How influential are music critics?

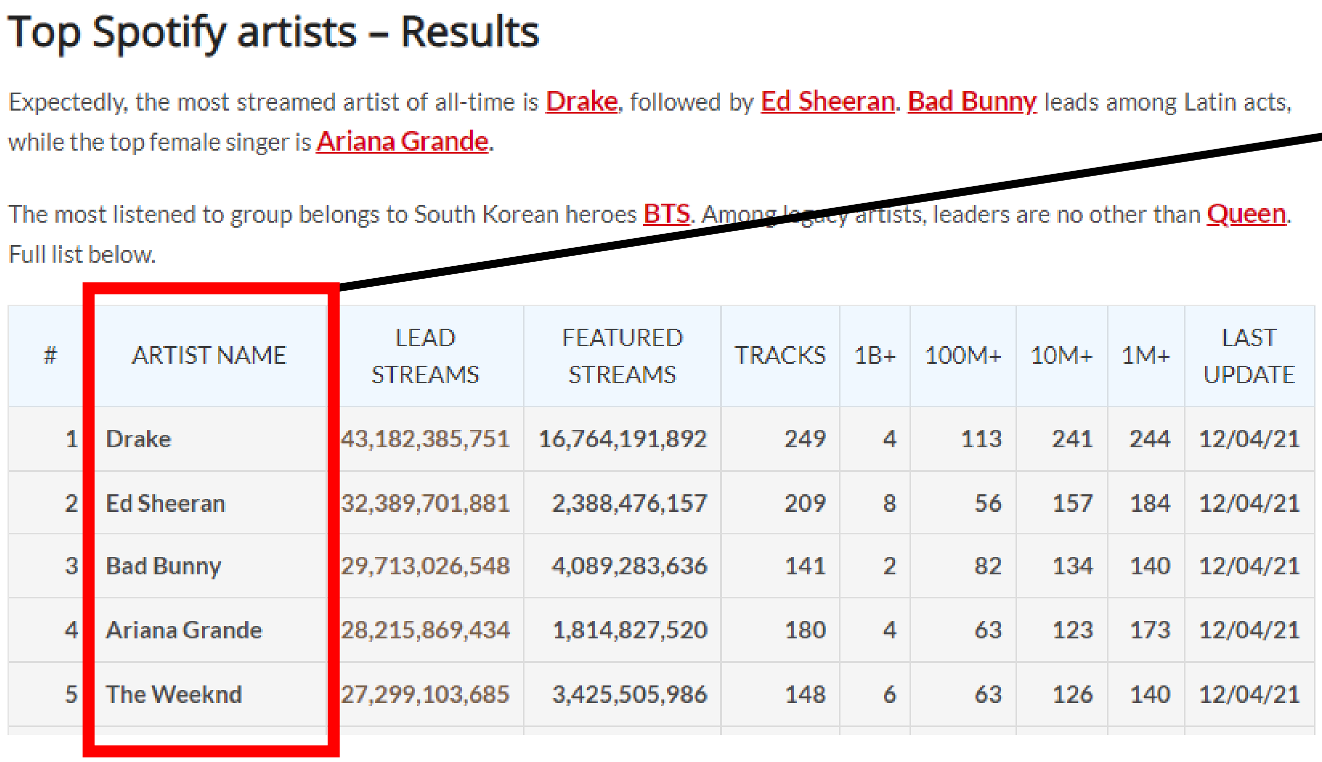
Gradon Nicholls

Dec 6, 2021

Motivation

- Economics: consumer demand
    - How does positive or negative reviews impact consumer demand for music?
        - Non-linearity? Both negative and positive reviews drive increased sales?
        - Does this differ for big name artists vs. smaller artists?
        - Does “new media” have more of an impact than “old media”
        - How correlated are customer reviews and critic reviews?
- Idea: regress number of listens of an album on critic scores

Data I – List of artist names



Top Spotify artists – Results

Expectedly, the most streamed artist of all-time is Drake, followed by Ed Sheeran. Bad Bunny leads among Latin acts, while the top female singer is Ariana Grande.

The most listened to group belongs to South Korean heroes BTS. Among legacy artists, leaders are no other than Queen. Full list below.

#	ARTIST NAME	LEAD STREAMS	FEATURED STREAMS	TRACKS	1B+	100M+	10M+	1M+	LAST UPDATE
1	Drake	43,182,385,751	16,764,191,892	249	4	113	241	244	12/04/21
2	Ed Sheeran	32,389,701,881	2,388,476,157	209	8	56	157	184	12/04/21
3	Bad Bunny	29,713,026,548	4,089,283,636	141	2	82	134	140	12/04/21
4	Ariana Grande	28,215,869,434	1,814,827,520	180	4	63	123	173	12/04/21
5	The Weeknd	27,299,103,685	3,425,505,986	148	6	63	126	140	12/04/21

