Package ‘taylor’

February 10, 2026

Title Lyrics and Song Data for Taylor Swift's Discography

Version 4.0.0

Description A comprehensive resource for data on Taylor Swift songs. Data is included for all officially released studio albums, extended plays (EPs), and individual singles are included. Data comes from 'Genius' (lyrics) and 'SoundStat' (song characteristics). Additional functions are included for easily creating data visualizations with color palettes inspired by Taylor Swift's album covers.

License MIT + file LICENSE

URL `https://taylor.wjakethompson.com`, `https://github.com/wjakethompson/taylor`

BugReports `https://github.com/wjakethompson/taylor/issues`

Depends R (>= 4.1.0)

Imports askpass, cli, dplyr, ggplot2, glue, httr2, lifecycle, methods, rlang, scales, spotifyr, tibble, tidyr, vctrs

Suggests bookdown, knitr, patchwork, rmarkdown, spelling, testthat (>= 3.0.0), vdiffr (>= 1.0.2), withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website bookdown

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.3

NeedsCompilation no

Author W. Jake Thompson [aut, cre, cph] (ORCID: <`https://orcid.org/0000-0001-7339-0300`>)

Maintainer W. Jake Thompson <`wjakethompson@gmail.com`>

Repository CRAN

Date/Publication 2026-02-10 15:40:02 UTC

Contents

`album_levels` *Correct ordering of Taylor Swift's albums*

Description

Easily create a factor variable for Taylor Swift's albums. Rather than specifying each album individually, you can use this shortcut vector that has already specified the ordering.

Usage

```
album_levels
```

Format

A vector of length 16. Each element is an album name, in an order that can be used for making factor variables.

Details

Albums are listed in release order, including the "Taylor's Version" releases. That means that *Fearless (Taylor's Version)* comes directly after *evermore*, rather than after *Taylor Swift* or the original *Fearless*.

Examples

```
library(ggplot2)
studio_albums <- subset(taylor_albums, !ep)

# by default, albums get plotted in alphabetical order
ggplot(studio_albums, aes(x = metacritic_score, y = album_name)) +
  geom_col()

# use `album_levels` to create a sensible factor variable
studio_albums$album_name <- factor(studio_albums$album_name,
                                   levels = album_levels)
ggplot(studio_albums, aes(x = metacritic_score, y = album_name)) +
  geom_col()
```

`album_palettes` *Color palettes based on Taylor Swift's albums*

Description

Premade color palettes based on Taylor Swifts album covers. For details on how to extend and shorten these palettes, or create your own color palette, see `color_palette()`.

Usage

```
album_palettes

album_compare
```

Format

A list of length 16. Each element contains a color palette for one of Taylor Swift's studio albums. The list elements are named according to the name of the album.

An object of class `taylor_color_palette` (inherits from `vctrs_vctr`, `character`) of length 16.

Source

Colors derived from album covers using 'Colormind'.

See Also

`color_palette()`

Examples

```
album_palettes

album_compare

album_palettes$evermore
```

`color_palette` *Create a custom color palette*

Description

This creates a character vector that represents palettes so when it is printed, it displays the palette colors.

Usage

```
color_palette(pal = character(), n = length(pal))

is_color_palette(pal)
```

Arguments

`pal`	• For `color_palette()`: A character vector of hexadecimal codes • For `is_color_palette()`: An object to test
`n`	The number of colors

Value

A color palette object.

Examples

```
# use color_palette() to extend or shorten an existing palette
color_palette(album_palettes$lover, n = 10)

color_palette(album_palettes$fearless, n = 10)

color_palette(album_palettes$red, n = 3)

# you can also define your own color palette
(my_pal <- color_palette(pal = c("#264653", "#2A9D8F", "#E9C46A",
                                 "#F4A261", "#E76F51")))

# and then use that palette for plotting
library(ggplot2)
ggplot(faithfuld) +
  geom_tile(aes(waiting, eruptions, fill = density)) +
  scale_fill_gradientn(colours = my_pal) +
  theme_minimal()
```

`eras_tour_surprise` *The Eras Tour Surprise Songs*

Description

A data set containing all of the surprise songs played on The Eras Tour through the first North American leg of the tour.

