Homework 01

Brown University

DATA 1010

Fall 2020

In [4]:
```
using Plots, Images, ImageMagick, CSV
default(legend = false)
```

Problem 1

The **rank** of a matrix is defined to be the dimension of the span of its columns.

(a) Show that the rank of $\mathbf{u}\mathbf{v}'$ is equal to 1, if $\mathbf{u}$ and $\mathbf{v}$ are any (column) vectors.

Suppose u is a vector of n entries and v is a vector of m entries. uv' produces a matrix of of n*m with the kth column equals $\mathbf{v}_k$ times u. Therefore any two columns i, j in uv' have the ratio $\frac{\mathbf{v}_i}{\mathbf{v}_j}$, which means that columns in uv' are linearly dependent and are all multiples of v. In conclusion, the rank of uv' is 1.

In [5]:
```
# generate 2 5*1 vectors with random entries several times and verify the rank of uv'
using Random, LinearAlgebra
for i in 1:100
    u = rand(5, 1)
    v = rand(5, 1)
    if rank(u * v') != 1
        println("False")
    end
end
println("True")
```

```
True
```

(b) Identify some countries whose flags are rank-1 matrices (where the correspondence between flags and matrices is to associate a flag image with a matrix of pixel values). You can have a look at various flags in the `flags` directory

flags for ru, at, ua are rank-1

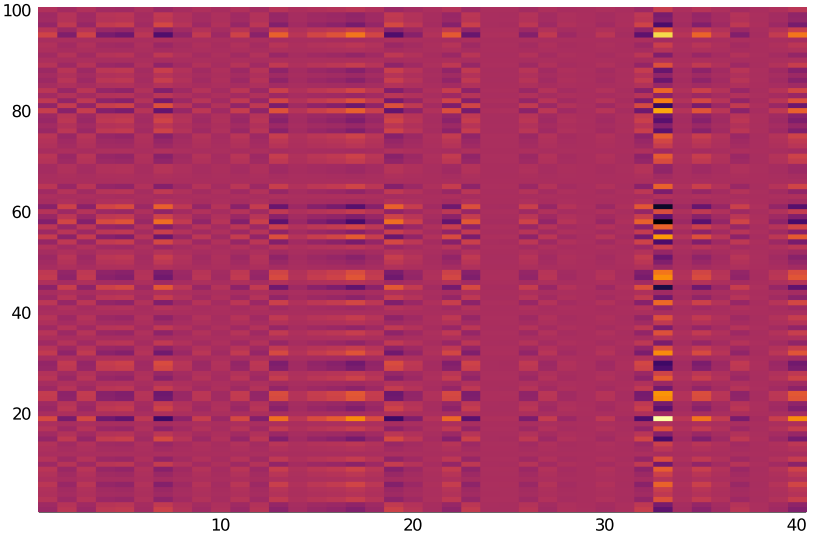
In [11]:
```
function flagrank(countrycode)
    img = RGBA{Normed{UInt8,8}}.(ImageMagick.load("flags/$(countrycode).png"))
    height, width = size(channelview(img))[2:3]
    rank(reshape(channelview(img), (4 * height, width)))
end
# println(flagrank("ru"))
# println(flagrank("at"))
# println(flagrank("mc"))
# println(flagrank("ye"))
# println(flagrank("ua"))
```

1

(c) Run the code below several times to see a handful of rank-1 matrix heatmaps. Describe qualitatively what random rank-1 matrices look like.

In [2]:
```
heatmap(randn(100) * randn(40)')
# The color in rank-1 matrix looks similar, mostly between red and dark blue.
# We can also see columns and rows with same color when we compare entry to
# entry.
```

Out[2]:



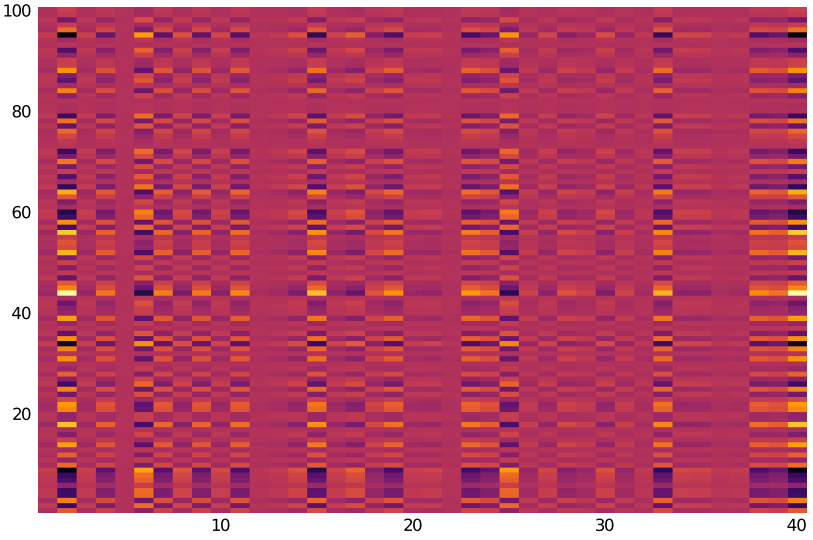
(d) Does the matrix A defined below look like a rank-1 matrix? Is it a rank-1 matrix?

In [4]:
```
A = randn(100) * randn(40)' + 0.001*randn(100, 40);
```

In [10]:
```
println(rank(A))
heatmap(A)
# it is not a rank-1 matrix. rank(A) is 40 This matrix look similar to
# matrix in problem c but its colors are more plentiful than those in
# rank-1 matrix. Because we add a noise in A, so the heatmap is more
# colorful.
```

```
40
```

Out[10]:



(e) Suppose we just had the matrix A, but not the expression that was used to generate it. How could we recover the information that this matrix is very close to a rank-1 matrix?

Running the folloing code, I find that the singular values in rank-1 matrix has only 1 entry that is nonzero while other entries are approximately zero(not zero because of overflow calculations). The singular values in the rank-1 matrix + some noise also have 1 entry that is much larger than the others while other terms are also very close to 0 but not as close as those in rank-1 matrix. Both the singular value matrix have rank 1. Therefore, we recover the information that the this matrix is very close to rank-1 matrix.

In [33]:
```
r1 = randn(100) * randn(40)'
noise = 0.001*randn(100, 40)
r2 = r1 + noise
s,v,d = svd(r1)
s2,v2,d2 = svd(r2)
println("rank of rank-1 matrix: ",rank(v[:,:]), "\nrank of rank-1 matrix plus noise: ",rank(v2[:,:]))
```

```
rank of rank-1 matrix: 1
rank of rank-1 matrix plus noise: 1
```

(f) What is the distribution of country flag image ranks? For consistency, we divide each rank value by the height of the matrix (the first dimension). Make a histogram of the resulting ratios. Which flag elements lead to an especially high rank?

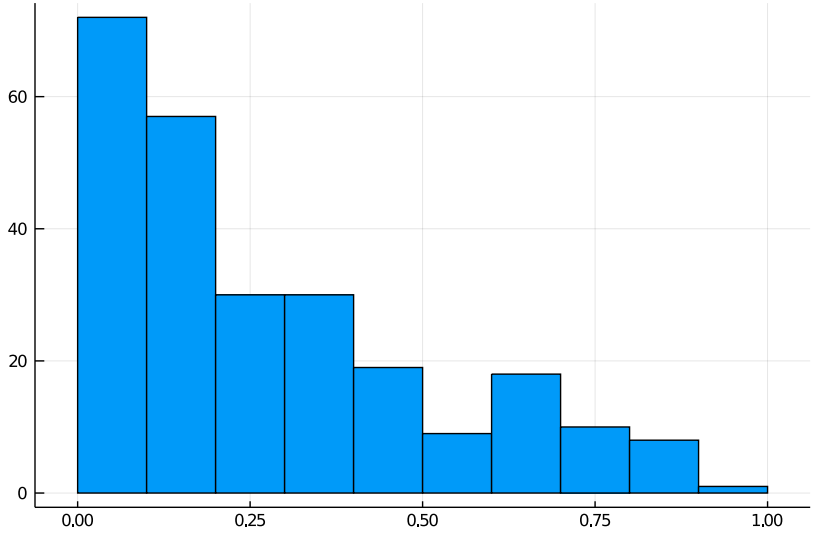
The distribution looks like a right-part of a normal distribution (right skewed) with most of the flags having low ranks and few of them having high ranks. Using the following code, I find out that flag with especially high rank such as "pm" have more colors than those with only one or two different colors. And the pattern of high-rank flag are much more complex than those of simple patterns like "ge". In conclusion, the complexity of patterns and number of different colors contributes proportionally to the rank of a flag.