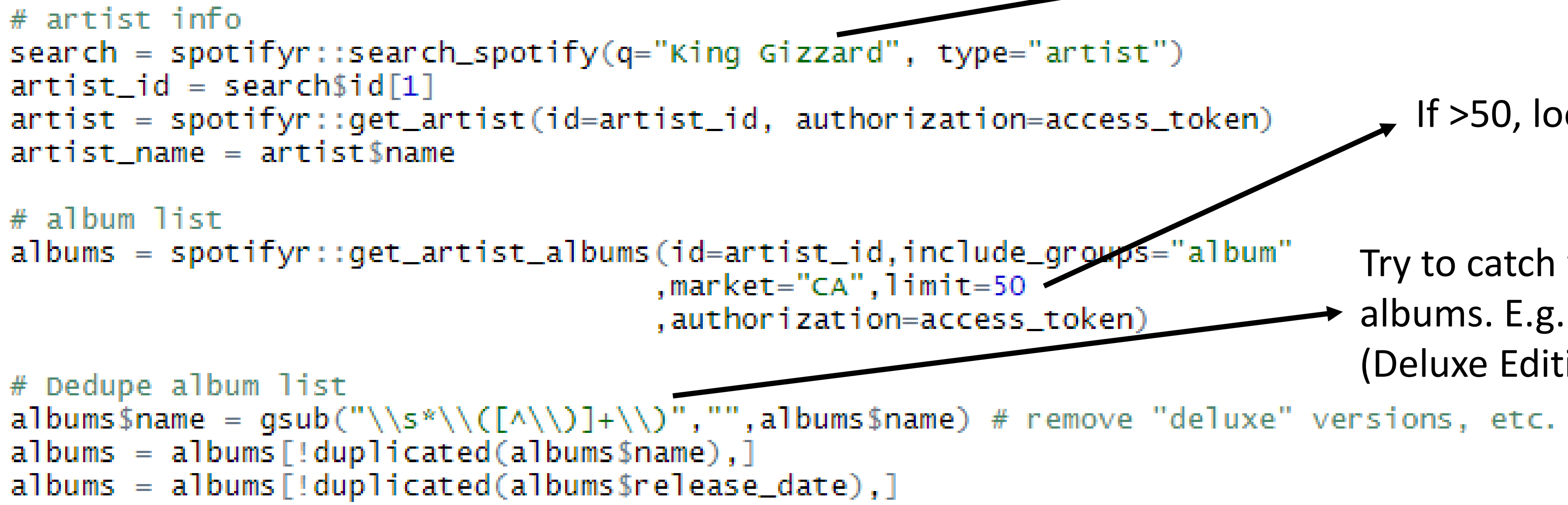
1,000 most-streamed artists on Spotify as of Nov 1, 2021

Source: https://chartmasters.org/most-streamed-artists-ever-on-spotify/

```
chartmasterurl = "https://chartmasters.org/most-streamed-artists-ever-on-spotify/"
chartwebpage = rvest::read_html(chartmasterurl)
artistnames = rvest::html_nodes(chartwebpage, xpath='//td[(((count(preceding-sibling::*) + 1) = 2) and parent::*)]')
artistnames = rvest::html_text(artistnames,trim=T)
```

Data II – Search artist on Spotify and get albums

- R package: spotifyr for spotify api

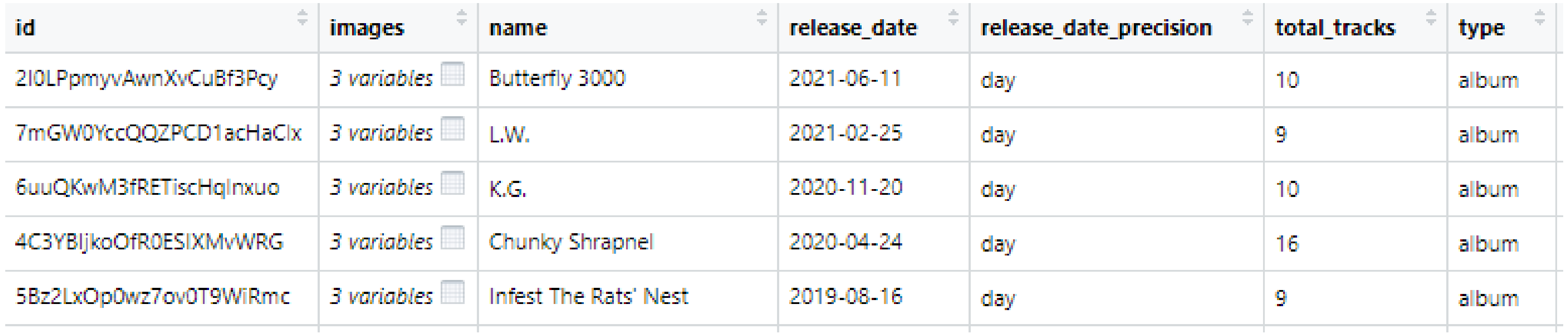


```
# artist info
search = spotifyr::search_spotify(q="King Gizzard", type="artist")
artist_id = search$id[1]
artist = spotifyr::get_artist(id=artist_id, authorization=access_token)
artist_name = artist$name

# album list
albums = spotifyr::get_artist_albums(id=artist_id,include_groups="album"
                                     ,market="CA",limit=50
                                     ,authorization=access_token)

# Dedupe album list
albums$name = gsub("\\s*\\([^\\)]+\\)","",albums$name) # remove "deluxe" versions, etc.
albums = albums[!duplicated(albums$name),]
albums = albums[!duplicated(albums$release_date),]
```

Take name from list

If >50, loop

Try to catch when there are repeat albums. E.g. "Album" and "Album (Deluxe Edition)"



id	images	name	release_date	release_date_precision	total_tracks	type
2I0LPpmyvAwnXvCuBf3Pcy	*3 variables*	Butterfly 3000	2021-06-11	day	10	album
7mGW0YccQQZPCD1acHaClx	*3 variables*	L.W.	2021-02-25	day	9	album
6uuQKwM3fRETiscHqlnxuo	*3 variables*	K.G.	2020-11-20	day	10	album
4C3YBljkoOfR0ESIXMvWRG	*3 variables*	Chunky Shrapnel	2020-04-24	day	16	album
5Bz2LxOp0wz7ov0T9WiRmc	*3 variables*	Infest The Rats' Nest	2019-08-16	day	9	album

