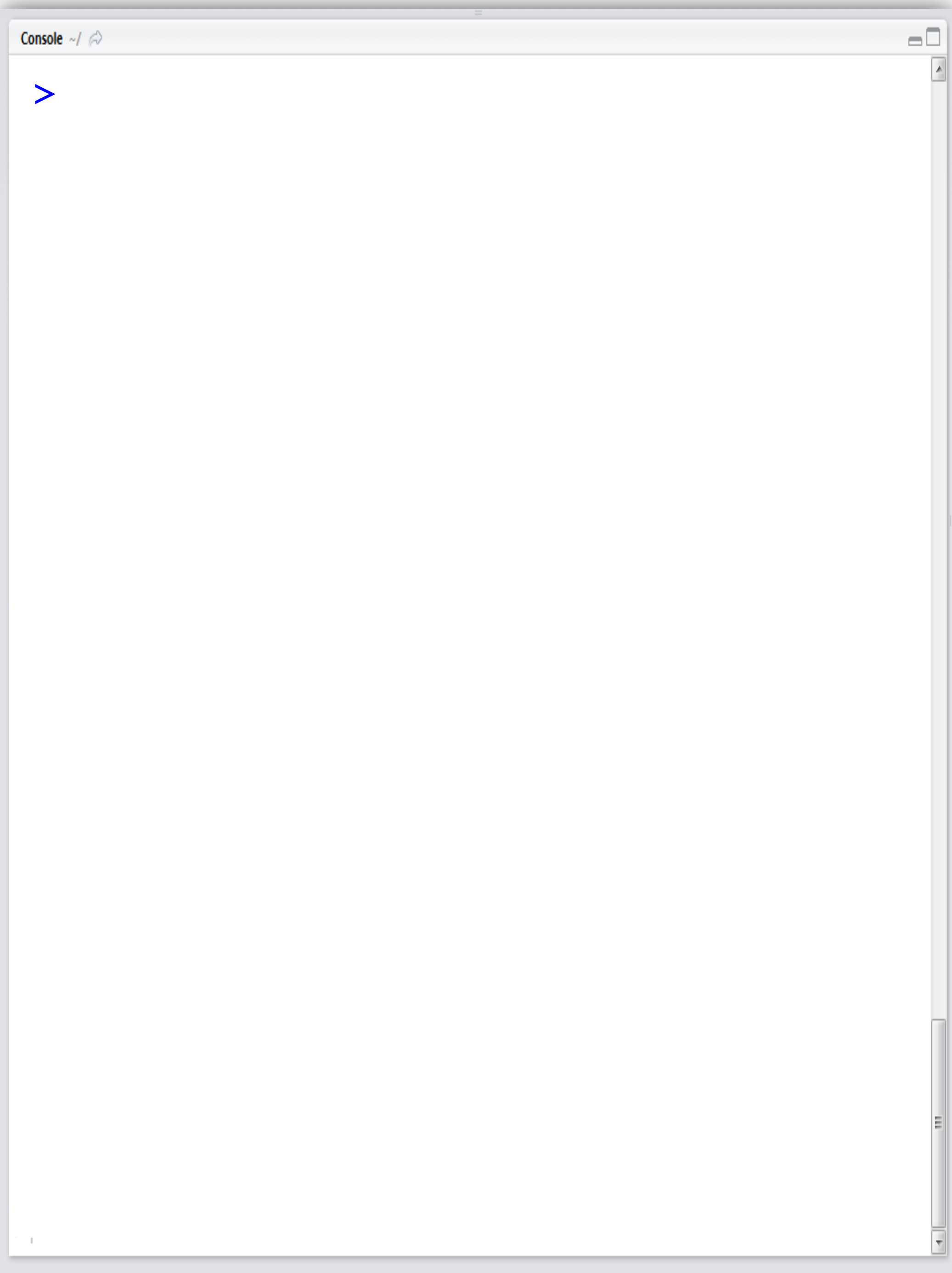
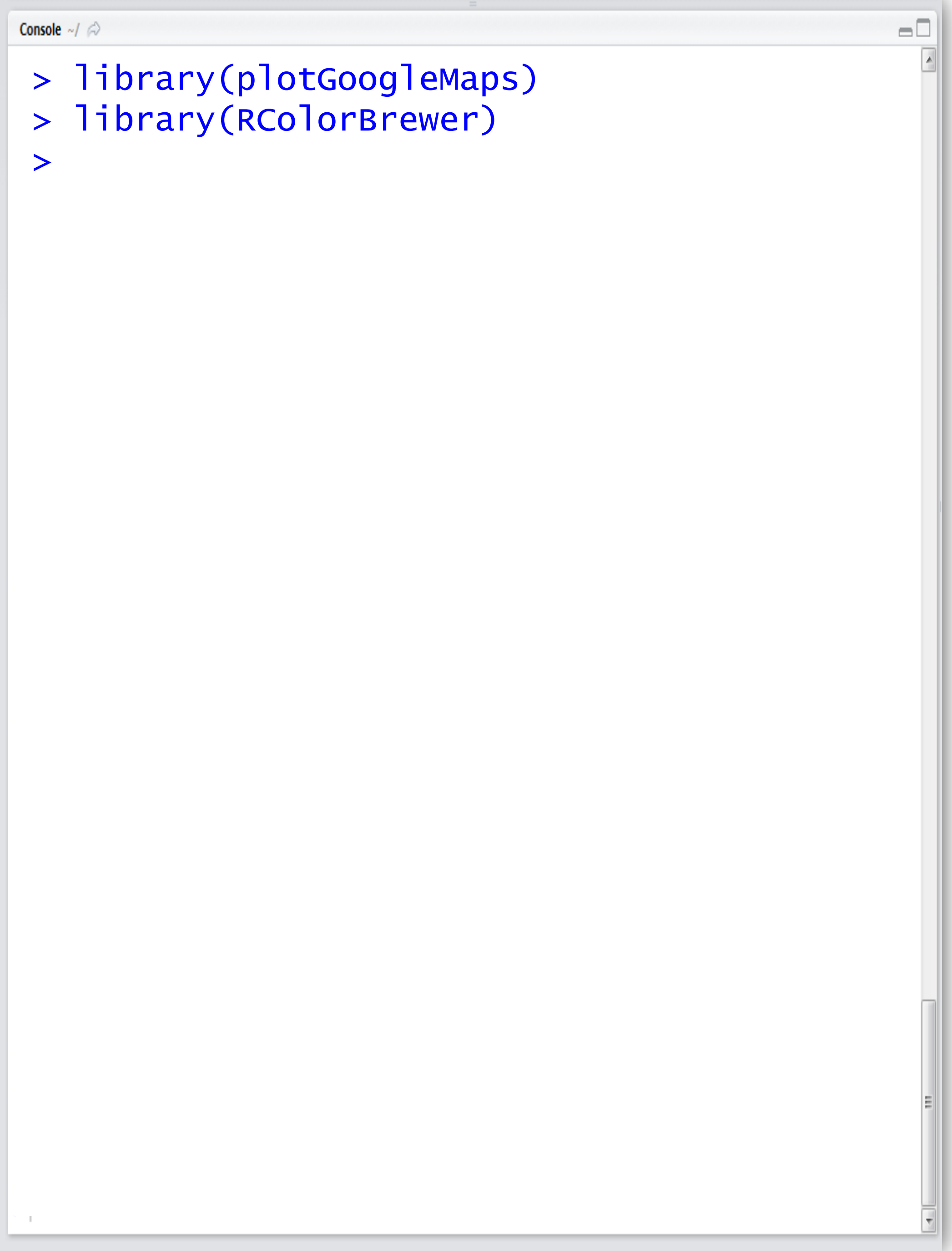
Interactive Mapping in R

Luba Gloukhov

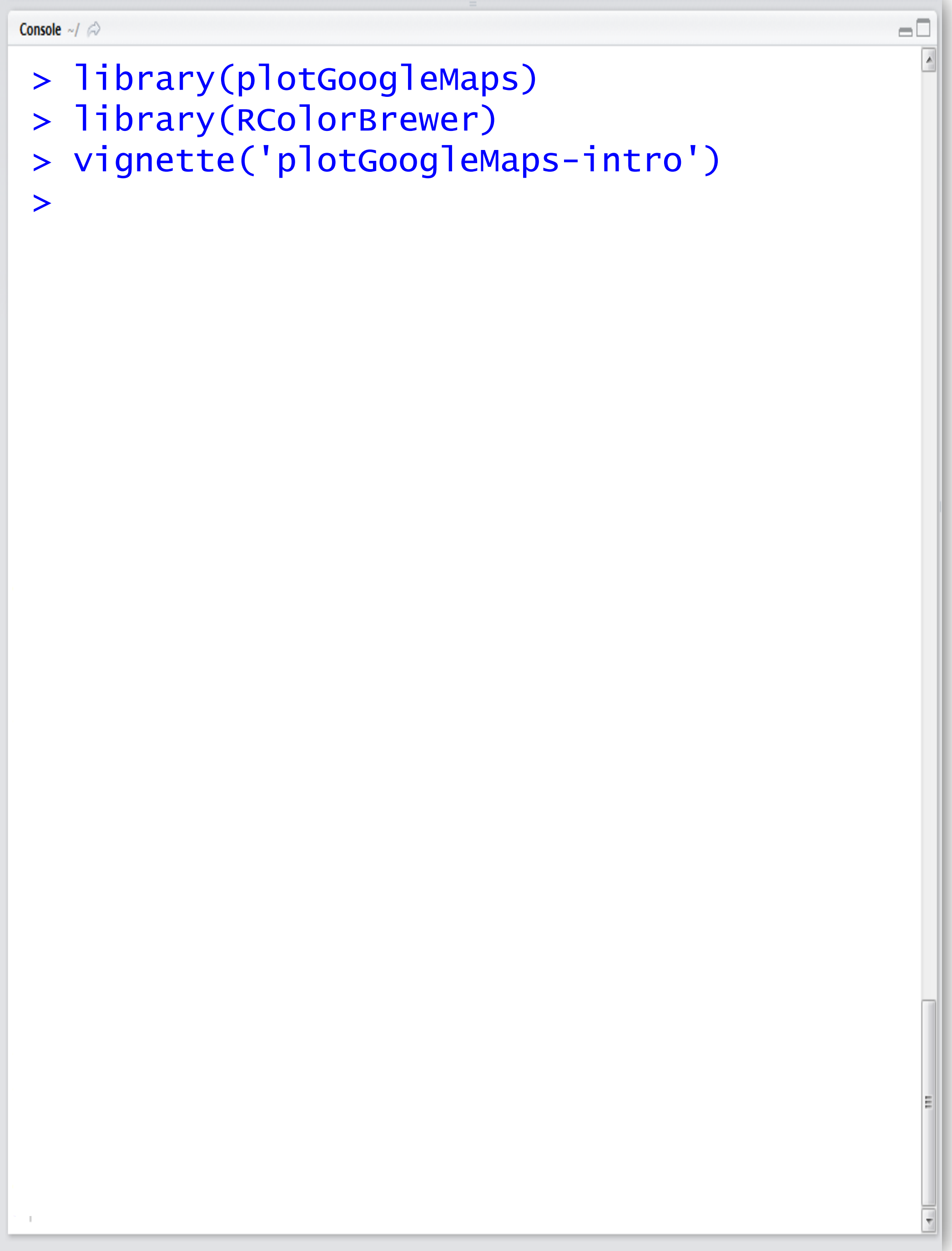


Console ~/

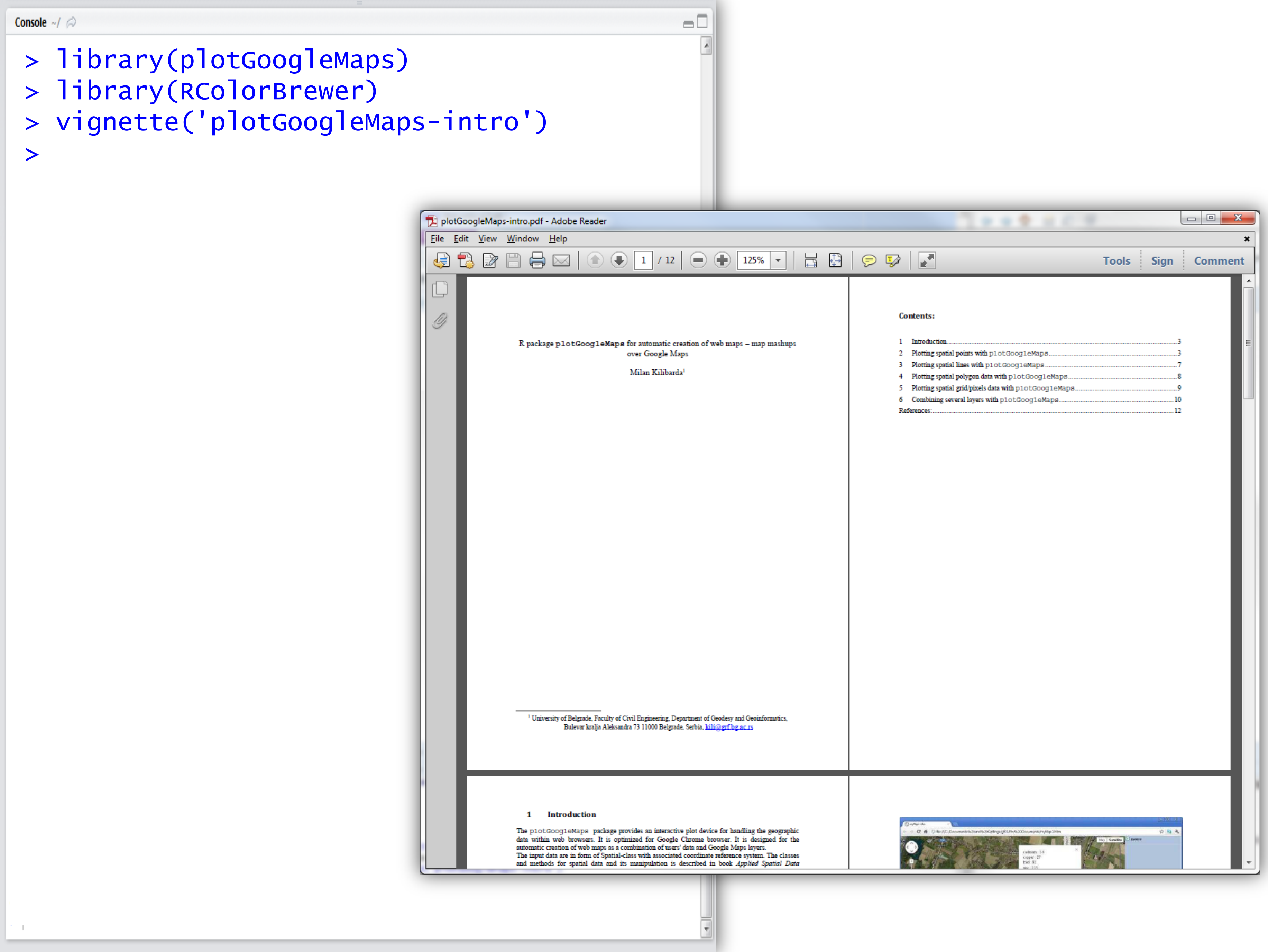
```
>
```



```
> library(plotGoogleMaps)
> library(RColorBrewer)
>
```



```
> library(plotGoogleMaps)
> library(RColorBrewer)
> vignette('plotGoogleMaps-intro')
>
```



Console ~/
plotGoogleMaps-intro.pdf - Adobe Reader
File Edit View Window Help
1 / 12
125%
Tools
Sign
Comment

```
> library(plotGoogleMaps)
> library(RColorBrewer)
> vignette('plotGoogleMaps-intro')
>
```

R package plotGoogleMaps for automatic creation of web maps – map mashups over Google Maps

Milan Kilibarda[1]

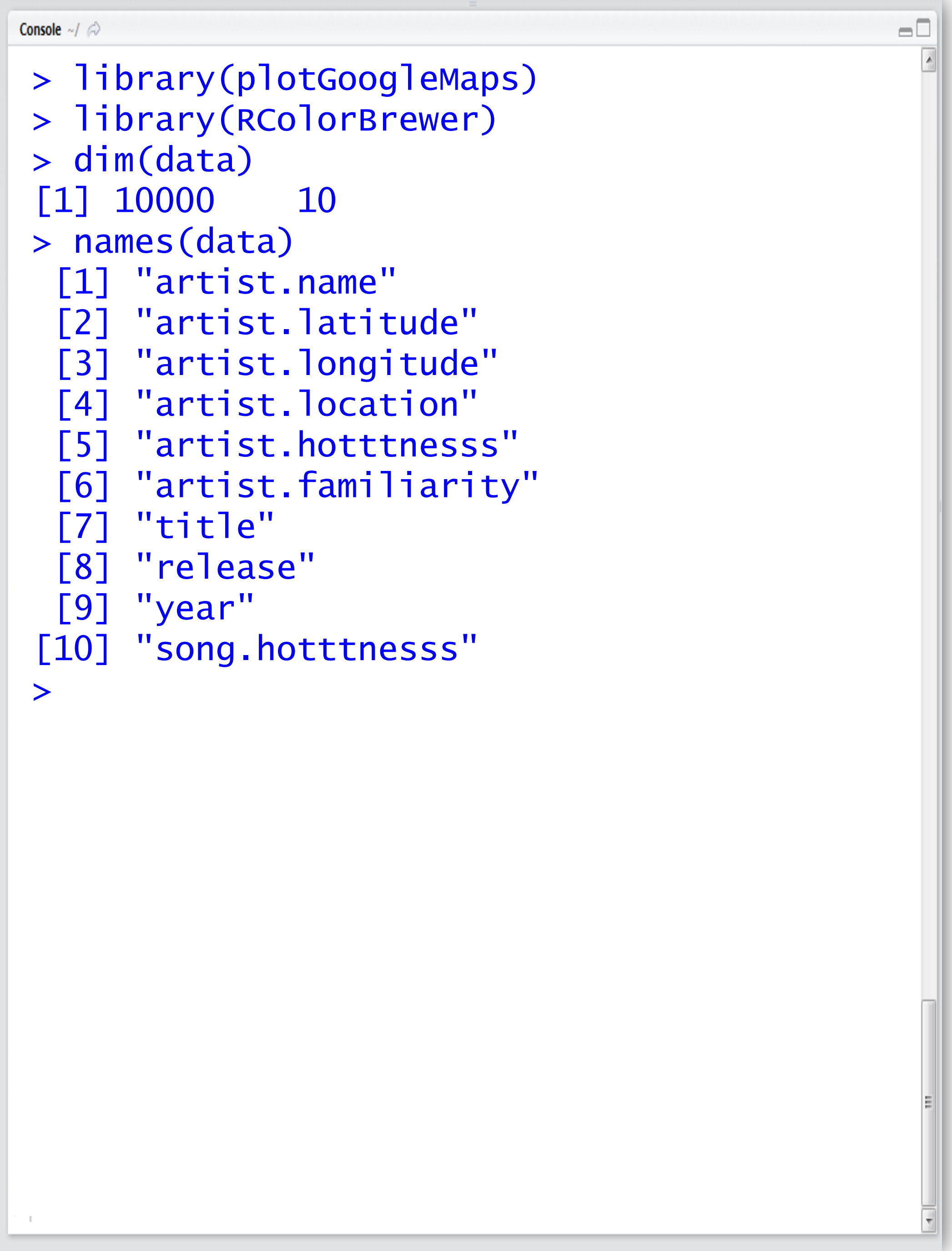
[1] University of Belgrade, Faculty of Civil Engineering, Department of Geodesy and Geoinformatics, Bulevar kralja Aleksandra 73 11000 Belgrade, Serbia, kili@grf.bg.ac.rs

Contents:

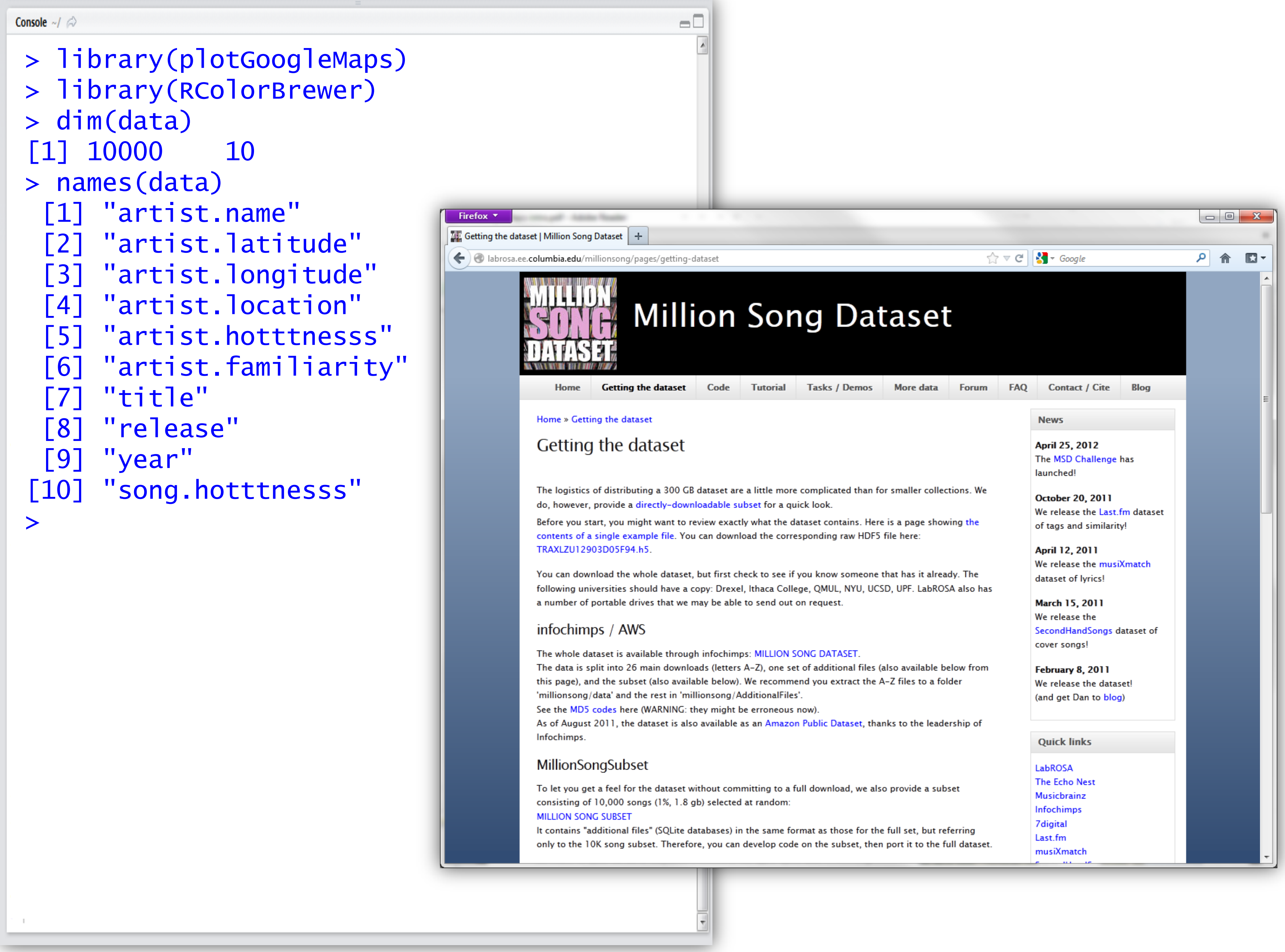
1 Introduction

The plotGoogleMaps package provides an interactive plot device for handling the geographic data within web browsers. It is optimized for Google Chrome browser. It is designed for the automatic creation of web maps as a combination of users' data and Google Maps layers.