In [7]:
```
using Plots, LinearAlgebra
country = []
rankratio = []
for filname in readdir("flags")
    code = split(filname, ".")[1]
    ratio = flagrankratio(code)
    push!(country, code)
    push!(rankratio, ratio)
end
```

In [11]:
```
maximum(rankratio)
println(country[argmax(rankratio)])
histogram(rankratio, bins=15)
```

```
pm
```

Out[11]:



Note: you should expect this computation to take a very long time to run. Also, every cell in the rest of this problem statement is there to help you; there are no further prompts.

In [6]:
```
"""
Compute the rank of the flag of the country with code `countrycode`, interpreting
each RGB triple as a 3×1 matrix block.
"""
function flagrankratio(countrycode)
    img = RGBA{Normed{UInt8,8}}.(ImageMagick.load("flags/$(countrycode).png"))
    height, width = size(channelview(img))[2:3]
    rank(reshape(channelview(img), (4 * height, width))) / size(img, 1)
end
```

Out[6]: flagrankratio

For example, the rank of the US flag image:

In [15]:
```
flagrankratio("us")
```

Out[15]: 0.06978470675575353

If we want to take a look at a particular flag, we can just load it:

In [7]:
```
ImageMagick.load("flags/us.png")
```

Out[7]:

Problem 2

Show that if $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ is linearly independent and $\{\mathbf{v}_1 + \mathbf{w}, \mathbf{v}_2 \ldots, \mathbf{v}_n\}$ is linearly dependent, then $\mathbf{w}$ is in the span of $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$.

Since $\{\mathbf{v}_1 + \mathbf{w}, \mathbf{v}_2 \ldots \mathbf{v}_n\}$ is linearly independent, we have $\mathbf{a}_1(\mathbf{v}_1 + \mathbf{w}) = \mathbf{a}_2\mathbf{v}_2 + \ldots + \mathbf{a}_n\mathbf{v}_n$ with $\mathbf{a}_1, \mathbf{a}_2 \ldots \mathbf{a}_n$ nontrivial. Therefore, $\mathbf{w} = \mathbf{v}_1 + \frac{\mathbf{a}_2}{\mathbf{a}_1} * \mathbf{v}_2 + \ldots + \frac{\mathbf{a}_n}{\mathbf{a}_1} * \mathbf{v}_n$ and $\mathbf{w}$ is in the span of $\{\mathbf{v}_1, \mathbf{v}_2 \ldots \mathbf{v}_n\}$ since $\frac{\mathbf{a}_2}{\mathbf{a}_1} \ldots \frac{\mathbf{a}_n}{\mathbf{a}_1}$ are non trivial.

Problem 3

Suppose that A is a square matrix each of whose rows has the same sum s. Show that s is an eigenvalue of A. Hint: don't use determinants; leverage the constant row sum to search for an eigenvector directly.

Note: as an example, the matrix

$$A = \begin{bmatrix} 2 & -2 & 4 \\ 1 & 1 & 2 \\ 0 & 3 & 1 \end{bmatrix}$$

has a constant row sum of 4.

Suppose v is a vector with entries that are same, Since every row of A has the same sum s, then A *v* = *s* v. Therefore, s is an eigenvalue of A with eigenvector v.

Problem 4

Consider the function $f(x, y) = xe^{-x^2+y} + \log\left(\frac{x}{1+y}\right)$.

(a) Find ∇f.

Gradient: [$e^{-x^2+y} - 2x^2e^{-x^2+y} + \frac{1}{x}, xe^{-x^2+y} - \frac{1}{1+y}$).