Data III – Last.FM

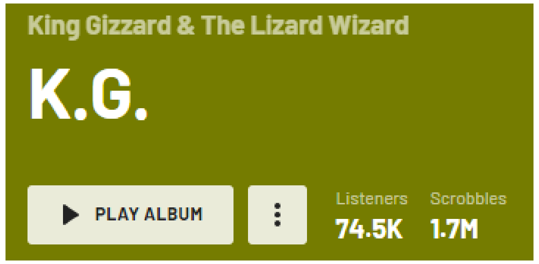
Spotify does NOT have number of listens! -> Rely on a proxy using Last.fm “scrobbles”

```
# get album info
example_album=spotifyr::get_album(id='6uuQKwM3fRETiscHqlnxuo'
                                  ,authorization=access_token)

# '&' and '+' and '#' cause issues
url_artist = stringr::str_replace_all(artist_name,"\\+","%2B")
url_album = stringr::str_replace_all(example_album$name,"\\+","%2B")
url_artist = stringr::str_replace_all(url_artist,"&","%26")
url_album = stringr::str_replace_all(url_album,"&","%26")
url_artist = stringr::str_replace_all(url_artist,"#","%23")
url_album = stringr::str_replace_all(url_album,"#","%23")

url = paste0("http://ws.audioscrobbler.com/2.0/",
             "?method=album.getinfo",
             "&api_key=",api_key,
             "&artist=", gsub(" ", "+", url_artist),
             "&album=", gsub(" ", "+", url_album),
             "&format=json")
data_json = httr::GET(url)
data_json = jsonlite::fromJSON(rawToChar(data_json$content))
album_scrobbles = as.integer(data_json$album$playcount)
```

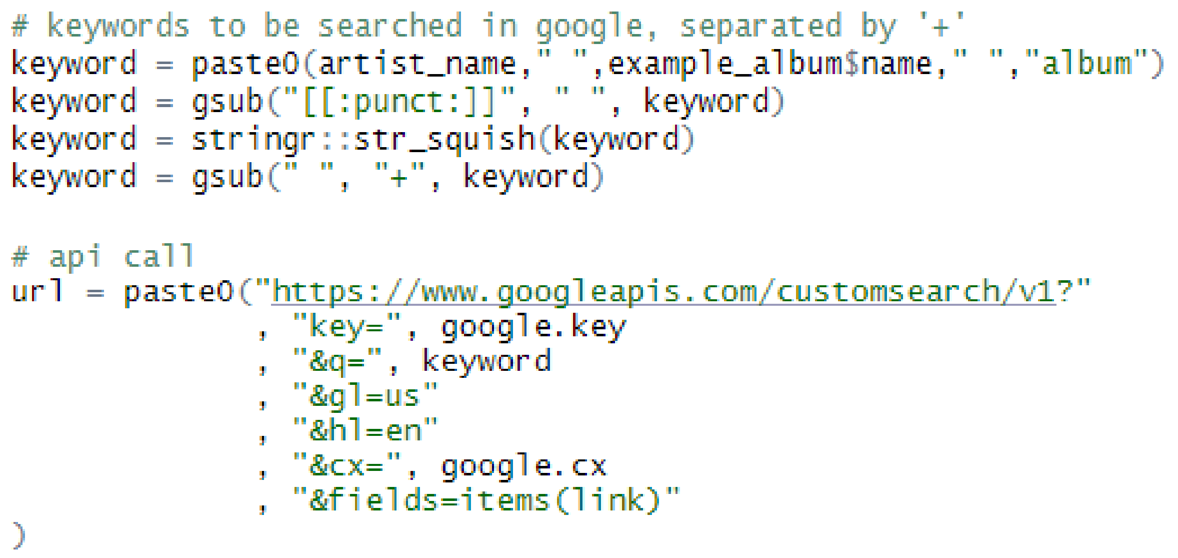
```
> album_name
[1] "K.G."
> artist_name
[1] "King Gizzard & The Lizard Wizard"
```

```
"http://ws.audioscrobbler.com/2.0/
?method=album.getinfo
&api_key=xxxx
&artist=King+Gizzard+%26+The+Lizard+Wizard
&album=K.G.&format=json"
```

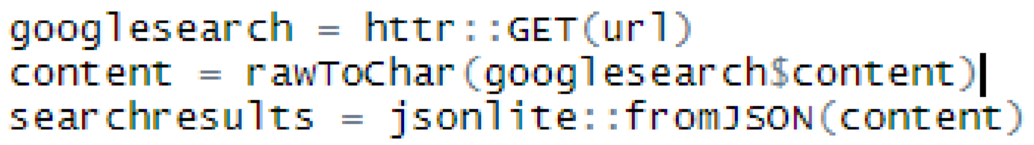
```
> album_scrobbles
[1] 1726282
```



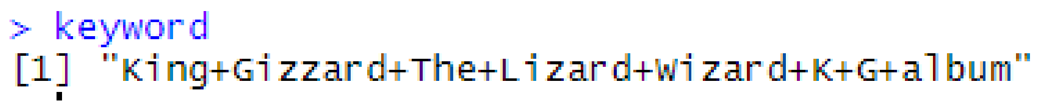
Data IV – Use Google api to search for Wikipedia article



```r
# keywords to be searched in google, separated by '+'
keyword = paste0(artist_name," ",example_album$name," ","album")
keyword = gsub("[[:punct:]]", " ", keyword)
keyword = stringr::str_squish(keyword)
keyword = gsub(" ", "+", keyword)

# api call
url = paste0("https://www.googleapis.com/customsearch/v1?"
             , "key=", google.key
             , "&q=", keyword
             , "&gl=us"
             , "&hl=en"
             , "&cx=", google.cx
             , "&fields=items(link)"
)
```