The input data are in form of Spatial-class with associated coordinate reference system. The classes and methods for spatial data and its manipulation is described in book *Applied Spatial Data*



```
> library(plotGoogleMaps)
> library(RColorBrewer)
> dim(data)
[1] 10000    10
> names(data)
 [1] "artist.name"
 [2] "artist.latitude"
 [3] "artist.longitude"
 [4] "artist.location"
 [5] "artist.hotttnesss"
 [6] "artist.familiarity"
 [7] "title"
 [8] "release"
 [9] "year"
[10] "song.hotttnesss"
>
```



```
> library(plotGoogleMaps)
> library(RColorBrewer)
> dim(data)
[1] 10000    10
> names(data)
 [1] "artist.name"
 [2] "artist.latitude"
 [3] "artist.longitude"
 [4] "artist.location"
 [5] "artist.hotttnesss"
 [6] "artist.familiarity"
 [7] "title"
 [8] "release"
 [9] "year"
[10] "song.hotttnesss"
>
```

Million Song Dataset

Home » Getting the dataset

Getting the dataset

The logistics of distributing a 300 GB dataset are a little more complicated than for smaller collections. We do, however, provide a directly-downloadable subset for a quick look.

Before you start, you might want to review exactly what the dataset contains. Here is a page showing the contents of a single example file. You can download the corresponding raw HDF5 file here: TRAXLZU12903D05F94.h5.

You can download the whole dataset, but first check to see if you know someone that has it already. The following universities should have a copy: Drexel, Ithaca College, QMUL, NYU, UCSD, UPF. LabROSA also has a number of portable drives that we may be able to send out on request.

infochimps / AWS

The whole dataset is available through infochimps: MILLION SONG DATASET.
The data is split into 26 main downloads (letters A-Z), one set of additional files (also available below from this page), and the subset (also available below). We recommend you extract the A-Z files to a folder 'millionsong/data' and the rest in 'millionsong/AdditionalFiles'.
See the MD5 codes here (WARNING: they might be erroneous now).
As of August 2011, the dataset is also available as an Amazon Public Dataset, thanks to the leadership of Infochimps.

MillionSongSubset

To let you get a feel for the dataset without committing to a full download, we also provide a subset consisting of 10,000 songs (1%, 1.8 gb) selected at random:
MILLION SONG SUBSET
It contains "additional files" (SQLite databases) in the same format as those for the full set, but referring only to the 10K song subset. Therefore, you can develop code on the subset, then port it to the full dataset.

News

April 25, 2012
The MSD Challenge has launched!

October 20, 2011
We release the Last.fm dataset of tags and similarity!

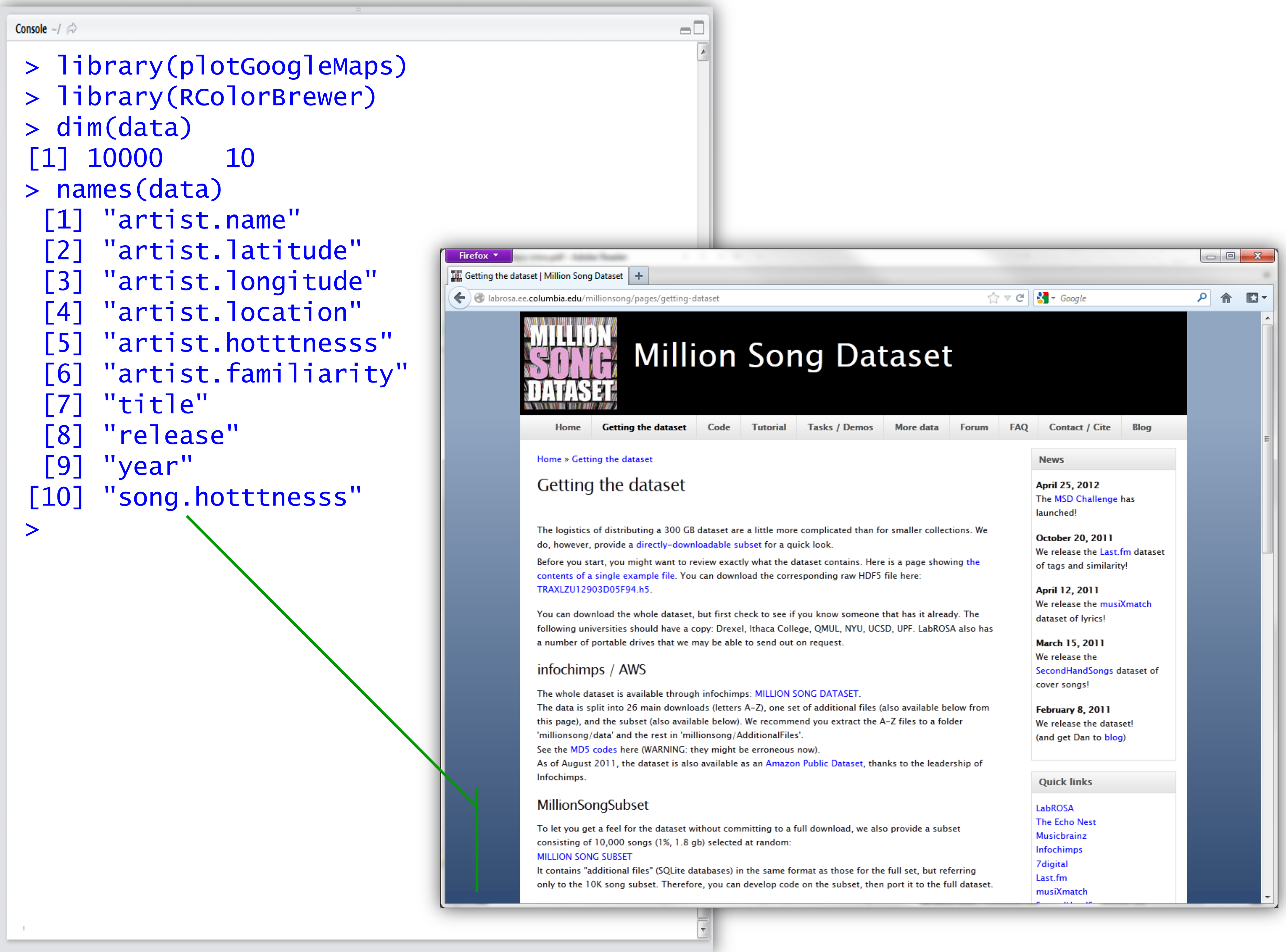
April 12, 2011
We release the musiXmatch dataset of lyrics!

March 15, 2011
We release the SecondHandSongs dataset of cover songs!

February 8, 2011
We release the dataset!
(and get Dan to blog)

Quick links

LabROSA
The Echo Nest
Musicbrainz
Infochimps
7digital
Last.fm
musiXmatch



```
> library(plotGoogleMaps)
> library(RColorBrewer)
> dim(data)
[1] 10000    10
> names(data)
 [1] "artist.name"
 [2] "artist.latitude"
 [3] "artist.longitude"
 [4] "artist.location"
 [5] "artist.hotttnesss"
 [6] "artist.familiarity"
 [7] "title"
 [8] "release"
 [9] "year"
[10] "song.hotttnesss"
>
```

Million Song Dataset

Home » Getting the dataset

Getting the dataset

The logistics of distributing a 300 GB dataset are a little more complicated than for smaller collections. We do, however, provide a directly-downloadable subset for a quick look.

Before you start, you might want to review exactly what the dataset contains. Here is a page showing the contents of a single example file. You can download the corresponding raw HDF5 file here: TRAXLZU12903D05F94.h5.

You can download the whole dataset, but first check to see if you know someone that has it already. The following universities should have a copy: Drexel, Ithaca College, QMUL, NYU, UCSD, UPF. LabROSA also has a number of portable drives that we may be able to send out on request.

infochimps / AWS

The whole dataset is available through infochimps: MILLION SONG DATASET.
The data is split into 26 main downloads (letters A-Z), one set of additional files (also available below from this page), and the subset (also available below). We recommend you extract the A-Z files to a folder 'millionsong/data' and the rest in 'millionsong/AdditionalFiles'.
See the MD5 codes here (WARNING: they might be erroneous now).
As of August 2011, the dataset is also available as an Amazon Public Dataset, thanks to the leadership of Infochimps.

MillionSongSubset

To let you get a feel for the dataset without committing to a full download, we also provide a subset consisting of 10,000 songs (1%, 1.8 gb) selected at random:
MILLION SONG SUBSET
It contains "additional files" (SQLite databases) in the same format as those for the full set, but referring only to the 10K song subset. Therefore, you can develop code on the subset, then port it to the full dataset.

News

April 25, 2012
The MSD Challenge has launched!

October 20, 2011
We release the Last.fm dataset of tags and similarity!

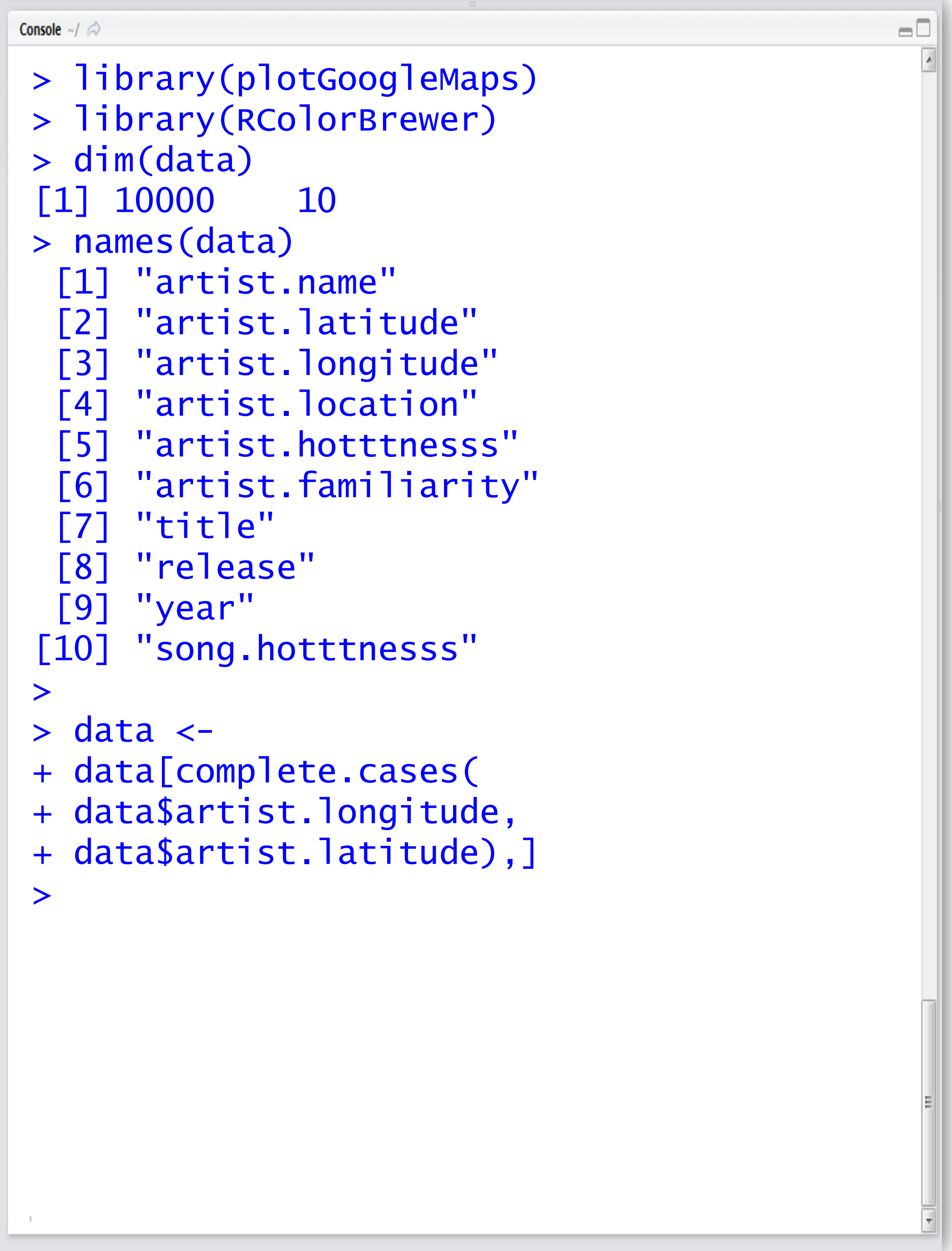
April 12, 2011
We release the musiXmatch dataset of lyrics!

March 15, 2011
We release the SecondHandSongs dataset of cover songs!

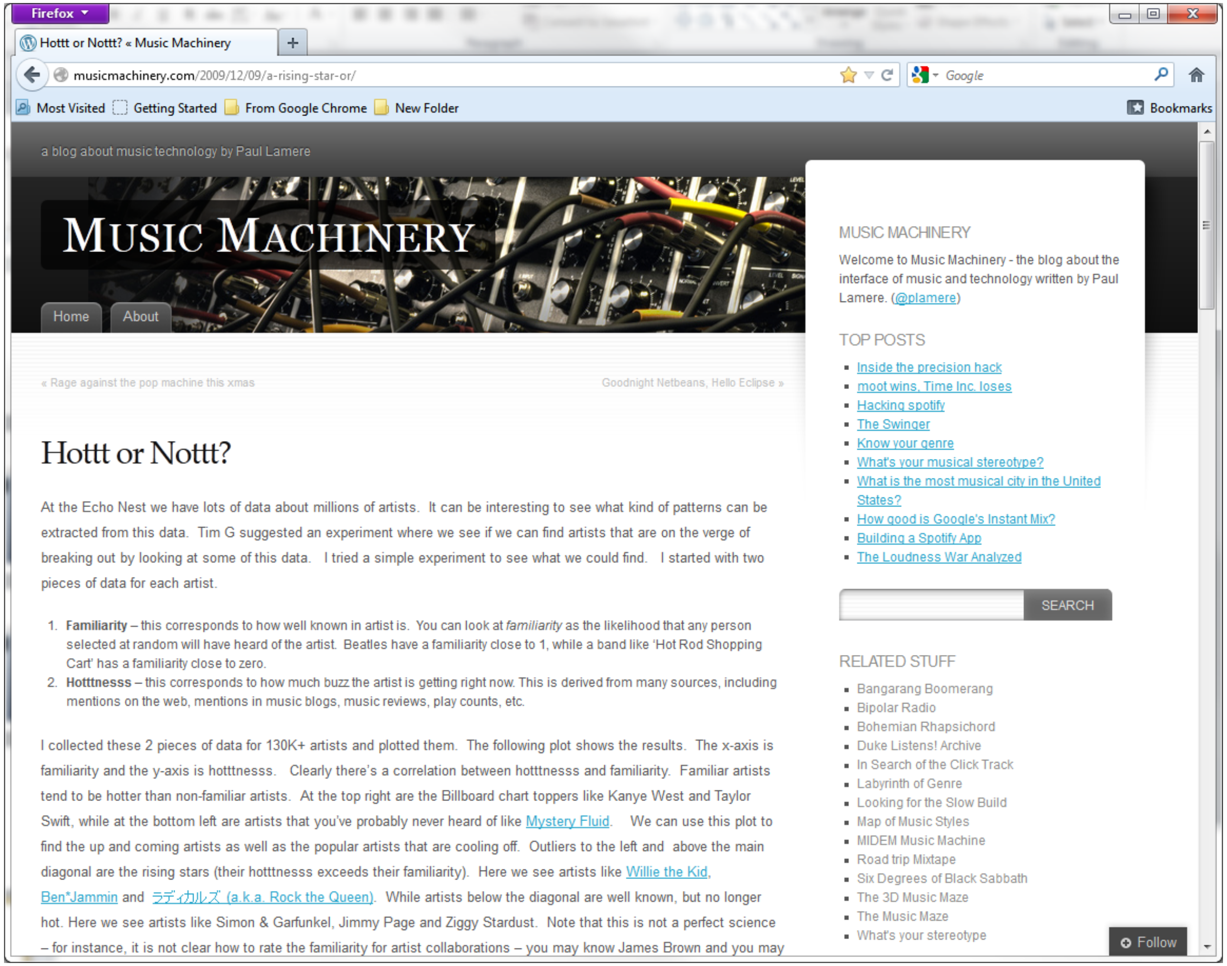
February 8, 2011
We release the dataset!
(and get Dan to blog)

Quick links

LabROSA
The Echo Nest
Musicbrainz
Infochimps
7digital
Last.fm
musiXmatch



```
> library(plotGoogleMaps)
> library(RColorBrewer)
> dim(data)
[1] 10000    10
> names(data)
 [1] "artist.name"
 [2] "artist.latitude"
 [3] "artist.longitude"
 [4] "artist.location"
 [5] "artist.hotttnesss"
 [6] "artist.familiarity"
 [7] "title"
 [8] "release"
 [9] "year"
[10] "song.hotttnesss"
>
> data <-
+ data[complete.cases(
+ data$artist.longitude,
+ data$artist.latitude),]
>
```



a blog about music technology by Paul Lamere

Music Machinery

« Rage against the pop machine this xmas

Goodnight Netbeans, Hello Eclipse »

Hottt or Nottt?

At the Echo Nest we have lots of data about millions of artists. It can be interesting to see what kind of patterns can be extracted from this data. Tim G suggested an experiment where we see if we can find artists that are on the verge of breaking out by looking at some of this data. I tried a simple experiment to see what we could find. I started with two pieces of data for each artist.

1. **Familiarity** – this corresponds to how well known in artist is. You can look at *familiarity* as the likelihood that any person selected at random will have heard of the artist. Beatles have a familiarity close to 1, while a band like 'Hot Rod Shopping Cart' has a familiarity close to zero.
2. **Hotttnesss** – this corresponds to how much buzz the artist is getting right now. This is derived from many sources, including mentions on the web, mentions in music blogs, music reviews, play counts, etc.

