. And nobody came between us no one could ever come above. She had me goin' crazy. Oh, I was starstruck. She woke me up daily. Don't need no Starbucks. She made my heart pound. And skip a beat when I see her in the street and. At school on the playground. But I really wanna see her on the weekend. She know she got me dazin'. 'Cause she was so amazin'. And now my heart is breakin'. But I just keep on sayin'. Baby, baby, baby oh. Like baby, baby, baby no. Like baby, baby, baby oh. I thought you'd always be mine (mine). Baby, baby, baby oh. Like baby, baby, baby no. Like baby, baby, baby ooh. I thought you'd always be mine. Now I'm gone. Yeah, yeah, yeah. Yeah, yeah, yeah (now I'm all gone). Yeah, yeah, yeah. Yeah, yeah, yeah (now I'm all gone). Yeah, yeah, yeah. Yeah, yeah, yeah (now I'm all gone). Gone, gone, gone, I'm gone.

Working With Text: tidytext Package

README.md

tidytext: Text mining using tidy tools

Authors: Julia Silge, David Robinson

R-CMD-check passing | CRAN 0.2.6 | CRAN OK | coverage 84% | DOI 10.5281/zenodo.4041568
JOSS 10.21105/joss.00037 | downloads 37K/month | downloads 1.2M

Using tidy data principles can make many text mining tasks easier, more effective, and consistent with tools already in wide use. Much of the infrastructure needed for text mining with tidy data frames already exists in packages like dplyr, broom, tidyr, and ggplot2. In this package, we provide functions and supporting data sets to allow conversion of text to and from tidy formats, and to switch seamlessly between tidy tools and existing text mining packages. Check out our book to learn more about text mining using tidy data principles.

Installation

You can install this package from CRAN:

```
install.packages("tidytext")
```

Or you can install the development version from GitHub with remotes:

```
library(remotes)
install_github("juliasilge/tidytext")
```

Working With Text: tidytext Package

Text Mining with R: A Tidy Approach

Search

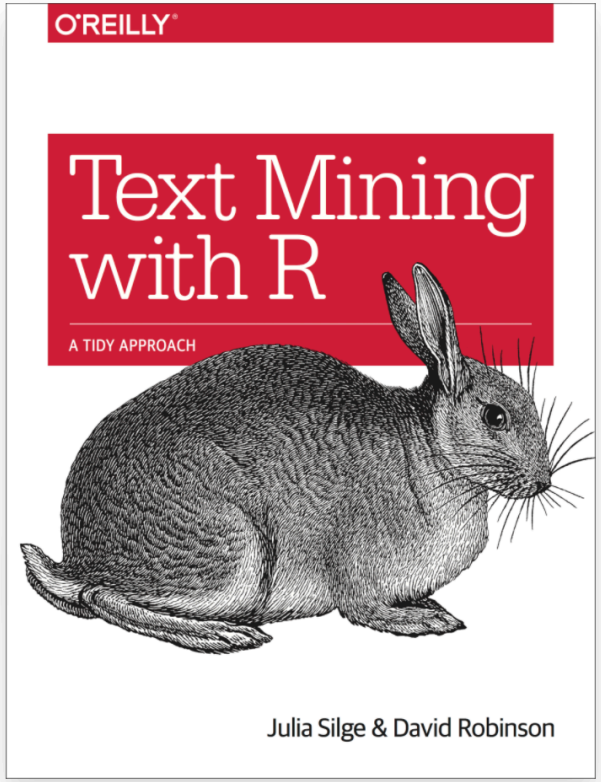
Table of contents

View book source

Welcome to Text Mining with R

This is the website for *Text Mining with R*! Visit the GitHub repository for this site, find the book at O'Reilly, or buy it on Amazon.

Preface »



Breakout Rooms

- Meet with groups for 5-10 minutes