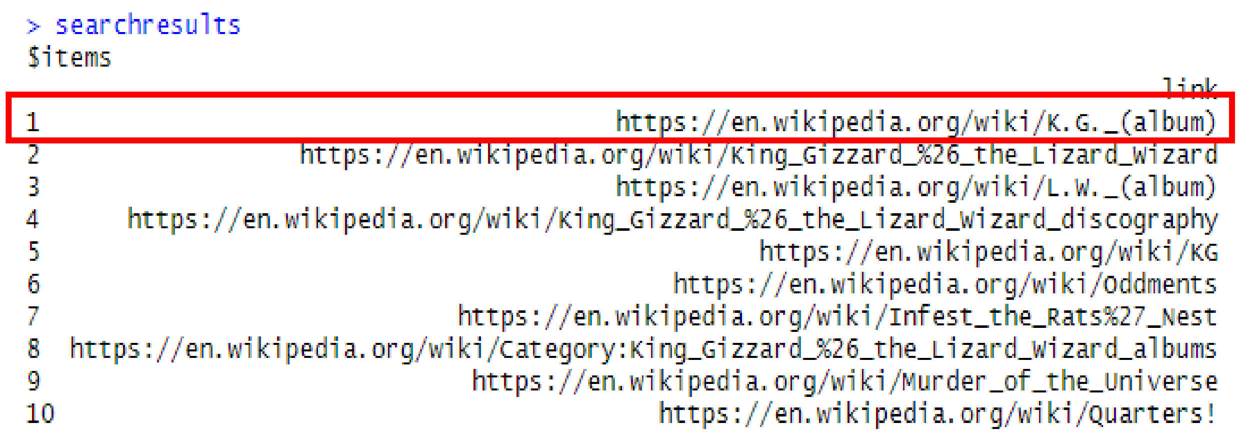


```r
googlesearch = httr::GET(url)
content = rawToChar(googlesearch$content)
searchresults = jsonlite::fromJSON(content)
```



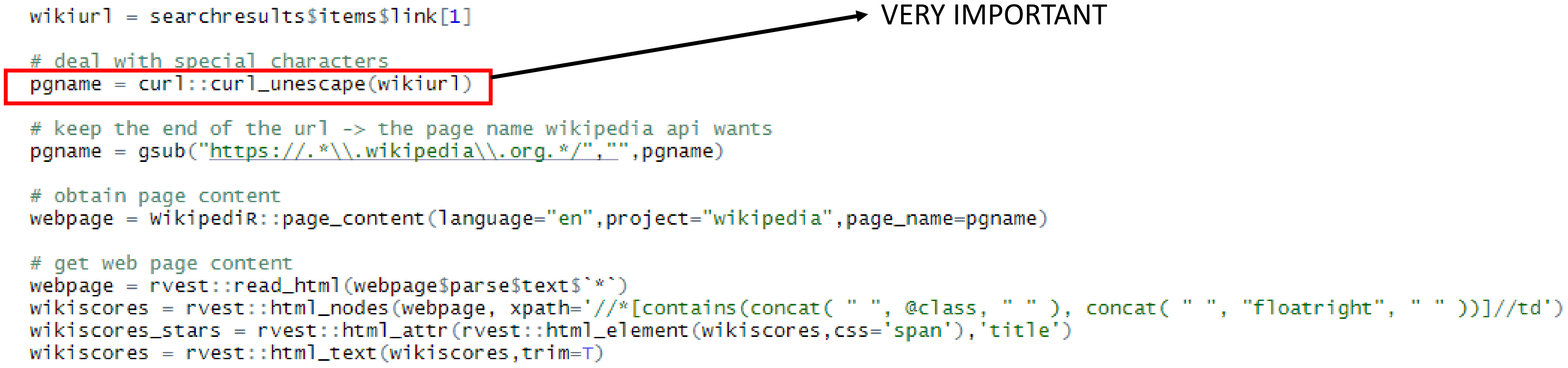
```
> keyword
[1] "King+Gizzard+The+Lizard+Wizard+K+G+album"
```

```
https://www.googleapis.com/customsearch/v1?
key=xxxxxxxx
&q=King+Gizzard+The+Lizard+Wizard+K+G+album
&gl=us
&hl=en
&cx=xxxxxxxx
&fields=items(link)
```



```
> searchresults
$items
                                                                                link
1                                               https://en.wikipedia.org/wiki/K.G._(album)
2                               https://en.wikipedia.org/wiki/King_Gizzard_%26_the_Lizard_Wizard
3                                               https://en.wikipedia.org/wiki/L.W._(album)
4                  https://en.wikipedia.org/wiki/King_Gizzard_%26_the_Lizard_Wizard_discography
5                                                       https://en.wikipedia.org/wiki/KG
6                                                 https://en.wikipedia.org/wiki/Oddments
7                                   https://en.wikipedia.org/wiki/Infest_the_Rats%27_Nest
8 https://en.wikipedia.org/wiki/Category:King_Gizzard_%26_the_Lizard_Wizard_albums
9                                    https://en.wikipedia.org/wiki/Murder_of_the_Universe
10                                               https://en.wikipedia.org/wiki/Quarters!
```