I collected these 2 pieces of data for 130K+ artists and plotted them. The following plot shows the results. The x-axis is familiarity and the y-axis is hotttnesss. Clearly there's a correlation between hotttnesss and familiarity. Familiar artists tend to be hotter than non-familiar artists. At the top right are the Billboard chart toppers like Kanye West and Taylor Swift, while at the bottom left are artists that you've probably never heard of like Mystery Fluid. We can use this plot to find the up and coming artists as well as the popular artists that are cooling off. Outliers to the left and above the main diagonal are the rising stars (their hotttnesss exceeds their familiarity). Here we see artists like Willie the Kid, Ben*Jammin and ラディカルズ (a.k.a. Rock the Queen). While artists below the diagonal are well known, but no longer hot. Here we see artists like Simon & Garfunkel, Jimmy Page and Ziggy Stardust. Note that this is not a perfect science – for instance, it is not clear how to rate the familiarity for artist collaborations – you may know James Brown and you may

MUSIC MACHINERY

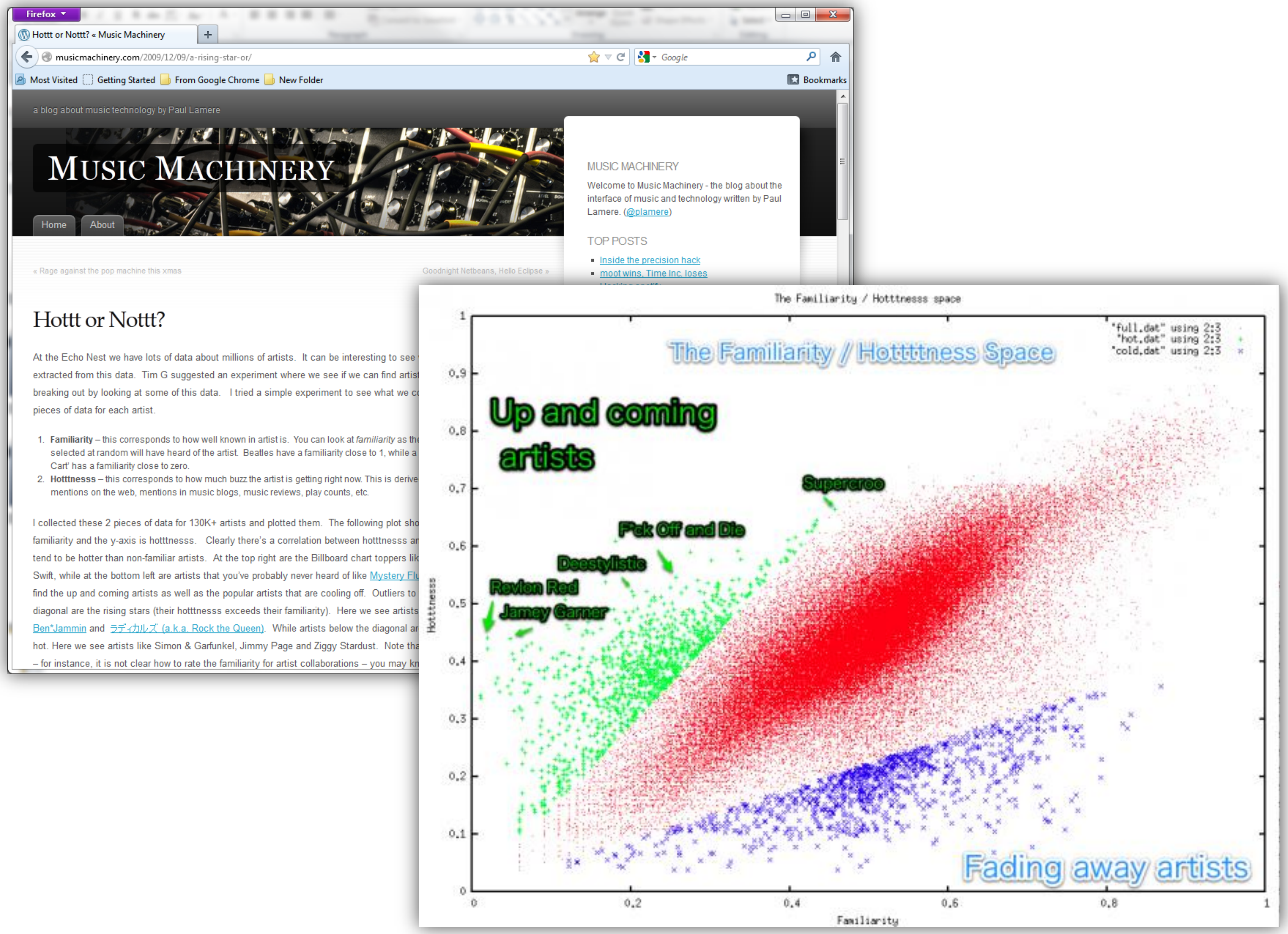
Welcome to Music Machinery - the blog about the interface of music and technology written by Paul Lamere. (@plamere)

TOP POSTS

- Inside the precision hack
- moot wins, Time Inc. loses
- Hacking spotify
- The Swinger
- Know your genre
- What's your musical stereotype?
- What is the most musical city in the United States?
- How good is Google's Instant Mix?
- Building a Spotify App
- The Loudness War Analyzed

RELATED STUFF

- Bangarang Boomerang
- Bipolar Radio
- Bohemian Rhapsichord
- Duke Listens! Archive
- In Search of the Click Track
- Labyrinth of Genre
- Looking for the Slow Build
- Map of Music Styles
- MIDEM Music Machine
- Road trip Mixtape
- Six Degrees of Black Sabbath
- The 3D Music Maze
- The Music Maze
- What's your stereotype



a blog about music technology by Paul Lamere

MUSIC MACHINERY

Home About

MUSIC MACHINERY

Welcome to Music Machinery - the blog about the interface of music and technology written by Paul Lamere. (@plamere)

TOP POSTS

- Inside the precision hack
- moot wins, Time Inc. loses

« Rage against the pop machine this xmas

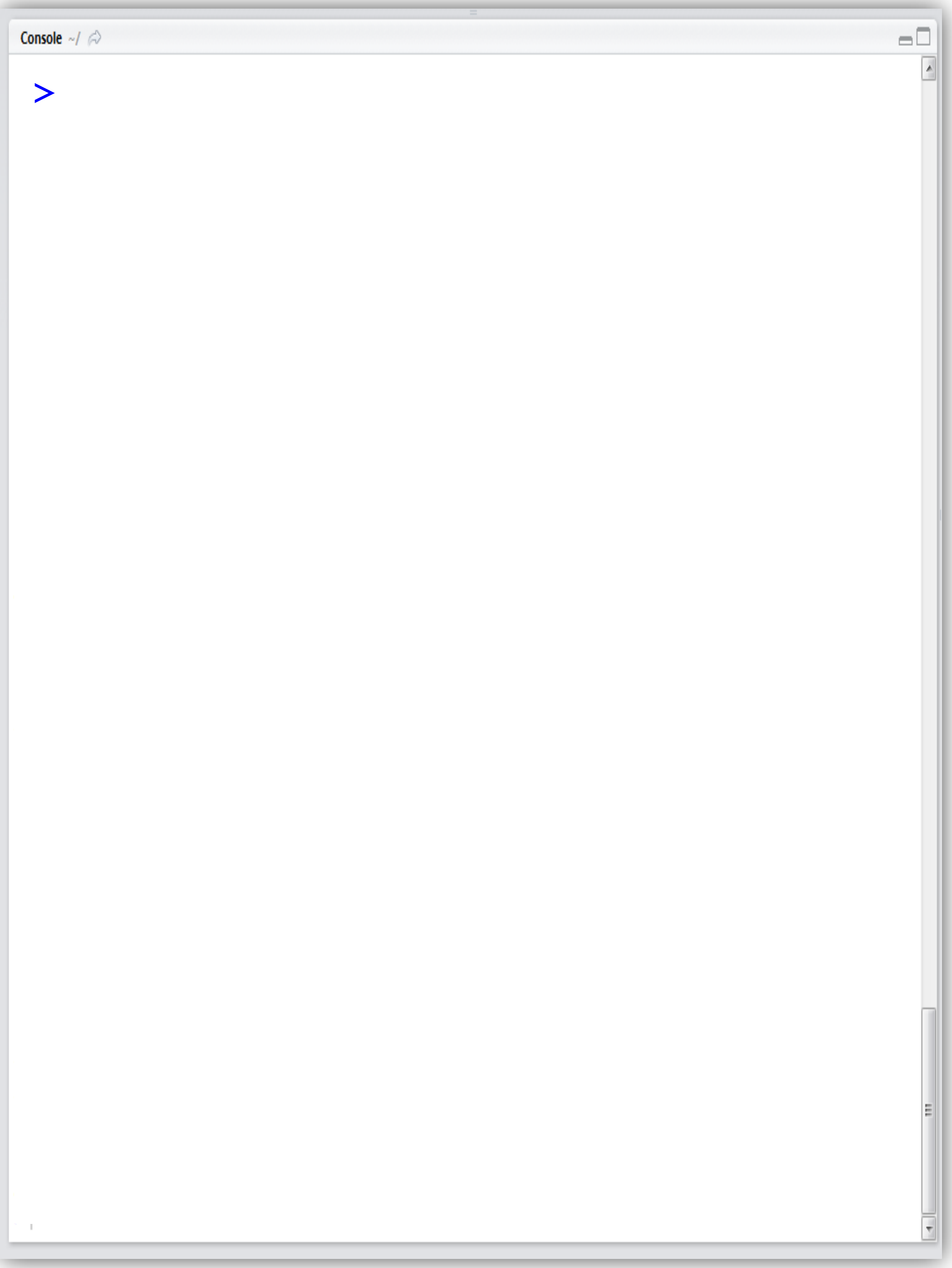
Goodnight Netbeans, Hello Eclipse »

Hottt or Nottt?

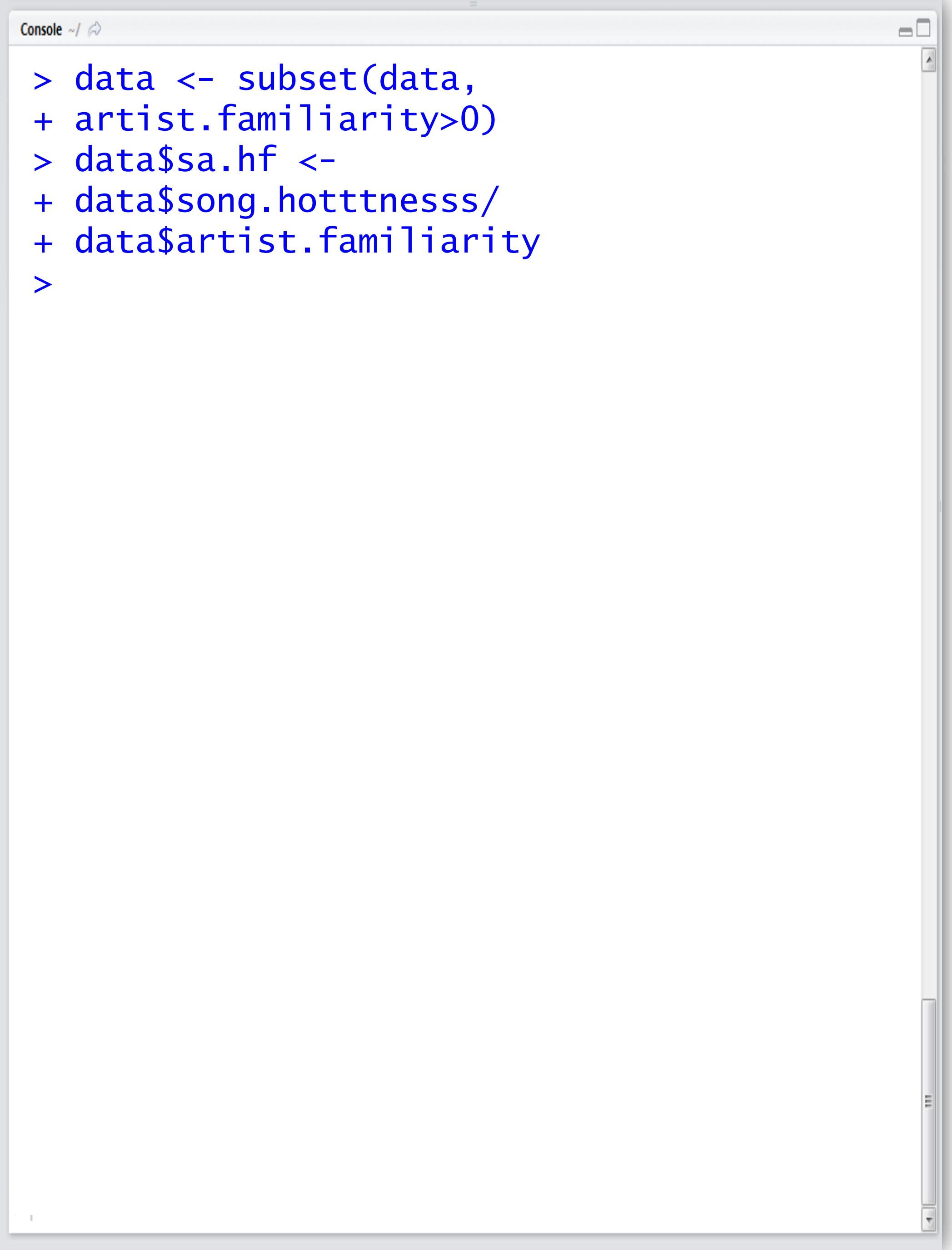
At the Echo Nest we have lots of data about millions of artists. It can be interesting to see
extracted from this data. Tim G suggested an experiment where we see if we can find artis
breaking out by looking at some of this data. I tried a simple experiment to see what we c
pieces of data for each artist.

1. **Familiarity** – this corresponds to how well known in artist is. You can look at *familiarity* as th
selected at random will have heard of the artist. Beatles have a familiarity close to 1, while a
Cart' has a familiarity close to zero.
2. **Hotttnesss** – this corresponds to how much buzz the artist is getting right now. This is derive
mentions on the web, mentions in music blogs, music reviews, play counts, etc.

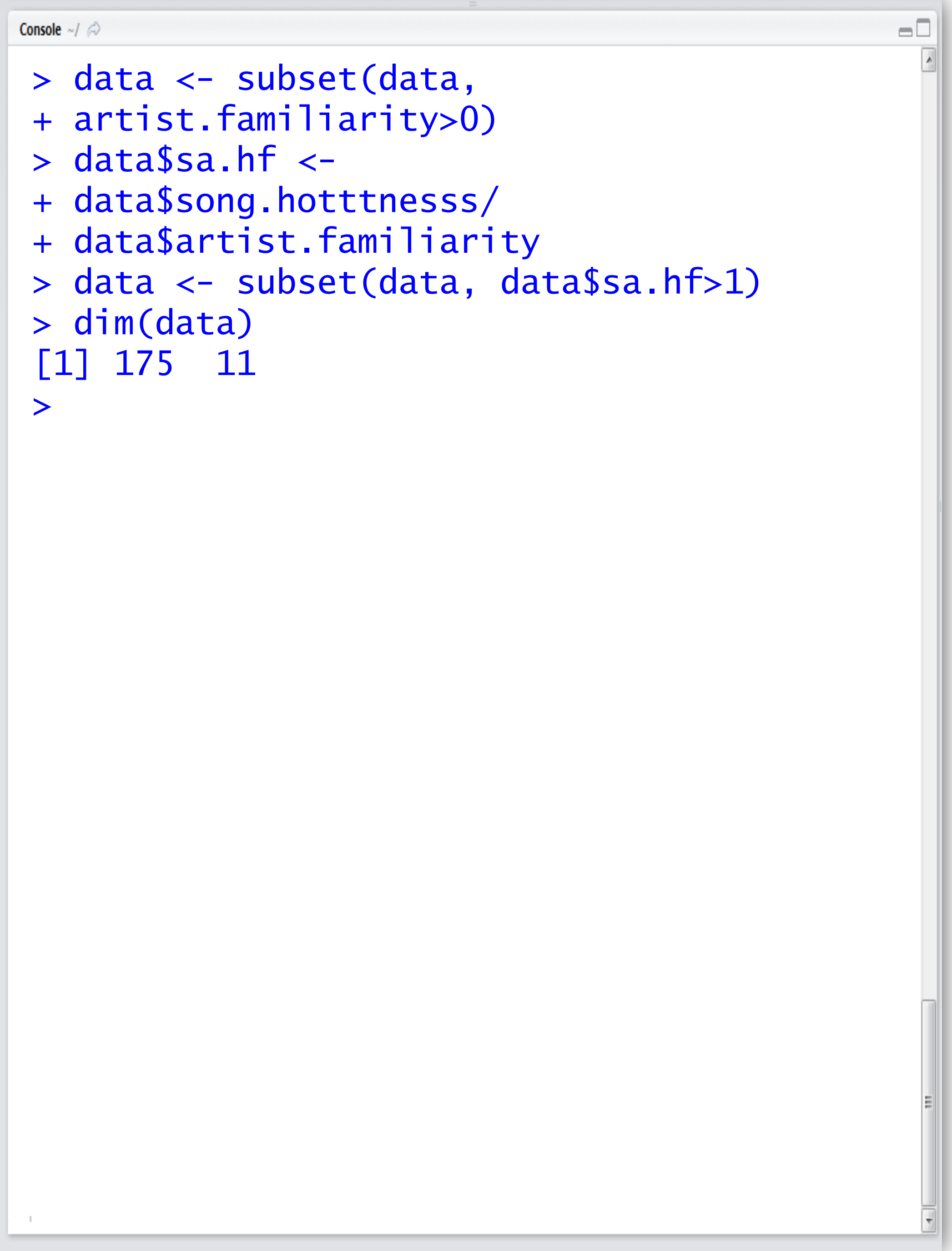
I collected these 2 pieces of data for 130K+ artists and plotted them. The following plot sho
familiarity and the y-axis is hotttnesss. Clearly there's a correlation between hotttnesss ar
tend to be hotter than non-familiar artists. At the top right are the Billboard chart toppers lik
Swift, while at the bottom left are artists that you've probably never heard of like Mystery Flu
find the up and coming artists as well as the popular artists that are cooling off. Outliers to
diagonal are the rising stars (their hotttnesss exceeds their familiarity). Here we see artists
Ben*Jammin and ラディカルズ (a.k.a. Rock the Queen). While artists below the diagonal ar
hot. Here we see artists like Simon & Garfunkel, Jimmy Page and Ziggy Stardust. Note tha
– for instance, it is not clear how to rate the familiarity for artist collaborations – you may kn


