But When Sweden Gets 12 Points From Norway, It's Clearly Just Good Taste: The Determinants of Eurovision Success

Lauren Gilbert

Database construction

I am attempting to use prior data from the Eurovision Song Contest to predict future outcomes (particularly the results of the 2019 edition of the contest, scheduled for 14-18 May).

The difficulty of this project will lie primarily in the data wranging, with data coming from various sources. The Kaggle dataset includes the scores to and from each country 1975-2018, but does not include language or lyrics information. For this, we must scrape the Wikipedia pages for each contest to collect language and title information. This is done below.

In the (forthcoming) analysis, I will be including covariates for:

- Distance between countries

To calculate this, I matched a Kaggle dataset on capitals to the countries competing in each contest. I then used the `geosphere` package to calculate capital-to-capital distance.

- Language of song

This is included in the Wikipedia data for each contest; I merely had to clean it up a bit (below). I have not yet decided how I will code multi-language songs.

- Themes (based on lyrics)

I originally attempted (unsuccessfully) to add lyrics using a Kaggle database of lyrics from Metrolyrics. However, this resulted in no matches, and further investigation revealed that artists were almost never tagged correctly, so this wasn't workable. I emailed (Diggiloo Thrush)[http://www.diggiloo.net/] for their database, and then realized that Eurovision World had set up their website in a very organized fashion; I added to my function matching Wikipedia information scraped lyrics from Eurovision World.

- Countries that are neighbors

I use the COW definition of neighbor, as coded in this Github JSON document.

Covariates I have considered but do not appear here:

- Religious/ethnic/linguistic similarity. I have not found a good dataset to use for these.
- "Song quality". Some papers use various measures of song quality; perhaps number of Youtube views prior to the contest would be good here? But that would mean I need numbers of youtube views *prior to the contest* for prior years, and clearly this would only work for Youtube-era songs. (I may write to Eurovision for this?)

