Experiments

1 Experiments

1.1 Correlation Between Foreign Language Knowledge and Listening Foreing Music

The main purpose of this experiment is to examine whether there is a relationship betweeen knowledge of foreign language and listining of foreign music ratios. We accepted English as a foreign language because of globalization. For doing this experiment, English knowledge statistics on country basis were required and we reached this data from Wikizero. This resource includes the ratio of English knowledge, which is an additional language for many countries. We created a table containing the rates of English knowledge of the countries that match the our data countries by extracting the countries whose native language is English.

Another aim in this experiment is to estimate the rate of English knowledge for other countries if the correlation is high for the relevant 24 countries.

("http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvTGlzdF9vZl9jb3VudHJpZXN

Table 1: Knowledge of English Ratios

RegionName	English_Knowledge_Ratio
Austria	73.00
Belgium	60.00
Brazil	5.00
Chile	9.53
Colombia	4.22
Costa Rica	8.20
Czech Republic	27.00
Denmark	86.00
Dominican Republic	60.00
Estonia	50.00
Finland	70.00
France	39.00
Germany	56.00
Greece	51.00
Italia	34.00
Latvia	46.00
Lithuania	38.00
Luxembourg	56.00
Mexico	11.60
Poland	37.00
Portugal	27.00
Slovakia	26.00
Spain	22.00
Sweden	86.00
Switzerland	61.28
The Netherlands	90.00
Turkey	17.00

Before we go into the correlation calculation, we need to create the variables such as region and track language that are missing in raw data but will be needed.

1.1.0.1 Detection of Region Language

In order to calculate the rate of listening the foreign music on a country basis, we need to know the national language(s) of the countries. We created the region language variable by using the table where ISO 3166-1 country codes match the ISO 639-1 language codes. ("https://wiki.openstreetmap.org/wiki/Nominatim/Country_Codes")

Table 2: Spotify Data With Region Language

Date	RegionName	RegionLanguage	TrackName	Streams
2017-01-01	Argentina	Spanish	Hasta la luna	35806
2017-01-01	Australia	English	Hello	14778
2017-01-01	Austria	German	Holz	3968
2017-01-01	Belgium	Dutch, Flemish, French, German	Someone Like You	2711
2017-01-01	Bolivia	Spanish, Quechua, Guaraní, Aymara	Side To Side	1440
2017-01-01	Brazil	Portuguese	Send My Love (To Your New Lover)	44361
2017-01-01	Canada	English, French	No Heart	34538
2017-01-01	Chile	Spanish	Faded	10379
2017-01-01	Colombia	Spanish	Hello	3406
2017-01-01	Costa Rica	Spanish	Hello	2119

1.1.1 Detection of Track Language

Another missing point was that we did not know the languages of the tracks in raw data. I have searched the Python and R libraries which give the language as output when the word or sentence is given as input. In R, CLD3 (Compact Language Detector 3) and textcat libraries can be used to detect language.

CLD3 is a neural network model for language identification. This package contains the inference code and a trained model. The inference code extracts character n-grams from the input text and computes the fraction of times each of them appears. The other library, textcat, has language profiles for 75 languages and some of the algorithms at its core have been used inside Spamassassin. It can be manipulated using language profiles and various options among which the selection of a distance function.

cld3 libraryu example

```r
library(cld3)
text <- c("Rechtdoor gaan, dan naar rechts.",
"Türkiye'nin üç tarafı denizlerle çevrili.",
"I live in the countryside",
"Questa frase non è scritta in Napoletano.",
"Das ist ein deutscher satz.",
"La vie est magnifique",
"El jugador está predispuesto a que será un partido complicado.",
"Jar kan ikke snakke Norsk","Bom dia")
detect_language(text)
```

```
## [1] "nl" "tr" "en" "it" "de" "fr" "es" "da" "pt"
```

textcat library example

```r
library(textcat)

text <- c("Rechtdoor gaan, dan naar rechts.",
"Türkiye'nin üç tarafı denizlerle çevrili.",
"I live in the countryside",
"Questa frase non è scritta in Napoletano.",
"Das ist ein deutscher satz.",
"La vie est magnifique",
"El jugador está predispuesto a que será un partido complicado.",
"Jar kan ikke snakke Norsk","Bom dia" )
textcat(text)
```

```
## [1] "dutch"      "turkish"    "english"    "italian"    "german"
## [6] "french"     "spanish"    "norwegian"  "indonesian"
```

After investigating the R libraries, we wanted to see what the language detection libraries in Python environment were and how they were compared to R. We found out that langdetect library is one derived directly from Google language detection. This Python library seems to be well alive and maintained. It claims it can detect 55 languages out of the box and upon a simple call to the function "detect" will return the two letter ISO 639-1 code of the language detect while a call to detect_lang will return a vector of probabilities strings. There are alternative detection libraray, langid, in Python as well as in R. This library pre-trained over a large number of languages (currently 97) and not sensitive to domain-specific features (e.g. HTML/XML markup). classify() function of this libratay can be used to get the most likely language and its "score".

```python
import langdetect
documents = ["Rechtdoor gaan, dan naar rechts.",
"Türkiye'nin üç tarafı denizlerle çevrili.",
"I live in the countryside",
"Questa frase non è scritta in Napoletano.",
"Das ist ein deutscher satz.",
"La vie est magnifique",
"El jugador está predispuesto a que será un partido complicado.",
"Jar kan ikke snakke Norsk","Bom dia"]

for line in documents:
 print(langdetect.detect_langs(line))
```

```
## [nl:0.9999962687431464]
## [tr:0.9999985591349785]
## [en:0.9999954632098351]
## [it:0.9999962068295052]
## [de:0.9999987233728493]
## [fr:0.9999960204145566]
## [es:0.999998513356376]
## [no:0.9999939668244664]
## [pt:0.9999968047446296]
```

```python
import langid
documents = ["Rechtdoor gaan, dan naar rechts.",
"Türkiye'nin üç tarafı denizlerle çevrili.",
"I live in the countryside",
"Questa frase non è scritta in Napoletano.",
"Das ist ein deutscher satz.",
"La vie est magnifique",
"El jugador está predispuesto a que será un partido complicado.",
"Jar kan ikke snakke Norsk","Bom dia"]

for line in documents:
 print(langid.classify(line))
```

```
## ('nl', -73.47373533248901)
## ('tr', -150.72091460227966)
## ('en', -28.477052211761475)
## ('it', -64.93842697143555)
## ('de', -154.69498538970947)
## ('fr', -47.38988637924194)
## ('es', -346.22576999664307)
## ('nb', -34.800633907318115)
## ('en', 9.061840057373047)
```

Comparing the results of the language sensing of the libraries in both R and python, we see that there is no big difference between the accuracy, but the langdetect library has reached the most correct result with the probability information. I also considered the Python's NLTK library (Natural Language Toolkit), but the scope of the power of this library seems to be beyond our quest for a quick language detection solution therefore I did not compared with others. As a result, using the langid.classify() function of the langdetect library of python, song languages have been identified with the probability of estimation. The tracks with the language estimation rate of 50 and below were extracted from the data in the experimental phase.