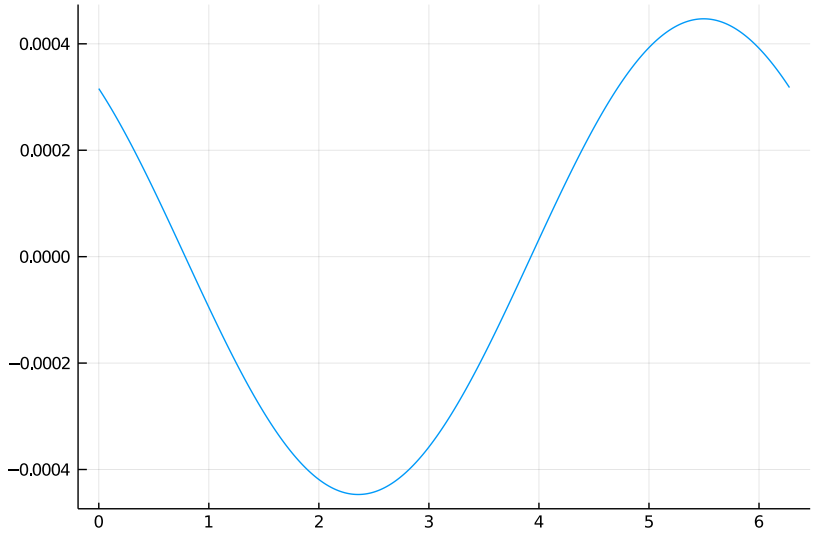
(b) Numerically identify f's direction of maximum increase at the point $(1, 0)$ by evaluating $f([1, 0] + \epsilon[\cos(\theta), \sin(\theta)])$ for some small value of ϵ for many equally-spaced θ values in the range $[0, 2\pi)$ and identifying the ones that produce the largest and smallest output values.

Hints: You want to make a plot showing the relationship between θ and the resulting value output by the function. Comment on the shape of the resulting graph; is the shape universal or idiosyncratic to this function? You also want to also investigate a few values of ϵ, to eludicate its role in the problem.

In [5]:
```
num = 1000
steps = [2*pi / num * i for i in 0:(num-1)]
f(x,y) = x*exp(-x^2+y) + log(x/(1+y))
e = 0.0005
points = [(cos(step), sin(step)) for step in steps]
output = [f(1 + e*x, e*y) - f(1,0) for (x, y) in points]
println(steps[argmin(output)])
println(steps[argmax(output)])
println(points[argmin(output)])
println(points[argmax(output)])
plot(steps, output)
```

```
2.356194490192345
5.497787143782138
(-0.7071067811865475, 0.7071067811865476)
(0.7071067811865474, -0.7071067811865477)
```

Out[5]:



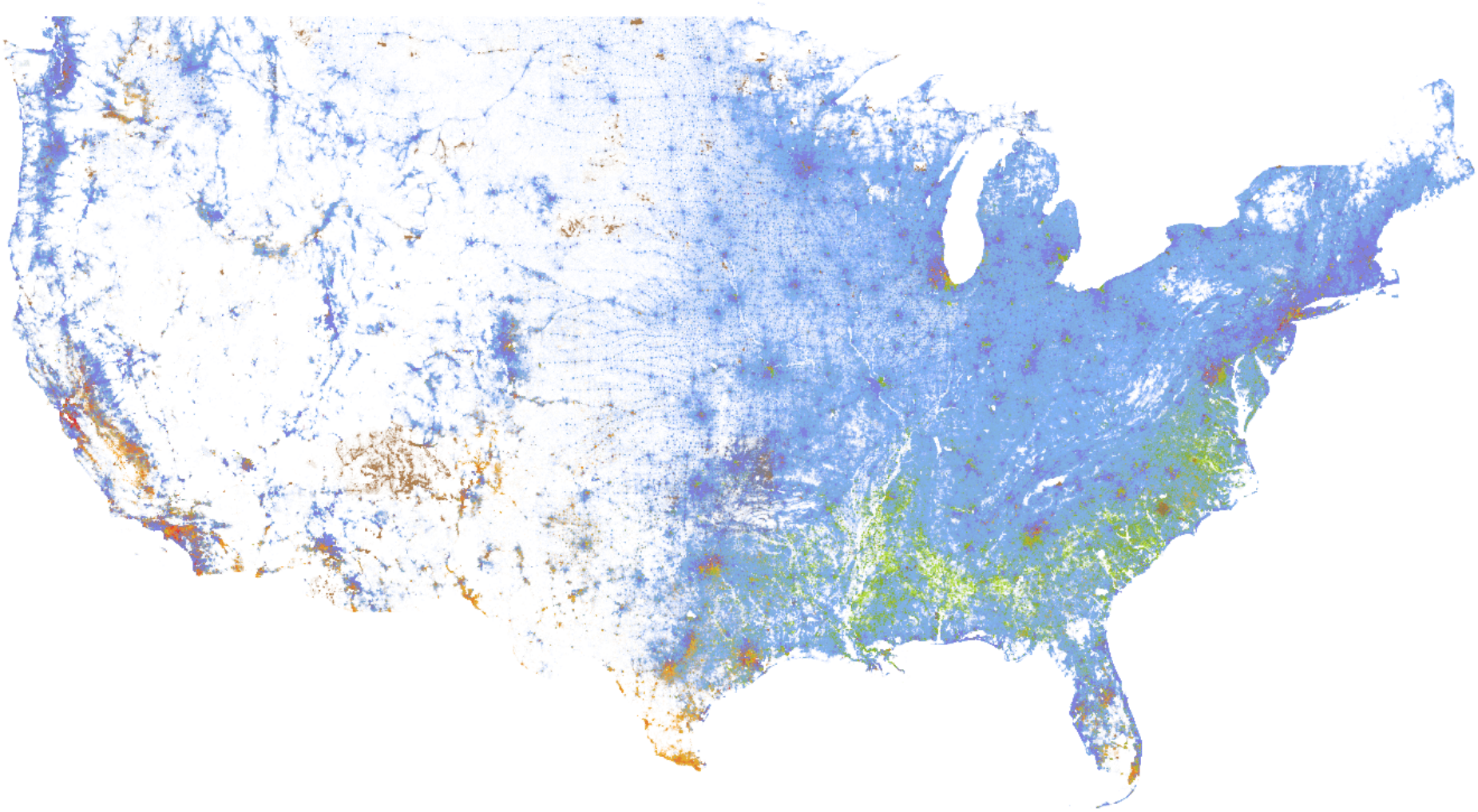
(c) Discuss the relationship between parts (a) and (b).