The construction of my database is below:

```
scores <- read.xlsx("eurovision_song_contest_1975_2018v2.xlsx")
scores <- subset(scores, scores$Final == "f")

addyear <- function(year, html)
{
  str <- paste("https://en.wikipedia.org/wiki/Eurovision_Song_Contest_", toString(year), sep = "")
```

```
  wikipedia_page<-read_html(str)
  section_of_wikipedia<-html_node(wikipedia_page, xpath=html)
  yeartable <- html_table(section_of_wikipedia)
  Year <- rep(year, nrow(yeartable))
  colnames(yeartable) <- c("Draw", "To", "Artist", "Song", "Language", "Place", "Points")
  yeartable <- cbind(yeartable, Year)
  for (i in 1:length(yeartable$To))
  {
    slug <- str_replace_all(tolower(yeartable$To[i])," ","-")
    slug <- ifelse(slug == "bosnia-and-herzegovina", "bosnia-herzegovina", slug)
    slug <- ifelse(slug == "macedonia", "north-macedonia", slug)
    slug <- ifelse(slug == "serbia-and-montenegro2", "serbia-montenegro", slug)
    slug <- ifelse(slug == "serbia-and-montenegro", "serbia-montenegro", slug)
    slug <- ifelse(slug == "croatia2", "croatia", slug)
    slug <- ifelse(slug == "west-germany", "germany", slug)
    url <- paste("https://eurovisionworld.com/eurovision/", toString(year), "/", slug, sep = "")
    webpage <- read_html(url)
    lyrics <- html_nodes(webpage, xpath = '//*[@id="lyrics_0"]')
    lyrics <- html_text(lyrics)
    yeartable$lyrics[i] <- lyrics
  }
  return(yeartable)
}

# Sadly I can't automate this because the table isn't in the same part of the page for every year.
process2018 <- addyear(2018, '//*[@id="mw-content-text"]/div/table[6]')
process2017 <- addyear(2017, '//*[@id="mw-content-text"]/div/table[6]')
process2016 <- addyear(2016, '//*[@id="mw-content-text"]/div/table[6]')
process2015 <- addyear(2015, '//*[@id="mw-content-text"]/div/table[6]')
process2014 <- addyear(2014, '//*[@id="mw-content-text"]/div/table[6]')
process2013 <- addyear(2013, '//*[@id="mw-content-text"]/div/table[6]')
process2012 <- addyear(2012, '//*[@id="mw-content-text"]/div/table[5]')
process2011 <- addyear(2011, '//*[@id="mw-content-text"]/div/table[6]')
process2010 <- addyear(2010, '//*[@id="mw-content-text"]/div/table[6]')
process2009 <- addyear(2009, '//*[@id="mw-content-text"]/div/table[6]')
process2008 <- addyear(2008, '//*[@id="mw-content-text"]/div/table[5]')
process2007 <- addyear(2007, '//*[@id="mw-content-text"]/div/table[3]')
process2006 <- addyear(2006, '//*[@id="mw-content-text"]/div/table[3]')
process2005 <- addyear(2005, '//*[@id="mw-content-text"]/div/table[3]')
process2004 <- addyear(2004, '//*[@id="mw-content-text"]/div/table[3]')
process2003 <- addyear(2003, '//*[@id="mw-content-text"]/div/table[3]')
process2002 <- addyear(2002, '//*[@id="mw-content-text"]/div/table[4]')
process2001 <- addyear(2001, '//*[@id="mw-content-text"]/div/table[2]')
process2000 <- addyear(2000, '//*[@id="mw-content-text"]/div/table[3]')
process1999 <- addyear(1999, '//*[@id="mw-content-text"]/div/table[3]')
process1998 <- addyear(1998, '//*[@id="mw-content-text"]/div/table[3]')
process1997 <- addyear(1997, '//*[@id="mw-content-text"]/div/table[3]')
process1996 <- addyear(1996, '//*[@id="mw-content-text"]/div/table[4]')
process1995 <- addyear(1995, '//*[@id="mw-content-text"]/div/table[3]')
process1994 <- addyear(1994, '//*[@id="mw-content-text"]/div/table[3]')
process1993 <- addyear(1993, '//*[@id="mw-content-text"]/div/table[4]')
process1992 <- addyear(1992, '//*[@id="mw-content-text"]/div/table[3]')
process1991 <- addyear(1991, '//*[@id="mw-content-text"]/div/table[4]')
```

```
process1990 <- addyear(1990, '//*[@id="mw-content-text"]/div/table[2]')
process1989 <- addyear(1989, '//*[@id="mw-content-text"]/div/table[2]')
process1988 <- addyear(1988, '//*[@id="mw-content-text"]/div/table[3]')
process1987 <- addyear(1987, '//*[@id="mw-content-text"]/div/table[2]')
process1986 <- addyear(1986, '//*[@id="mw-content-text"]/div/table[2]')
process1985 <- addyear(1985, '//*[@id="mw-content-text"]/div/table[4]')
process1984 <- addyear(1984, '//*[@id="mw-content-text"]/div/table[3]')
process1983 <- addyear(1983, '//*[@id="mw-content-text"]/div/table[3]')
process1982 <- addyear(1982, '//*[@id="mw-content-text"]/div/table[2]')
process1981 <- addyear(1981, '//*[@id="mw-content-text"]/div/table[3]')
process1980 <- addyear(1980, '//*[@id="mw-content-text"]/div/table[3]')
process1979 <- addyear(1979, '//*[@id="mw-content-text"]/div/table[2]')
process1978 <- addyear(1978, '//*[@id="mw-content-text"]/div/table[2]')
process1977 <- addyear(1977, '//*[@id="mw-content-text"]/div/table[2]')
process1976 <- addyear(1976, '//*[@id="mw-content-text"]/div/table[2]')
process1975 <- addyear(1975, '//*[@id="mw-content-text"]/div/table[2]')

songtitles <- rbind(process2018, process2017, process2016, process2015, process2014, process2013, proces

songtitles <- songtitles %>% distinct()

# Wikipedia is not always consistent
songtitles$Language <- revalue(songtitles$Language, c("Serbo-Croatian2" = "Serbo-Croatian", "Italian3" =

# Country names are even less consistent.
scores$To <- revalue(scores$To, c("The Netherlands"="Netherlands", "Bosnia & Herzegovina"="Bosnia and H
songtitles$To <- revalue(songtitles$To, c("West Germany"="Germany", "Serbia and Montenegro2" = "Serbia

eurovision <- merge(scores, songtitles, by=c("Year", "To"), all.x=TRUE)

capitals <- read.csv("concap.csv")
eurovision <- merge(eurovision, capitals, by.x="From", by.y = "CountryName")
eurovision <- merge(eurovision, capitals, by.x="To", by.y = "CountryName")
eurovision$dist <- distGeo(data.frame(eurovision$CapitalLongitude.x, eurovision$CapitalLatitude.x), data
```

```
json_data <- fromJSON(file = "country_adj_fullname.json")
json_data$Serbia <- c("Hungary", "Romania", "Bulgaria", "Macedonia", "Croatia", "Bosnia and Herzegovina
names(json_data)[names(json_data) == "Russian Federation"] <- "Russia"
eurovision$neighbor <- 0
for (i in 1:length(eurovision$To))
{
  if (eurovision$From[i] %in% json_data[[(match(eurovision$To[i], names(json_data)))]])
  {
    eurovision$neighbor[i] <- 1
  }
}
```

