k15-2.png	39 minutes ago
k15-3.png	39 minutes ago
k5-1.png	39 minutes ago
k5-2.png	39 minutes ago
k5-3.png	39 minutes ago
k7-1.png	39 minutes ago
k7-2.png	39 minutes ago
k7-3.png	39 minutes ago
q3-1.png	27 minutes ago
q3-2.png	27 minutes ago

BIS634 Yuechen Liu HW4

Exercise 1 (25 points):

1) Implement a two-dimensional version of the gradient descent algorithm to find optimal choices of a and b. (7 points)

Answer: Let the error function be L (loss function) and parameter θ:[a, b] The goal is to find an optimal set of θ parameters to minimize L, as follows: θ^*=arg(min)L(θ) In this task, we do not need to care about the input and output. We can directly obtain loss through Request, so we only need to adjust the parameter gradient. Let the initial parameter be θ(0):[a(0), b(0)]. As a result, I implemented the gradient descent funtion. It takes the f function, gradf funtion, and our inital guess of t, the step size alpha, and the EPS (Epsilon is a very small number to prevent any division by zero in the implementation. Citation: https://www.cs.toronto.edu/~guerzhoy/411/lec/W02/python/graddescent2d.html

2) Explain how you estimate the gradient given that you cannot directly compute the derivative (3 points)

Answer: Without knowing the exact function, we know the changing trend between parameters and loss, and we can calculate the gradient by calculating the derivative of loss and parameters (a, b). Loss decreases faster in the opposite direction of the gradient.

3) Identify any numerical choices -- including but not limited to stopping criteria -- you made (3 points), and justify why you think they were reasonable choices (3 points).

Answer: In this task, when the number of iterations is less than the maximum 1000, the training is stopped. Another option is to check if the loss continues to decrease or the gradient changes near 0. The reasons for choosing these two methods are as follows: When the number of iterations is enough, the probability of obtaining the local or global minimum value of loss function by gradient descent method is increased and stops when the number of max iterations is reached. In the case of unfixed iteration steps, when iteration reaches the global or local minimum value, the gradient continues to fluctuate near the minimum value due to the existence of learning rate, and loss does not continue to decrease, so it stops.

4) It so happens that this error function has a local minimum and a global minimum. Find both locations (i.e. a, b values) querying the API as needed (5 points) and identify which corresponds to which (2 point).

Answer: After training, we can find that the local minimum is 1.10000049995 when a = 0.2155, b = 0.6885; the global minimum is 1.00000149999 when a =0.7115, b = 0.1685.

5) Briefly discuss how you would have tested for local vs global minima if you had not known how many minima there were. (2 points)

Answer:

1. If I had not known how many minima there were, I would try different corners and center, just like I did; I chose (0.99,0.01),(0.01,0.99)(0.01,0.01),(0.99,0.99), and (0.5,0.5) to test. Initializing multiple models with multiple different parameters is to starting the search from multiple different initial points, which may fall into different local minima, from which selection may obtain a result closer to the global minimum.
2. Simulated annealing accepts worse results than the current solution with some probability at each step, thus contributing to the jump out of the local minimum. In the iterative process of each step, the probability of accepting the "suboptimal solution" should gradually decrease with the passage of time, so as to ensure the stability of the algorithm.
3. Use random gradient descent. Different from the standard gradient descent method, the stochastic factor is added into the gradient calculation. Thus, even if it falls into the local minimum, its calculated gradient may not be zero, giving it a chance to jump out of the local minimum and continue the search.

Exercise 2 (25 points):

Readme.md

work with our dataset (5 points).

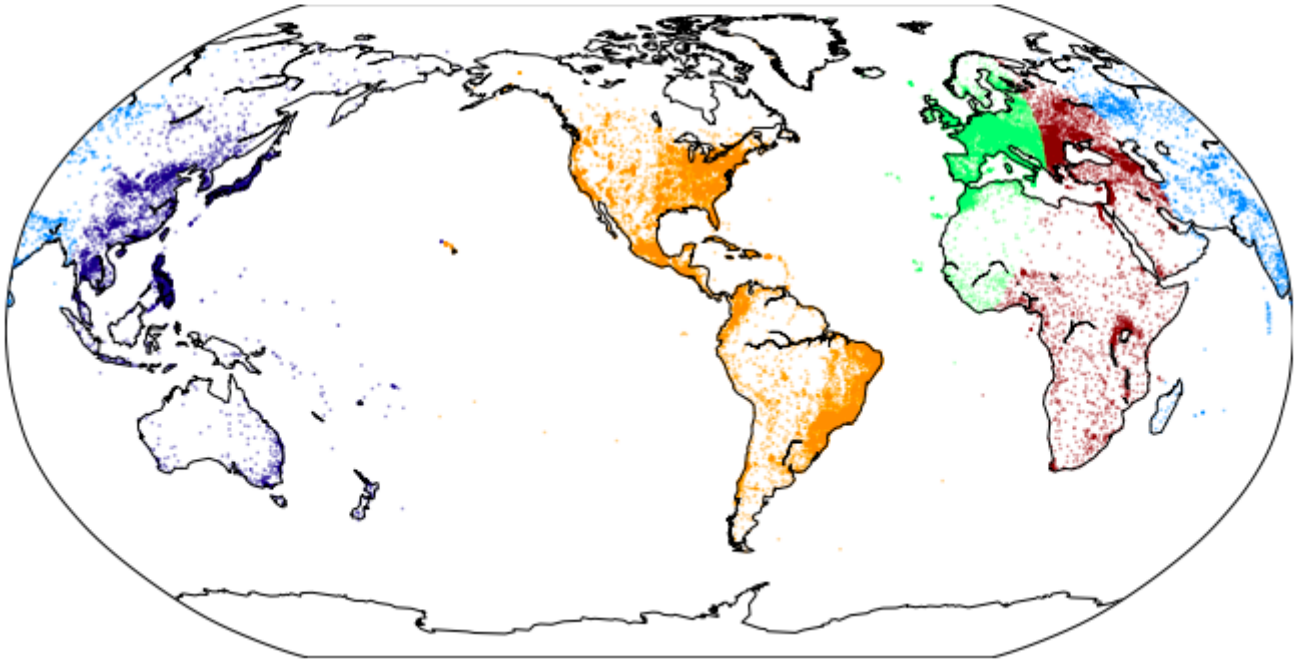
Answer: I modified k-means code by removing all normalization since we need the accurate numbers; besides, I added the Haversine function from https://stackoverflow.com/questions/4913349/haversine-formula-in-python-bearing-and-distance-between-two-gps-points