Table 3: Spotify Data With Track Language

Date	RegionName	RegionLanguage	TrackName	TrackLanguage	Score
2017-01-01	Turkey	Turkish	How Deep Is Your Love	Dutch, Flemish	0.5714291
2017-01-01	Turkey	Turkish	Olsun	German	0.5714281
2017-01-01	Turkey	Turkish	Gir Kanıma	Turkish	0.9999990
2017-01-01	Turkey	Turkish	Bu Kız	Turkish	0.9999987
2017-01-01	Turkey	Turkish	Rap God	Indonesian	0.9996486
2017-01-01	Turkey	Turkish	Kimsenin Suçu Yok	Turkish	0.7142843
2017-01-01	Turkey	Turkish	Dokunma	Finnish	0.9999969
2017-01-01	Turkey	Turkish	Dynamite (feat. Pretty Sister)	English	0.8571397
2017-01-01	Turkey	Turkish	I Got It Bad (And That Ain't Good)	English	0.9999990
2017-01-01	Turkey	Turkish	Bir Pazar Kahvaltısı	Turkish	0.9999967

1.1.2 Language Flagging

At this stage, we created the flag variable by assigning 0 to the different ones and 1 to the same ones to show whether the languages of the songs and the local languages are the same. For countries with multiple official languages, we assigned 1 if the language of the song is one of the official languages.

Table 4: Flagged Spotify Data

Date	RegionName	RegionLanguage	TrackName	TrackLanguage	flag
2017-01-01	Turkey	Turkish	How Deep Is Your Love	English	0
2017-01-01	Turkey	Turkish	Olsun	German	0

Date	RegionName	RegionLanguage	TrackName	TrackLanguage	flag
2017-01-01	Turkey	Turkish	Gir Kanıma	Turkish	1
2017-01-01	Turkey	Turkish	Bu Kız	Turkish	1
2017-01-01	Turkey	Turkish	Rap God	English	0
2017-01-01	Turkey	Turkish	Kimsenin Suçu Yok	Turkish	1
2017-01-01	Turkey	Turkish	Dokunma	English	0
2017-01-01	Turkey	Turkish	Dynamite (feat. Pretty Sister)	English	0
2017-01-01	Turkey	Turkish	I Got It Bad (And That Ain't Good)	English	0
2017-01-01	Turkey	Turkish	Bir Pazar Kahvaltısı	Turkish	1

1.1.3 Correlation

We had foreign language knowledge, but we had to calculate the rate of listening to music in different languages by using the generated flag variable. For each country, we created this variable by dividing the number of music in the different language by the total number of music. Afterwards, we calculated the correlation between the rate of knowledge of English and listening to foreign music by using cor() function of the stats library.

Table 5: Region With Ratios

Region_Name	Listening_of_Foreign_Music	Knowledge_of_English
Austria	87.98760	73.00
Belgium	87.39992	60.00
Brazil	77.57919	5.00
Chile	72.75056	9.53
Colombia	72.44015	4.22
Costa Rica	81.54319	8.20
Czech Republic	97.80651	27.00
Denmark	93.13645	86.00
Dominican Republic	74.86827	60.00
Estonia	99.97659	50.00
Finland	70.30003	70.00
France	78.67907	39.00
Germany	82.85483	56.00
Greece	99.64317	51.00
Italia	77.28756	34.00
Latvia	99.40596	46.00
Lithuania	99.92853	38.00
Luxembourg	96.77577	56.00
Mexico	71.67718	11.60
Poland	94.40887	37.00
Portugal	95.09867	27.00
Slovakia	99.39794	26.00
Spain	71.40123	22.00
Sweden	91.56221	86.00
Switzerland	86.30028	61.28
The Netherlands	90.38581	90.00
Turkey	72.96744	17.00

```
## [1] "Correlation: 0.353178118303188"
```

1.2 Paired t-test

In this experiment, it is aimed to measure whether there is a difference between the types of music that is listened during the weekends and week days. First of all we had to identify the genres therefore we searched the Spotify APIs that give tracks genre. Because we have track ids we started to investigate Tracks API of Spotify. Then, we noticed that we couldn't reach the tracks genres directly from the Track API which gives various information with the track such as artist, album, release date, available regions. While researching other apis, we found that we can access the artists genre from the Artists API. We decided to create the track genre varaible by accepting the artists genre as the tracks genre. "https://developer.spotify.com/console/artists/"

1.2.1 Getting Artist Id from Spotify Tracks API

Data files can be stored locally in our system or in our working directory, or a scenario where these files are stored on a remote server. Remote servers just the part of a remotely located computer that is optimized to process requests. In the cases that the data is stored on the remote servers, it is no longer available in expected file formats, such as .txt, .csv, .excel, and the most common form in which data is stored on the Web can be json, xml, html.When we want to access and work on Web Data, we can call the corresponding API using HTTP clients in R.

HTTP: Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers. There are many possible HTTP methods used to reach an API, below are the most commonly used:

GET: is used to request data from a specified resource.

POST: is used to send data to a server to create/update a resource.

In order to request and parse the data, we made use of R's httr and rjson libraries.

The httr library was builted on the six basic http methods: GET (), PATCH (), POST (), HEAD (), PUT (), and DELETE (). Each request returns a response () object that provides access to various information. The output of the request can be used as a raw vecor, character vector or as well as parsed to an R object for html, xml, json, png and jpeg. Configuration functions make it easier to modify the request in common ways: set_cookies(), add_headers(), authenticate(), use_proxy(), verbose(), timeout(), content_type(), accept(), progress(). This library supports for OAuth 1.0 and 2.0 with oauth1.0_token() and oauth2.0_token().

In our case, we have only used the GET method because we would not any create or update. In the request we made with the Get() function, we gave the the spotify track api url and authorization header as a parameter. The HTTP Authorization request header contains the credentials to authenticate a user agent with a server to not encounter 401 Unauthorized status response.

We also used the fromJSON() function of the rjson package to make meaningful returned json objects. JSON (JavaScript Object Notation) is a way to store information in an organized, easy-to-access manner. rjosn package converts JSON objects into R objects and vice-versa.

The following example shows how to retrieve variety of track informations from the spotify track api.

```r
library(magrittr)
library(dplyr)
library(rjson)
library(httr)
library(knitr)

url <- "https://api.spotify.com/v1/tracks/494OU6M7NOf4ICYb4zWCf5"

get_data <- GET(url, add_headers(Authorization=" Bearer BQCbq-K7uCK9uCkqS_X7M5Ut
HyT4rpMCFOPtGIKLqtw7QMfvulZvgYzNW_WxjTjCkNXuJoN1nPdyOlzYJu45qLxNZ3mpUMHbAO5Z19UY_
gS9Yys86H1AFJZBQSrU5UK3CTKKENDWsQVCaB80SHIyBqTpQlIBo5L4-8GBTr8lJebCG9CueJZYg30A1X
LY602BWbWvxB7nPqtzMnQCCM3iCKc8anQTld8PbKRRzn2gAREMYIZ-HD4qiCZj5qgQirGndNhrirCgWZd
2LTSISFh4JSFv5lmEBc585KE"))

track_name <- fromJSON(get_data %>% as.character())$name
artist_name <- fromJSON(get_data %>% as.character())$artists[[1]]$name
artist_id <- fromJSON(get_data %>% as.character())$artists[[1]]$id
album_name <- fromJSON(get_data %>% as.character())$album$name
album_release_date <- fromJSON(get_data %>% as.character())$album$release_date
album_total_tracks <- fromJSON(get_data %>% as.character())$album$total_tracks

print(c(paste0("Track Name: ", track_name),
        paste0("Artist Name: ", artist_name),
        paste0("Artist Name: ", artist_name),
        paste0("Artist Id: ", artist_id),
        paste0("Album Name: ", album_name),paste0("Album Release Date: ", album_release_date),
        paste0("Album Total Tracks Count: ", album_total_tracks)))
```

```
## [1] "Track Name: "                "Artist Name: "
## [3] "Artist Name: "               "Artist Id: "
## [5] "Album Name: "                "Album Release Date: "
## [7] "Album Total Tracks Count: "
```