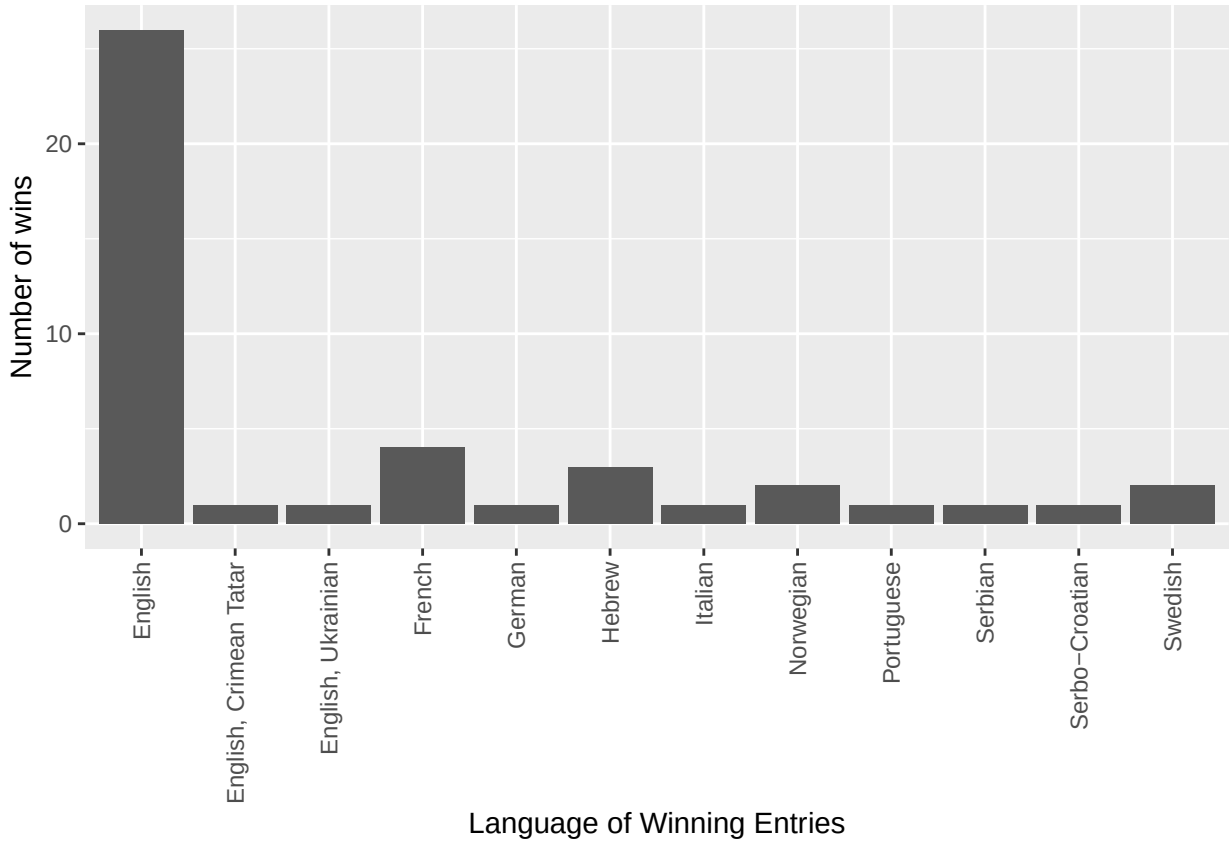
This produces a dataframe with each score that each country gave, the distance to the recieving country, the title of the song that was awarded points, the language it was performed in, its overall performance in the contest, and its lyrics. It also codes if the points were given to a neighbor.