2,3,4) Visualize your results with a color-coded scatter plot (5 points); be sure to use an appropriate map projection (i.e. do not simply make x=longitude and y=latitude; 5 points). Use this algorithm to cluster the cities data for k=5, 7, and 15. Run it several times to get a sense of the variation of clusters for each k (share your plots) (5 points)

Answer: I run k =5, k =7, and k=15, each for 3 times to see the difference, and I use Basemap projection. Citation: https://matplotlib.org/basemap/users/robin.html

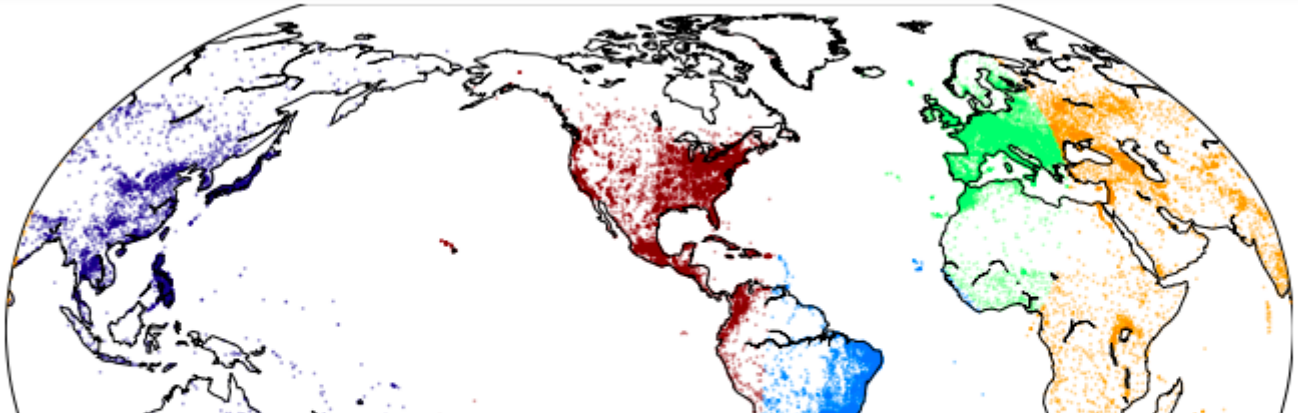
```
centers(df,5)#map1
```

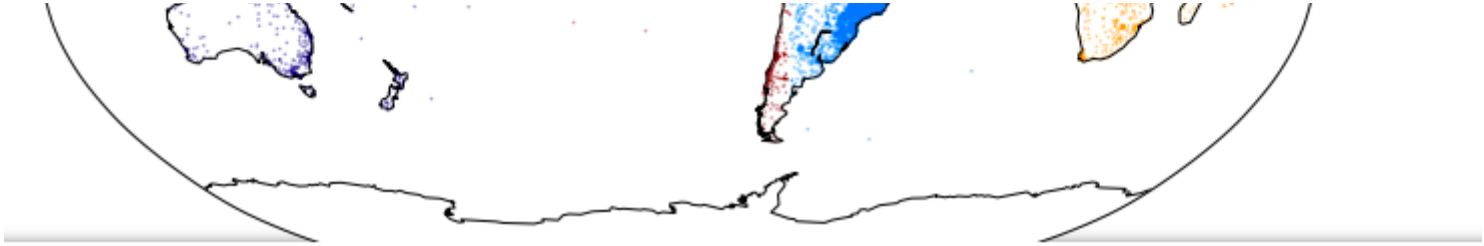
```
<ipython-input-311-b23bdc214fbc>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
-a-view-versus-a-copy
```