Data V – Get wiki page from Wikipedia api

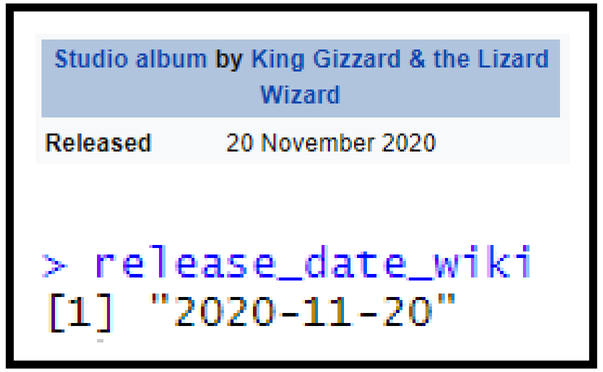


```r
wikiurl = searchresults$items$link[1]

# deal with special characters
pgname = curl::curl_unescape(wikiurl)

# keep the end of the url -> the page name wikipedia api wants
pgname = gsub("https://.*\\.wikipedia\\.org.*/","",pgname)

# obtain page content
webpage = WikipediR::page_content(language="en",project="wikipedia",page_name=pgname)

# get web page content
webpage = rvest::read_html(webpage$parse$text$`*`)
wikiscores = rvest::html_nodes(webpage, xpath='//*[contains(concat( " ", @class, " " ), concat( " ", "floatright", " " ))]//td')
wikiscores_stars = rvest::html_attr(rvest::html_element(wikiscores,css='span'),'title')
wikiscores = rvest::html_text(wikiscores,trim=T)
```

VERY IMPORTANT

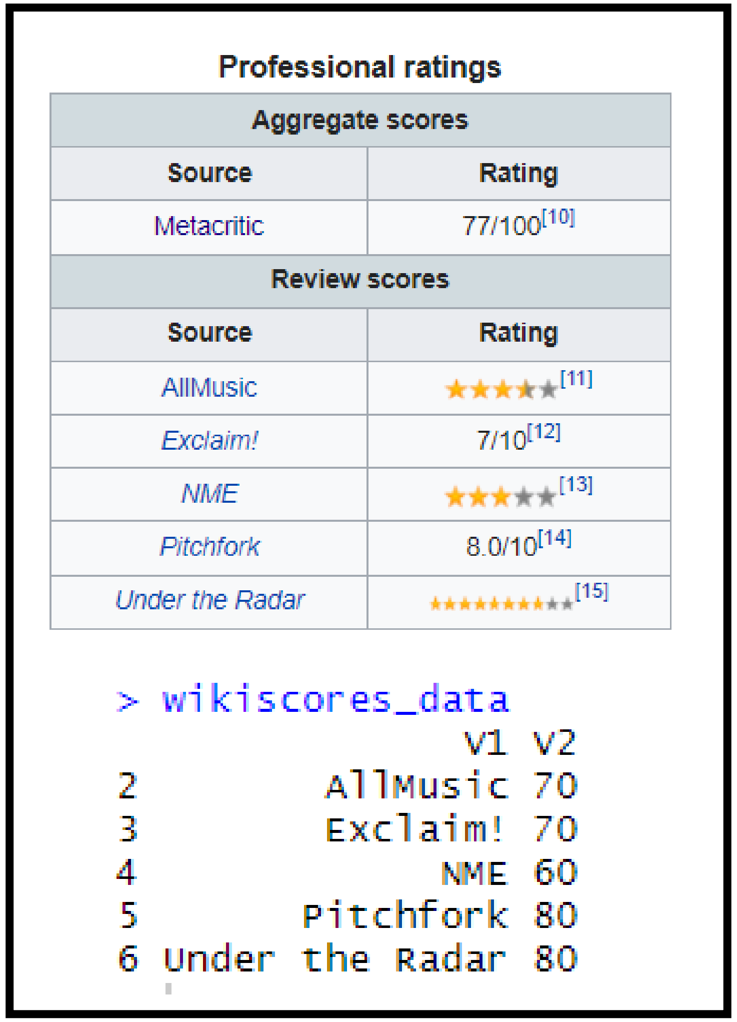
Data VI – Parse Wikipedia page

```r
# some formatting for the scores -> convert them all into standard numeric out of 100
wikiscores_data = as.data.frame(matrix(wikiscores,ncol=2,byrow=T))
wikiscores_stars_data = as.data.frame(matrix(wikiscores_stars,ncol=2,byrow=T))
# deal with specific review websites
selector = (!is.na(wikiscores_stars_data$V2) & !(tolower(wikiscores_data$V1)=="tom hull â·· on the web"))
wikiscores_data$V2[selector] = wikiscores_stars_data$V2[selector]
wikiscores_data = wikiscores_data[wikiscores_data$V1!="Metacritic",] # this is a review aggregator
wikiscores_data = wikiscores_data[wikiscores_data$V1!="AnyDecentMusic?",] # this is a review aggregator
# remove [] and content inside
wikiscores_data$V2 = gsub("\\[.*?\\]","",wikiscores_data$V2)
# remove () but NOT CONTENT (sometimes the score is contained within ()
wikiscores_data$V2 = stringr::str_replace(wikiscores_data$V2,"\\(","")
wikiscores_data$V2 = stringr::str_replace(wikiscores_data$V2,"\\)","")
# fix ill-behaved special characters
wikiscores_data$V2 = sapply(wikiscores_data$V2 , function(x) utils::URLencode(x) )
wikiscores_data$V2 = stringr::str_replace(wikiscores_data$V2,"%E2%88%92","-")
wikiscores_data$V2 = stringr::str_replace(wikiscores_data$V2,"%E2%80%93","-")
wikiscores_data$V2 = stringr::str_replace(wikiscores_data$V2,"%E2%80%94","-")
wikiscores_data$V2 = sapply(wikiscores_data$V2 , function(x) utils::URLdecode(x) )
wikiscores_data$V2 = stringr::str_replace(wikiscores_data$V2,"\\?\\^\\'","-")
# covert letter grade to numeric
wikiscores_data$V2 = sapply(stringr::str_trim(wikiscores_data$V2) , lettergrade_num)
# remove unit of measure (stars or discs)
wikiscores_data$V2 = stringr::str_replace(wikiscores_data$V2,"stars","")
wikiscores_data$V2 = stringr::str_replace(wikiscores_data$V2,"discs","")
# remove %'s
wikiscores_data$V2 = stringr::str_replace(wikiscores_data$V2,"%","")
# remove remaining scores that contain no numbers
wikiscores_data = wikiscores_data[grepl("\\d",wikiscores_data$V2),]

# convert to numeric
wikiscores_data$V2 = sapply(wikiscores_data$V2,
                        function(x) tryCatch(eval(parse(text=x)),
                                             error=function(e){},
                                             finally=NULL) )
# final set of scores to put in dataset
wikiscores_data = wikiscores_data[!sapply(wikiscores_data$V2,is.null),]
wikiscores_data$V2 = as.numeric(wikiscores_data$V2)
wikiscores_data$V2[wikiscores_data$V2<=1] = 100*wikiscores_data$V2[wikiscores_data$V2<=1]


## CRITIC SCORES FROM WIKIPEDIA ##########################################
release_date_wiki = rvest::html_nodes(webpage, xpath='//*[contains(concat( " ", @class, " " ), concat( " ", "published", " " ))]')
release_date_wiki = rvest::html_text(release_date_wiki,trim=T)
release_date_wiki = release_date_wiki[2]
```