Some Graphs

I haven't done much analysis yet (as constructing the dataset itself was a significant amount of work). I, however, did make a couple quick language graphs:

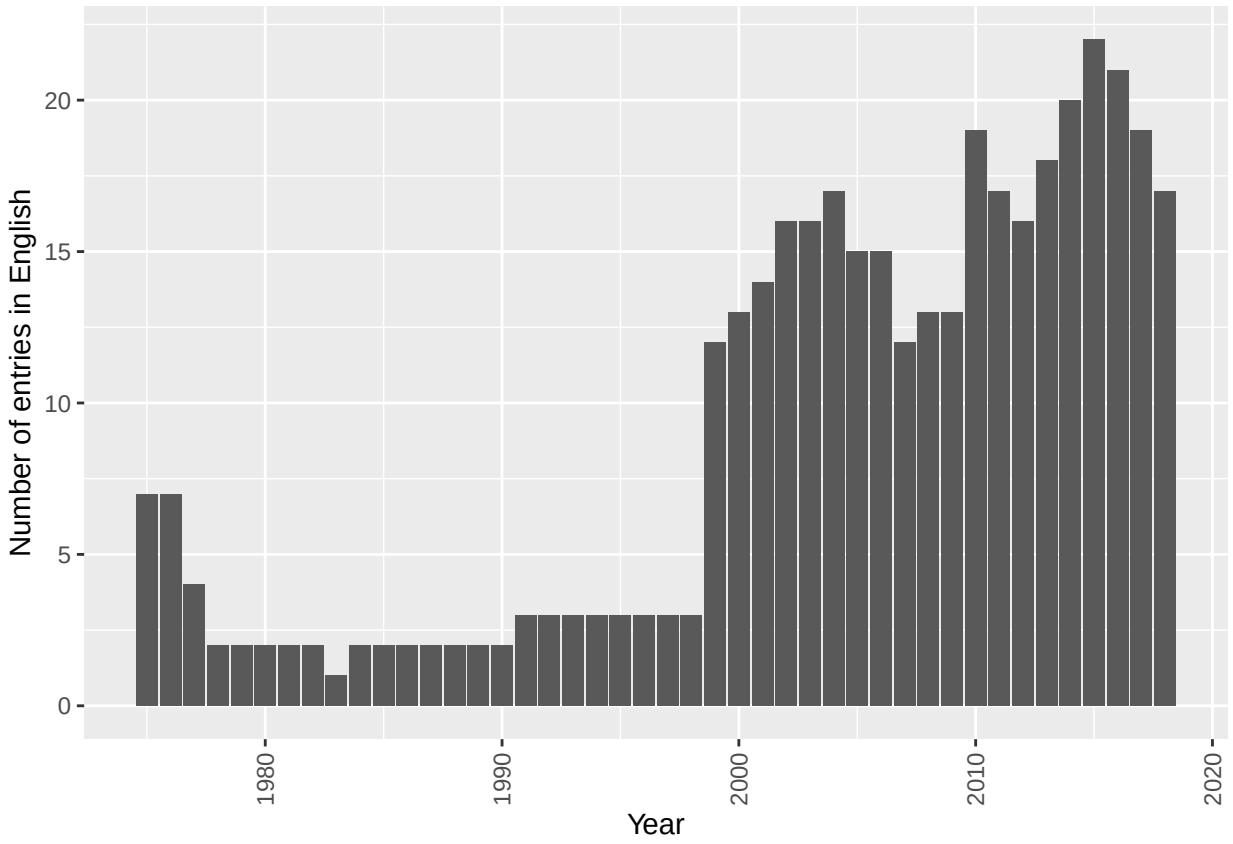
```
winners <- songtitles %>% filter(Place == 1)
inenglish <- songtitles %>% group_by(Year) %>% filter(Language == "English")

ggplot(winners, aes(x=winners$Language)) +
  geom_bar() +
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5)) + xlab("Language of Winning Entries") + yl
```



Most winners of the contest perform in English. Many countries have realized this in the last 15 years and have shifted towards submitting songs in English:

```
ggplot(inenglish, aes(x=inenglish$Year)) +
  geom_bar() +
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5)) + xlab("Year") + ylab("Number of entries i
```