TrackName	Artist	ArtistId
La Noche No Es para Dormir	Mano Arriba	4YBAOrBF9vBB9inOLtpRzp
Carnavalintro	CHANO	14lvrkdaXxggonwjKZpePM
Hellow Cotto	Mueva Records	79DSCFvlXAsjytJCUMZ4tb
Llamame Mas Temprano	Mano Arriba	4YBAOrBF9vBB9inOLtpRzp
Rockstar	Mueva Records	79DSCFvlXAsjytJCUMZ4tb
Tu Ritmo	Los Bonnitos	7d1kmnhwauydKmuAvjgQrH
Le Hace Falta un Beso	Agapornis	27Yc5RzJf27tJfqezJnHY1
Loquita	Marama	4GepMkTgrIZECoCC55vqjW
Vete	Khea	4m6ubhNsdwF4psNf3R8kwR
Era Tranquila	Marama	4GepMkTgrIZECoCC55vqjW

1.2.2 Getting Artist Genre from Spotify Artist API

After identifying artist ids, we get the artists genres by requesting the artists api such as in the previous experiment. Where the artist has more than one genres that are revieced from artist api , we assumed the determined first type as the artist genre hereby that is track genre.

```
url <- "https://api.spotify.com/v1/artists/04gDigrS5kc9YWfZHwBETP"

get_data <- GET(url, add_headers(Authorization= "Bearer BQCbq-K7uCK9uCkqS
_X7M5UtHyT4rpMCFOPtGIKLqtw7QMfvulZvgYzNW _WxjTjCkNXuJoN1nPdy0lzYJu45qLxNZ
3mpUMHbAO5Z19UY_gS9Yys86H1AFJZBQSrU5UK3CTKKENDWsQVCaB80SHIyBqTpQlIBo5L4-8
GBTr8lJebCG9CueJZYg30A1XLY602BWbWvxB7nPqtzMnQCCM3iCKc8anQTld8PbKRRzn2gAREM
YIZ-HD4qiCZj5qgQirGndNhrirCgWZd2LTSISFh4JSFv5lmEBc585KE"))
```

```r
artist_name <- fromJSON(get_data %>% as.character())$name
artist_genres <- fromJSON(get_data %>% as.character())$genre
total_followers_of_artist <- fromJSON(get_data %>% as.character())$followers$total
artist_popularity <- fromJSON(get_data %>% as.character())$popularity

print(c(paste0("Artist Name: ", artist_name),
        paste0("Artist Genres: ",artist_genres),
        paste0("Total Followers: ",total_followers_of_artist),
        paste0("Artist Popularity: ", artist_popularity )))
```

```
## [1] "Artist Name: "       "Artist Genres: "     "Total Followers: "
## [4] "Artist Popularity: "
```

Artist	ArtistGenre
Mano Arriba	cumbia pop
CHANO	argentine telepop
Mueva Records	
Mano Arriba	cumbia pop
Mueva Records	
Los Bonnitos	argentine telepop
Agapornis	cumbia pop
Marama	cumbia pop
Khea	argentine hip hop
Marama	cumbia pop
Paulo Londra	argentine hip hop
Agapornis	cumbia pop
Marama	cumbia pop
El Polaco	cumbia pop
Charly GarcÃa	argentine rock
Duki	argentine hip hop
Lali	argentine telepop
Marama	cumbia pop
Duki	argentine hip hop
Soda Stereo	argentine rock

1.2.3 Paired t-test

Firstly, we categorized the dates as weekend and weekday by utilizing the is.weekend() that is R chron package function.

Date	Weekend/Weekday
1/1/2017	Weekends
1/2/2017	Weekday
1/3/2017	Weekday
1/4/2017	Weekday
1/5/2017	Weekday
1/6/2017	Weekday
1/7/2017	Weekends
1/8/2017	Weekends
1/9/2017	Weekday

Date	Weekend/Weekday
1/10/2017	Weekday
1/11/2017	Weekday
1/12/2017	Weekday
1/13/2017	Weekday
1/14/2017	Weekends
1/15/2017	Weekends
1/16/2017	Weekday
1/17/2017	Weekday
1/18/2017	Weekday
1/19/2017	Weekday
1/20/2017	Weekday

Then, we found stream ratios of the genres grouped by region and weekend/weekday variables.

Artist_Genres	Region_Name	Weekends	Weekday
432hz	Australia	1.0000000	0.0000000
432hz	Czech Republic	0.3357027	0.6642973
432hz	Switzerland	0.0000000	1.0000000
a cappella	Australia	0.5359496	0.4640504
a cappella	Austria	0.4130078	0.5869922
a cappella	Belgium	0.6791493	0.3208507
a cappella	Canada	0.3825578	0.6174422
a cappella	Czech Republic	0.4779878	0.5220122
a cappella	Denmark	0.7969019	0.2030981
a cappella	Germany	0.4540402	0.5459598
a cappella	Switzerland	0.4743002	0.5256998
acid jazz	Austria	0.0000000	1.0000000
acid jazz	Czech Republic	0.0000000	1.0000000
acid jazz	Denmark	0.0000000	1.0000000
acid jazz	Switzerland	0.0000000	1.0000000
acoustic pop	Australia	0.3386148	0.6613852
acoustic pop	Austria	0.6876492	0.3123508
acoustic pop	Belgium	0.8446372	0.1553628
acoustic pop	Colombia	0.0000000	1.0000000
acoustic pop	Costa Rica	0.1017864	0.8982136

we used the t.test() function of stats package to calculate paired t test coefficients.

```
##
##  Paired t-test
##
## data:  df$Weekday and df$Weekends
## t = 29.756, df = 1743, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.3260165 0.3720269
## sample estimates:
## mean of the differences
##               0.3490217
```

1.3 Visualization

Maps are used in a variety of fields to express data in an appealing and interpretive way. Data can be expressed into simplified patterns, and this data interpretation is generally lost if the data is only seen through a spread sheet. Maps can add important context by incorporating many variables into an easy to read and applicable context. Maps are also very important in the information world because they can quickly allow the public to gain better insight so that they can stay informed.

1.3.1 Choropleth Map

In this experiment, we created choropleth maps to compare the listening rates of pop, latin, rock and rap/hiphop genres based on 53 countries in data. A choropleth map displays divided geographical areas or regions that are coloured in relation to a numeric variable. It allows to study how a variable evolutes along a territory. It is a powerful and widely used data visualization technique. The nueric variable uses colour progression to represent itself in each region of the map. Typically, this can be a blending from one colour to another, a single hue progression, transparent to opaque, light to dark or an entire colour spectrum. While creating choropleth maps we utilized R's two mapping packages: ggplot2, viridis. As in the previous sections, let's deep into the details of these packages.