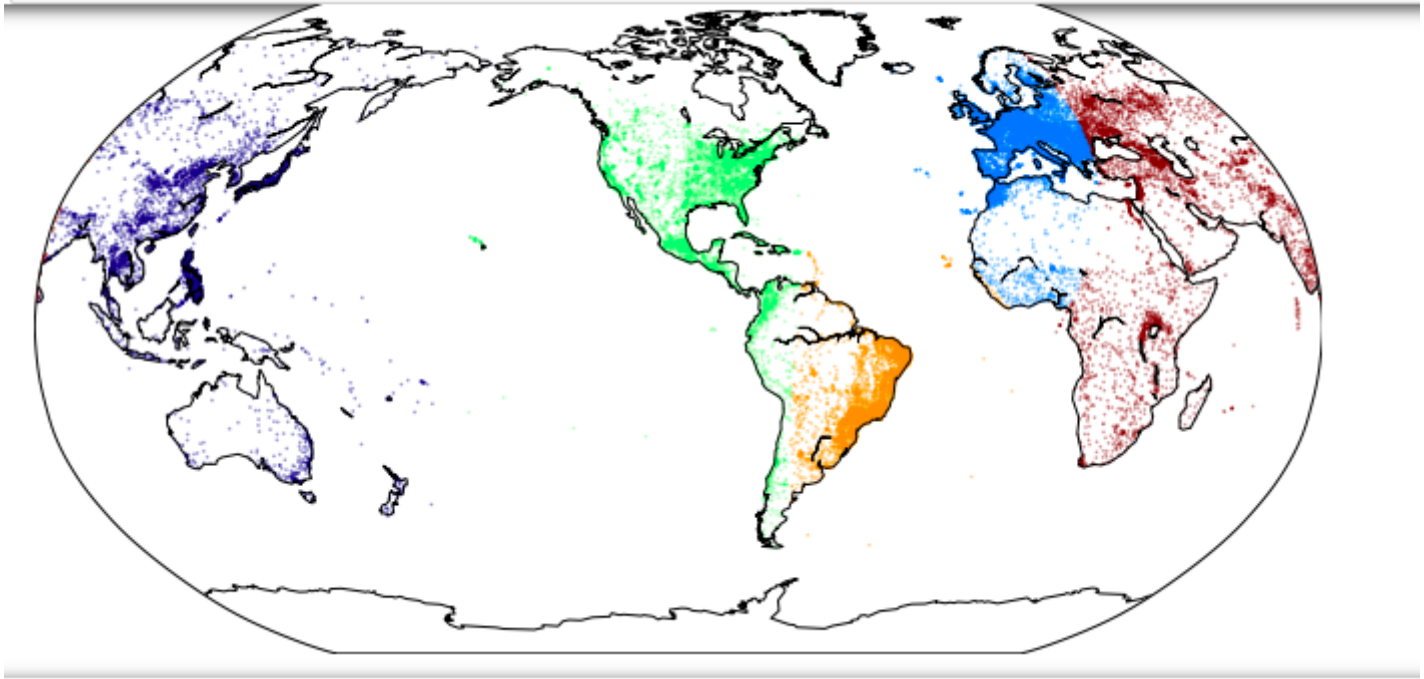
When k =5

```
centers(df,5)#map2
```

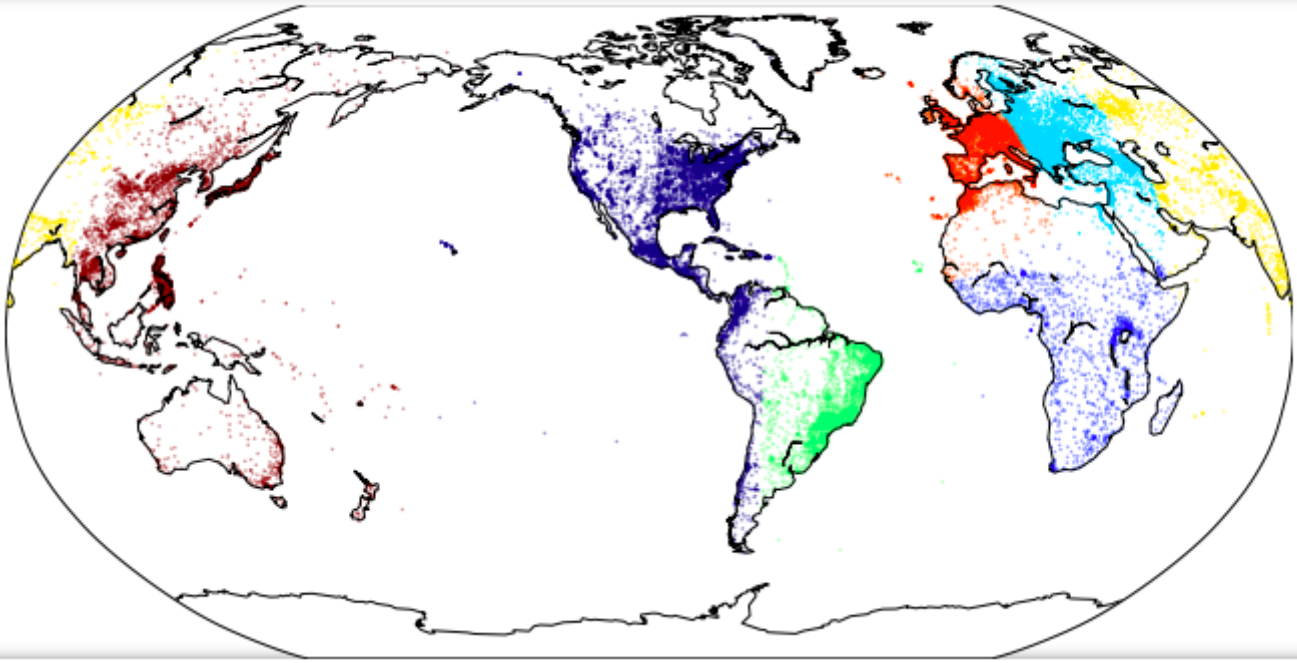




```
centers(df,5)#map3
```



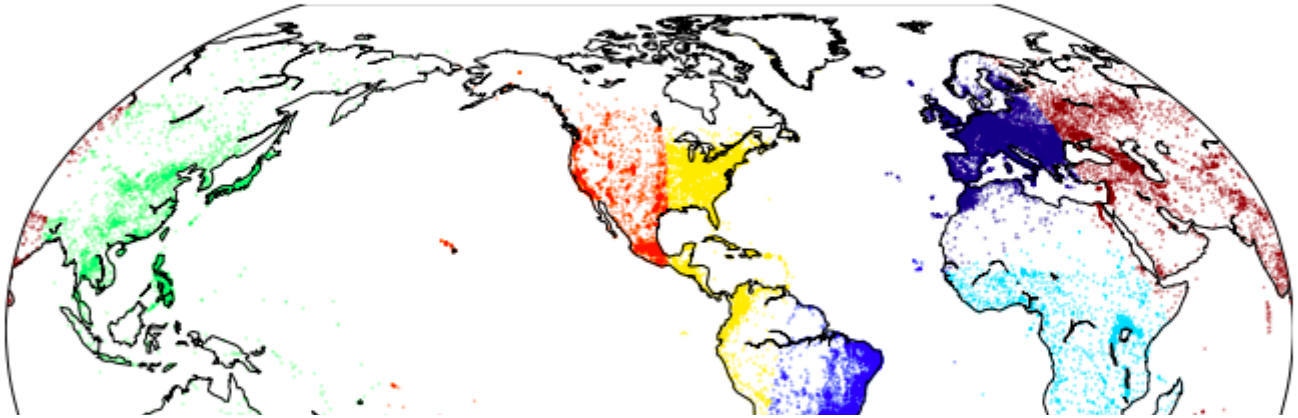
```
centers(df,7)#map4
```

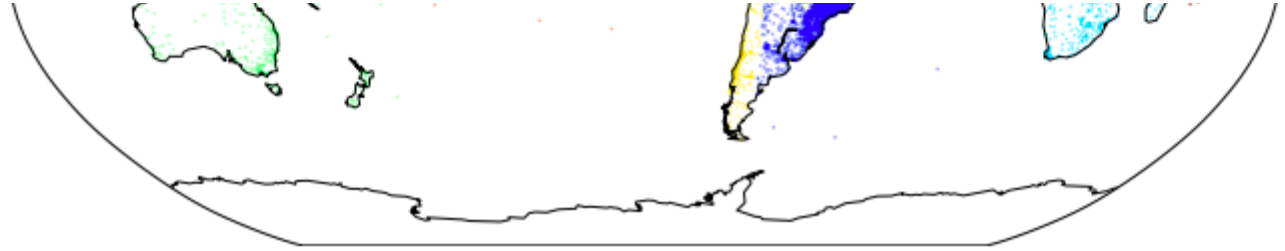


When k = 7

```
centers(df,7)#map5
```

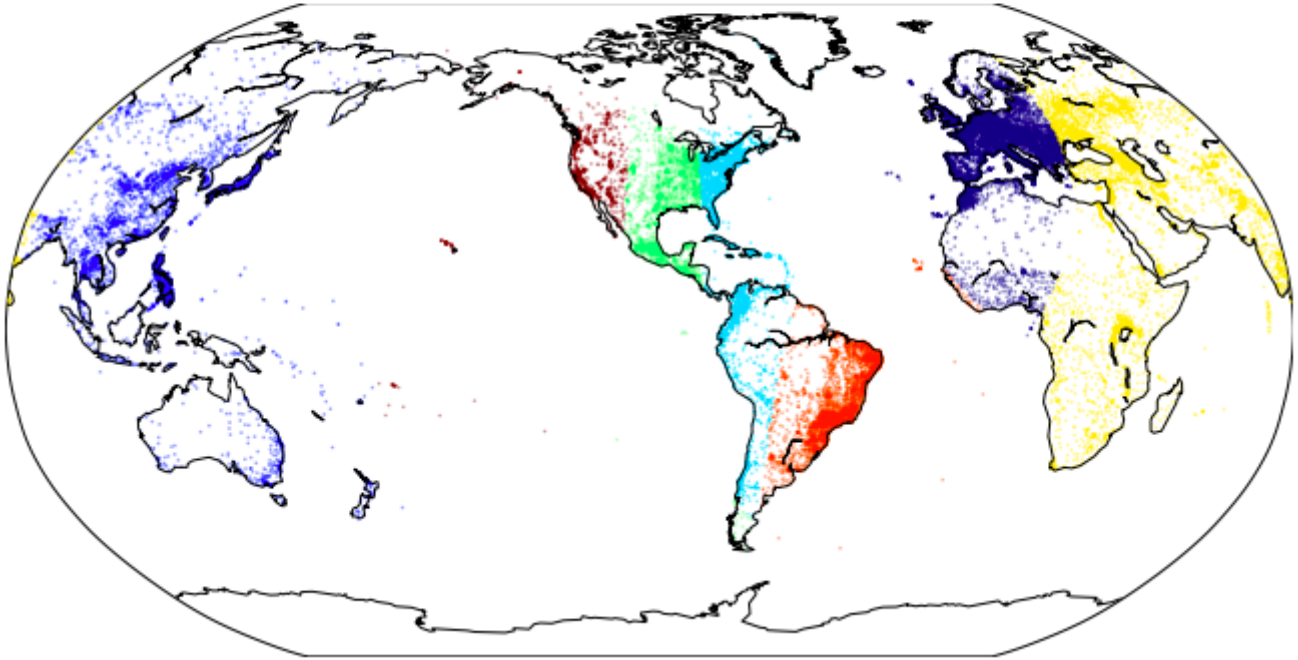