Lyric analysis

We do some quick lyric analysis (though this will not actually impact our final model):

```
english <- subset(songtitles, songtitles$Language == "English")
english$lyrics <- as.character(english$lyrics)
corpus <- corpus(english$lyrics, docvars=english)
doc.features <- dfm(corpus, remove=c(stopwords("english")), stem=F, remove_punct=T)
textplot_wordcloud(doc.features)
```

```
## Warning in graphics::strwidth(word[i], cex = size[i]): conversion failure
## on 'şekerim' in 'mbcsToSbcs': dot substituted for <c5>
```

```
## Warning in graphics::strwidth(word[i], cex = size[i]): conversion failure
## on 'şekerim' in 'mbcsToSbcs': dot substituted for <9f>
```

```
## Warning in text.default(x1, y1, word[i], cex = (1 + adjust) * size[i],
## offset = 0, : conversion failure on 'şekerim' in 'mbcsToSbcs': dot
## substituted for <c5>
```

```
## Warning in text.default(x1, y1, word[i], cex = (1 + adjust) * size[i],
## offset = 0, : conversion failure on 'şekerim' in 'mbcsToSbcs': dot
## substituted for <9f>
```

```
## Warning in text.default(x1, y1, word[i], cex = (1 + adjust) * size[i],
## offset = 0, : font metrics unknown for Unicode character U+015f
```

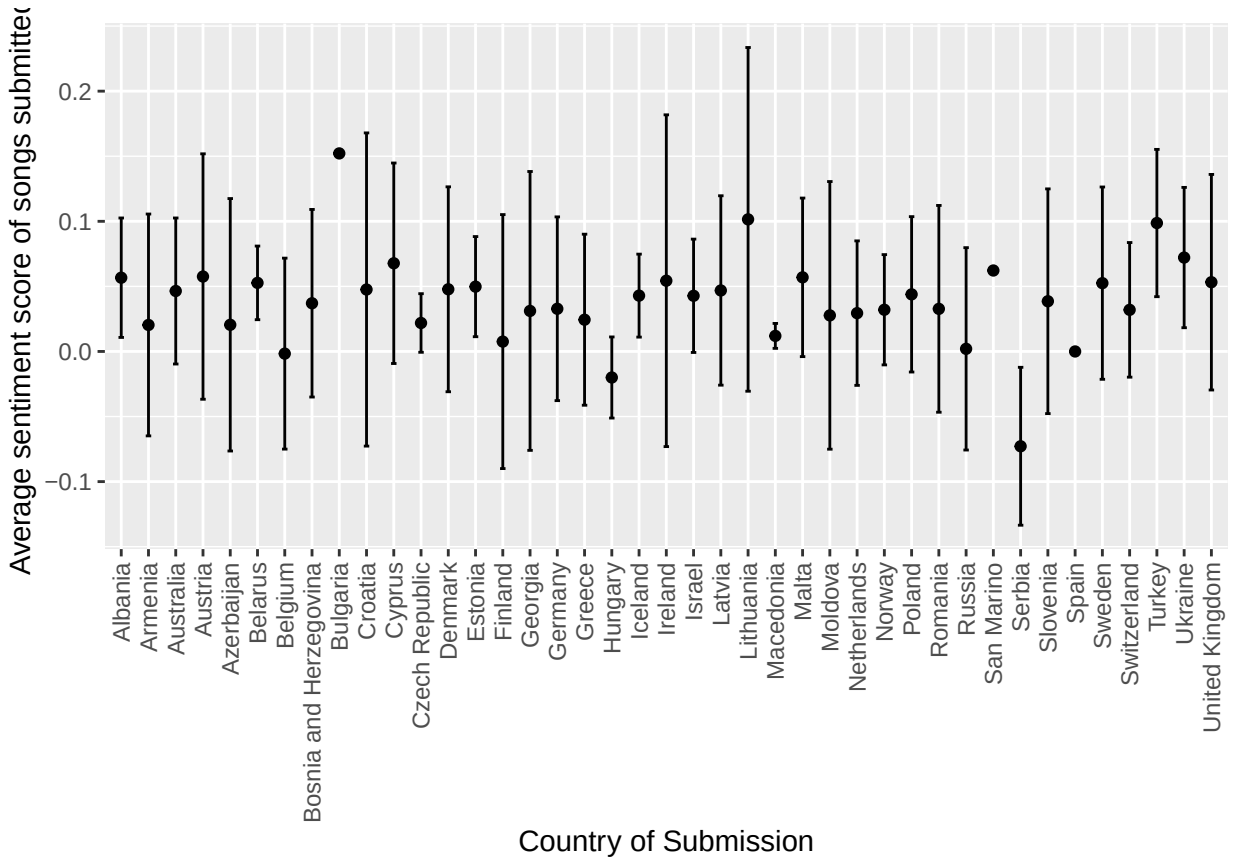

```
vocab <- colnames(dfm_weight(doc.features))
words <- read.csv("AFINN111.txt", sep="\t", header=F)
words <- words[words$V1%in%vocab,]
dfm1 <- dfm_weight(doc.features)[,colnames(dfm_weight(doc.features))%in%words$V1]
dfm1 <- dfm1[,order(colnames(dfm1))]
words <- words[order(words$V1),]
weights <- words$V2%*%t(dfm1)
len <- ntoken(english$lyrics)
corpus$sent <- weights[1,]/len
test <- cbind(english, corpus$sent)
```