Usage

```
eras_tour_surprise
```

Format

A tibble with 298 rows and 9 variables:

- `leg`: The leg of the tour (e.g., North America, South America, etc.).
- `date`: The date of the show in ISO 8601 format (yyyy-mm-dd).
- `city`: The location of the show. For US shows, the location is the city and state. For international shows, the location is the city and country.
- `night`: The show number within each city.
- `dress`: The color of the dress Taylor wore on the given night.
- `instrument`: The instrument used to play the song (guitar or piano).
- `song`: The track name of the primary surprise song.
- `mashup`: Additional songs included in a mashup with the primary song.
- `guest`: The special guest (if any) that joined Taylor to play the song.

`get_reccobeats_audio_features` *Reccobeats audio features*

Description

Access the Reccobeats API to get audio features for tracks.

Usage

```
get_reccobeats_audio_features(track_id)
```

Arguments

`track_id`	The Spotify ID for a track.

Value

- `get_reccobeats_audio_features()` returns a tibble with track audio features, including:
  - `danceability`: Suitability for dancing (0.0 to 1.0). Higher values indicate more rhythmically engaging tracks.
  - `energy`: Intensity and liveliness (0.0 to 1.0). Higher values indicate more energetic tracks.
  - `loudness`: Average loudness in decibels (dB). Typically ranges between -60 and 0 dB.
  - `speechiness`: Presence of spoken words (0.0 to 1.0). Values above 0.66 indicate mostly speech.
  - `acousticness`: Confidence (0.0 to 1.0) that the track is acoustic. Higher values indicate more natural sounds.
  - `instrumentalness`: Likelihood of no vocals (0.0 to 1.0). Values above 0.5 suggest instrumental tracks.
  - `liveness`: Probability of a live audience (0.0 to 1.0). Values above 0.8 strongly suggest a live track.
  - `valence`: Emotional tone (0.0 to 1.0). Higher values indicate a happier mood, lower values a sadder one.
  - `tempo`: Estimated tempo in beats per minute (BPM). Typically ranges between 0 and 250.
  - `key`: The key the track is in. Integers map to pitches using standard Pitch Class notation. If no key was detected, the value is -1.
  - `mode`: Mode indicates the modality (major or minor) of a track. Major is represented by 1 and minor is 0.
  - `key_name`: Corresponds directly to the key, but the integer is converted to the key name using Pitch Class notation (e.g., 0 becomes `C`).
  - `mode_name`: Corresponds directly to the mode, but the integer is converted to the mode name (e.g., 0 becomes `minor`).
  - `key_mode`: A combination of the `key_name` and `mode_name` variables (e.g., `C minor`).

See Also

Other API access: `get_soundstat_audio_features()`, `get_spotify_track_info()`

Examples

```
# So High School
get_reccobeats_audio_features(track_id = "7Mts0OfPorF4iwOomvfqn1")
```

---

get_soundstat_audio_features

SoundStat audio features

Description

Access the SoundStat API to get audio features for tracks.

Usage

```
get_soundstat_audio_features(
  track_id,
  convert_values = FALSE,
  api_key = get_soundstat_api_key()
)
```

Arguments

`track_id`	The Spotify ID for a track.
`convert_values`	Logical. For SoundStat features, should audio features be converted to the Spotify scale. See details for conversion formulas.
`api_key`	A SoundStat API key, e.g., `get_soundstat_api_key()`.

Details

Due to differences in algorithms and methodologies, the SoundStat audio features are on a slightly different scale than the audio features that were originally included in taylor prior to the changes to the Spotify API. We can convert the SoundStat values to the Spotify scale using the formulas in the SoundStat docs:

```
acousticness = sound_value * 0.005
energy = sound_value * 2.25
instrumentalness = sound_value * 0.03
loudness = -(1 - sound_value) * 14
```

To automatically perform these conversions, set `convert_values = TRUE`.