```
<ipython-input-311-b23bdc214fbc>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/use
-a-view-versus-a-copy
```



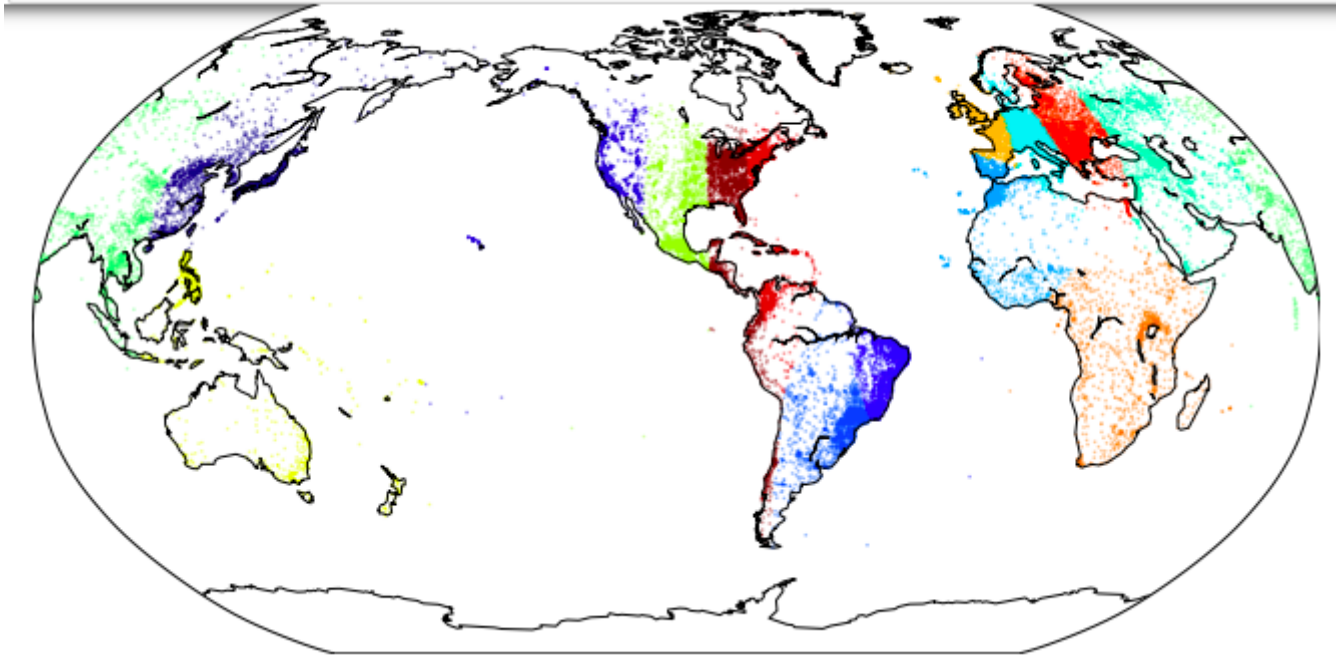


```
centers(df,7)#map6
```

```
<ipython-input-311-b23bdc214fbc>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
-a-view-versus-a-copy
```