The ggplot2 package, created by Hadley Wickham, offers a powerful graphics language for creating elegant and complex plots. Origianlly based on Leland Wilkinson's The Grammar of Graphics, ggplot2 allows us to create graphs that represent both univariate and multivariate numerical and categorical data in a straightforward manner. Grouped variables can be represented by color, symbol, size, and transparency. ggplot2 graphics are built step by step by adding new elements. Adding layers allows for extensive flexibility and customization of plots. To build a ggplot, we used the following template that can be used for different types of plots:

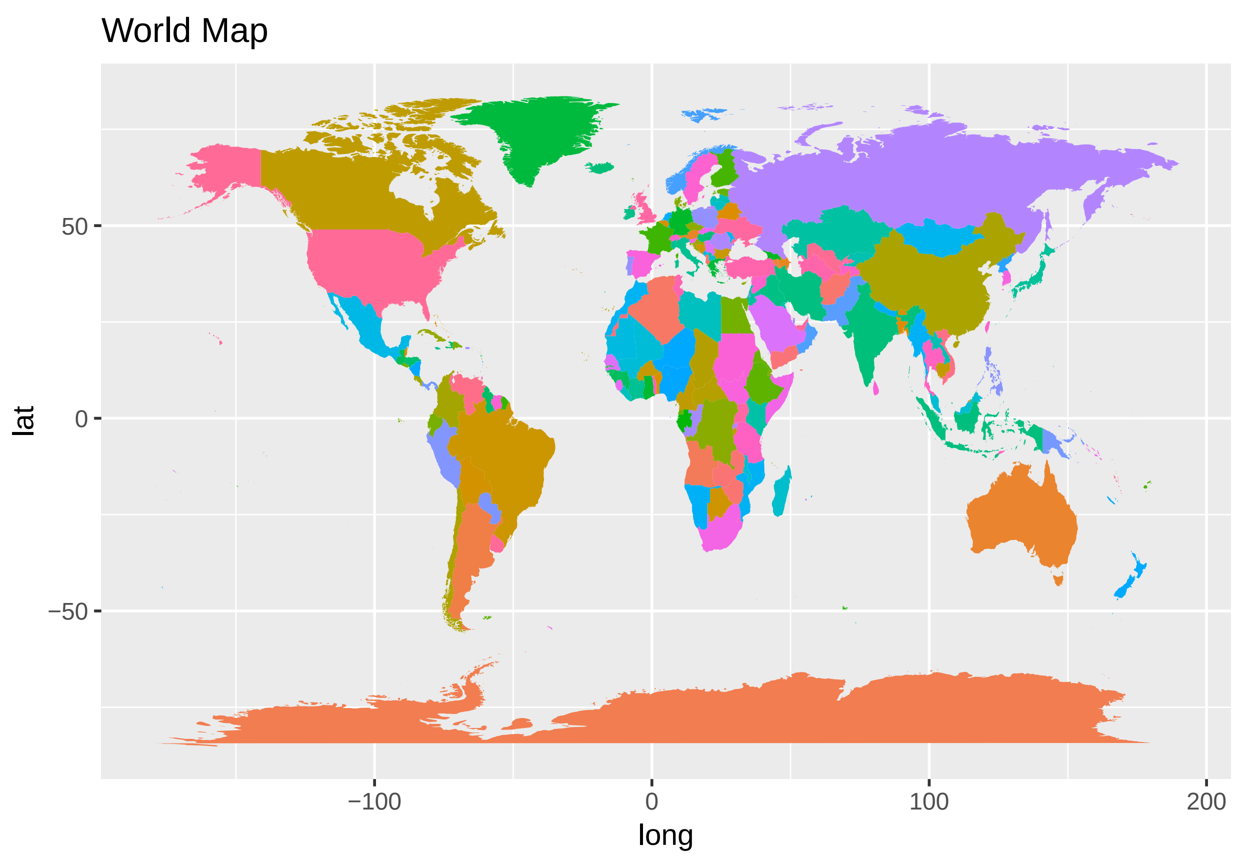
ggplot(data = , mapping = aes(MAPPINGS)) + GEOM_FUNCTION()

Knowing what elements are required to enhance data is key into making effective maps. Basic elements of a map that can be considered are polygon, points, lines, and text. Polygons are very similar to paths except that the start and end points are connected and the inside is coloured by fill. Lines are considered to be linear shapes that are not filled with any aspect, such as highways, streams, or roads. Points are used to specify specific positions, such as city or landmark locations. We used polygons element (geom_polygon()) to frame and color countries with latitude and longitude that are defined in aes parameter.

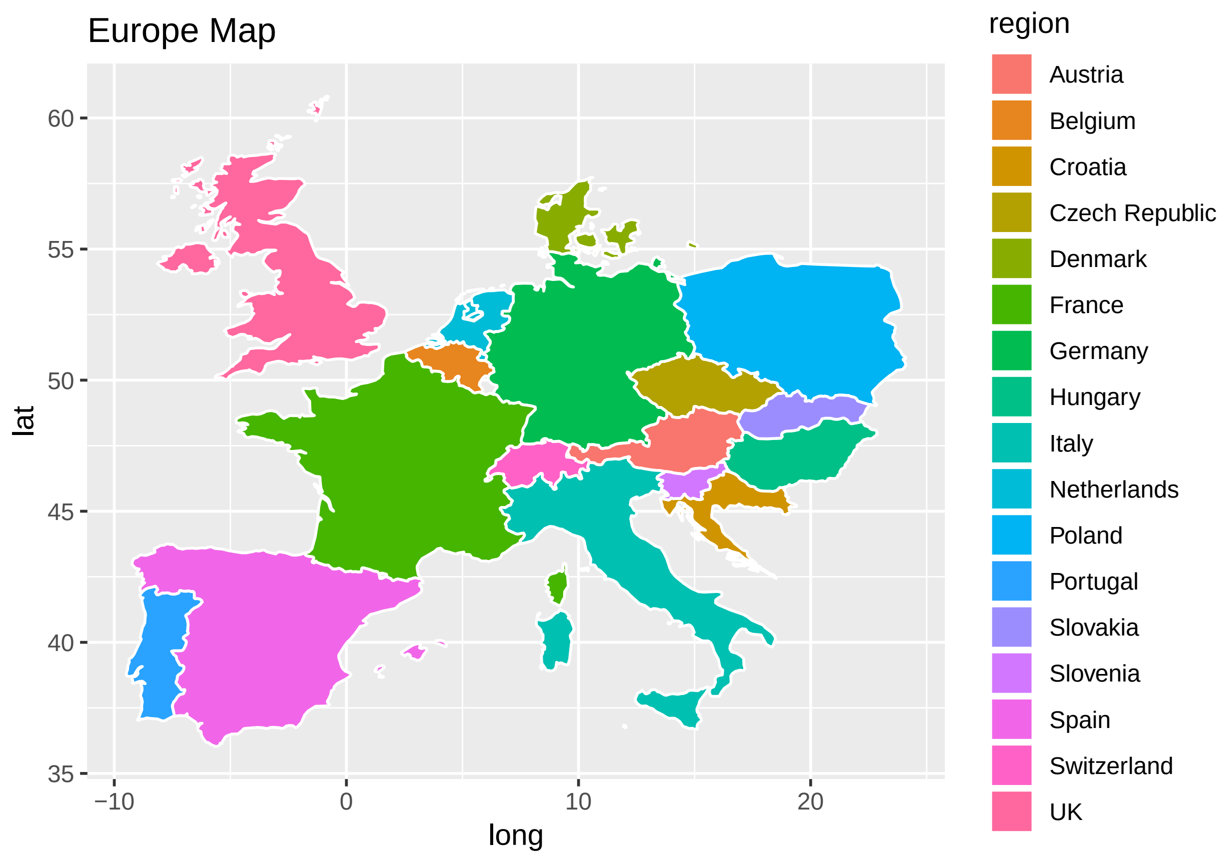
map_data() generates the map data with logitude and latitiude grouped by subregion. It requires the maps package. Specific continents countries, world map data can be created with this function.