Value

A tibble with track audio features, including:

- `danceability`: Danceability score (0-1).
- `energy`: Energy level (0-1).
- `loudness`: Loudness level (0-1).
- `acousticness`: Acousticness score (0-1).
- `instrumentalness`: Instrumentalness score (0-1).
- `valence`: Mood/positiveness (0-1).
- `tempo`: Track tempo in beats per minute (BPM).
- `key`: Track key (0-11).
- `mode`: Mode (0 - minor, 1 - major).
- `key_name`: Corresponds directly to the key, but the integer is converted to the key name using Pitch Class notation (e.g., 0 becomes `C`).
- `mode_name`: Corresponds directly to the mode, but the integer is converted to the mode name (e.g., 0 becomes `minor`).
- `key_mode`: A combination of the `key_name` and `mode_name` variables (e.g., `C minor`).

See Also

Other API access: `get_reccobeats_audio_features()`, `get_spotify_track_info()`

Examples

```
get_soundstat_audio_features(track_id = "7Mts0OfPorF4iwOomvfqn1")
```

---

get_spotify_track_info *Spotify track information*

Description

Access the Spotify API to get metadata for audio tracks.

Usage

```
get_spotify_track_info(track_id, api_key = get_spotify_access_token())
```

Arguments

`track_id`	The Spotify ID for a track.
`api_key`	A Spotify access token, from `get_spotify_access_token()`.

Value

A tibble with track metadata, including:

- `album_name`: The name of the album the track appears on (if relevant).
- `track_name`: The name of the track.
- `artist`: The artist of the track.
- `featuring`: The artist(s) featured on the track (if relevant).
- `duration_ms`: Duration of the track in milliseconds.
- `explicit`: Logical. Does the track contain explicit lyrics (`TRUE`) or not (`FALSE`).

See Also

Other API access: `get_reccobeats_audio_features()`, `get_soundstat_audio_features()`

Examples

```
# So High School
get_spotify_track_info(track_id = "7Mts0OfPorF4iwOomvfqn1")
```

`scale_colour_taylor_d` *Taylor Swift colour scales based on album cover palettes*

Description

Taylor Swift colour scales based on album cover palettes

Usage

```
scale_colour_taylor_d(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
  album = "Lover",
  aesthetics = "colour"
)

scale_color_taylor_d(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
  album = "Lover",
  aesthetics = "colour"
)

scale_fill_taylor_d(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
  album = "Lover",
  aesthetics = "fill"
)

scale_colour_taylor_c(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
  album = "Lover",
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "colour"
)

scale_color_taylor_c(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
  album = "Lover",
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "colour"
)

scale_fill_taylor_c(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
  album = "Lover",
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "fill"
)

scale_colour_taylor_b(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
  album = "Lover",
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "coloursteps",
  aesthetics = "colour"
)
```

```
scale_color_taylor_b(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
  album = "Lover",
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "coloursteps",
  aesthetics = "colour"
)

scale_fill_taylor_b(
  ...,
  alpha = 1,
  begin = 0,
  end = 1,
  direction = 1,
  album = "Lover",
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "coloursteps",
  aesthetics = "fill"
)
```

Arguments

`...`	Other arguments passed on to `ggplot2::discrete_scale()`, `ggplot2::continuous_scale()`, or `ggplot2::binned_scale()` to control name, limits, breaks, labels and so forth.
`alpha`	The alpha transparency, a number in [0,1], see argument alpha in `hsv`.
`begin, end`	The (corrected) hue in `[0,1]` at which the color map begins and ends.
`direction`	Sets the order of colors in the scale. If 1, the default, colors are ordered from darkest to lightest. If -1, the order of colors is reversed.
`album`	A character string indicating the album that should be used for the palette.
`aesthetics`	Character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with. This can be useful, for example, to apply colour settings to the `colour` and `fill` aesthetics at the same time, via `aesthetics = c("colour", "fill")`.
`values`	if colours should not be evenly positioned along the gradient this vector gives the position (between 0 and 1) for each colour in the `colours` vector. See `rescale()` for a convenience function to map an arbitrary range to between 0 and 1.

space	colour space in which to calculate gradient. Must be "Lab" - other values are deprecated.
na.value	Missing values will be replaced with this value.
guide	A function used to create a guide or its name. See `guides()` for more information.