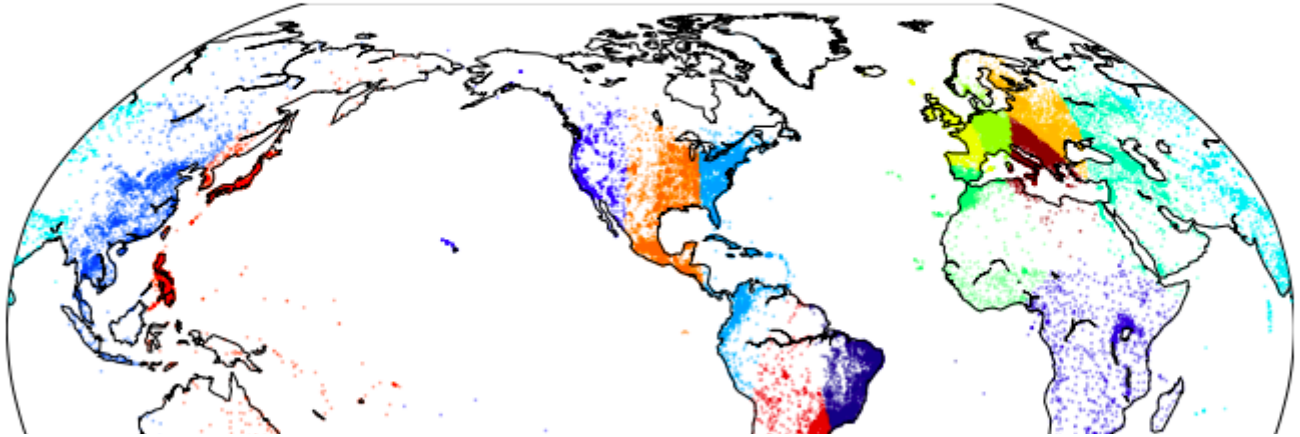
```
centers(df,15)#map7
```

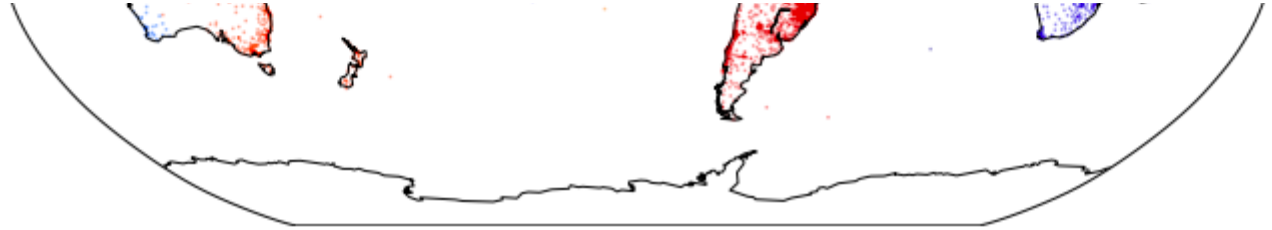


When k = 15

```
centers(df,15)#map8
```

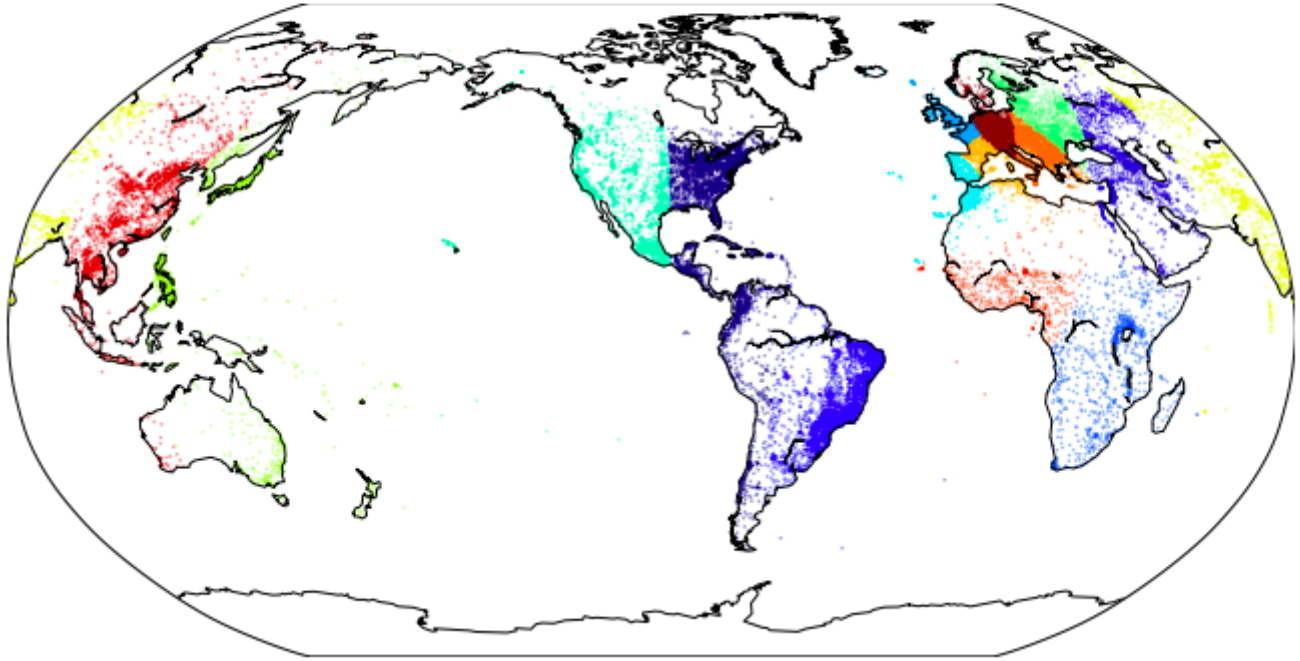
```
<ipython-input-311-b23bdc214fbc>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/use
-a-view-versus-a-copy
```





```
centers(df,15)#map9
```

```
<ipython-input-311-b23bdc214fbc>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/use
-a-view-versus-a-copy
```



5) Comment briefly on the diversity of results for each k. (5 points)

Answer:

1. I run k = 5 three times. As showing in the map, the division of cities in Asia and Europe change a lot(map 1 has four divisions in this two areas; map 2 and 3 has three divisions). Besides, South America and North America are separated in the map 2&3.