geom_polygon() creates the map. In the following examples, the countries are colored by assigning the region to the fill argument.

```r
library(ggplot2)
require(maps)

worldmap = map_data("world")

ggplot(worldmap, aes(long, lat, group=group, fill=region)) +
geom_polygon(show.legend = F) +
ggtitle("World Map")
```



```
some.eu.countries <- c(
"Portugal", "Spain", "France", "Switzerland", "Germany",
"Austria", "Belgium", "UK", "Netherlands",
"Denmark", "Poland", "Italy",
"Croatia", "Slovenia", "Hungary", "Slovakia",
"Czech republic"
)

europemap <- map_data("world", region = some.eu.countries)

ggplot(europemap, aes(x = long, y = lat)) +
geom_polygon(aes( group = group, fill = region),color = "white")+
ggtitle("Europe Map")
```



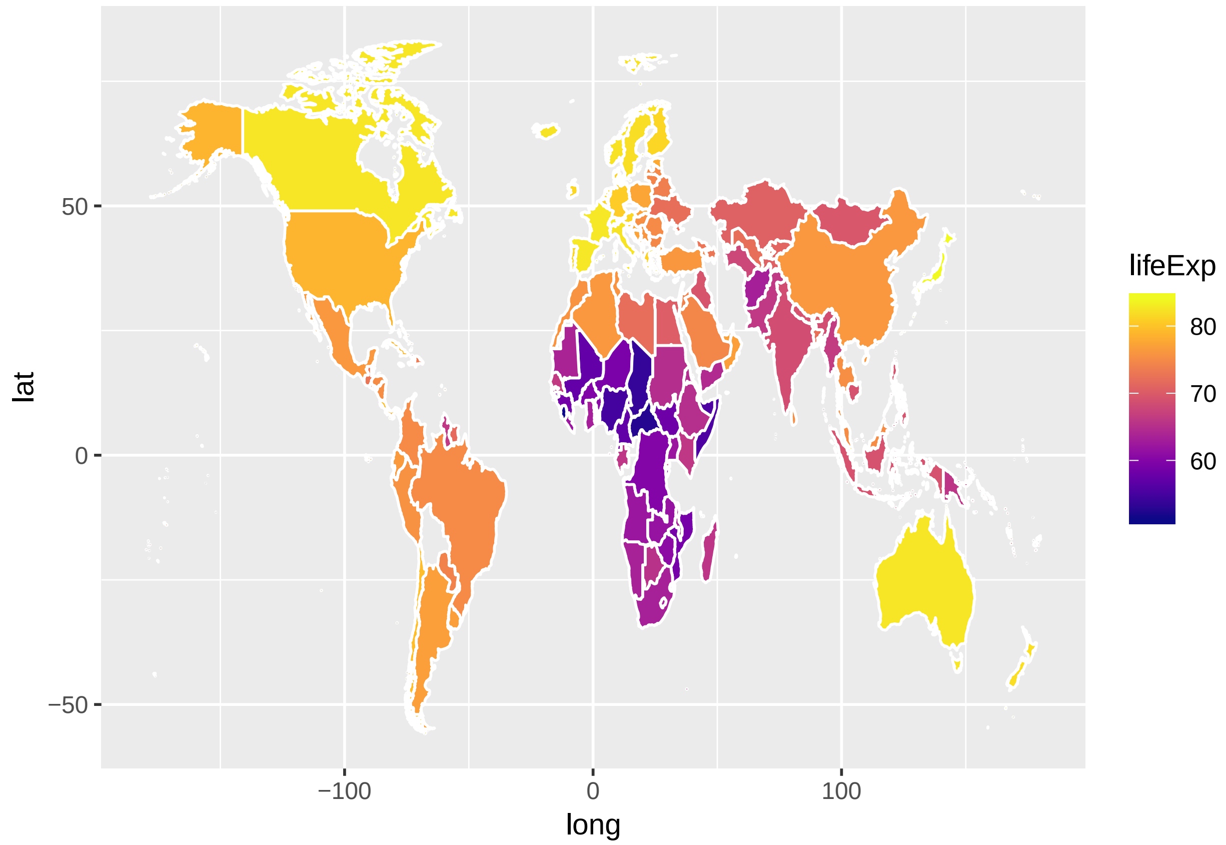
The viridis package presents various color scales for maps. From the these color scales ‘viridis’, ‘magma’, ‘plasma’, and ‘inferno’are moved from Python’s plotting library matplot and ’cividis’ was developed by Jamie R, Nuñez and Sean M. Colby. As using the scale_fill_viridis() function of the viridis package, we created a world map that colors the countries according to the percentage of countries’ life expectancy at birth in 2015. We reached this data from WHO (World Health Organozation) data frame using the WHO package. We see that life expectancy at birth increases as passing from purple to yellow. For instance, we can infer from this map that life expectancy rates of neighboring countries are similar.

```r
library(WHO)
library(viridis)

life.exp <- get_data("WHOSIS_000001")
life.exp <- life.exp %>%
filter(year == 2015 & sex == "Both sexes") %>%
select(country, value) %>%
rename(region = country, lifeExp = value) %>%
mutate(region = ifelse(region == "United States of America", "USA", region) )

world_map <- map_data("world")
life.exp.map <- left_join(life.exp, world_map, by = "region")

kable(life.exp %>% arrange(region) %>% head(15))
```