At point [1,0], ∇f is $[-e^{-1}+1, e^{-1}-1]$ It means the direction of the largest increase and the opposite of it is the direction of the largest decrease. It matches with what we find in (b):(-0.7071067811865475, 0.7071067811865476) for smallest, (0.7071067811865474, -0.7071067811865477) for largest. They represent the same directions

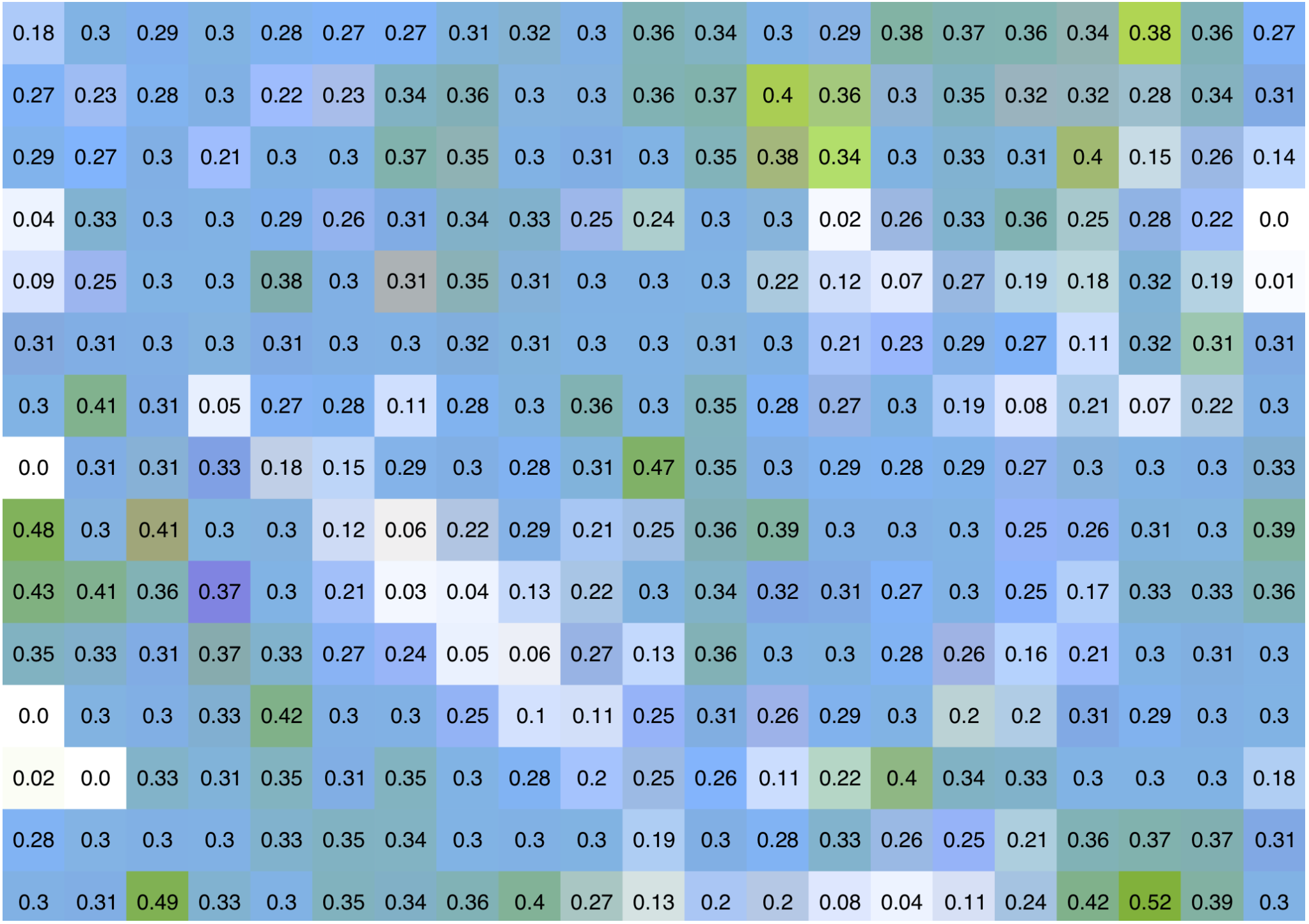
Problem 5

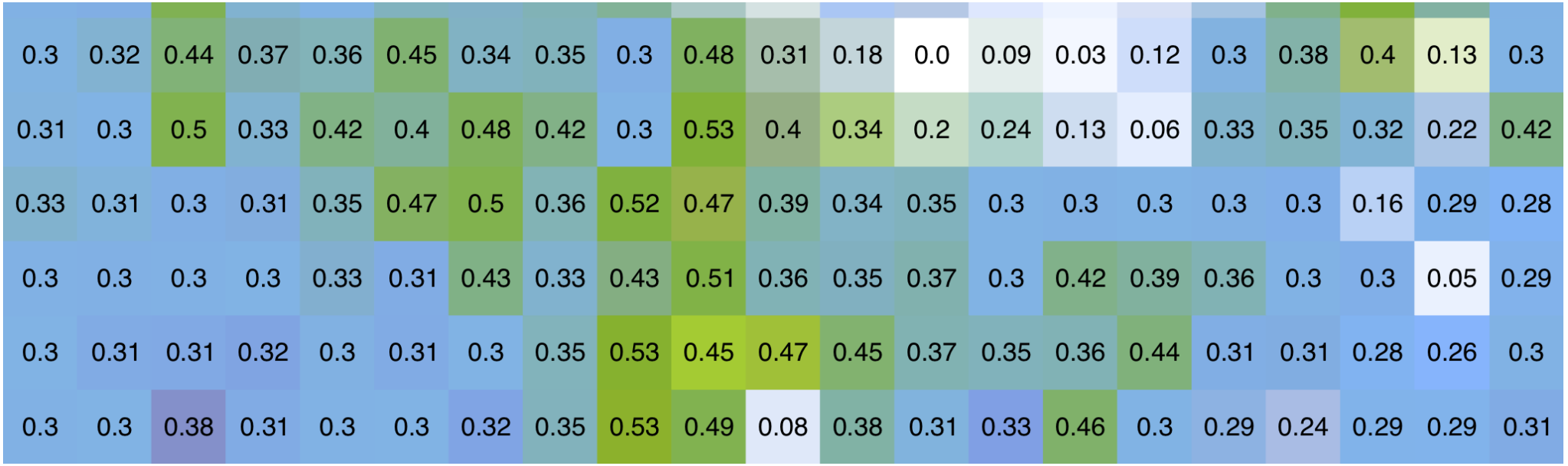
In this problem, we will explore an important multivariable calculus idea which was not emphasized in the multivariable calculus review day.

The map below was obtained by placing one dot at the residence of each person in the contiguous United States in the 2010 US Census (with color indicating race identification) and then representing each of many small squares in a grid with the average of the colors of the dots in the square.