Console ~/

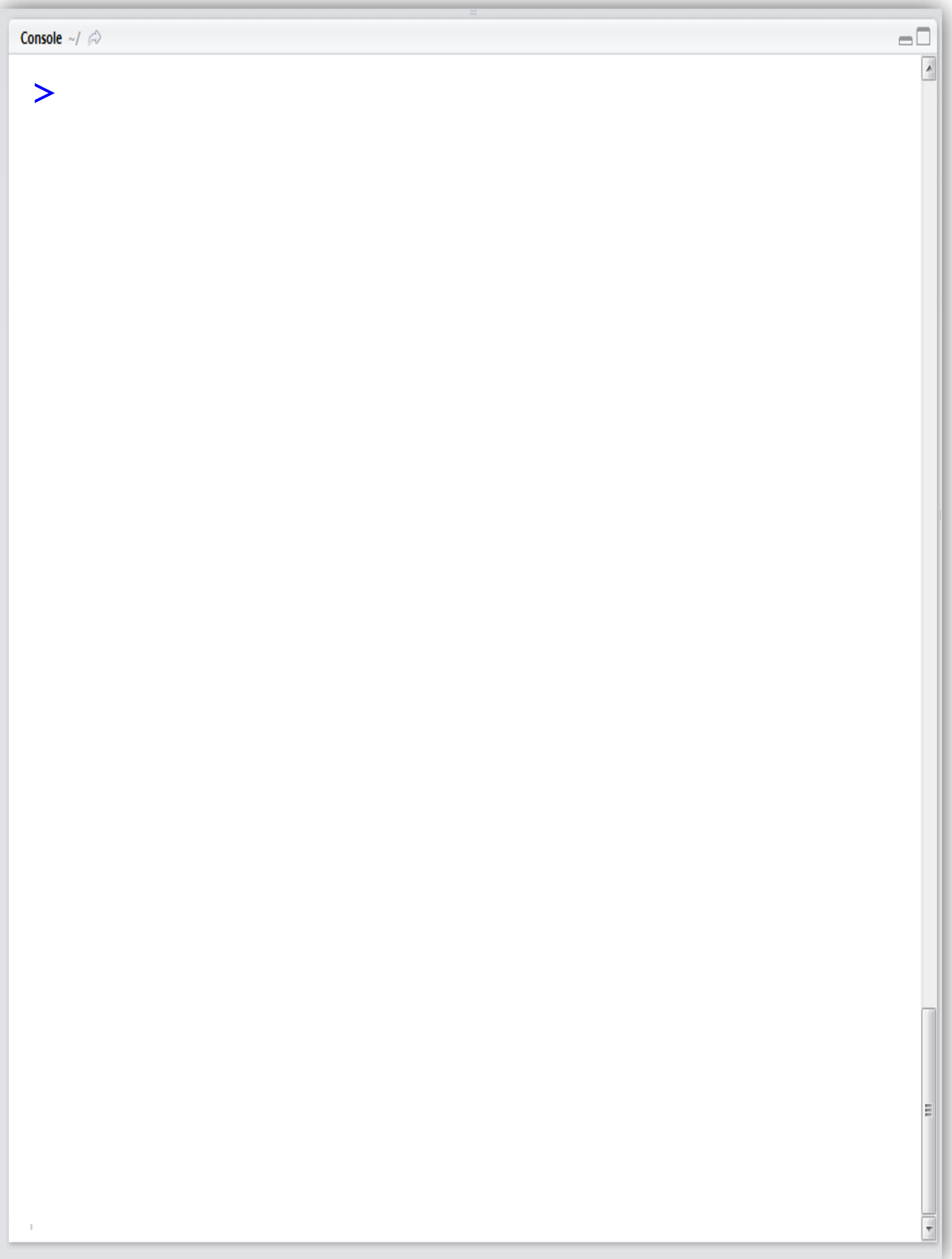
```
>
```



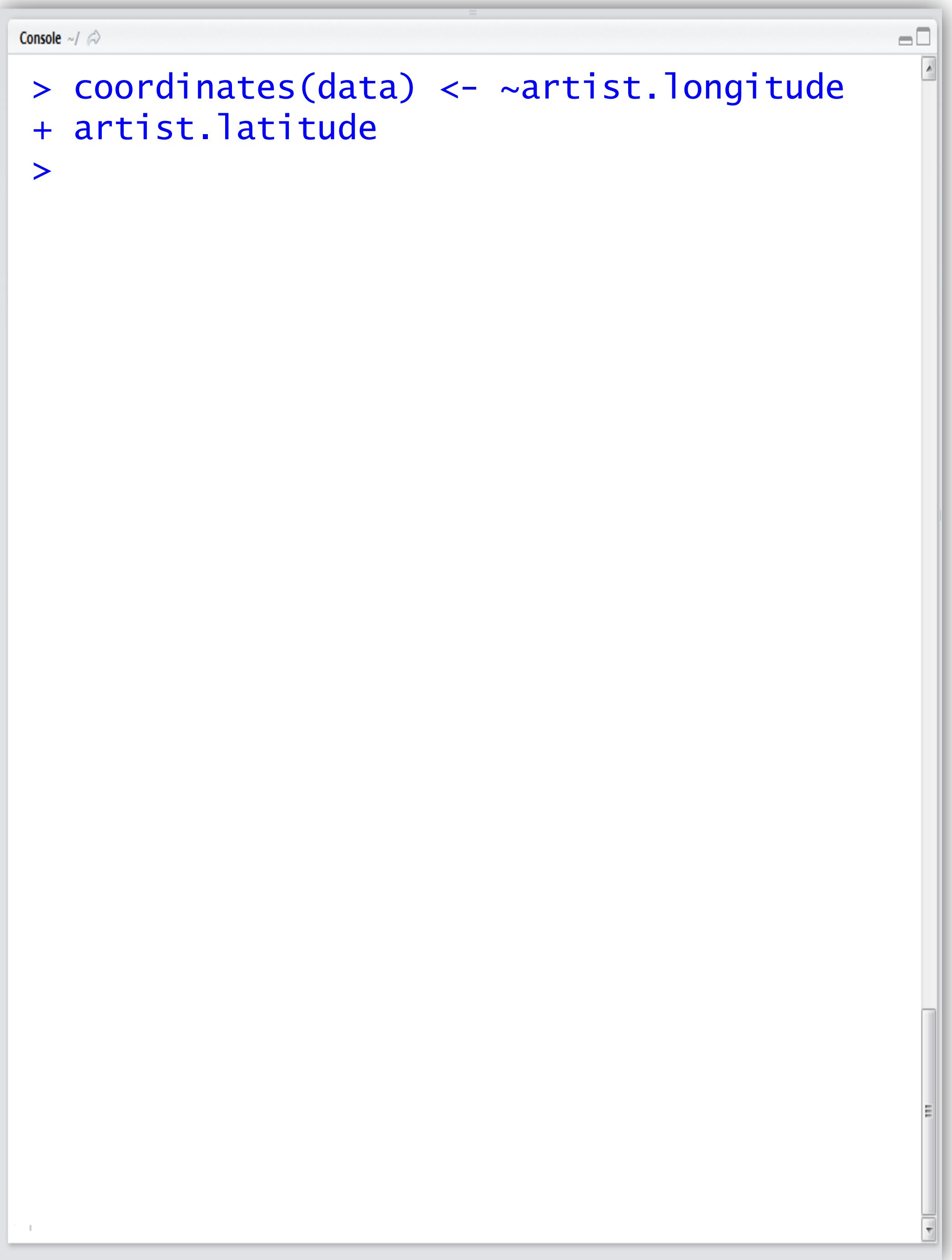
```
> data <- subset(data,
+ artist.familiarity>0)
> data$sa.hf <-
+ data$song.hotttnesss/
+ data$artist.familiarity
>
```



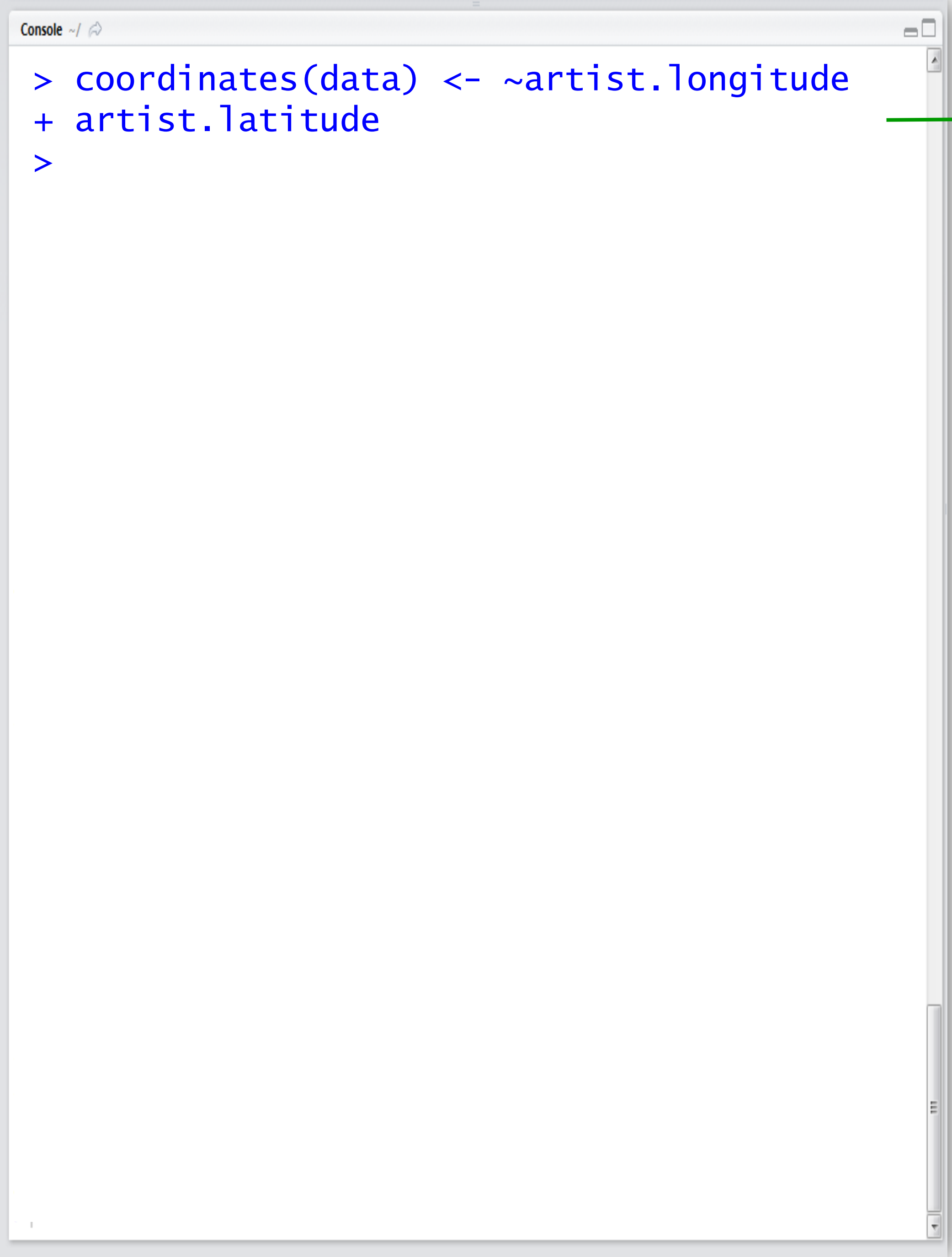
```
> data <- subset(data,
+ artist.familiarity>0)
> data$sa.hf <-
+ data$song.hotttnesss/
+ data$artist.familiarity
> data <- subset(data, data$sa.hf>1)
> dim(data)
[1] 175  11
>
```


Console ~/

```
>
```

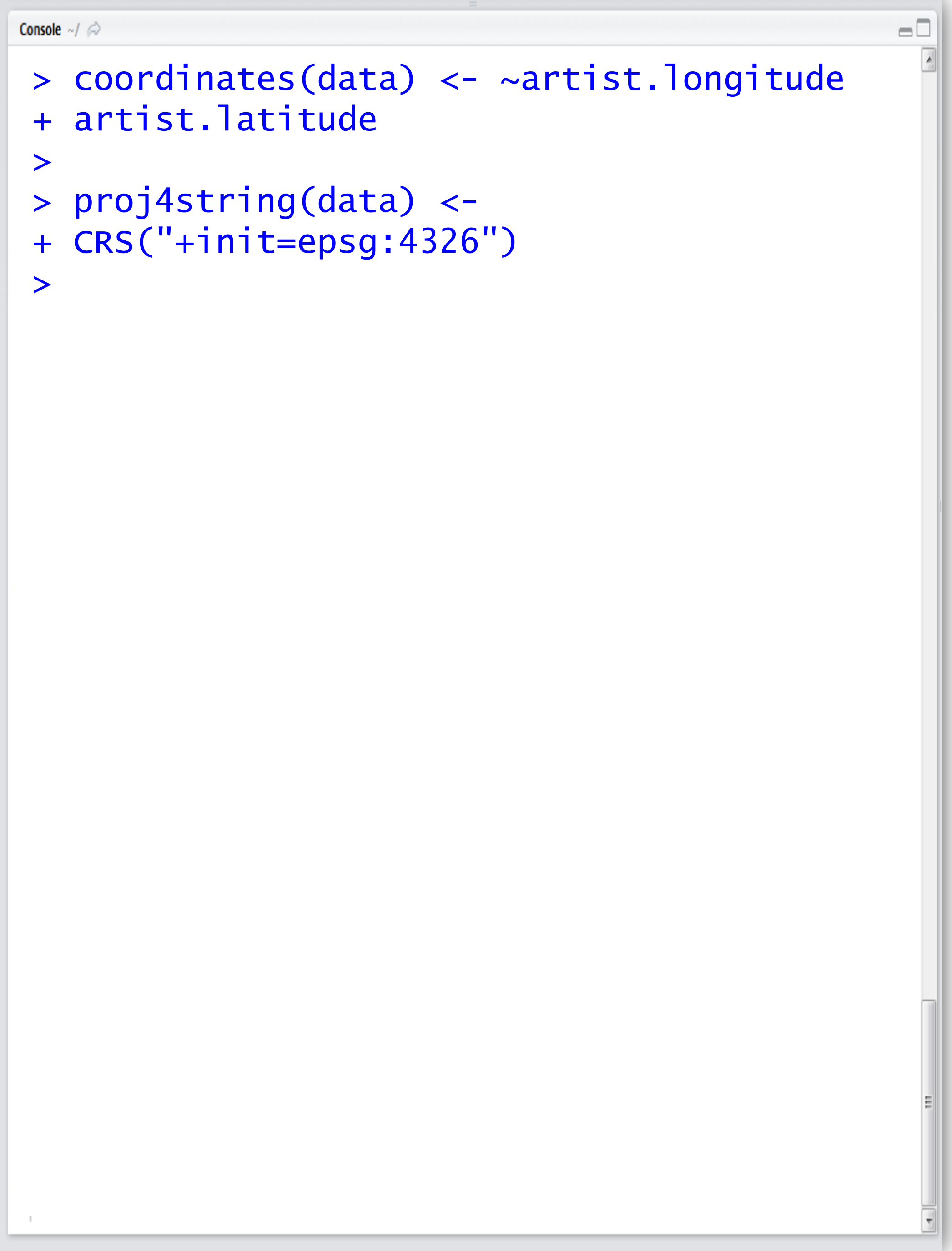


```
> coordinates(data) <- ~artist.longitude
+ artist.latitude
>
```

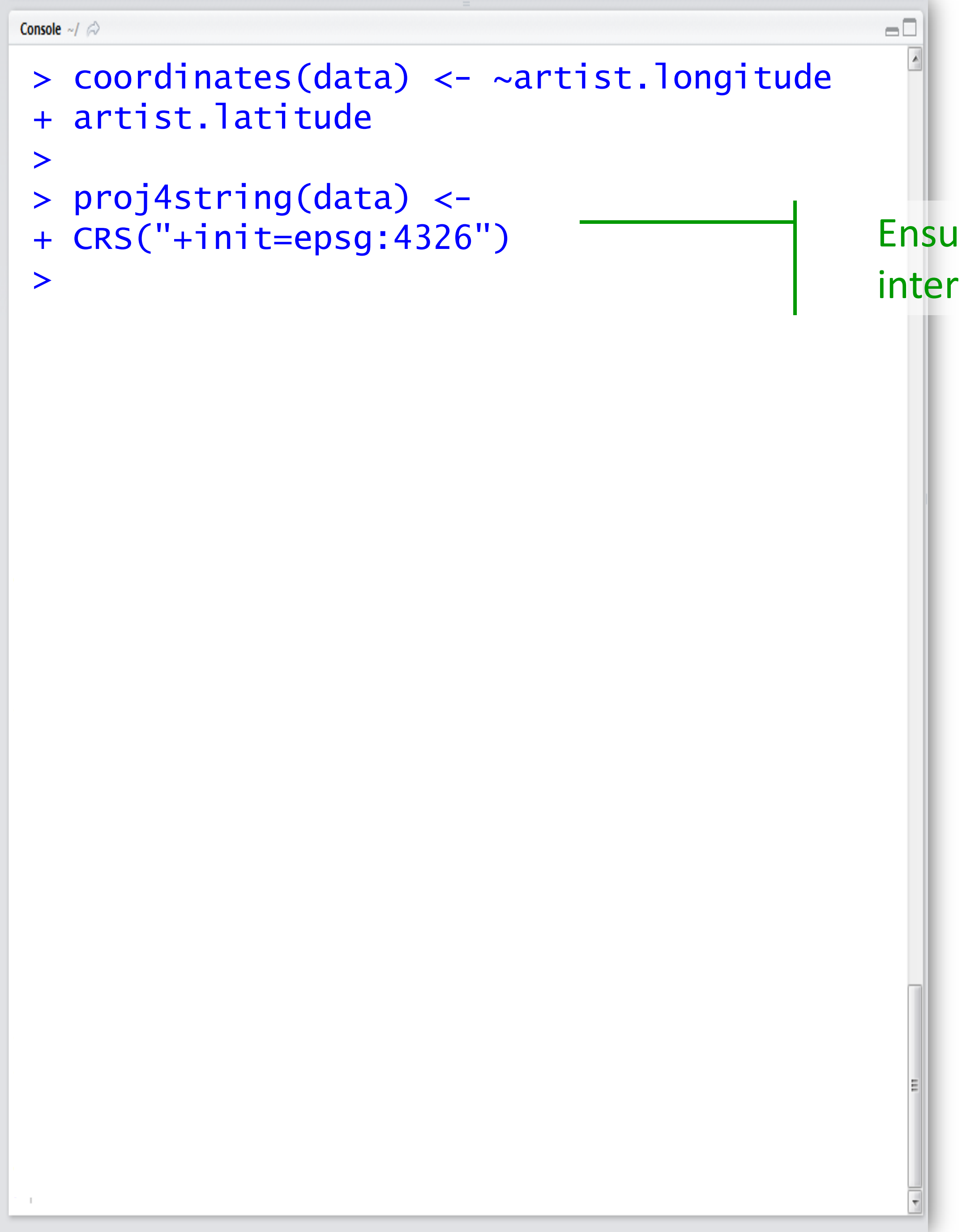


```
> coordinates(data) <- ~artist.longitude
+ artist.latitude
>
```

Convert data into a Spatial Points Data Frame.

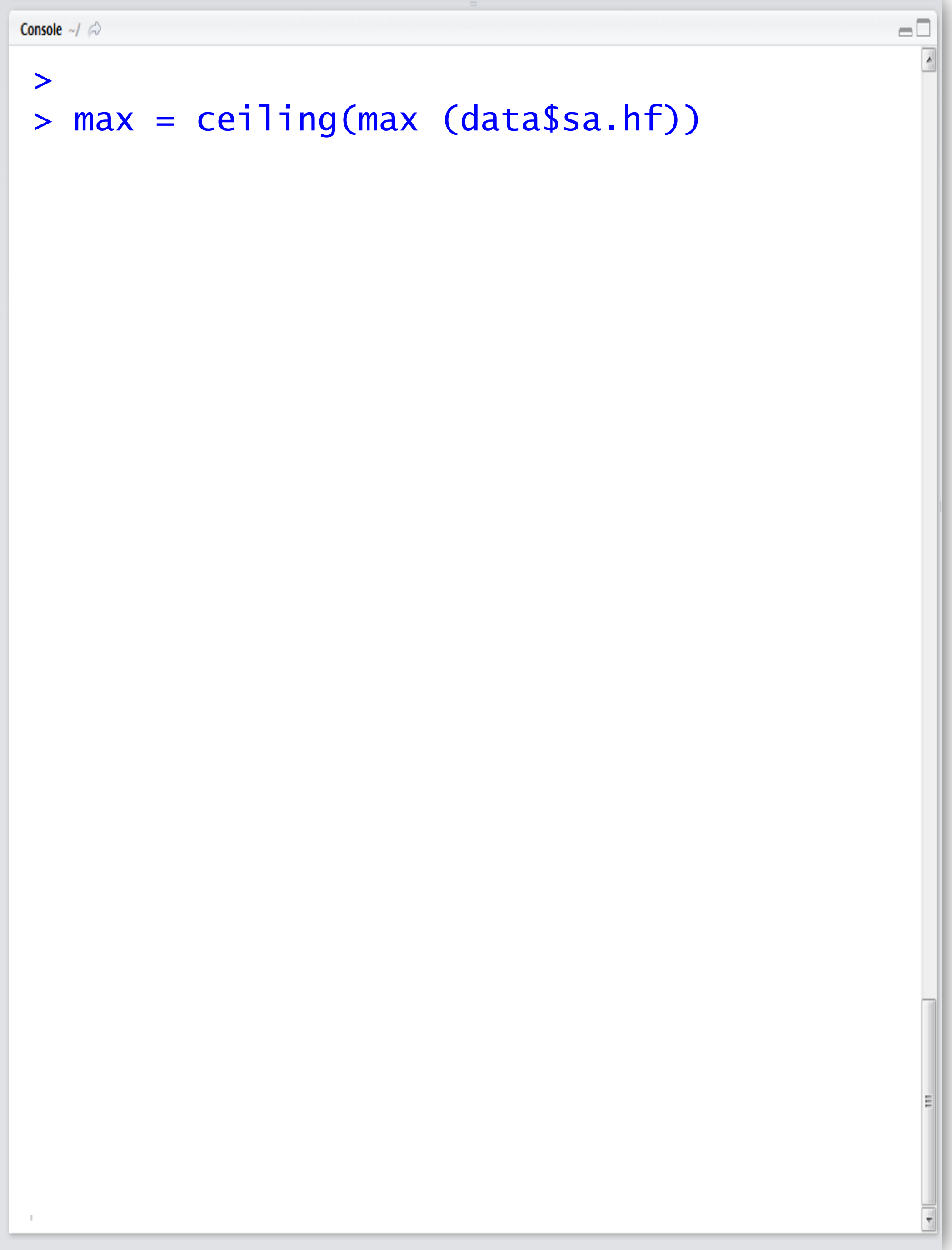


```
> coordinates(data) <- ~artist.longitude
+ artist.latitude
>
> proj4string(data) <-
+ CRS("+init=epsg:4326")
>
```