Studio album by King Gizzard & the Lizard Wizard	
Released	20 November 2020

```
> release_date_wiki
[1] "2020-11-20"
```



Professional ratings

Aggregate scores	
Source	**Rating**
Metacritic	77/100[10]
Review scores	
Source	**Rating**
AllMusic	★★★★☆[11]
Exclaim!	7/10[12]
NME	★★★☆☆[13]
Pitchfork	8.0/10[14]
Under the Radar	★★★★★★★★☆☆[15]

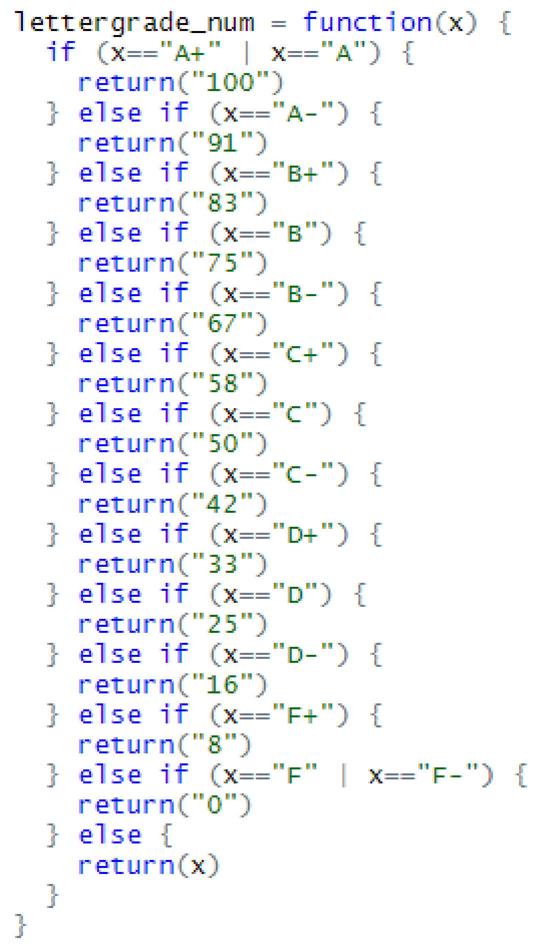
```
> wikiscores_data
              V1 V2
2       AllMusic 70
3       Exclaim! 70
4            NME 60
5      Pitchfork 80
6 Under the Radar 80
```

Aside: how to amalgamate scores?

- Numeric scores scaled to 100
  - e.g. 3.5 stars out of 4 -> 87.5

- Letter grades assigned scores based on Metacritic rules.