This squares with the sterotype about Eurovision songs always being love songs.

```
bycountry <- test %>% group_by(To) %>% dplyr::summarise(instances = n(), score = mean(`corpus$sent`), s
bycountry <- bycountry %>% arrange(desc(instances))

ggplot(bycountry, aes(x=as.factor(To), y=score)) + geom_point() + geom_errorbar(aes(ymin=score-sd, ymax=
                  position=position_dodge(0.05)) + theme(axis.text.x=element_text(angle=90,hjust=1,vjust=
```

```
## Warning: Removed 2 rows containing missing values (geom_errorbar).
```



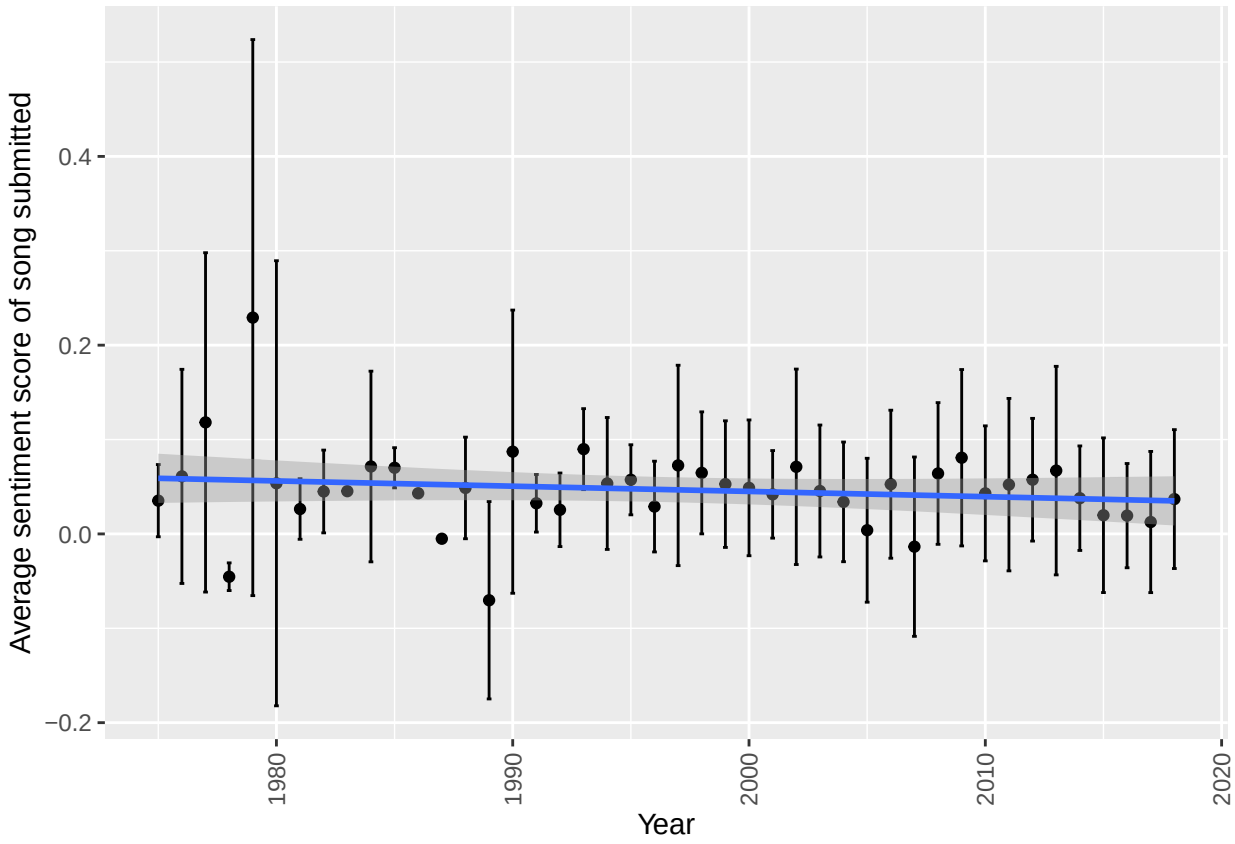
Everyone is neutral to cheerful, except Serbia (and to a less extent, Hungary).

```
byyear <- test %>% group_by(Year) %>% dplyr::summarise(instances = n(), score = mean(`corpus$sent`), sd
byyear <- byyear %>% arrange(desc(instances))



ggplot(byyear, aes(x=Year, y=score)) + geom_point() + geom_errorbar(aes(ymin=score-sd, ymax=score+sd),
                position=position_dodge(0.05)) + geom_smooth(method="lm", se=TRUE, fullrange=FALSE, le
```

```
## Warning: Removed 1 rows containing missing values (geom_errorbar).
```