Suppose you're given the 764×1366 matrix which represent how darkly colored in each little square is (see the figure to the right for a small piece of this grid). These numbers are roughly proportional to the number of people living in each square. Here's a zoomed-in view of this matrix:


0.18 0.3 0.29 0.3 0.28 0.27 0.27 0.31 0.32 0.3 0.36 0.34 0.3 0.29 0.38 0.37 0.36 0.34 0.38 0.36 0.27
0.27 0.23 0.28 0.3 0.22 0.23 0.34 0.36 0.3 0.3 0.36 0.37 0.4 0.36 0.3 0.35 0.32 0.32 0.28 0.34 0.31
0.29 0.27 0.3 0.21 0.3 0.3 0.37 0.35 0.3 0.31 0.3 0.35 0.38 0.34 0.3 0.33 0.31 0.4 0.15 0.26 0.14
0.04 0.33 0.3 0.3 0.29 0.26 0.31 0.34 0.33 0.25 0.24 0.3 0.3 0.02 0.26 0.33 0.36 0.25 0.28 0.22 0.0
0.09 0.25 0.3 0.3 0.38 0.3 0.31 0.35 0.31 0.3 0.3 0.3 0.22 0.12 0.07 0.27 0.19 0.18 0.32 0.19 0.01
0.31 0.31 0.3 0.3 0.31 0.3 0.3 0.32 0.31 0.3 0.3 0.31 0.3 0.21 0.23 0.29 0.27 0.11 0.32 0.31 0.31
0.3 0.41 0.31 0.05 0.27 0.28 0.11 0.28 0.3 0.36 0.3 0.35 0.28 0.27 0.3 0.19 0.08 0.21 0.07 0.22 0.3
0.0 0.31 0.31 0.33 0.18 0.15 0.29 0.3 0.28 0.31 0.47 0.35 0.3 0.29 0.28 0.29 0.27 0.3 0.3 0.3 0.33
0.48 0.3 0.41 0.3 0.3 0.12 0.06 0.22 0.29 0.21 0.25 0.36 0.39 0.3 0.3 0.3 0.25 0.26 0.31 0.3 0.39
0.43 0.41 0.36 0.37 0.3 0.21 0.03 0.04 0.13 0.22 0.3 0.34 0.32 0.31 0.27 0.3 0.25 0.17 0.33 0.33 0.36
0.35 0.33 0.31 0.37 0.33 0.27 0.24 0.05 0.06 0.27 0.13 0.36 0.3 0.3 0.28 0.26 0.16 0.21 0.3 0.31 0.3
0.0 0.3 0.3 0.33 0.42 0.3 0.3 0.25 0.1 0.11 0.25 0.31 0.26 0.29 0.3 0.2 0.2 0.31 0.29 0.3 0.3
0.02 0.0 0.33 0.31 0.35 0.31 0.35 0.3 0.28 0.2 0.25 0.26 0.11 0.22 0.4 0.34 0.33 0.3 0.3 0.3 0.18
0.28 0.3 0.3 0.3 0.33 0.35 0.34 0.3 0.3 0.3 0.19 0.3 0.28 0.33 0.26 0.25 0.21 0.36 0.37 0.37 0.31
0.3 0.31 0.49 0.33 0.3 0.35 0.34 0.36 0.4 0.27 0.13 0.2 0.2 0.08 0.04 0.11 0.24 0.42 0.52 0.39 0.3



Using these data, approximate the population-weighted center of mass of the contiguous United States. Feel free to answer in terms of the matrix dimensions (e.g., how many pixels down and how many to the right of the top left corner of the matrix).

In [64]:
```
pixeldensities = Matrix(CSV.read("US_Matrix.csv"))
# Gray.(pixeldensities)
allone = findall(x->x>0, pixeldensities)
zero = findall(x->x==0, pixeldensities)
existing_value = [pixeldensities[i] for i in allone]
mass = sum(existing_value)
x_bar = 0
y_bar = 0
for i in allone
    x_bar = x_bar + pixeldensities[i] * i[2]
    y_bar = y_bar + pixeldensities[i] * i[1]
end
xe = x_bar / mass
ye = y_bar / mass
xe, ye
# the center of mass locate at (772.4303338084652, 359.5156757258744)
# which is 772th square horizontally and 359th square veritcally to
# the top left corner of the matrix
```

Out[64]:
```
(772.4303338084652, 359.5156757258744)
```

In [ ]:
```

```