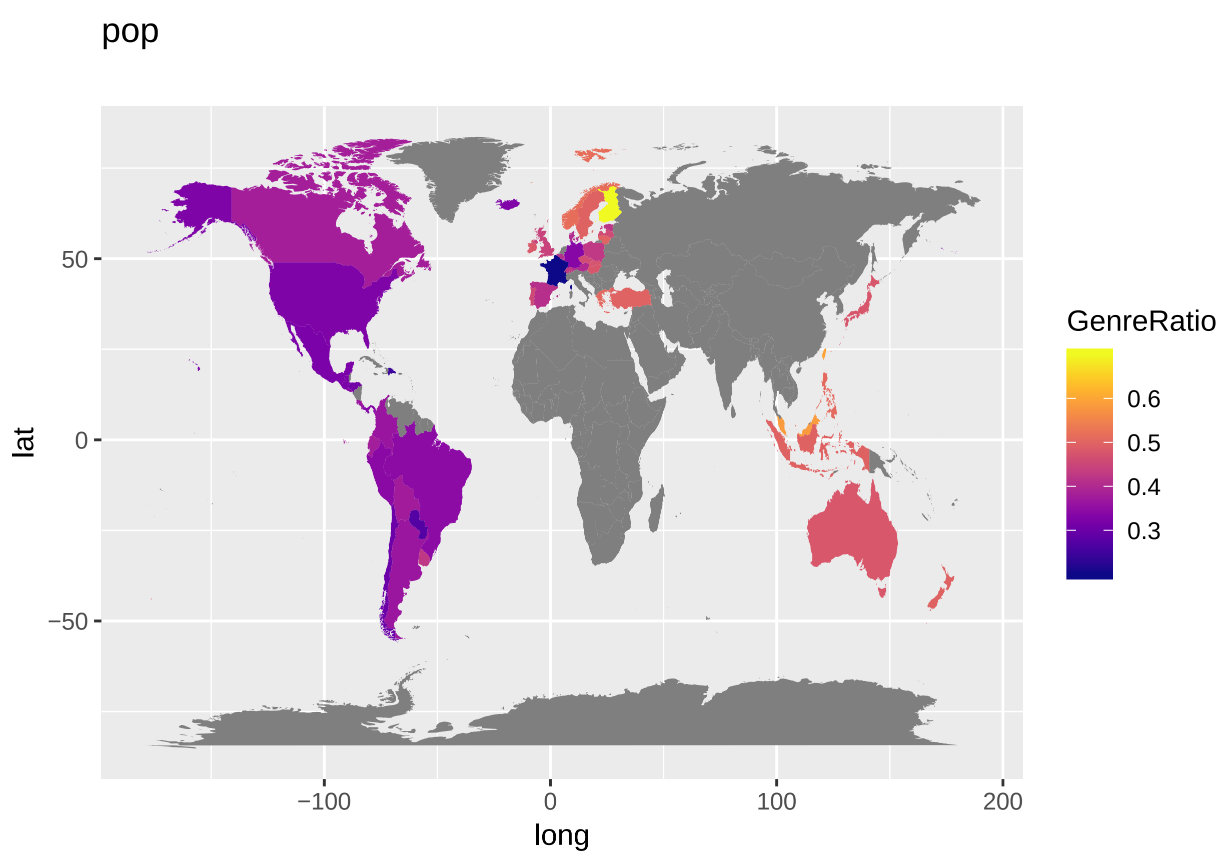
region	lifeExp
Afghanistan	63.2

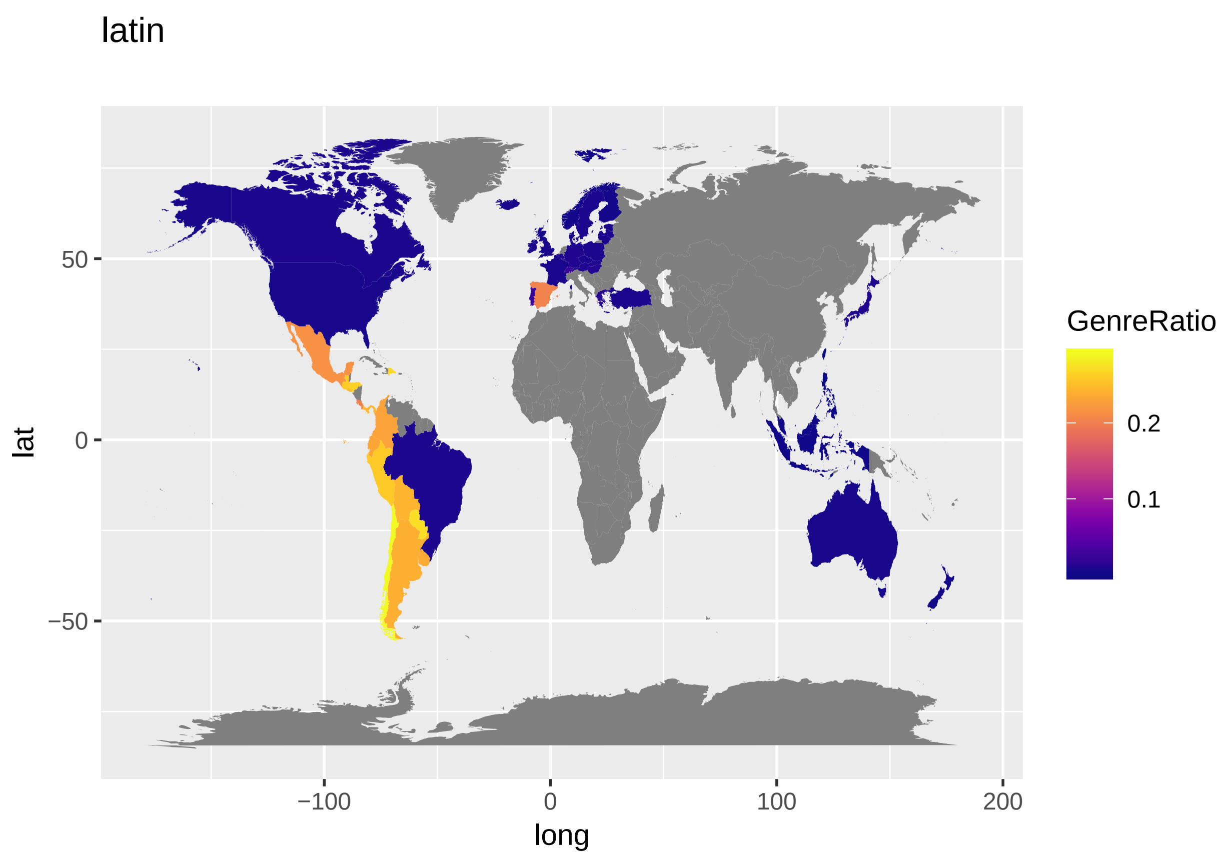
region	lifeExp
Albania	76.1
Algeria	76.2
Angola	62.2
Antigua and Barbuda	75.0
Argentina	76.8
Armenia	74.6
Australia	82.6
Austria	81.4
Azerbaijan	72.9
Bahamas	75.6
Bahrain	78.8
Bangladesh	72.2
Barbados	75.5
Belarus	73.8

```r
ggplot(life.exp.map, aes(long, lat, group = group))+
geom_polygon(aes(fill = lifeExp ), color = "white")+
scale_fill_viridis(option = "plasma")
```