- Compute simple unweighted summary statistics
  - Mean, median, sd, IQR, min, max



```
lettergrade_num = function(x) {
  if (x=="A+" | x=="A") {
    return("100")
  } else if (x=="A-") {
    return("91")
  } else if (x=="B+") {
    return("83")
  } else if (x=="B") {
    return("75")
  } else if (x=="B-") {
    return("67")
  } else if (x=="C+") {
    return("58")
  } else if (x=="C") {
    return("50")
  } else if (x=="C-") {
    return("42")
  } else if (x=="D+") {
    return("33")
  } else if (x=="D") {
    return("25")
  } else if (x=="D-") {
    return("16")
  } else if (x=="F+") {
    return("8")
  } else if (x=="F" | x=="F-") {
    return("0")
  } else {
    return(x)
  }
}
```

Source: [How We Create the Metascore Magic - Metacritic]

Summary Stats – Number of review scores

```
          0 |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |      3,094       43.89       43.89
          1 |        831       11.79       55.68
          2 |        375        5.32       61.00
          3 |        385        5.46       66.46
          4 |        274        3.89       70.35
          5 |        275        3.90       74.25
          6 |        229        3.25       77.50
          7 |        207        2.94       80.44
          8 |        187        2.65       83.09
          9 |        188        2.67       85.76
        10+ |      1,004       14.24      100.00
------------+-----------------------------------
      Total |      7,049      100.00
```

$\left.\begin{array}{l} 3 \\ \vdots \\ 10+ \end{array}\right\}$ 484 artists with 2,749 albums

Model

Artist fixed effect

# days album has been out (spotify)

$$\boxed{\log s_{ij} = \alpha_0 + \alpha_i + \gamma \log n_{ij} + \log score_{ij} + \epsilon_{ij}}$$

Scrobbles (Last.fm)

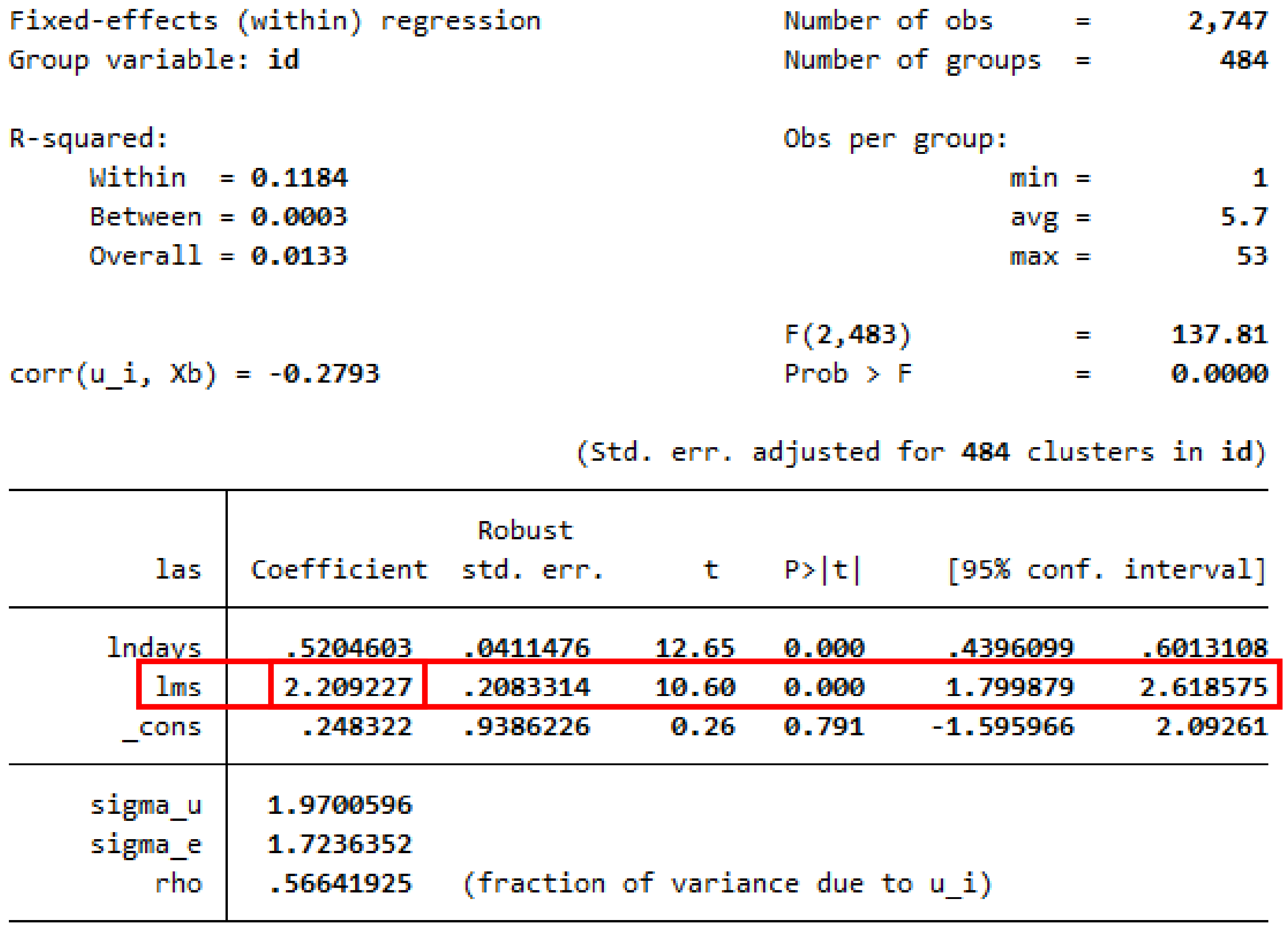
Aggregated scores (wikipedia)

$$\boxed{\begin{aligned} i &= \text{Artist} \\ j &= \text{Album} \end{aligned}}$$

$$\boxed{\epsilon_i \sim N(0, \Sigma_i)}$$

$$\boxed{\log s_{ij} = \alpha_0 + \alpha_i + \log s_{i,j-1} + \gamma \log n_{ij} + \log score_{ij} + \epsilon_{ij}}$$