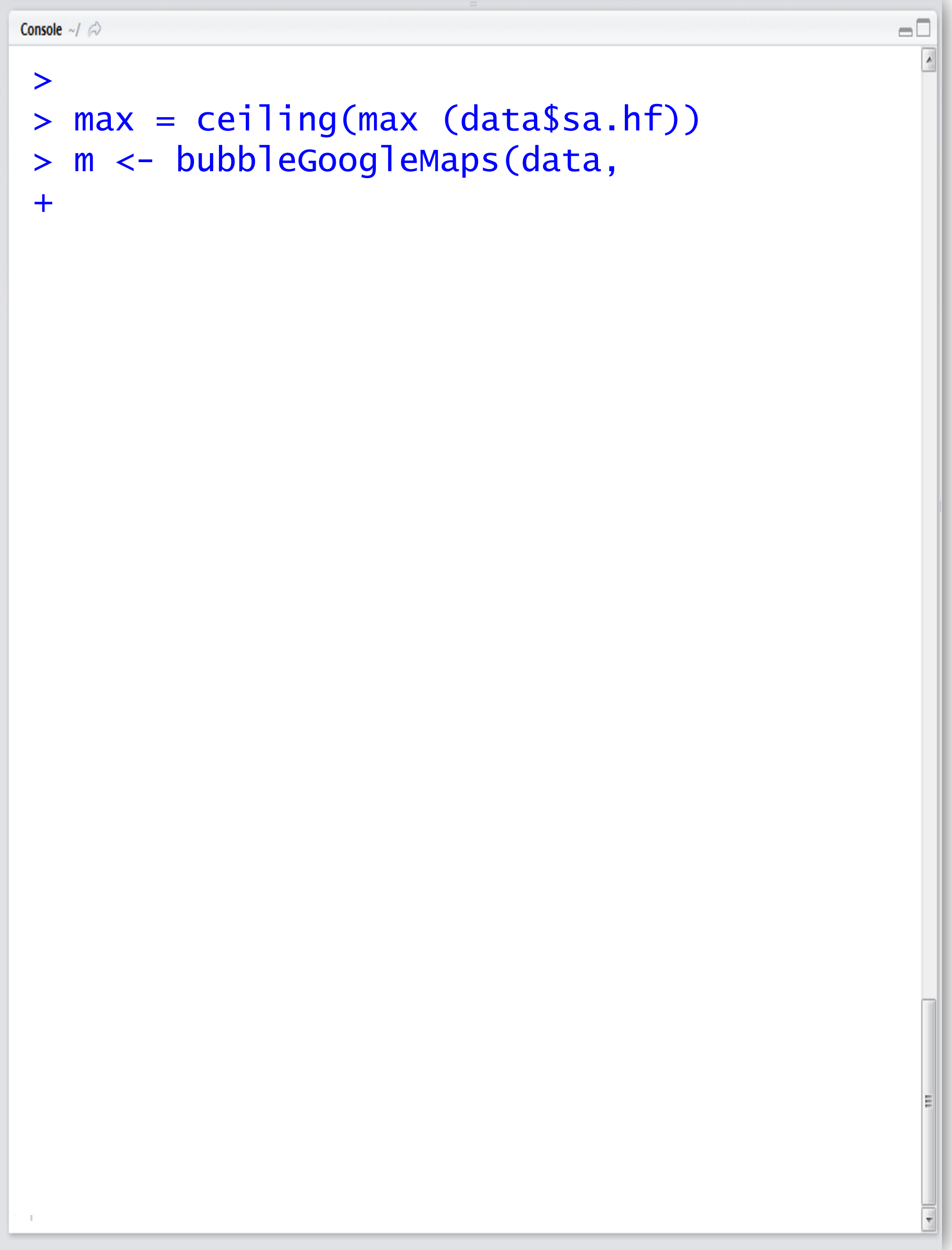


```
> coordinates(data) <- ~artist.longitude
+ artist.latitude
>
> proj4string(data) <-
+ CRS("+init=epsg:4326")
>
```

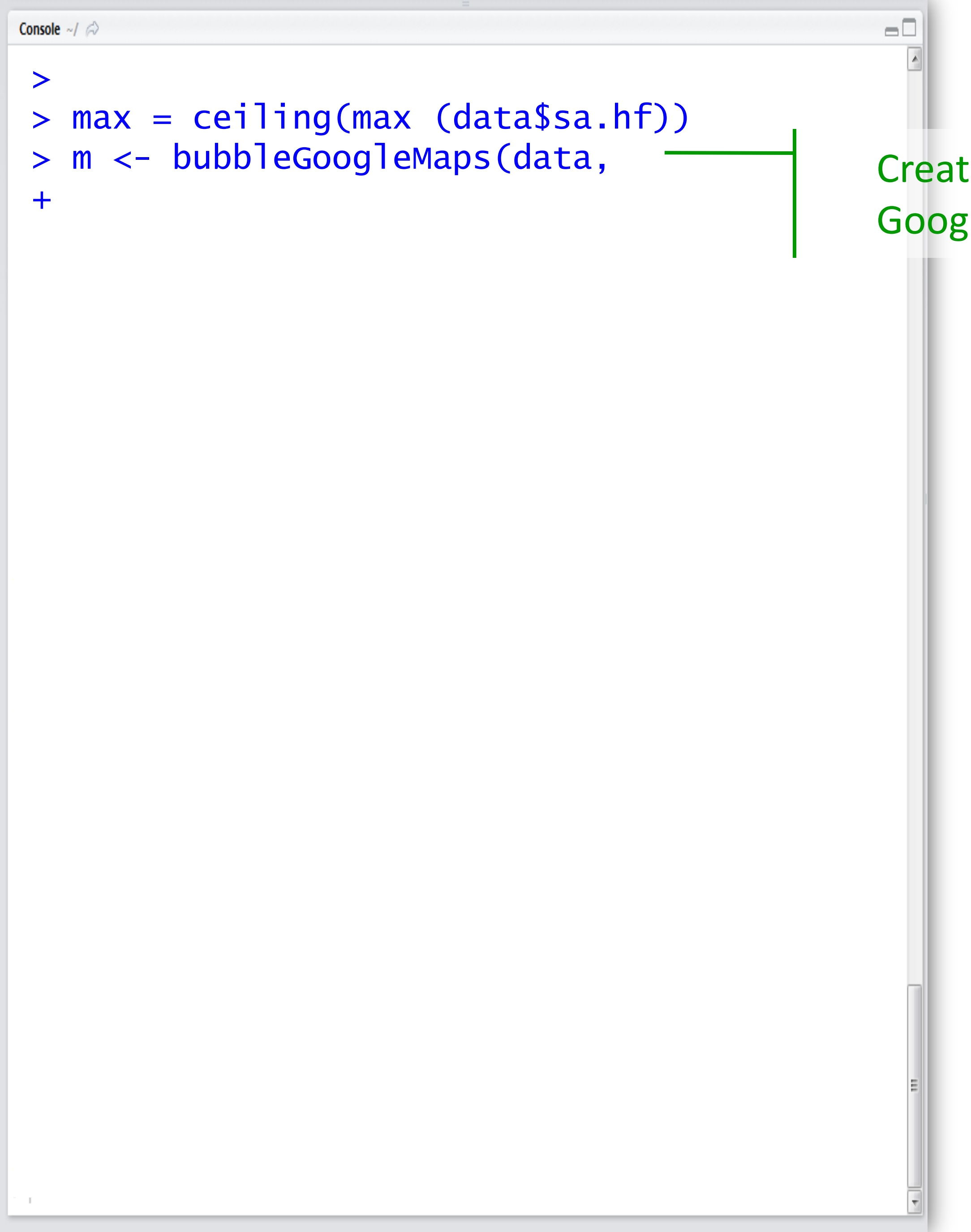
Ensure that the coordinates will be interpreted as longitudes and latitudes.



```
>
> max = ceiling(max (data$sa.hf))
```

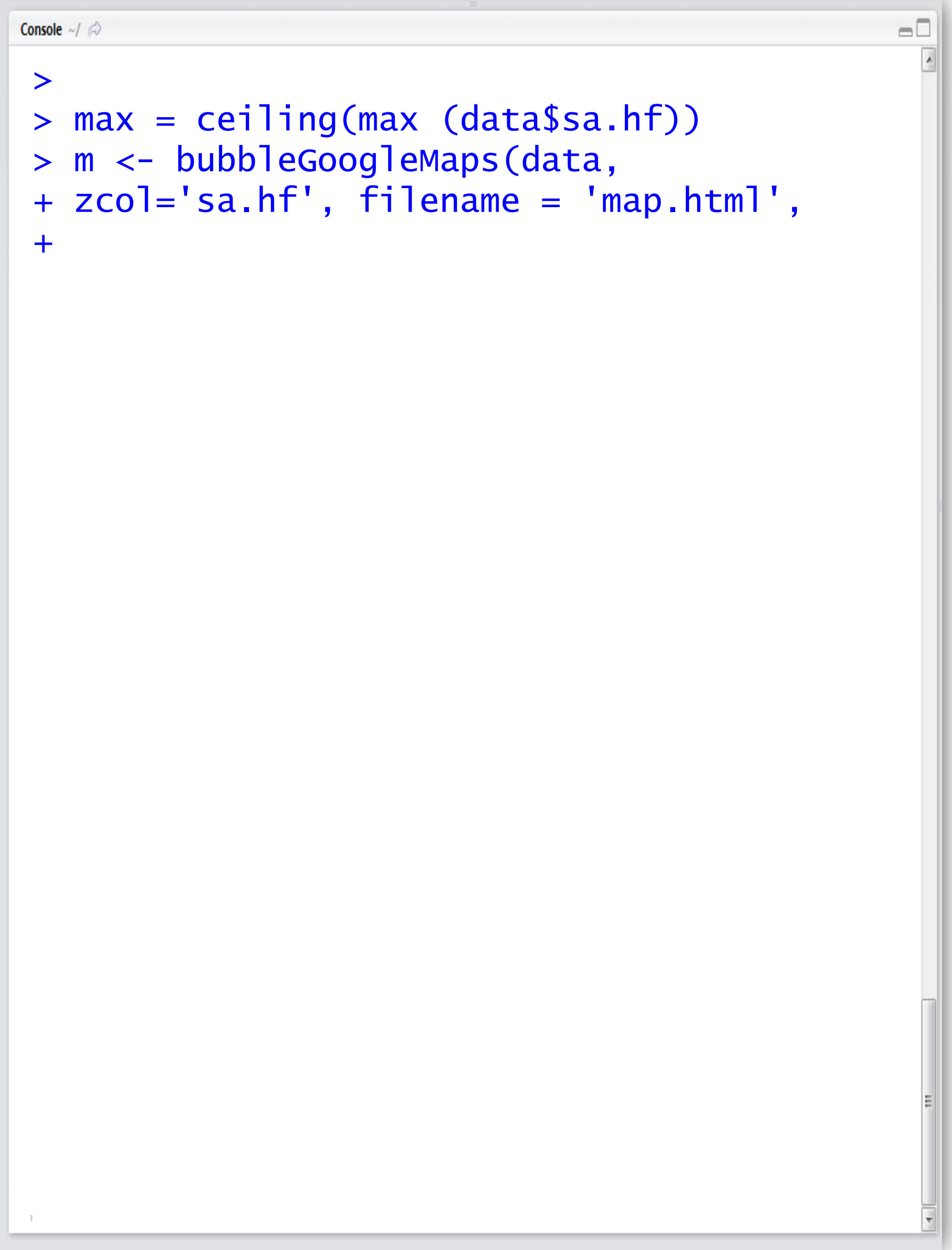


```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+
```

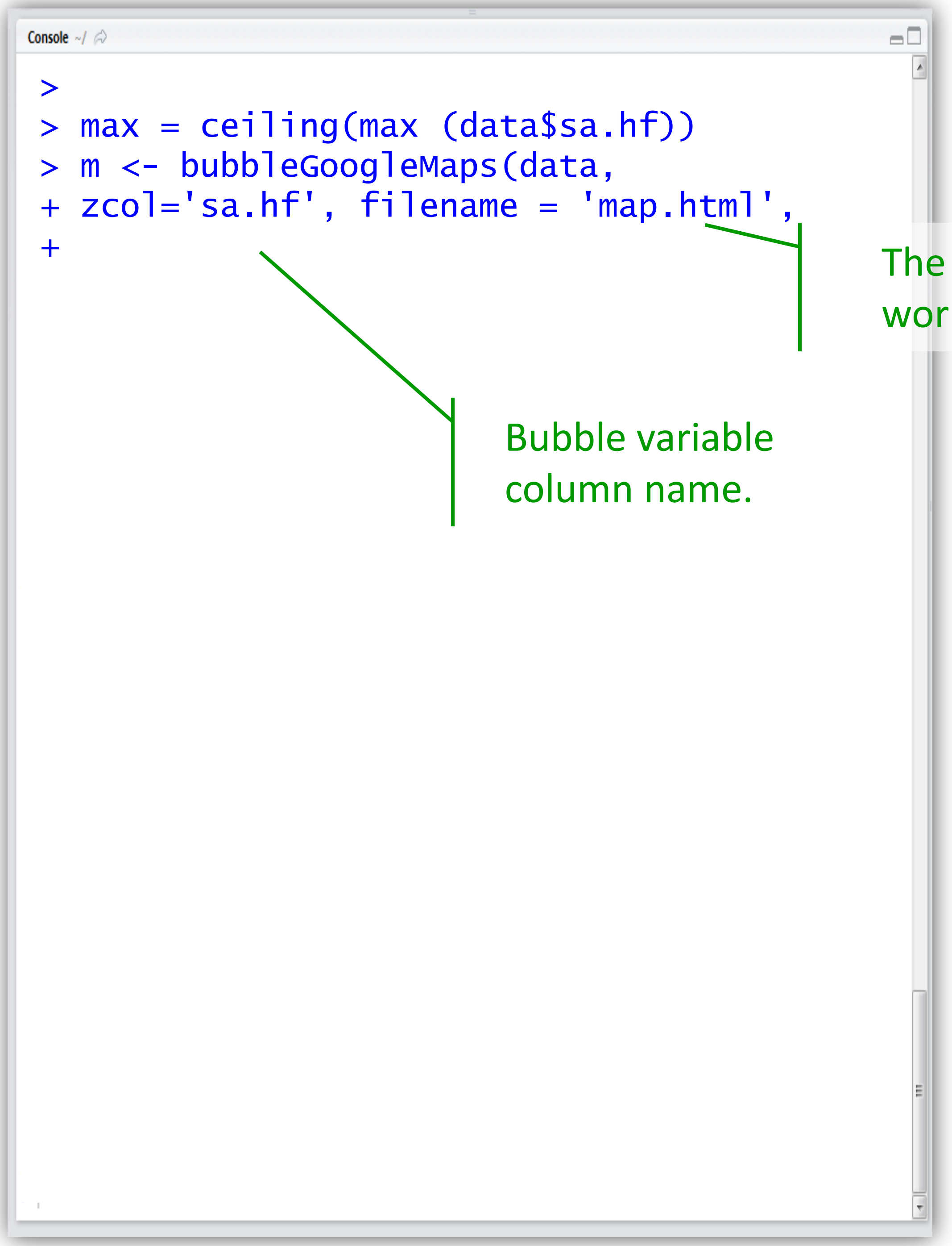


```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+
```

Create a bubble plot of spatial data on Google Maps.



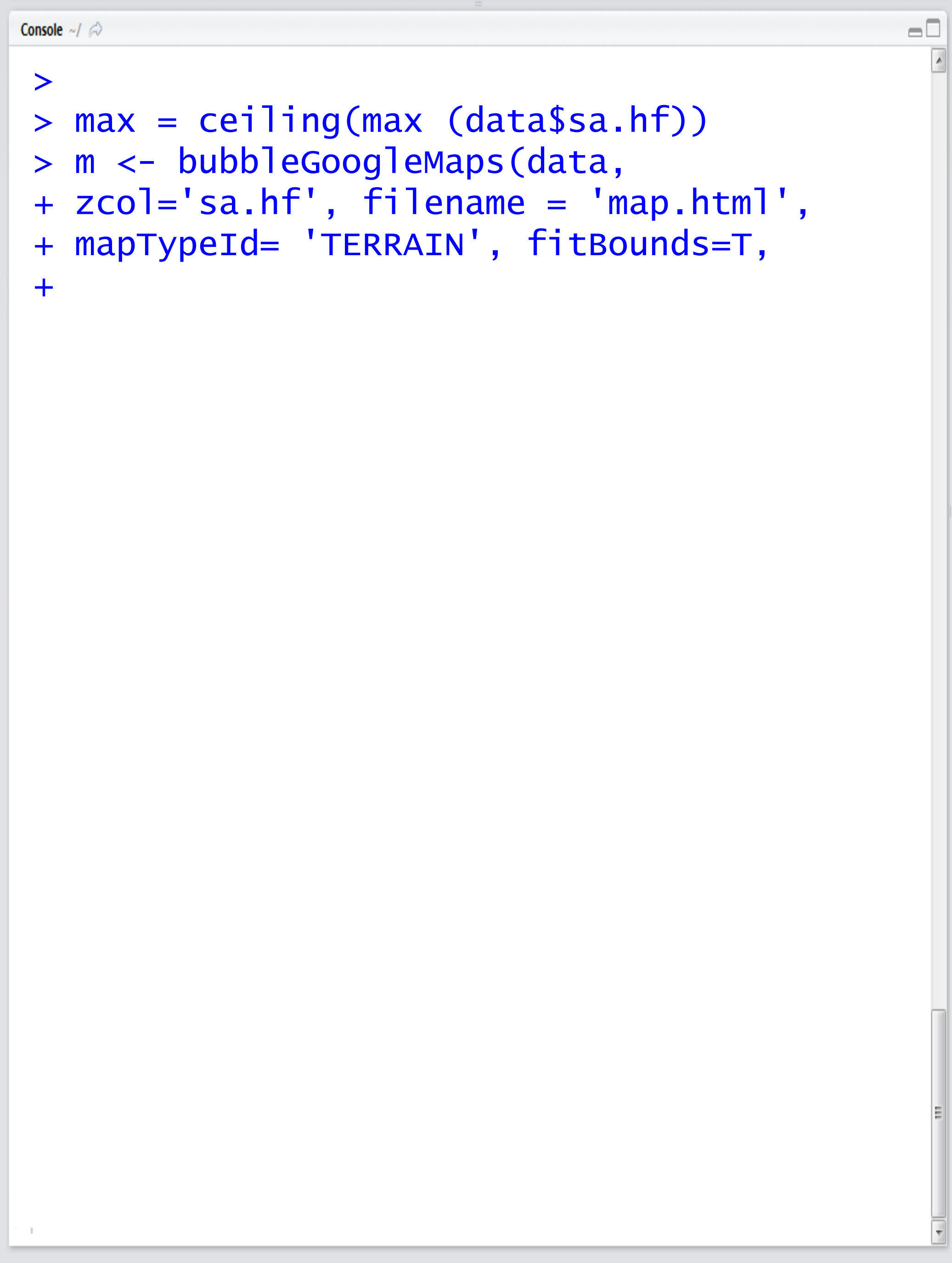
```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+ zcol='sa.hf', filename = 'map.html',
+
```



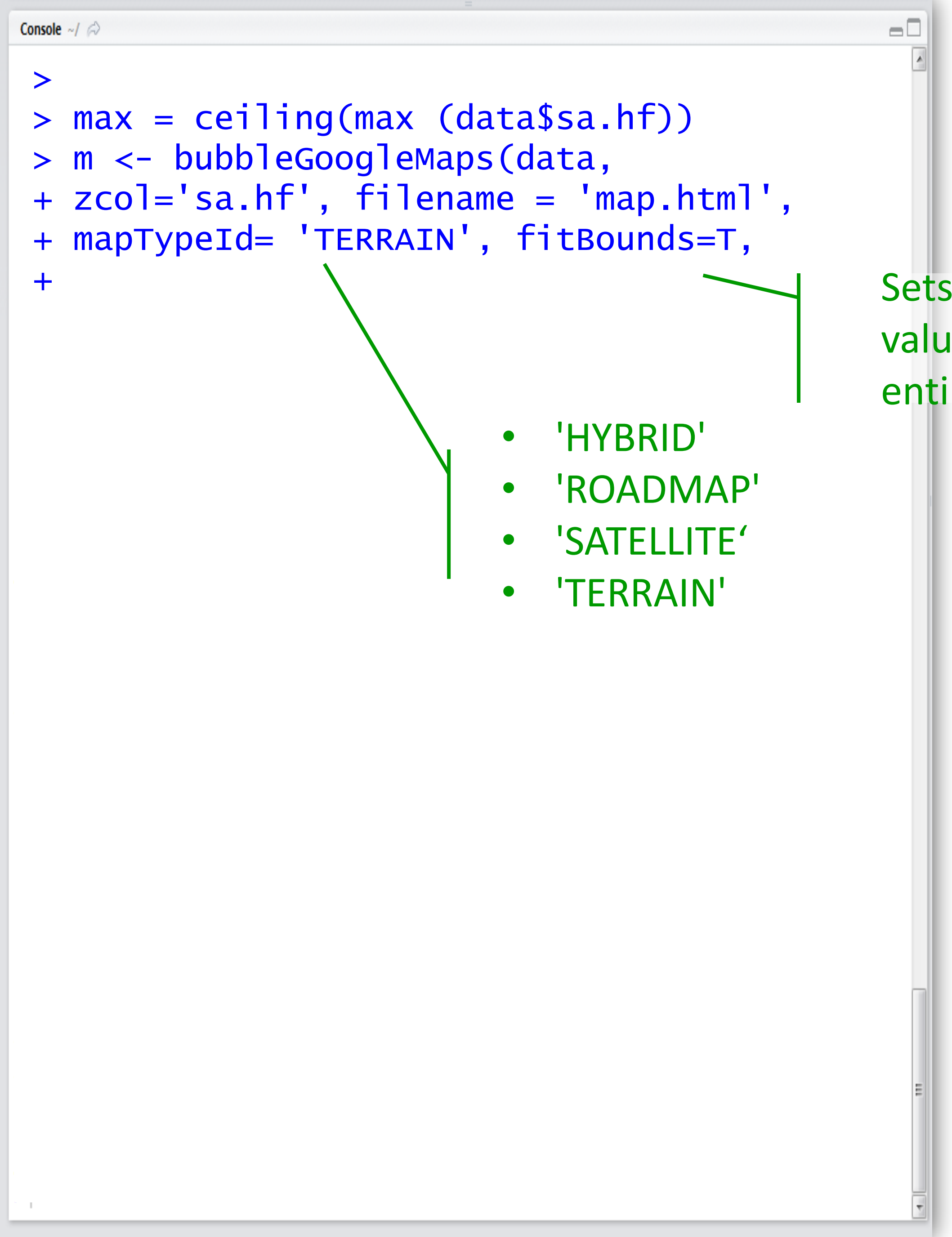
```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+ zcol='sa.hf', filename = 'map.html',
+
```

The name of the output file to be saved in working directory

Bubble variable column name.