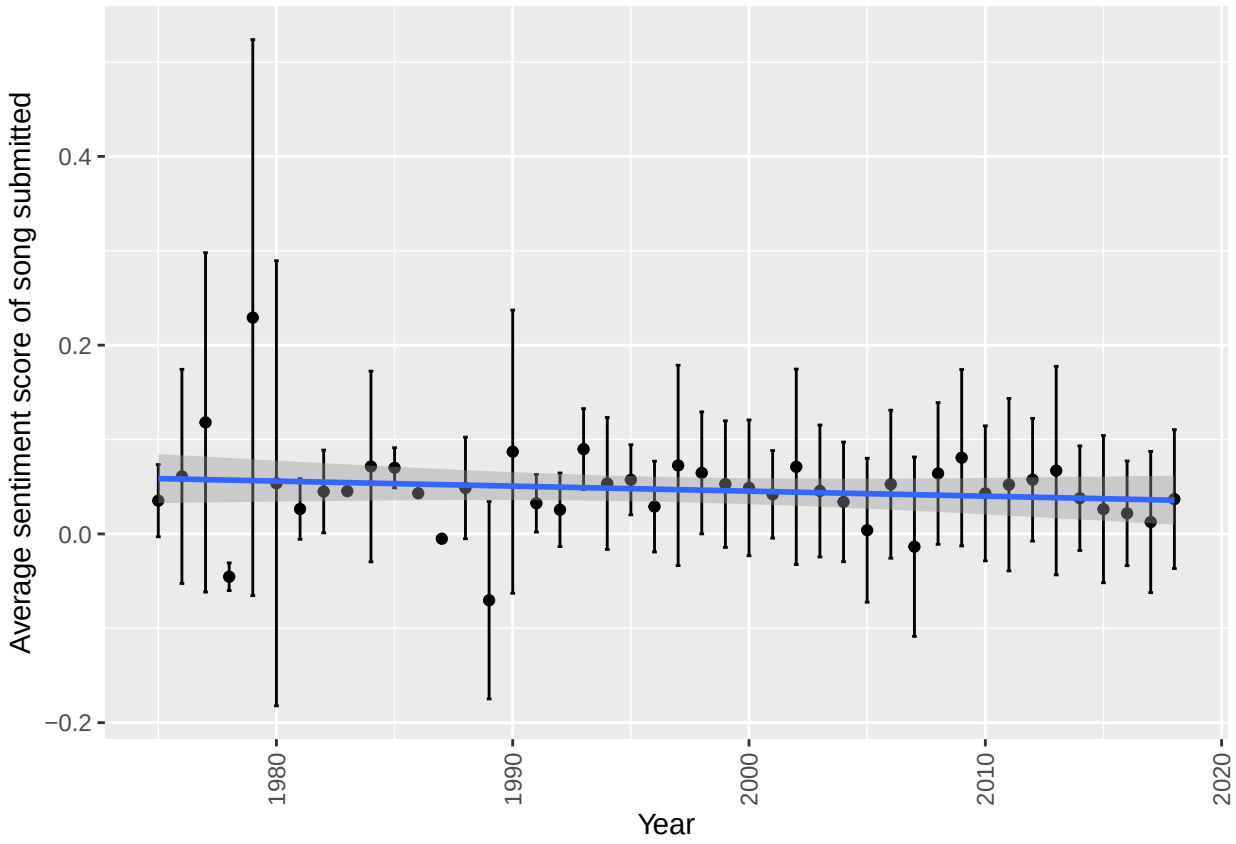
Songs are not getting more cheerful with time; if anything, they are getting less so. But we want to see if this is just Serbia's fault.

```
test <- subset(test, test$To != "Serbia")

byyear <- test %>% group_by(Year) %>% dplyr::summarise(instances = n(), score = mean(`corpus$sent`), sd
byyear <- byyear %>% arrange(desc(instances))

ggplot(byyear, aes(x=Year, y=score)) + geom_point() + geom_errorbar(aes(ymin=score-sd, ymax=score+sd),
                position=position_dodge(0.05)) + geom_smooth(method="lm", se=TRUE, fullrange=FALSE, le
```

```
## Warning: Removed 1 rows containing missing values (geom_errorbar).
```