Results – Regress $\log s_{ij}$ on $\log score_{ij}$



```
Fixed-effects (within) regression               Number of obs     =      2,747
Group variable: id                              Number of groups  =        484

R-squared:                                      Obs per group:
     Within  = 0.1184                                         min =          1
     Between = 0.0003                                         avg =        5.7
     Overall = 0.0133                                         max =         53

                                                F(2,483)          =     137.81
corr(u_i, Xb) = -0.2793                         Prob > F          =     0.0000

                                 (Std. err. adjusted for 484 clusters in id)
------------------------------------------------------------------------------
             |               Robust
         las | Coefficient  std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
      lndays |   .5204603   .0411476    12.65   0.000     .4396099    .6013108
         lms |   2.209227   .2083314    10.60   0.000     1.799879    2.618575
       _cons |    .248322   .9386226     0.26   0.791    -1.595966     2.09261
-------------+----------------------------------------------------------------
     sigma_u |  1.9700596
     sigma_e |  1.7236352
         rho |  .56641925   (fraction of variance due to u_i)
------------------------------------------------------------------------------
```

Example: King Gizzard's K.G. Album

Mean review score = 70 -> log(score) = 4.24
Scrobbles = 1.7mil -> log(s) = 14.3

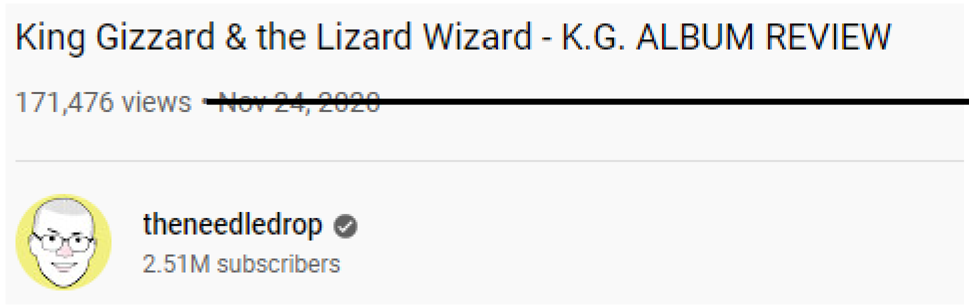
Mean review score = 80 -> log(score) = 4.38
E[log(s)|score=80] = 14.3 + (4.38-4.24) = 14.44
=> s = exp(14.44) = 1.9mil

So improving review scores by 10 points results in ~100,000 extra (lifetime) scrobbles.

If each listen is $0.003, then then they would earn an additional $300 from streaming.

Future work

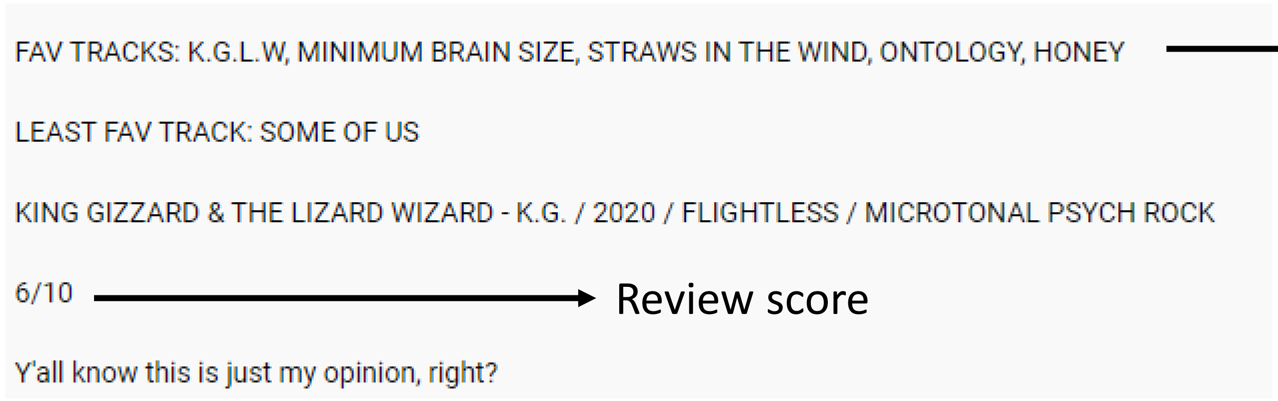
- Additional variables of interest
  - Demographics from wikipedia
  - Additional info on tracks from Spotify (genre, tempo, key, etc.)
- Review scores from The Needle Drop -> “new” vs “traditional” media



King Gizzard & the Lizard Wizard - K.G. ALBUM REVIEW

171,476 views · Nov 24, 2020 → Views

theneedledrop
2.51M subscribers



FAV TRACKS: K.G.L.W, MINIMUM BRAIN SIZE, STRAWS IN THE WIND, ONTOLOGY, HONEY → Track recommendations

LEAST FAV TRACK: SOME OF US

KING GIZZARD & THE LIZARD WIZARD - K.G. / 2020 / FLIGHTLESS / MICROTONAL PSYCH ROCK

6/10 → Review score

Y'all know this is just my opinion, right?