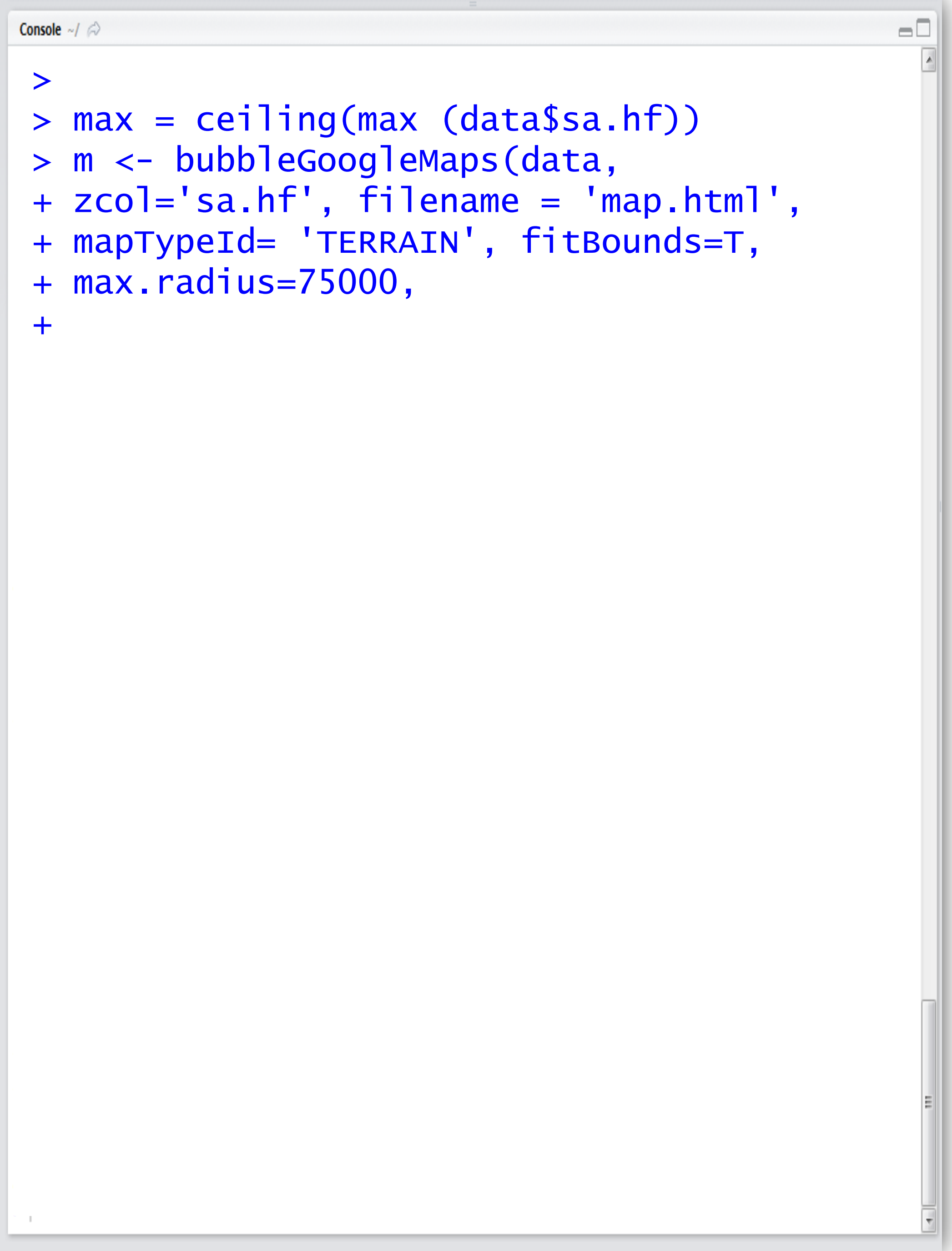
```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+ zcol='sa.hf', filename = 'map.html',
+ mapTypeId= 'TERRAIN', fitBounds=T,
+
```



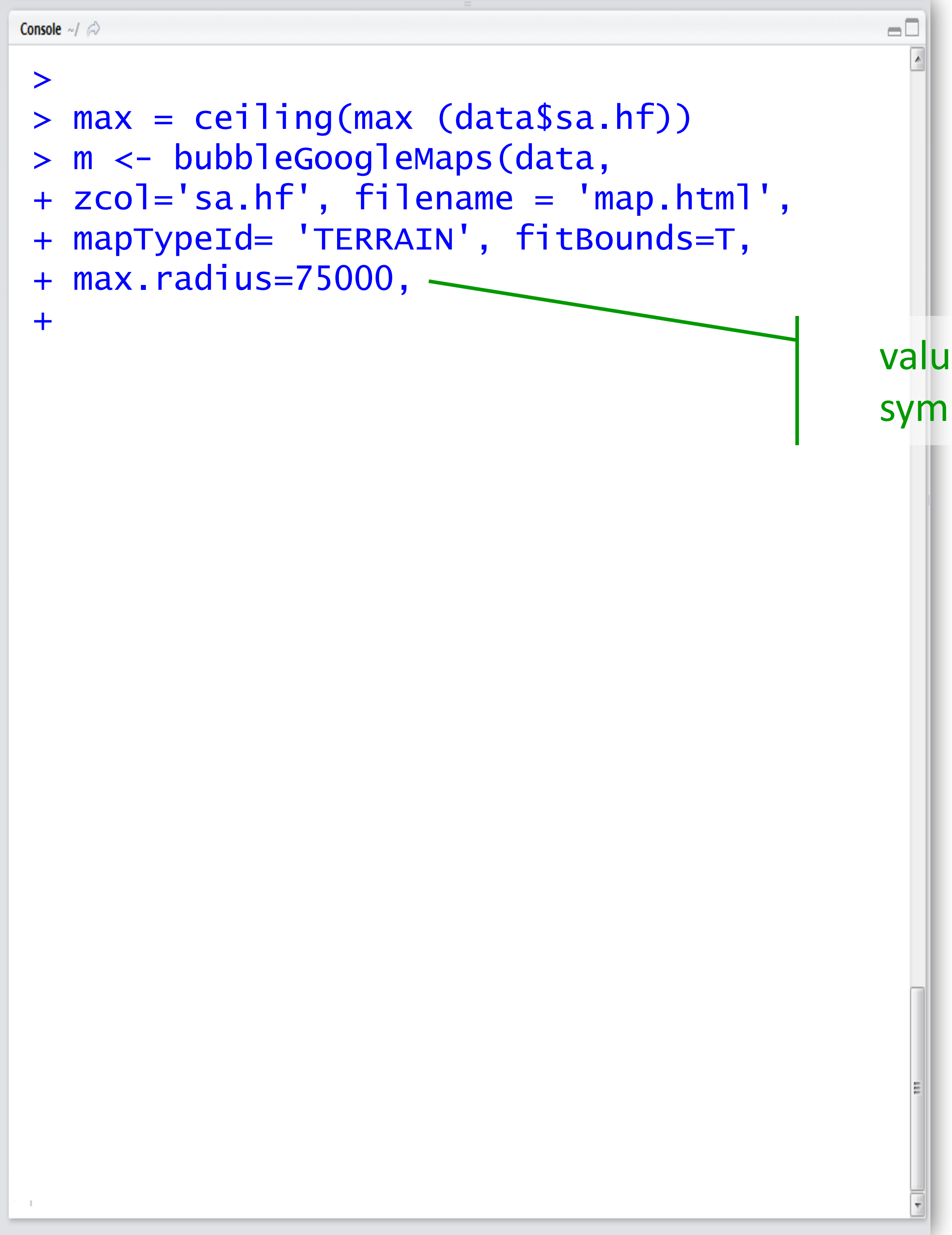
```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+ zcol='sa.hf', filename = 'map.html',
+ mapTypeId= 'TERRAIN', fitBounds=T,
+
```

Sets the maps to fit to the boundary box values of spacial object, in our case – entire world.

- 'HYBRID'
- 'ROADMAP'
- 'SATELLITE‘
- 'TERRAIN'

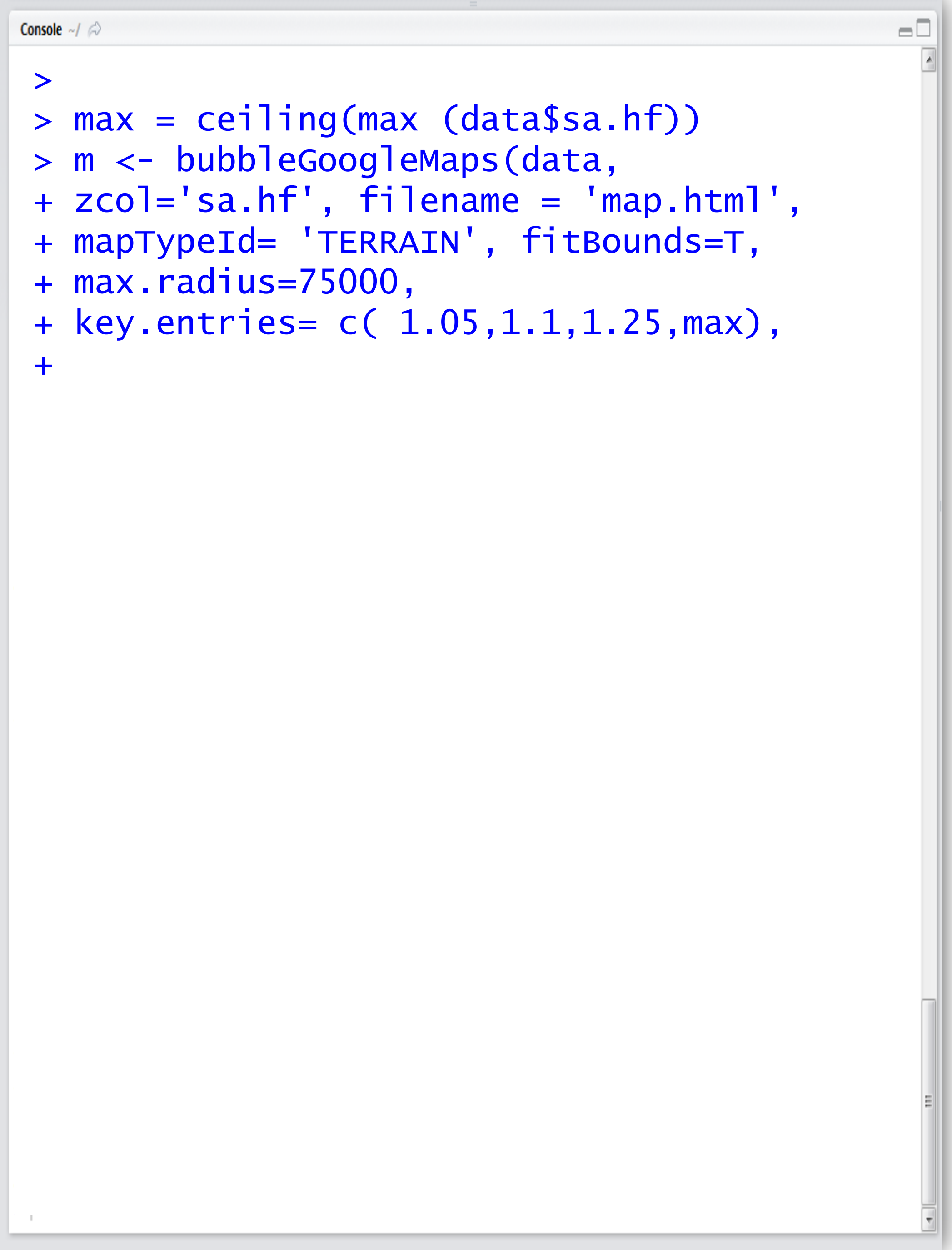


```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+ zcol='sa.hf', filename = 'map.html',
+ mapTypeId= 'TERRAIN', fitBounds=T,
+ max.radius=75000,
+
```

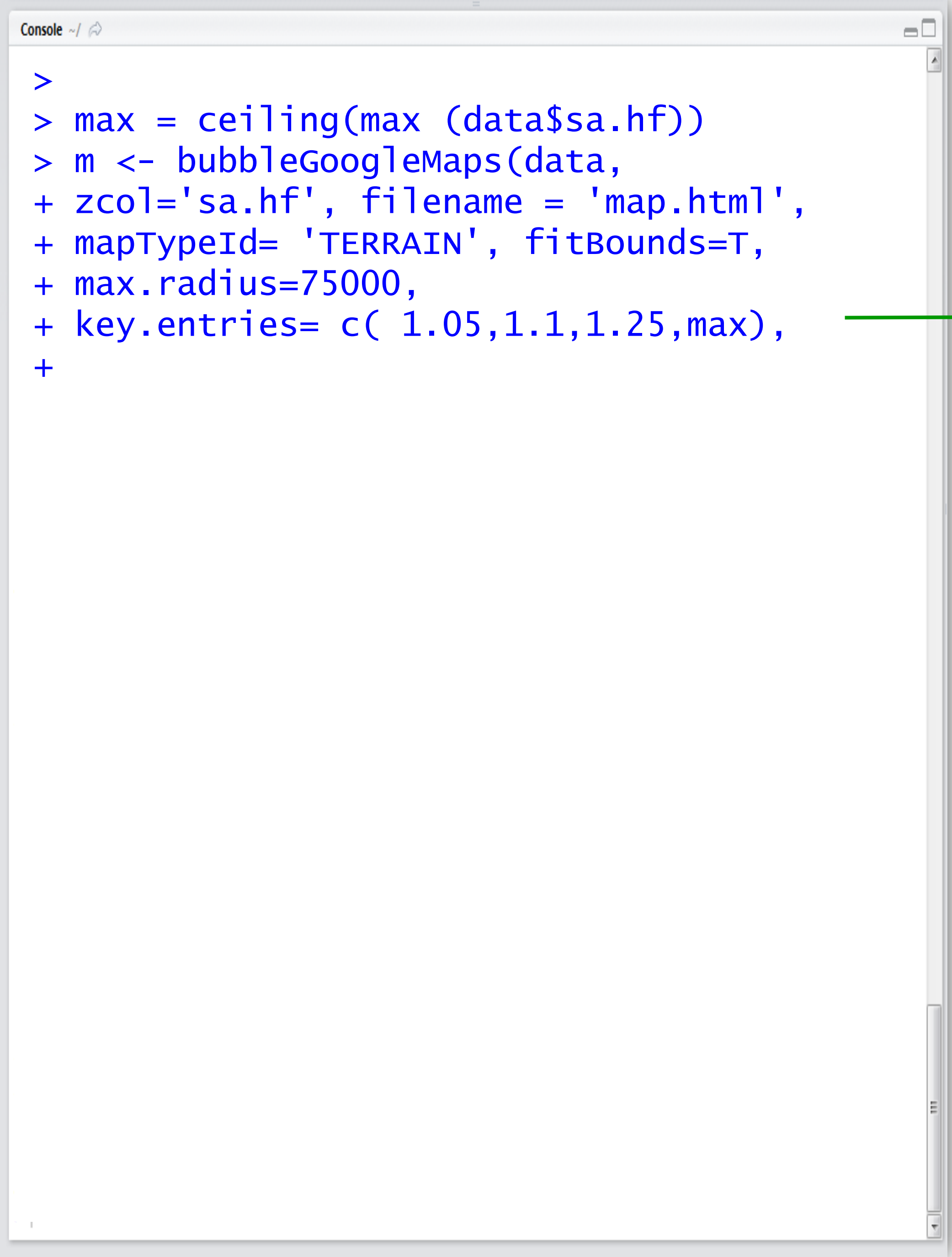


```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+ zcol='sa.hf', filename = 'map.html',
+ mapTypeId= 'TERRAIN', fitBounds=T,
+ max.radius=75000,
+
```

value for largest circle (the plotting symbols) in meters

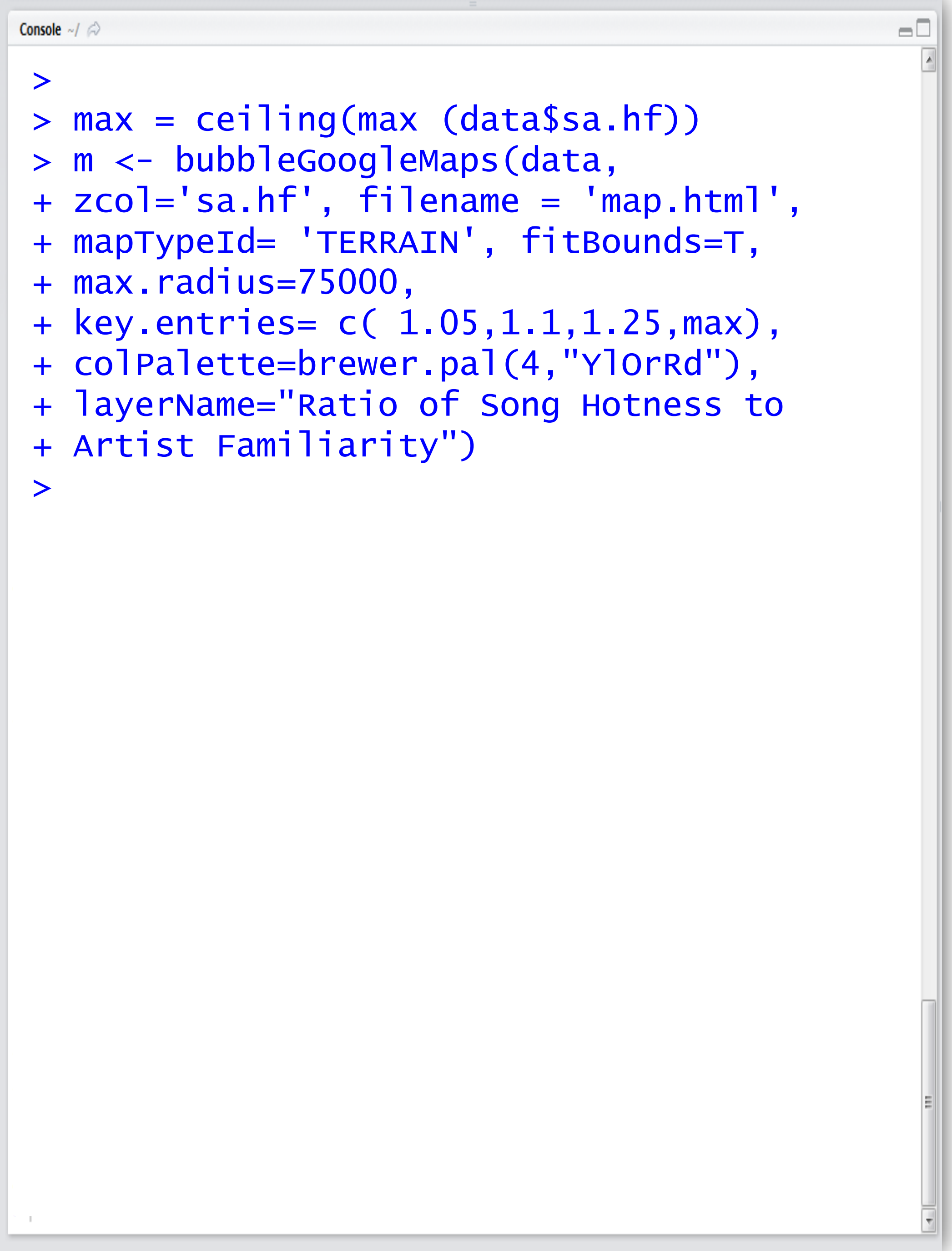


```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+ zcol='sa.hf', filename = 'map.html',
+ mapTypeId= 'TERRAIN', fitBounds=T,
+ max.radius=75000,
+ key.entries= c( 1.05,1.1,1.25,max),
+
```