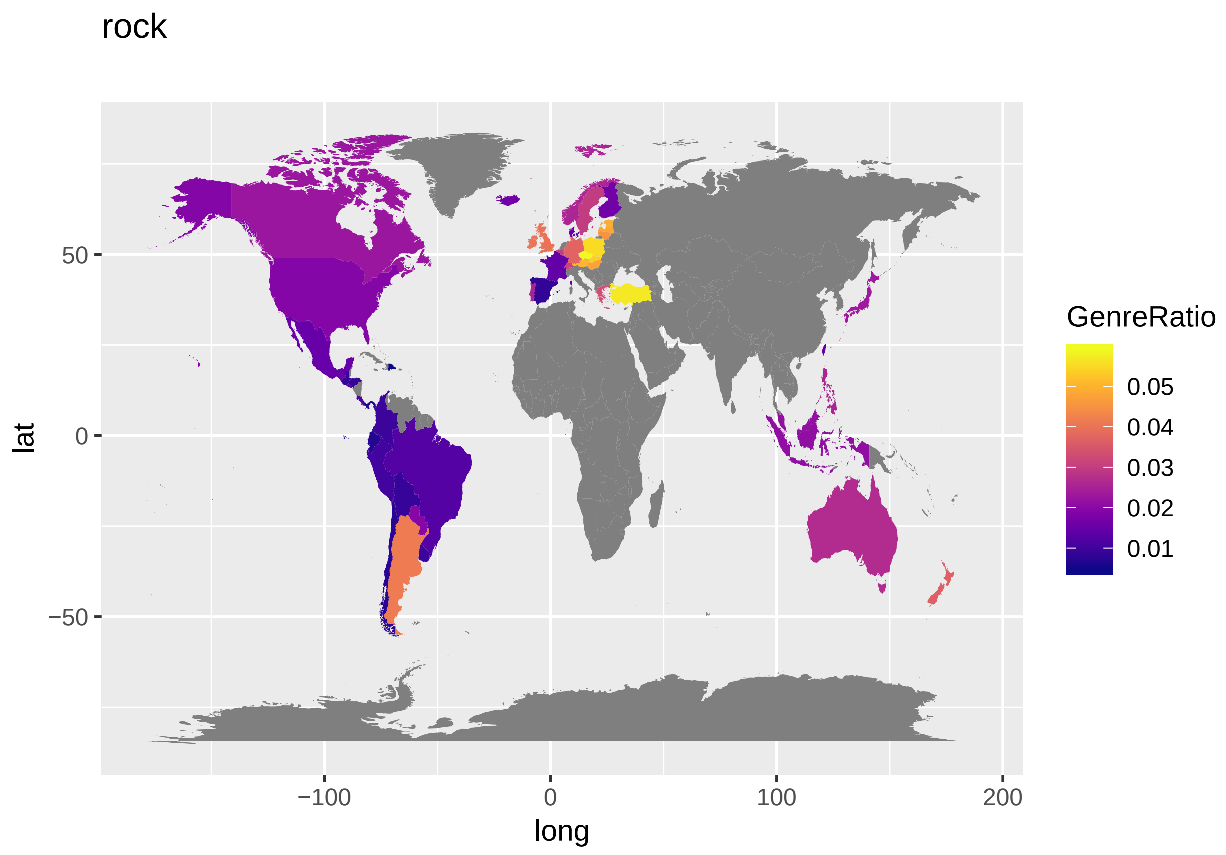


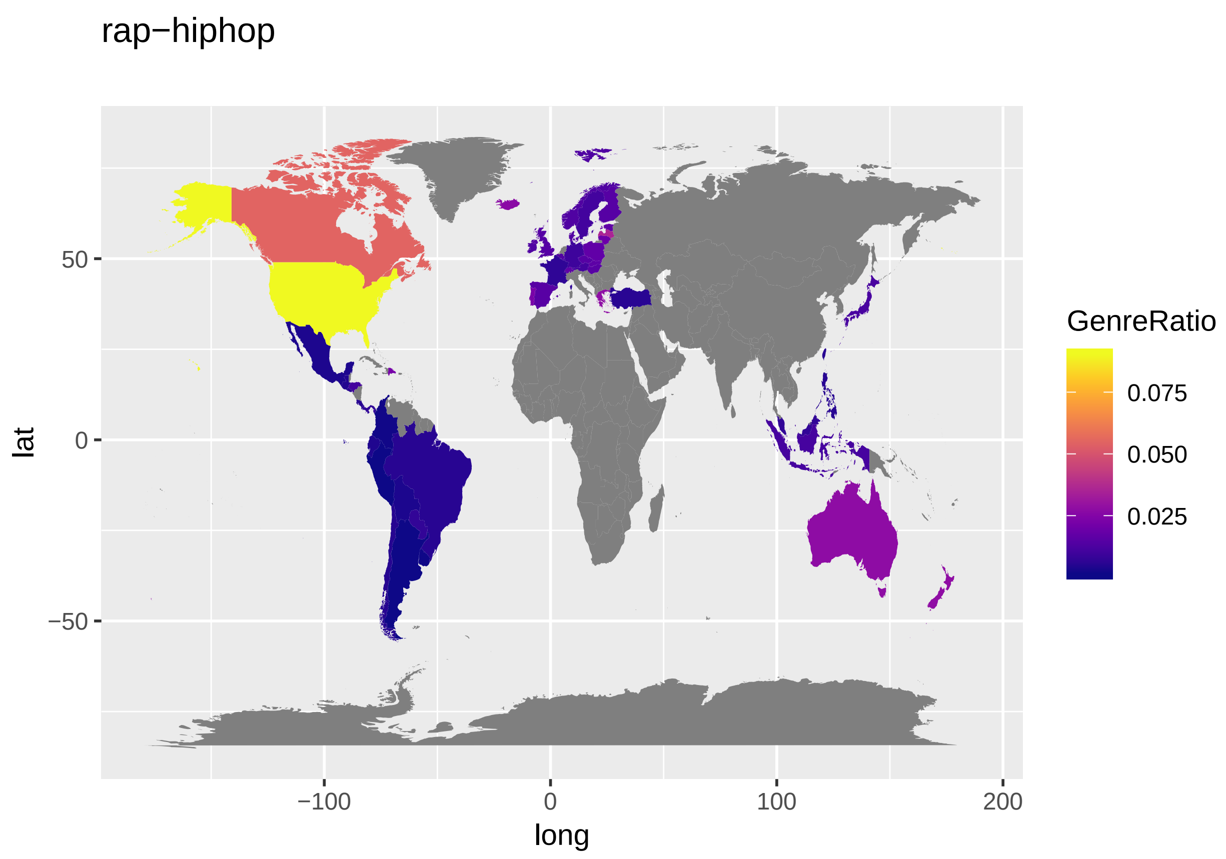
Consequently, using the methods exemplified, we generated four world maps that show the listening rates of pop, latin, rock and rap genres in the countries. we have calculated the ratios by gathering the sub-types of pop, rock, latin, rap-hiphop in these 4 types.