Value

A color scale for use in plots created with `ggplot2::ggplot()`.

Examples

```
# use taylor_d with discrete data
library(ggplot2)
(p <- ggplot(taylor_album_songs, aes(x = valence, y = energy)) +
   geom_point(aes(color = mode_name), size = 2) +
   theme_bw())
p + scale_color_taylor_d()

# change scale label
p + scale_fill_taylor_d("Mode of Track")

# select album palette to use, see `?album_palettes` for more details
lover <- subset(taylor_album_songs, album_name == "Lover")
(p <- ggplot(lover, aes(x = valence, y = track_name)) +
   geom_col(aes(fill = track_name)) +
   theme_minimal())
p + scale_fill_taylor_d(album = "Lover")
p + scale_fill_taylor_d(album = "evermore")

# use taylor_c with continuous data
(p <- ggplot(taylor_album_songs, aes(valence, energy)) +
   geom_point(aes(color = danceability)) +
   theme_minimal())
p + scale_color_taylor_c(album = "Fearless")
p + scale_color_taylor_c(album = "folklore")
```

`scale_fill_albums` *Taylor Swift colour scale for album comparisons*

Description

A convenience wrapper for comparing albums with color. In contrast to `scale_fill_taylor_d()` and `scale_colour_taylor_d()`, `scale_fill_albums()` and `scale_colour_albums()` use a single palette, with one color per album. Specifically, the `album_compare` palette is used to apply a color associated with each album.

Usage

```
scale_fill_albums(
  ...,
  aesthetics = "fill",
  breaks = waiver(),
  limits = force,
  na.value = NA
)

scale_colour_albums(
  ...,
  aesthetics = "colour",
  breaks = waiver(),
  limits = force,
  na.value = NA
)

scale_color_albums(
  ...,
  aesthetics = "colour",
  breaks = waiver(),
  limits = force,
  na.value = NA
)
```

Arguments

`...`	Other arguments to be passed to `ggplot2::discrete_scale()`
`aesthetics`	The names of the aesthetics that this scale works with.
`breaks`	One of: • `NULL` for no breaks • `waiver()` for the default breaks (the scale limits) • A character vector of breaks • A function that takes the limits as input and returns breaks as output. Also accepts rlang lambda function notation.
`limits`	One of: • `NULL` to use the default scale values • A character vector that defines possible values of the scale and their order • A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang lambda function notation.
`na.value`	If `na.translate = TRUE`, what aesthetic value should the missing values be displayed as? Does not apply to position scales where `NA` is always placed at the far right.

Value

A color scale for use in plots created with `ggplot2::ggplot()`.

Examples

```
library(ggplot2)
studio <- subset(taylor_albums, !is.na(metacritic_score))

# create a plot that we want to color or fill by album
ggplot(studio, aes(x = metacritic_score, y = album_name)) +
  geom_col(aes(fill = album_name))

# apply a color inspired by each album cover
ggplot(studio, aes(x = metacritic_score, y = album_name)) +
  geom_col(aes(fill = album_name)) +
  scale_fill_albums()

# even when the axis or levels are rearranged, the correct color is applied
studio$album_name <- factor(studio$album_name, levels = album_levels)
ggplot(studio, aes(x = metacritic_score, y = album_name)) +
  geom_col(aes(fill = album_name)) +
  scale_fill_albums()
```

`soundstat-api` *SoundStat API helpers*

Description

Set and retrieve a SoundStat API key

Usage

```
get_soundstat_api_key()

set_soundstat_api_key(key = NULL)
```

Arguments

`key`	A SoundStat API key.

Value

- `get_soundstat_api_key()` returns a previously stored API key.
- `set_soundstat_api_key()` is called for side effects only.

Examples

```
get_soundstat_api_key()
```

`spotify-api` *Spotify API helpers*

Description

Set and retrieve Spotify client information.

Usage

```
get_spotify_access_token()

set_spotify_api_key(id = NULL, secret = NULL)
```

Arguments

`id`	A Spotify Client ID.
`secret`	A Spotify Client Secret.