2. I run k =7 three times. As showing in the map, the clusters changing is focusing on the divisions of Europe and North America(map4 has different divisions of Europed with map5&6; the division of North America is different in all three maps).
3. I run k = 15 three times. As showing in the map, the diversity of centers is varied in the three maps. For map7, there are divisions in North America, whereas in map 8&9, the separation in North America decreased, similarly to South America and Africa; Oceania and South Asia have more separation of cities.

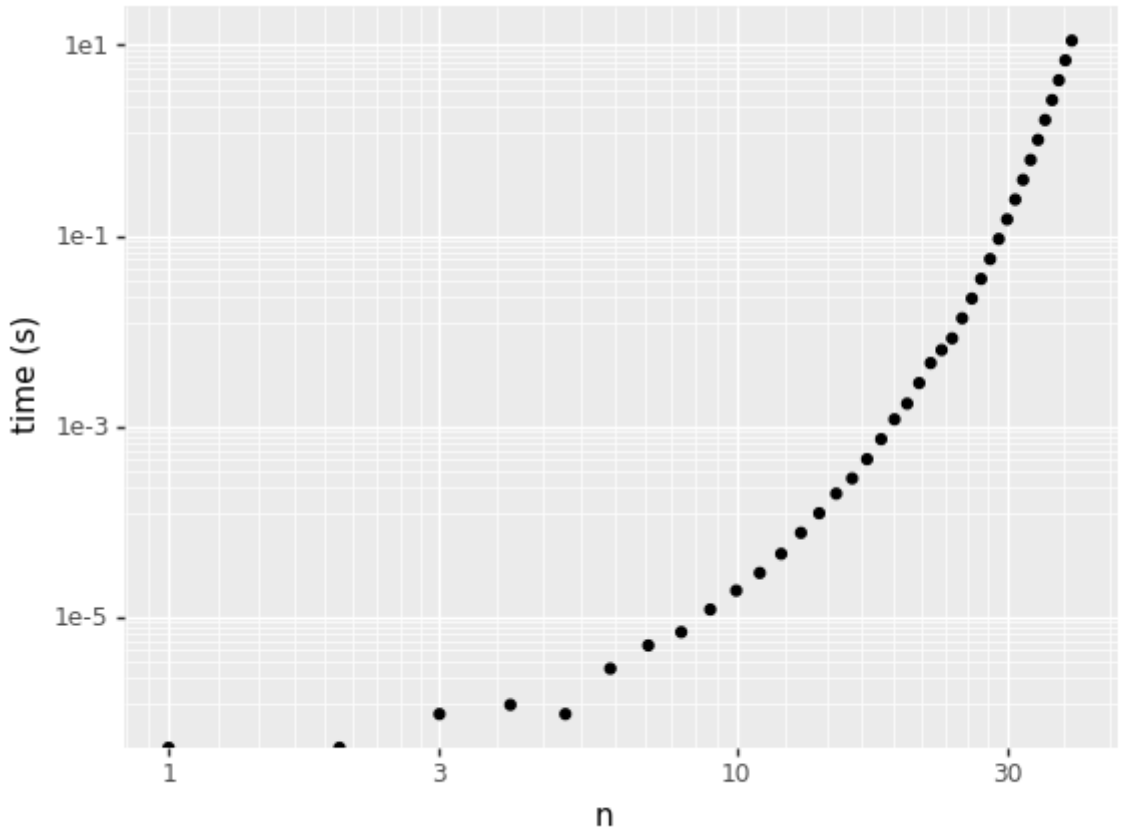
Exercise 3 (25 points):