pop
lat
50
0
−50
−100
0
100
200
long
GenreRatio
0.6
0.5
0.4
0.3

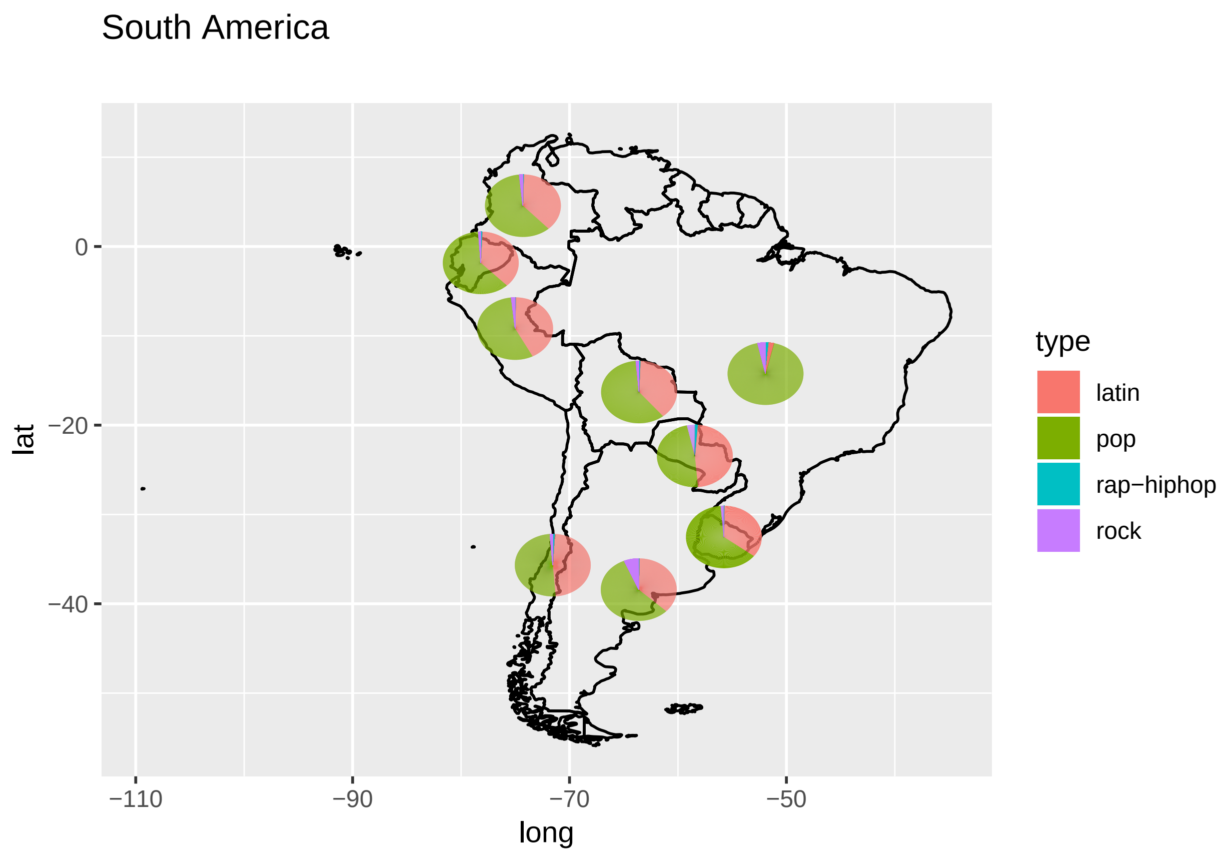


latin
GenreRatio
0.2
0.1
lat
long
50
0
−50
−100
0
100
200


rock
50
0
−50
lat
−100
0
100
200
long
GenreRatio
0.05
0.04
0.03
0.02
0.01


rap−hiphop
lat
50
0
−50
long
−100
0
100
200
GenreRatio
0.075
0.050
0.025

1.3.2 Pie Chart



1.4 Markov Chain

Using the markov model we have mentioned in the theory section, we predicted what is the most listened genre in the global for each week. First of all, for 53 weeks, we determined the most listened genre from the most listened track in the world. The popular genres in 53 weeks: pop, dance pop, latin and conscious hip hop.

To create Markov Chain transition matrix, we also created the Iterative_Genre variable that indicate the most listened genre of the previous week.

Track.Name	Region	week	Streams	Artist	Genre	Iterative_Genre
rockstar	global	1	28766763	Post Malone	pop	NA
Shape of You	global	2	53773041	Ed Sheeran	pop	pop
Shape of You	global	3	49253768	Ed Sheeran	pop	pop
Shape of You	global	4	50497666	Ed Sheeran	pop	pop
Shape of You	global	5	51854841	Ed Sheeran	pop	pop
Shape of You	global	6	52095532	Ed Sheeran	pop	pop
Shape of You	global	7	52674784	Ed Sheeran	pop	pop
Shape of You	global	8	44048323	Ed Sheeran	pop	pop
Shape of You	global	9	54537516	Ed Sheeran	pop	pop
Shape of You	global	10	52303245	Ed Sheeran	pop	pop
Shape of You	global	11	54248026	Ed Sheeran	pop	pop
Shape of You	global	12	49743302	Ed Sheeran	pop	pop

Track.Name	Region	week	Streams	Artist	Genre	Iterative_Genre
Shape of You	global	13	46810116	Ed Sheeran	pop	pop
Shape of You	global	14	43528509	Ed Sheeran	pop	pop
Shape of You	global	15	40543473	Ed Sheeran	pop	pop
HUMBLE.	global	16	38910624	Kendrick Lamar	conscious hip hop	pop
Despacito - Remix	global	17	43255538	Luis Fonsi	dance pop	conscious hip hop
Despacito - Remix	global	18	47659504	Luis Fonsi	dance pop	dance pop
Despacito - Remix	global	19	49525027	Luis Fonsi	dance pop	dance pop
Despacito - Remix	global	20	52084161	Luis Fonsi	dance pop	dance pop

In R, table() function build a contingency table of the counts at each combination of factor levels by using the cross-classifying factors. To compute the transition counts of genres in 53 weeks , we used the table function that produces below table.

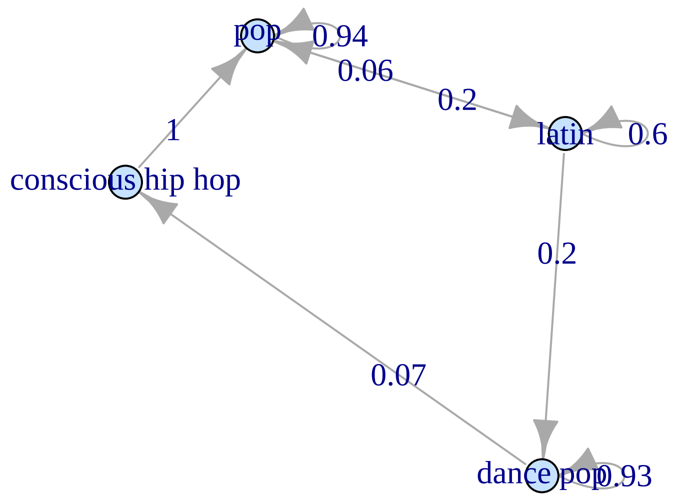
	conscious hip hop	dance pop	latin	pop
conscious hip hop	0	0	0	1
dance pop	1	13	0	0
latin	0	1	3	1
pop	0	0	2	31

After then, we created the transition ratios matrix by dividing the values in the resulting table by the sum of rows. Transitio ratios table describing the probabilities of moving from first week to last week. The entries in the first row of the table represent the probabilities for the various kinds of genres following a conscious hip hop. Similarly, the entries in the second, third and last rows represent the probabilities for the various kinds of genres following dance pop, latin and pop genres, respectively.

	conscious hip hop	dance pop	latin	pop
conscious hip hop	0.0000000	0.0000000	0.0000000	1.0000000
dance pop	0.0714286	0.9285714	0.0000000	0.0000000
latin	0.0000000	0.2000000	0.6000000	0.2000000
pop	0.0000000	0.0000000	0.0606061	0.9393939

We took advantage of R' markovchain package that provides classes, methods and function for handling Discrete Time Markov Chains (DTMC), performing probabilistic analysis and fitting. The markovchain objects can be created with new() function, as the following code shows. One of the R's core functions new() returns a newly allocated object from the class identified by the first argument. This function checks that the matrix to be a transition matrix, i.e., all entries to be probabilities and either all rows or all columns to sum up to one and the columns and rows names of transition matrix to be defined and to coincide with states vector slot. https://cran.r-project.org/web/packages/markovchain/vignettes/an_introduction_to_markovchain_package.pdf

```r
library(markovchain)

dtmcA <- new("markovchain",transitionMatrix=matrix,
states=,
name="MarkovChain")

plot(dtmcA)
```



After finding the transition probabilities of genres, we estimated the week's genres by using sample() function of R base that takes a sample of the specified size from the elements of x using either with or without replacement. The prob argument of the function weighs the obtaining the elements of the vector being sampled according to the given probabilities. Finally, we have found the accuracy value of the experiment by dividing the correct estimates to the all.

week	Genre	PredictedGenre
1	pop	pop
2	pop	pop
3	pop	pop
4	pop	pop
5	pop	latin
6	pop	pop
7	pop	pop
8	pop	pop
9	pop	pop
10	pop	pop
11	pop	pop
12	pop	pop
13	pop	pop
14	pop	pop
15	pop	pop
16	conscious hip hop	pop
17	dance pop	dance pop
18	dance pop	dance pop
19	dance pop	dance pop

week	Genre	PredictedGenre
20	dance pop	dance pop

```
## [1] "Accuracy: 0.849056603773585"
```