Value

- `get_spotify_api_key()` returns a previously stored Client ID and Secret.
- `set_spotify_api_key()` is called for side effects only.

Examples

```
get_spotify_access_token()
```

`taylor_albums` *Data for Taylor Swift's studio albums and EPs*

Description

A data set containing the names of Taylor's official releases, the album type, and release date.

Usage

```
taylor_albums
```

Format

A tibble with 18 rows and 5 variables:

- `album_name`: The name of the album. `NA` if the song was released separately from one of Taylor's studio albums or EPs.
- `ep`: Logical. Is the album a full studio album (`FALSE`) or an extended play (`TRUE`).
- `album_release`: The date the album was released, in the ISO-8601 format (YYYY-MM-DD).
- `metacritic_score`: The official album rating from metacritic.
- `user_score`: The user rating from metacritic.

Details

This data set includes all official studio albums and EPs with new tracks. This means that compilations or EPs that are a subset of the original albums are not included (e.g., *folklore: the escapism chapter*, *folklore: the sleepless nights chapter*, etc.)

Source

https://en.wikipedia.org/wiki/Taylor_Swift_albums_discography

https://www.metacritic.com/person/taylor-swift

`taylor_all_songs` *Data for Taylor Swift songs*

Description

A data set containing lyrics to, and characteristics of, all officially released Taylor Swift songs. This includes albums, EPs, and individually released singles.

Usage

```
taylor_all_songs

taylor_album_songs
```

Format

`taylor_all_songs` is a tibble with 384 rows and 26 variables. Each row is one song.

- `album_name`: The name of the album. `NA` if the song was released separately from one of Taylor's studio albums or EPs.
- `ep`: Logical. Is the album a full studio album (`FALSE`) or an extended play (`TRUE`).
- `album_release`: The date the album was released, in the ISO-8601 format (YYYY-MM-DD).
- `track_number`: The order of the song on the album or EP.
- `track_name`: The name of the song.

- `artist`: The name of the song artist. Usually Taylor Swift, but will show other artists for songs that Taylor is only featured on.
- `featuring`: Any artists that are featured on the track.
- `bonus_track`: Logical. Is the track only present on a deluxe edition of the album (`TRUE`) or is does it also appear on the standard version (`FALSE`).
- `promotional_release`: The date the song was released as a promotional single, in the ISO-8601 format (YYYY-MM-DD).. `NA` if the song was never released as a promotional single.
- `single_release`: The date the song was released as an official single, in the ISO-8601 format (YYYY-MM-DD). `NA` if the song was never released as an official single.
- `track_release`: The date the song was first publicly released. This is the earliest of `album_release`, `promotional_release`, and `single_release`.

The next set of variables come from the SoundStat API. See the documentation at `https://soundstat.info/` for complete details.

- `danceability`: How suitable a track is for dancing. `0.0` = least danceable, `1.0` = most danceable.
- `energy`: Perceptual measure of intensity and activity. `0.0` = least energy, `1.0` = most energy.
- `loudness`: Loudness of track in decibels (dB), averaged across the track.
- `acousticness`: Confidence that the track is acoustic. `0.0` = low confidence, `1.0` = high confidence.
- `instrumentalness`: Confidence that the track is an instrumental track (i.e., no vocals). `0.0` = low confidence, `1.0` = high confidence.
- `valence`: Musical positiveness conveyed by the track. `0.0` = low valence (e.g., sad, depressed, angry), `1.0` = high valence (e.g., happy, cheerful, euphoric).
- `tempo`: Estimated tempo of the track in beats per minute (BPM).
- `duration_ms`: Duration of the track in milliseconds.
- `explicit`: Logical. Does the track contain explicit lyrics (`TRUE`) or not (`FALSE`).
- `key`: The key the track is in. Integer maps to standard Pitch Class notation.
- `mode`: Modality of a track (major/minor). `0` = minor, `1` = major.

Finally, the last set of variables includes those calculated from the SoundStat API data, and a list-column containing song lyrics.