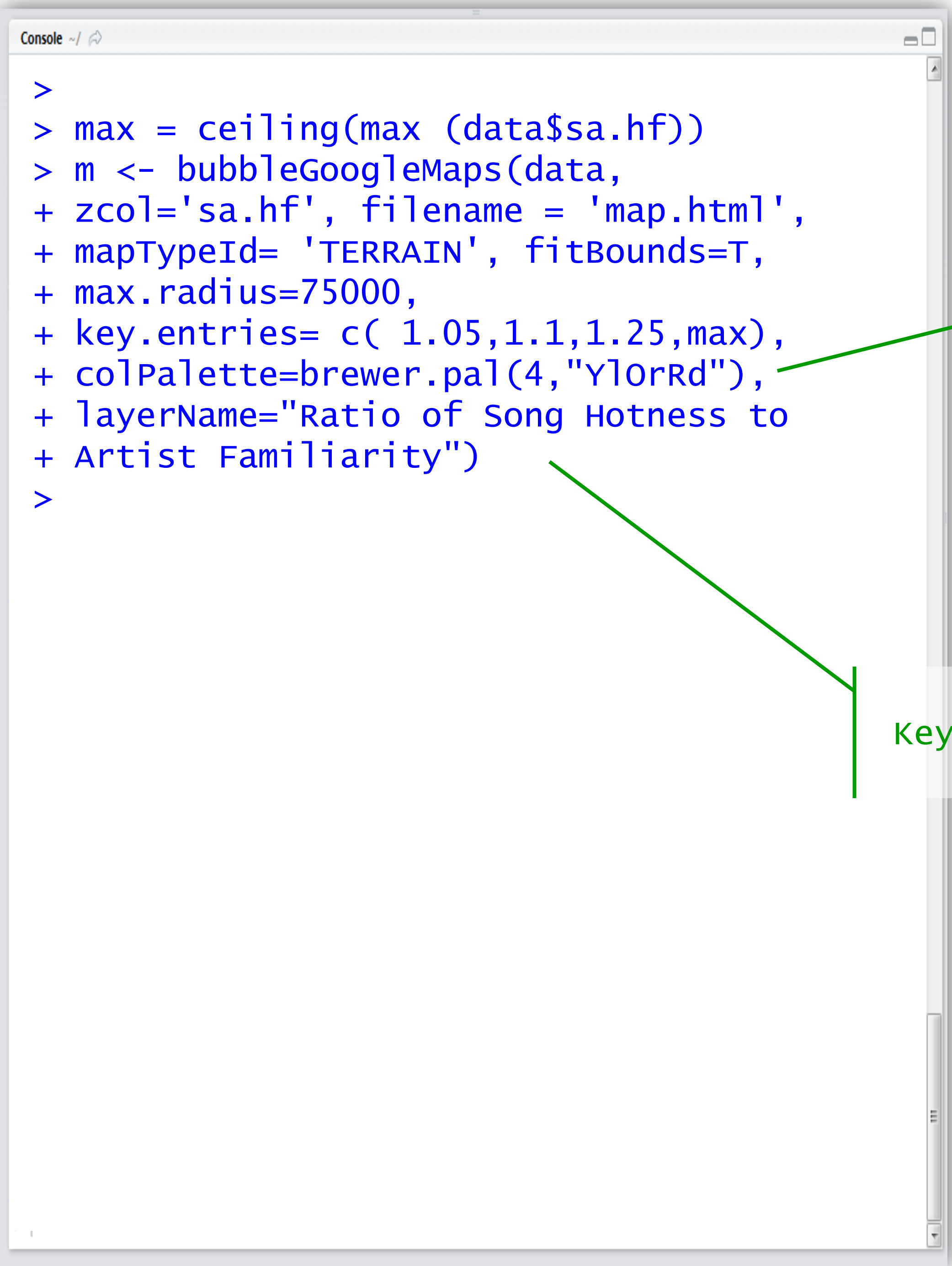


```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+ zcol='sa.hf', filename = 'map.html',
+ mapTypeId= 'TERRAIN', fitBounds=T,
+ max.radius=75000,
+ key.entries= c( 1.05,1.1,1.25,max),
+
```

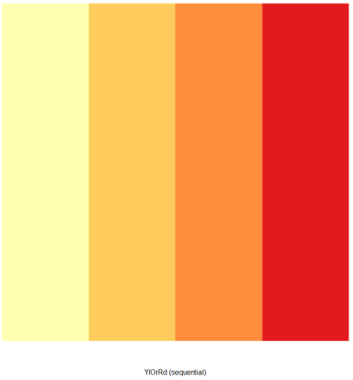
Key Values. These are upper endpoints. Do not include lowest value.



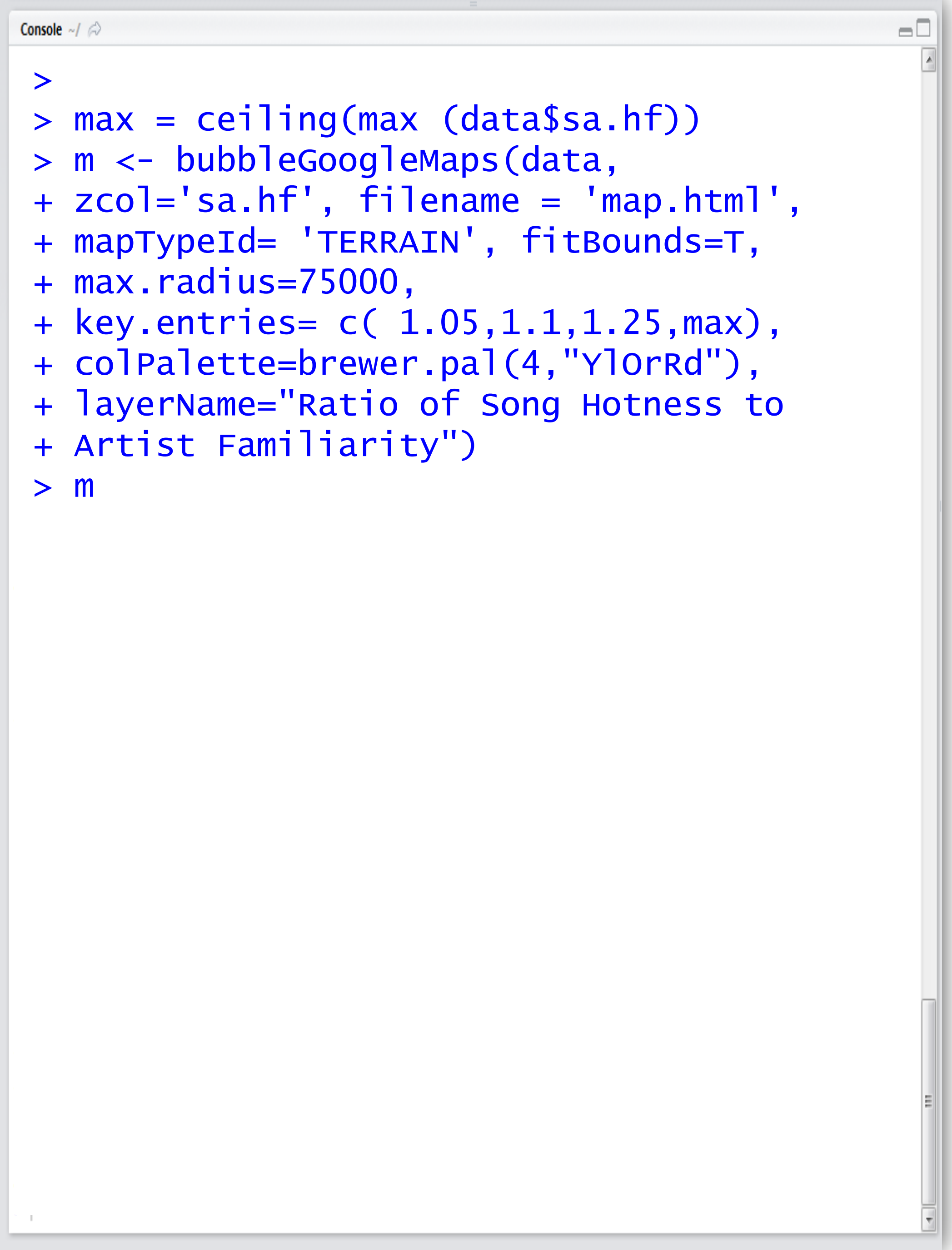
```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+ zcol='sa.hf', filename = 'map.html',
+ mapTypeId= 'TERRAIN', fitBounds=T,
+ max.radius=75000,
+ key.entries= c( 1.05,1.1,1.25,max),
+ colPalette=brewer.pal(4,"YlOrRd"),
+ layerName="Ratio of Song Hotness to
+ Artist Familiarity")
>
```



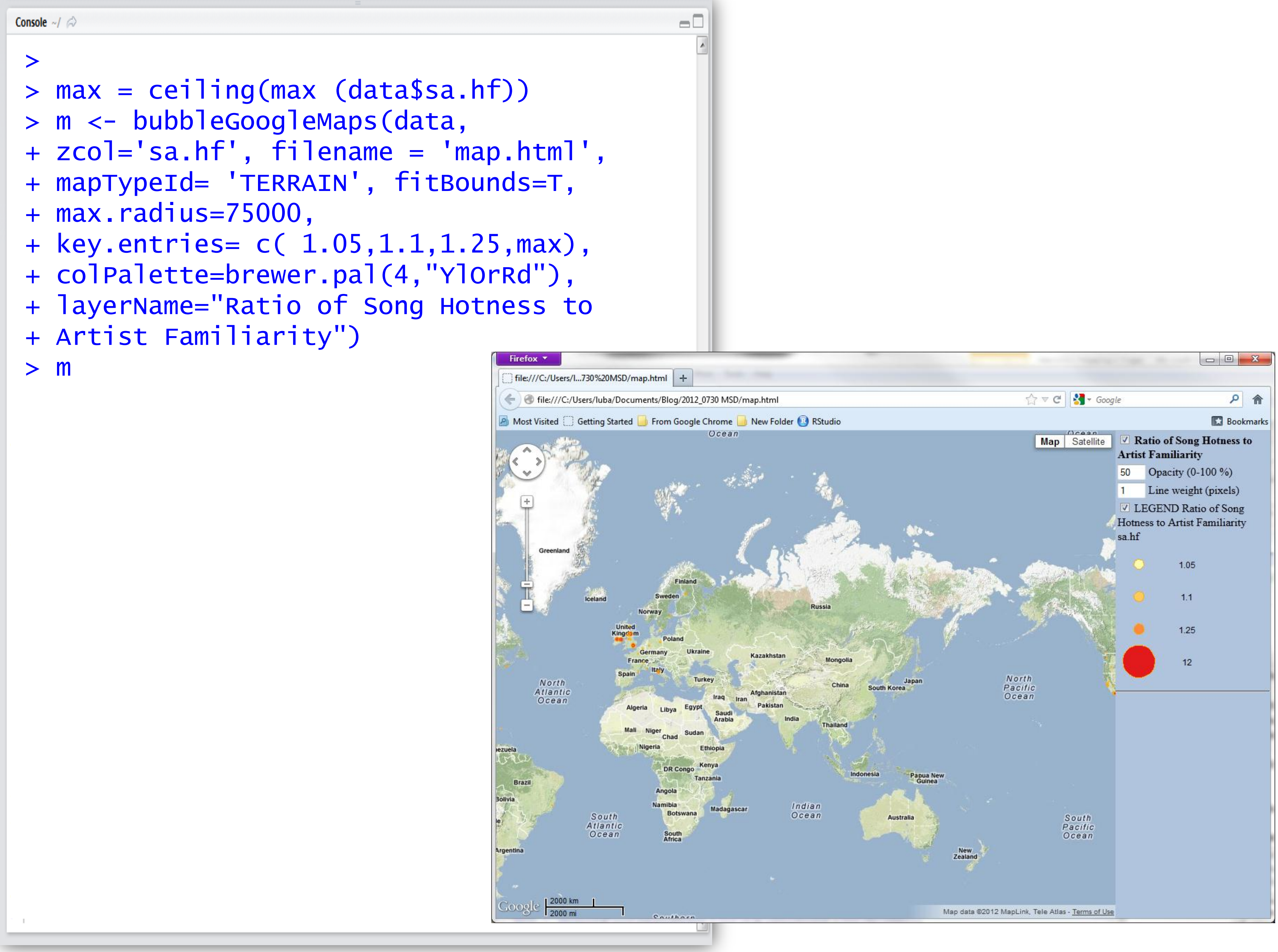
```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+ zcol='sa.hf', filename = 'map.html',
+ mapTypeId= 'TERRAIN', fitBounds=T,
+ max.radius=75000,
+ key.entries= c( 1.05,1.1,1.25,max),
+ colPalette=brewer.pal(4,"YlOrRd"),
+ layerName="Ratio of Song Hotness to
+ Artist Familiarity")
>
```



Key Header



```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+ zcol='sa.hf', filename = 'map.html',
+ mapTypeId= 'TERRAIN', fitBounds=T,
+ max.radius=75000,
+ key.entries= c( 1.05,1.1,1.25,max),
+ colPalette=brewer.pal(4,"YlOrRd"),
+ layerName="Ratio of Song Hotness to
+ Artist Familiarity")
> m
```



Console ~/

```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+ zcol='sa.hf', filename = 'map.html',
+ mapTypeId= 'TERRAIN', fitBounds=T,
+ max.radius=75000,
+ key.entries= c( 1.05,1.1,1.25,max),
+ colPalette=brewer.pal(4,"YlOrRd"),
+ layerName="Ratio of Song Hotness to
+ Artist Familiarity")
> m
```

file:///C:/Users/luba/Documents/Blog/2012_0730 MSD/map.html

☑ Ratio of Song Hotness to Artist Familiarity

50 Opacity (0-100 %)

1 Line weight (pixels)

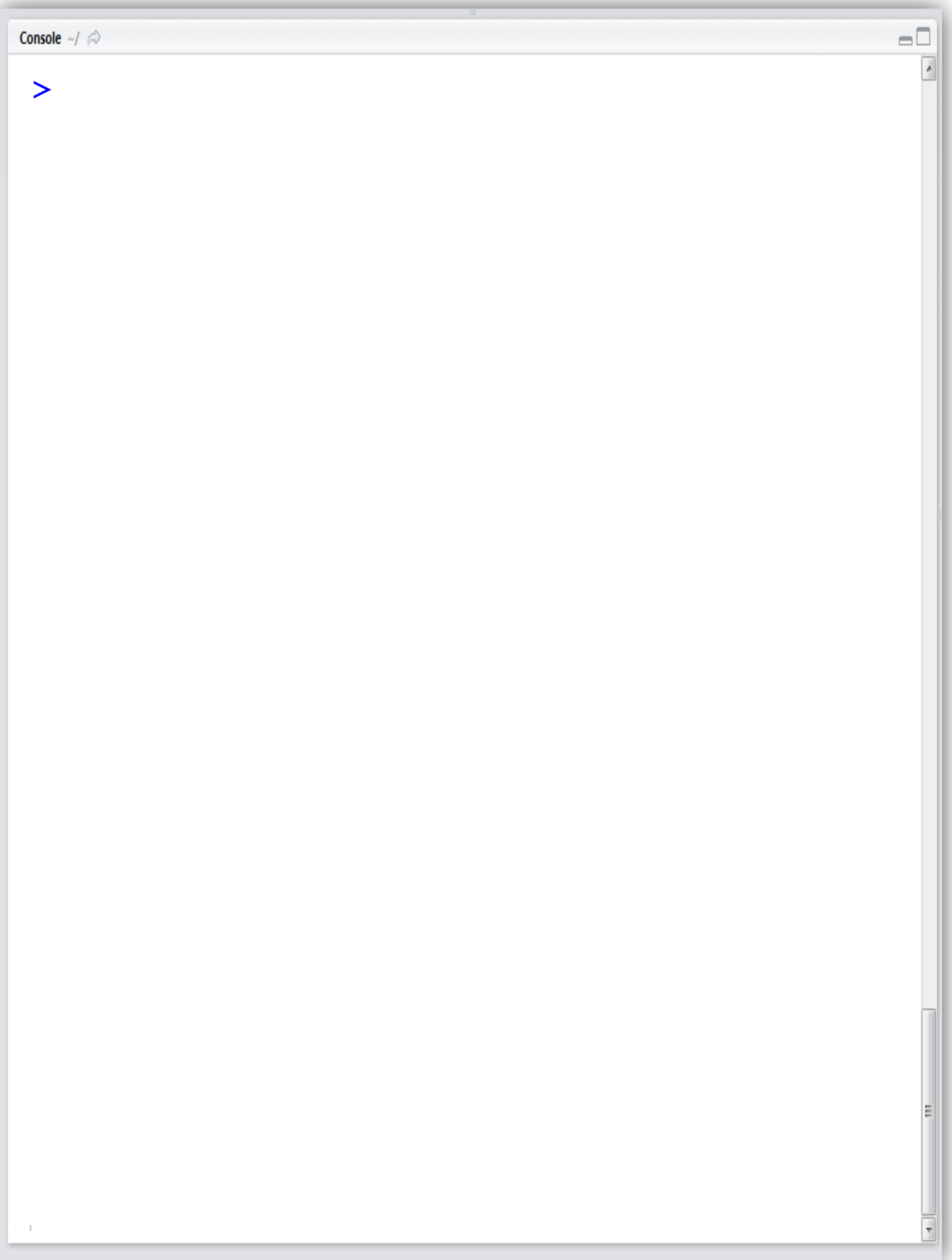
☑ LEGEND Ratio of Song Hotness to Artist Familiarity sa.hf