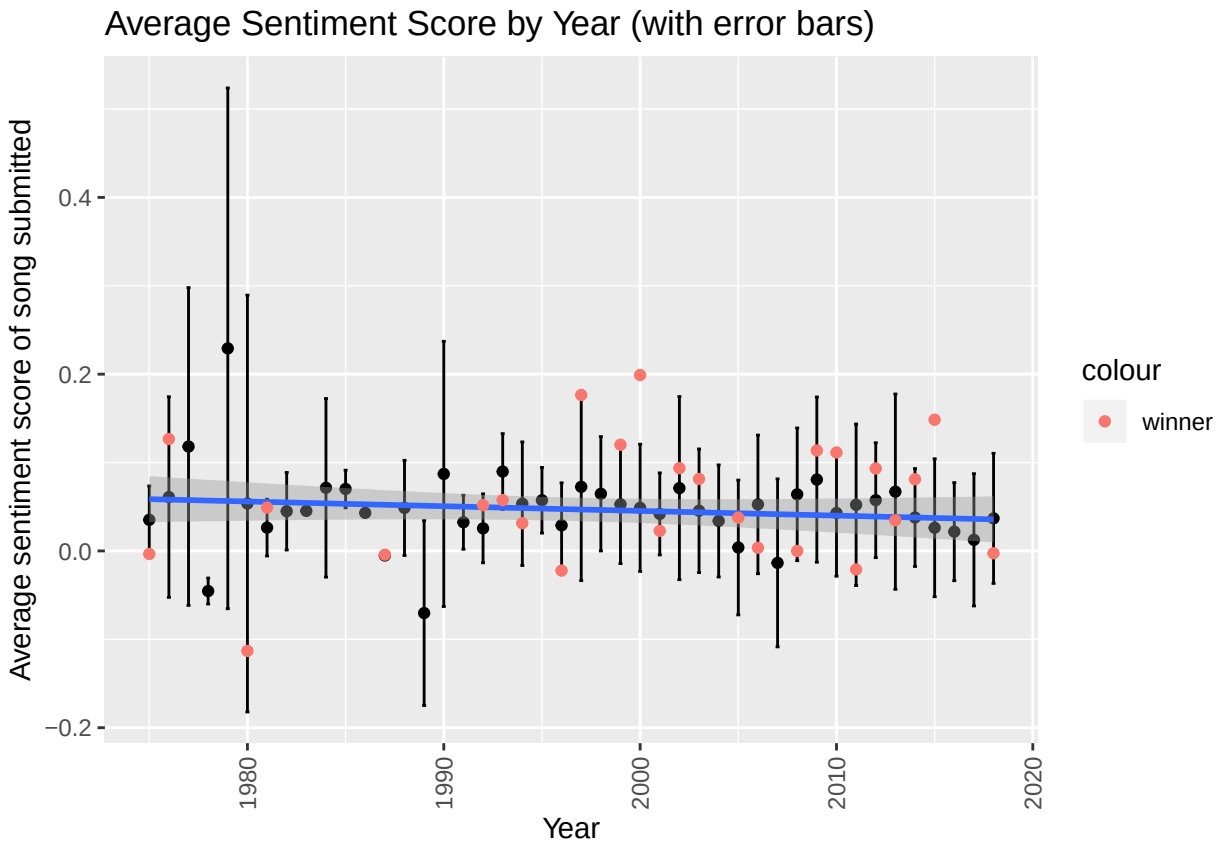
No, the trend is robust to the removal of Serbia. How does the average sentiment score compare to the winning score?

```
winner <- subset(test, test$Place == 1)
new <- left_join(byyear, winner, by = "Year")

ggplot(new, aes(x=Year, y=score)) +  geom_point(aes(y = score)) + geom_errorbar(aes(ymin=score-sd, ymax=
```

```
## Warning: Removed 1 rows containing missing values (geom_errorbar).
```

```
## Warning: Removed 18 rows containing missing values (geom_point).
```



There seems little evidence that the winner (coral) is more cheerful than the average song (black).