- `key_name`: Corresponds directly to the `key`, but the integer is converted to the key name using Pitch Class notation (e.g., `0` becomes `C`).
- `mode_name`: Corresponds directly to the `mode`, but the integer is converted to the mode name (e.g., `0` becomes `minor`).
- `key_mode`: A combination of the `key_name` and `mode_name` variables (e.g., `C minor`).
- `lyrics`: A list-column containing the lyrics to each song. The lyrics can be unnested with `tidyr::unnest()` (i.e., `tidyr::unnest(taylor_all_songs, lyrics)`). Each element is a data frame with 4 variables. :
  - `line`: The line number of the song.
  - `lyric`: The lyric for the given line.

- `element`: The element of the song the line and lyric belong to, as defined by `https://genius.com/` (e.g., `Verse 1`, `Chorus`, etc.).
- `element_artist`: The artist performing the element. Usually `Taylor Swift`, but other artists appear if they are featured on the track (e.g., `HAIM` is featured on *no body, no crime*).

`taylor_album_songs` is a tibble containing the same 26 variables, but only with 332 rows, one for each song from an official studio album (see Details).

Details

`taylor_all_songs` contains all songs in Taylor's discography. Lyrics come from Genius, and songs characteristics come from the SoundStat API. Some data is known to be missing. The Beautiful Eyes EP is not available on any streaming service, and therefore has no data from the SoundStat API. Similarly, the song *American Girl*, a cover of the Tom Petty original, was released exclusively on Rhapsody (now Napster), and therefore also does not have data from the SoundStat API.

For songs released separately from Taylor's official albums or EPs, full album information is not included. For example, *I Don't Wanna Live Forever* was released as part of the *Fifty Shades Darker (Original Motion Picture Soundtrack)*. However, the `album_release` and `track_number` columns for this song are `NA`.

Songs are only included one time. For example, if a song appears on both the standard and deluxe version of an album, there is only one record of the song in the data set. Similarly, compilations are not included. For example, following the release of *folklore*, Taylor released several EPs that were subsets of the original *folklore* album (e.g., *folklore: the escapism chapter*, *folklore: the sleepless nights chapter*, etc.). These are not included. Finally, all bonus tracks (with the exception of voice memos or similar) are included; however, for consistency, the album name is always the shortened, common name. For example, *the lakes* is a bonus track from *folklore (deluxe edition)*, but the `album_name` is listed only as *folklore*. The `bonus_track` variable can be used to determine which songs appeared on the standard version of an album vs. a deluxe or platinum edition.

`taylor_album_songs` contains the same information as `taylor_all_songs`, but only for Taylor's official studio albums. Thus, `taylor_album_songs` is a subset of `taylor_all_songs`, with EPs and individual singles excluded.

Source

`https://genius.com/artists/Taylor-swift`

`https://soundstat.info`

`taylor_examples` *Determine if code is executed interactively or in pkgdown*

Description

Used for determining examples that shouldn't be run on CRAN, but can be run for the pkgdown website.

Usage

```
taylor_examples()
```

Value

A logical value indicating whether or not the examples should be run.

Examples

```
taylor_examples()
```

`taylor_sitrep` *Situation report for APIs*

Description

Check configuration status for Spotify and SoundStat APIs.

Usage

```
taylor_sitrep()
```

Value

Called for side effects.

Examples

```
taylor_sitrep()
```

`title_case` *Convert string to title case*

Description

Capitalize the first letter of each word, and convert the remaining string to lower case.

Usage

```
title_case(string)
```

Arguments

`string`	String to modify.

Value

A character string with the first letter of each word capitalized.

Examples

```
title_case("taylor swift")
title_case("Taylor Swift")
title_case("TAYLOR SWIFT")
```

`translate_bracelet` *Translate a friendship bracelet*

Description

Receive a friendship bracelet at the Eras Tour but can't figure out what lyrics the bracelet has abbreviated? Now you can find out!

Usage

```
translate_bracelet(abbr)
```

Arguments

`abbr`	The abbreviated lyrics (i.e., the first letter of each word).

Value

A character vector with the name of the song and the abbreviated line.

Examples

```
translate_bracelet("bykmosftvfw")

translate_bracelet("kimbkiagkitbimhotw")
```

Index