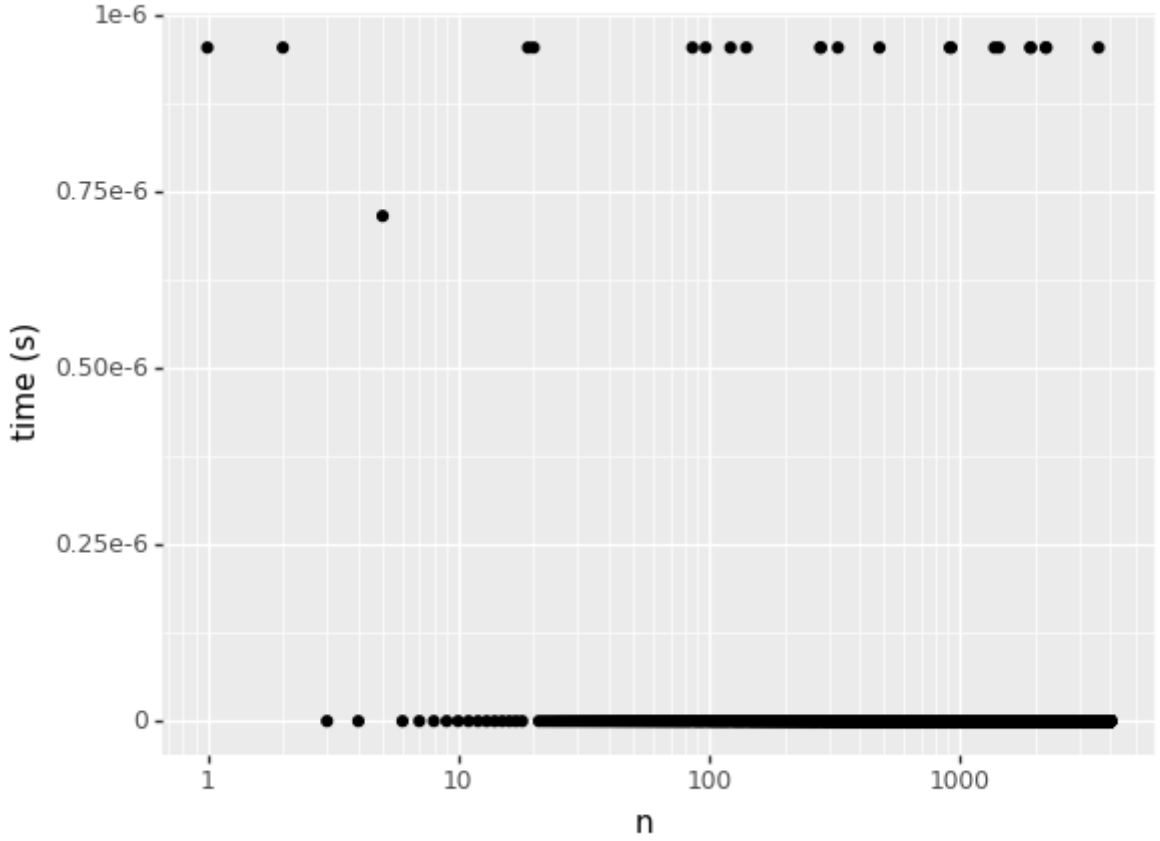
1) Implement both (yes, I know, I gave you implementations on the slides, but try to do this exercise from scratch as much as possible)

Answer: I modified the original function by taking out the time.time function, so it won't need to rewrite and can be directly used.

2,3,4,5) Time them as functions of n (5 points), and display this in the way you think is best (5 points). Discuss your choices (e.g. why those n and why you're displaying it that way; 5 points) and your results (5 points).

Answer: This is how I display the two functions.





I display the functions by setting x axis as n, y axis as time, to better displaying the time consumption. I use time.time to calculate the time consuming for each function. In the first function, I set n = 40, and we can see in the first graph, the time consuming is increasing. The time consuming is 2.25s/it in the first function. When I use lru_cache in the second function, according to the graph, we can see that the time consuming is approximate to 0(some outliers, but they are to small to ignore), and the time consuming is 192899.78it/s. As a result, we can conclude that using lru_cache is much more efficient. The reason that using lru_cache more efficient is that there are generally memory items that are added once and never used again, and there are items that are added and used frequently. LRU is much more likely to keep the frequently-used items in memory. Citation: https://stackoverflow.com/questions/2058403/why-is-lru-better-than-fifo

Exercise 4 (25 points)

Implement a function that takes two strings and returns an optimal local alignment (6 points) and score (6 points) using the Smith-Waterman algorithm; insert "-" as needed to indicate a gap (this is part of the alignment points). Your function should also take and correctly use three keyword arguments with defaults as follows: match=1, gap_penalty=1, mismatch_penalty=1 (6 points). Here, that is a penalty of one will be applied to match scores for each missing or changed letter.Test it, and explain how your tests show that your function works. Be sure to test other values of match, gap_penalty, and mismatch_penalty (7 points).

Answer: I first create a function called 'best_matrix' to create the matrix based on the two sequences. The function includes the matrix with the max score, max score position, and the matrix with socre. Citation: https://tiefenauer.github.io/blog/smith-waterman/ Next, I create a function called 'go', to actually show how to move in the matrix by recording the position in the matrix. If "diagonal", means match; if "up", means a gap in sequence b; if "left", means a gap in sequence a. Citation: https://stackoverflow.com/questions/23400317/smith-waterman-algorithm-to-generate-matrix-in-python Besides, I create a function called 'traceback' to decide which way to traceback in the matrix. It has three conditions, which are left, up, and diagonal. Align is inserted "-" as needed to indicate a gap. Citation: https://stackoverflow.com/questions/12666494/how-do-i-decide-which-way-to-backtrack-in-the-smith-waterman-algorithm

Finally, I create a function called 'align' to combine the three functions and get the final align.