● 1.05

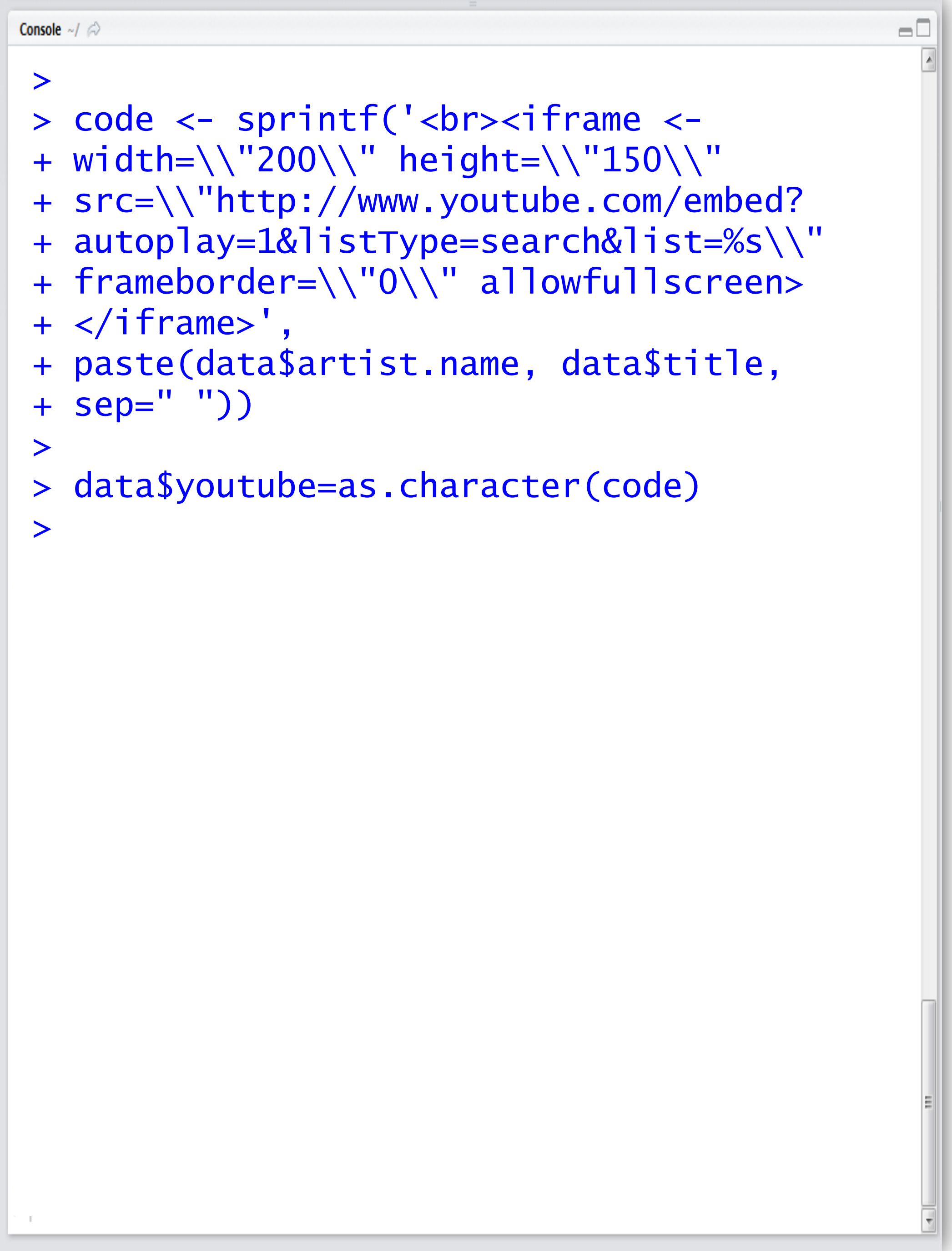
● 1.1

● 1.25

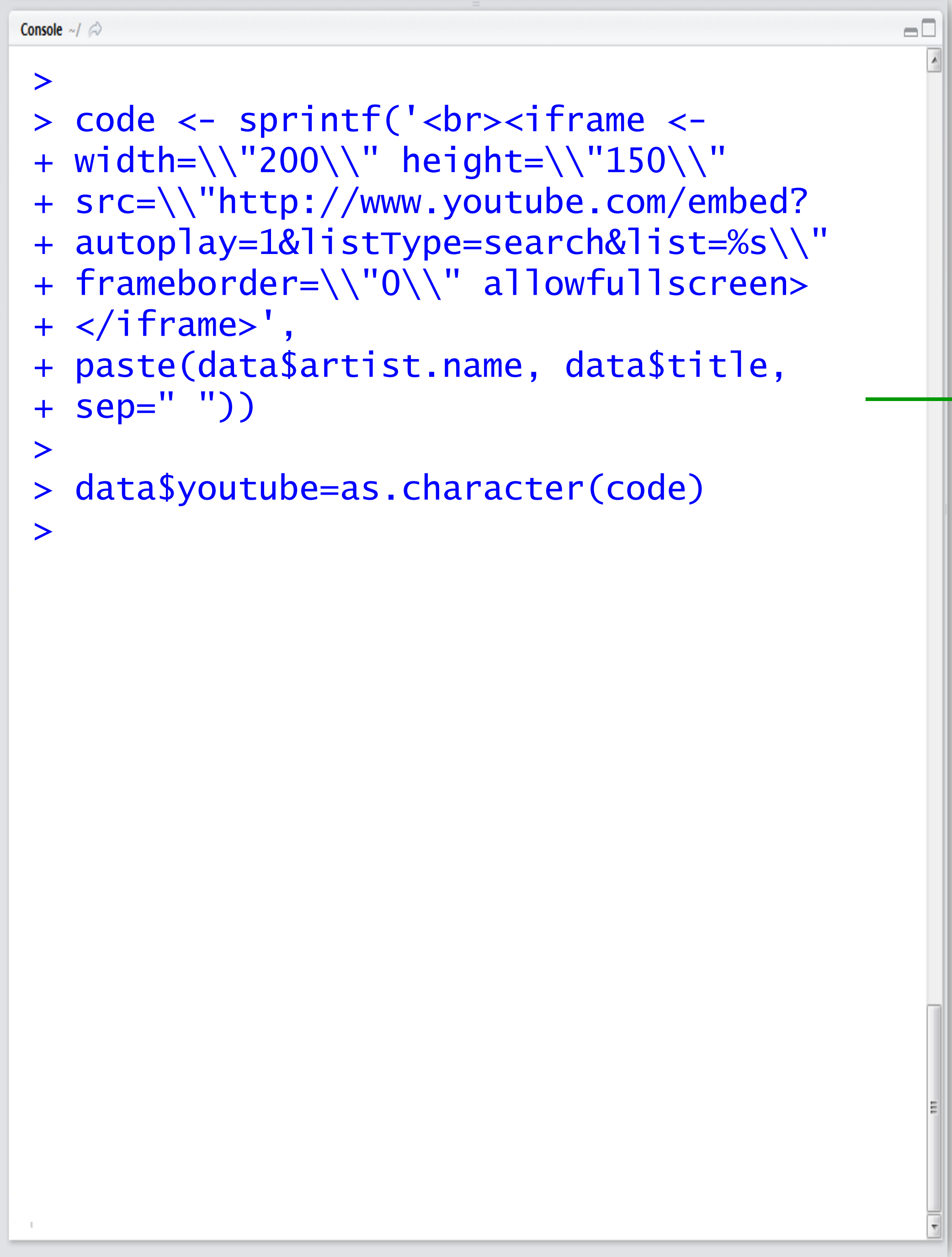
● 12


Console ~/

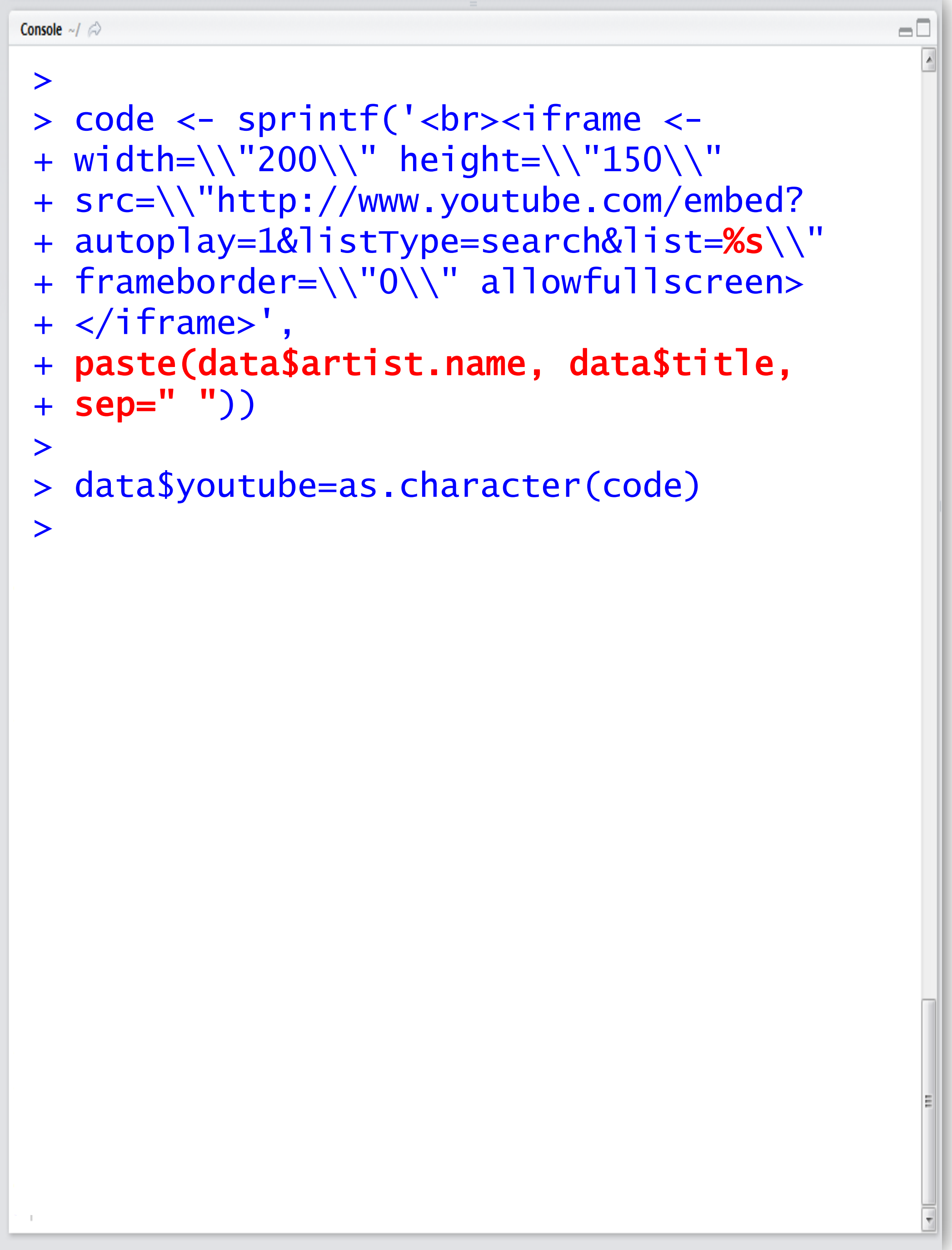
```
>
```



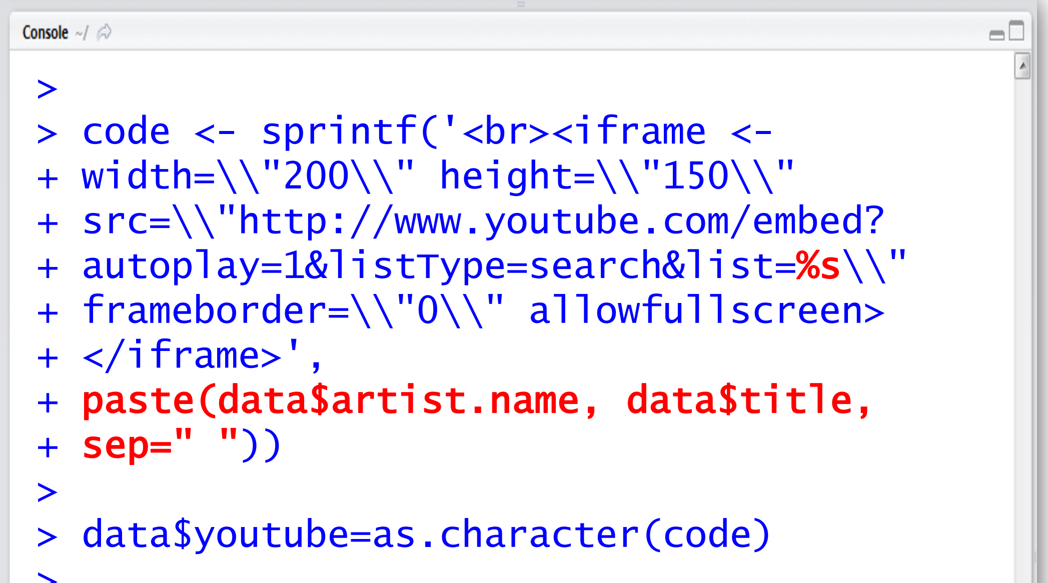
```
>
> code <- sprintf('<br><iframe <-
+ width=\\"200\\" height=\\"150\\"
+ src=\\"http://www.youtube.com/embed?
+ autoplay=1&listType=search&list=%s\\"
+ frameborder=\\"0\\" allowfullscreen>
+ </iframe>',
+ paste(data$artist.name, data$title,
+ sep=" "))
>
> data$youtube=as.character(code)
>
```



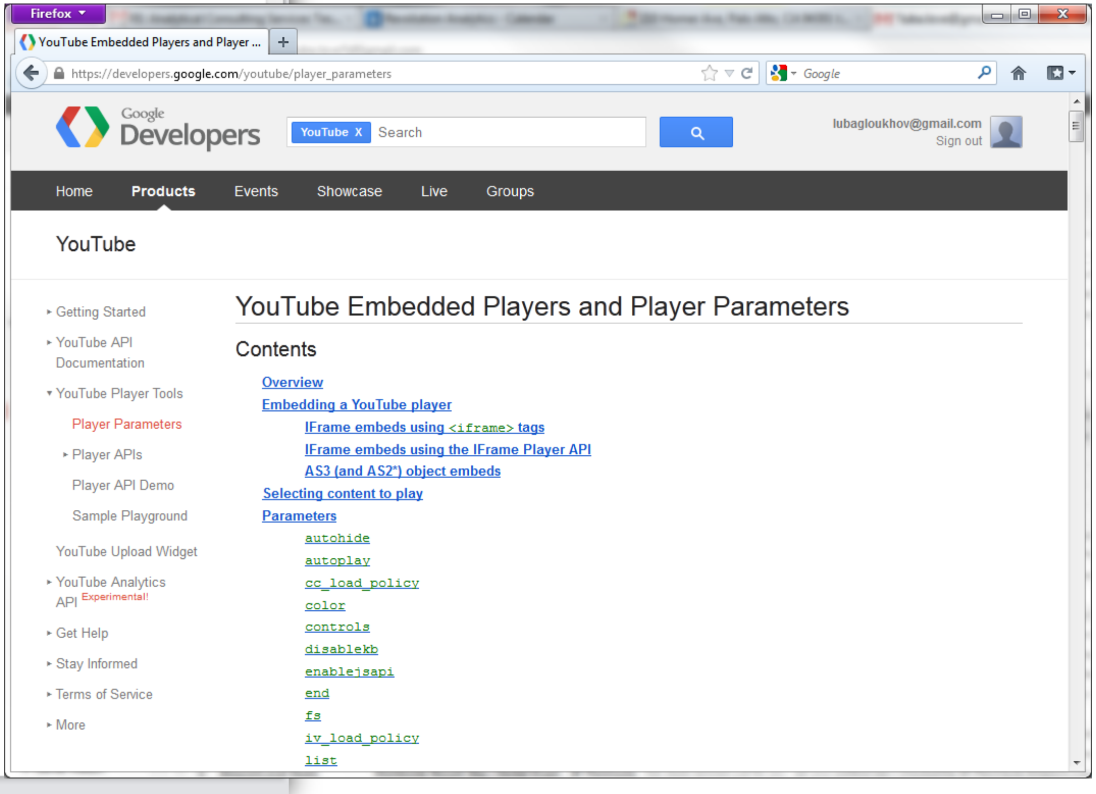
```
>
> code <- sprintf('<br><iframe <-
+ width=\\"200\\" height=\\"150\\"
+ src=\\"http://www.youtube.com/embed?
+ autoplay=1&listType=search&list=%s\\"
+ frameborder=\\"0\\" allowfullscreen>
+ </iframe>',
+ paste(data$artist.name, data$title,
+ sep=" "))
>
> data$youtube=as.character(code)
>
```



```
>
> code <- sprintf('<br><iframe <-
+ width=\\"200\\" height=\\"150\\"
+ src=\\"http://www.youtube.com/embed?
+ autoplay=1&listType=search&list=%s\\"
+ frameborder=\\"0\\" allowfullscreen>
+ </iframe>',
+ paste(data$artist.name, data$title,
+ sep=" "))
>
> data$youtube=as.character(code)
>
```



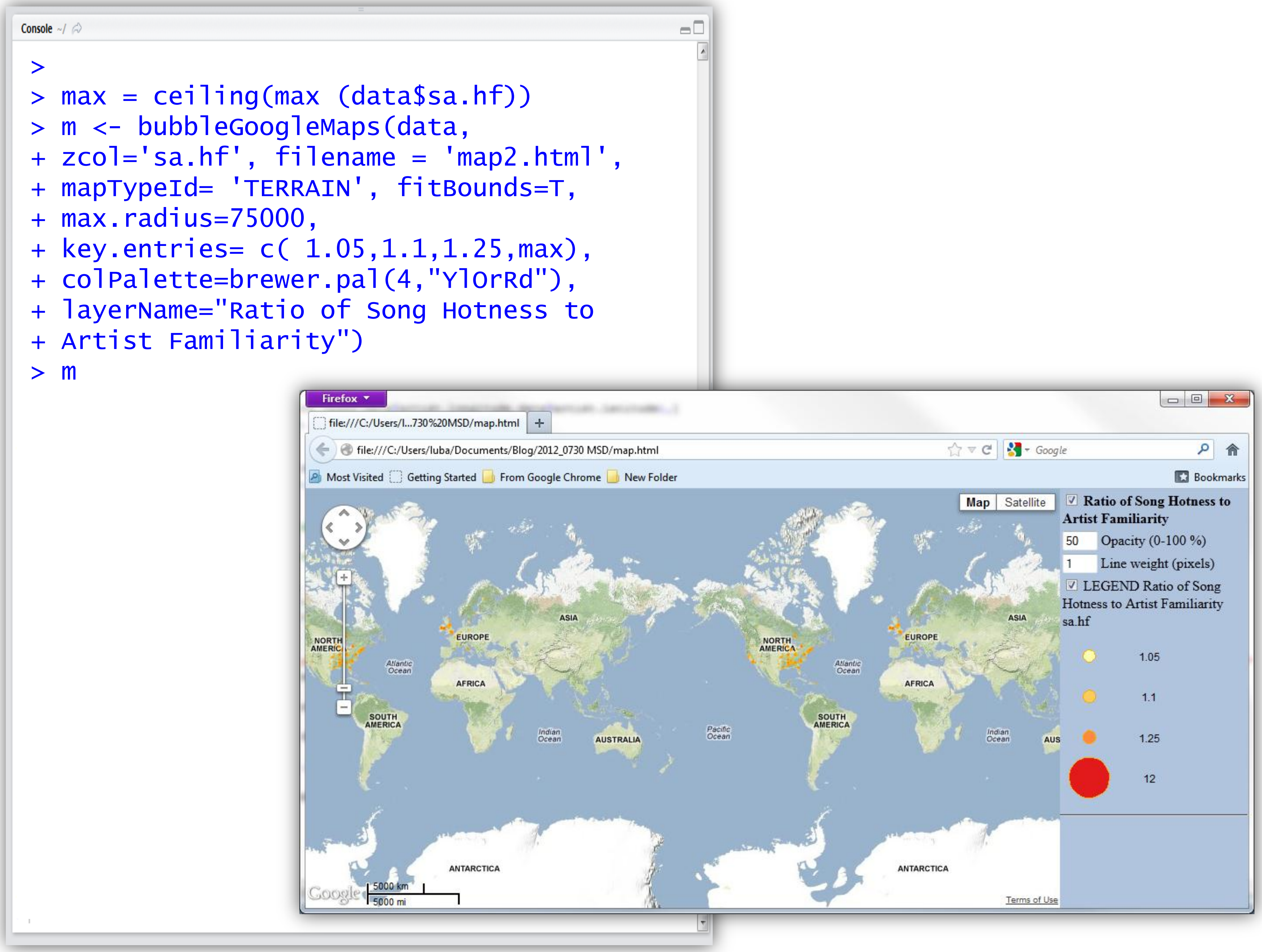
```
>
> code <- sprintf('<br><iframe <-
+ width=\\"200\\" height=\\"150\\"
+ src=\\"http://www.youtube.com/embed?
+ autoplay=1&listType=search&list=%s\\"
+ frameborder=\\"0\\" allowfullscreen>
+ </iframe>',
+ paste(data$artist.name, data$title,
+ sep=" "))
>
> data$youtube=as.character(code)
>
```



YouTube Embedded Players and Player Parameters

Contents

- Overview
- Embedding a YouTube player
  - IFrame embeds using <iframe> tags
  - IFrame embeds using the IFrame Player API
  - AS3 (and AS2*) object embeds
- Selecting content to play
- Parameters
  - autohide
  - autoplay
  - cc_load_policy
  - color
  - controls
  - disablekb
  - enablejsapi
  - end
  - fs
  - iv_load_policy
  - list



```
>
> max = ceiling(max (data$sa.hf))
> m <- bubbleGoogleMaps(data,
+ zcol='sa.hf', filename = 'map2.html',
+ mapTypeId= 'TERRAIN', fitBounds=T,
+ max.radius=75000,
+ key.entries= c( 1.05,1.1,1.25,max),
+ colPalette=brewer.pal(4,"YlOrRd"),
+ layerName="Ratio of Song Hotness to
+ Artist Familiarity")
> m
```

☑ **Ratio of Song Hotness to Artist Familiarity**

50 Opacity (0-100 %)

1 Line weight (pixels)

☑ LEGEND Ratio of Song Hotness to Artist Familiarity
sa.hf

●	1.05
●	1.1
●	1.25
●	12

Resources

plotGoogleMaps Vignette

http://cran.r-project.org/web/packages/plotGoogleMaps/vignettes/plotGoogleMaps-intro.pdf

R-Based Interfaces to Google Maps

http://maths.anu.edu.au/~johnm/r/spatial/googlemaps.pdf

http://www.maths.anu.edu.au/~johnm/wkshp/R/RTour-rc33.R

plotGoogleMaps - A Simple Solution for Geological Survey Web Mapping

http://e-science.amres.ac.rs/TP36035/wp-content/uploads/2012/06/PLOTGOOGLEMAPS_full.pdf

Million Song Dataset

http://labrosa.ee.columbia.edu/millionsong/pages/getting-dataset

Music Machinery Blog - Hottt or Nottt

http://musicmachinery.com/2009/12/09/a-rising-star-or/

Google Dev - YouTube Embedded Player Parameters

https://developers.google.com/youtube/player_parameters