To test if functions are correct, I compared with the official website: http://rna.informatik.uni-freiburg.de/Teaching/index.jsp?toolName=Smith-Waterman and the answer is the same. Say, when match=1, gap_penalty=1, mismatch_penalty=1 Mine:

```
align('tgcatcgagaccctacgtgac', 'actagacctagcatcgac', 1, 1, 1)
```

```
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0]
 [0 0 1 0 0 0 0 1 1 0 0 0 2 1 0 1 0 0 1]
 [0 1 0 0 1 0 1 0 0 0 1 0 1 3 2 1 0 1 0]
 [0 0 0 1 0 0 0 0 0 1 0 0 0 2 4 3 2 1 0]
 [0 0 1 0 0 0 0 1 1 0 0 0 1 1 3 5 4 3 2]
 [0 0 0 0 0 1 0 0 0 0 0 1 0 0 2 4 6 5 4]
 [0 1 0 0 1 0 2 1 0 0 1 0 0 1 1 3 5 7 6]
 [0 0 0 0 0 2 1 1 0 0 0 2 1 0 0 2 4 6 6]
 [0 1 0 0 1 1 3 2 1 0 1 1 1 2 1 1 3 5 5]
 [0 0 2 1 0 0 2 4 3 2 1 0 2 1 1 2 2 4 6]
 [0 0 1 1 0 0 1 3 5 4 3 2 1 1 0 2 1 3 5]
 [0 0 1 0 0 0 0 2 4 4 3 2 3 2 1 1 1 2 4]
 [0 0 0 2 1 0 0 1 3 5 4 3 2 2 3 2 1 1 3]
 [0 1 0 1 3 2 1 0 2 4 6 5 4 3 2 2 1 2 2]
 [0 0 2 1 2 2 1 2 1 3 5 5 6 5 4 3 2 1 3]
 [0 0 1 1 1 3 2 1 1 2 4 6 5 5 4 3 4 3 2]
 [0 0 0 2 1 2 2 1 0 2 3 5 5 4 6 5 4 3 2]
 [0 0 0 1 1 2 1 1 0 1 2 4 4 4 5 5 6 5 4]
 [0 1 0 0 2 1 3 2 1 0 2 3 3 5 4 4 5 7 6]
 [0 0 2 1 1 1 2 4 3 2 1 2 4 4 4 5 4 6 8]]
seq1 =  agacccta-cgt-gac , seq2 =  agacc-tagcatcgac , score = 8
```

Website(cannot see the complete matrix in the screenshot, but can see the result):

C_2	T_3	A_4	G_5	A_6	C_7	C_8	T_9	A_{10}	G_{11}	C_{12}	A_{13}	T_{14}	C_{15}	G_{16}	A_{17}	C_{18}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
1	0	0	0	0	1	1	0	0	0	2	1	0	1	0	0	1
0	0	1	0	1	0	0	0	1	0	1	3	2	1	0	1	0
0	1	0	0	0	0	0	1	0	0	0	2	4	3	2	1	0
1	0	0	0	0	1	1	0	0	0	1	1	3	5	4	3	2
0	**0**	0	1	0	0	0	0	0	1	0	0	2	4	6	5	4
0	0	**1**	0	2	1	0	0	1	0	0	1	1	3	5	7	6
0	0	0	**2**	1	1	0	0	0	2	1	0	0	2	4	6	6
0	0	1	1	**3**	2	1	0	1	1	1	2	1	1	3	5	5
2	1	0	0	**2**	4	3	2	1	0	2	1	1	2	2	4	6
1	1	0	0	1	**3**	5	4	3	2	1	1	0	2	1	3	5
1	0	0	0	0	2	**4**	4	3	2	3	2	1	1	1	2	4
0	2	1	0	0	1	3	**5**	4	3	2	2	3	2	1	1	3
0	1	3	2	1	0	2	4	**6**	**5**	4	3	2	2	1	2	2
2	1	2	2	1	2	1	3	5	5	**6**	5	4	3	2	1	3
1	1	1	3	2	1	1	2	4	6	5	**5**	4	3	4	3	2
0	2	1	2	2	1	0	2	3	5	5	4	**6**	**5**	4	3	2
0	1	1	2	1	1	0	1	2	4	4	4	5	5	**6**	5	4
0	0	2	1	3	2	1	0	2	3	3	5	4	4	5	**7**	6
2	1	1	1	2	4	3	2	1	2	4	4	4	5	4	6	**8**

2	1	0	0	2	4	3	2	1	0	2	1	1	2	2	4	6
1	1	0	0	1	3	5	4	3	2	1	1	0	2	1	3	5
1	0	0	0	0	2	4	4	3	2	3	2	1	1	1	2	4
0	2	1	0	0	1	3	5	4	3	2	2	3	2	1	1	3
0	1	3	2	1	0	2	4	6	5	4	3	2	2	1	2	2
2	1	2	2	1	2	1	3	5	5	6	5	4	3	2	1	3
1	1	1	3	2	1	1	2	4	6	5	5	4	3	4	3	2
0	2	1	2	2	1	0	2	3	5	5	4	6	5	4	3	2
0	1	1	2	1	1	0	1	2	4	4	4	5	5	6	5	4
0	0	2	1	3	2	1	0	2	3	3	5	4	4	5	7	6
2	1	1	1	2	4	3	2	1	2	4	4	4	5	4	6	8

Download Table

Results
You can select a result to get the related traceback.

```
AGACCCTA_CGT_GAC
*** **** *|* ***
AGA_CCTAGCATCGAC
---
AGACCCTA_CGT_GAC
**** *** *|* ***
AGAC_CTAGCATCGAC
---
AGACCCTA_CGT_GAC
***** ** *|* ***
AGACC_TAGCATCGAC
```

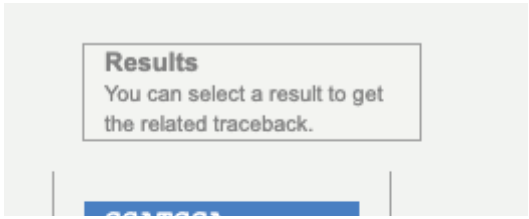
When match=1, gap_penalty=2, mismatch_penalty=1 Mine:

```
In [257]: align('tgcatcgagaccctacgtgac', 'actagacctagcatcgac', 1, 1, 2)
```

```
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0]
 [0 0 1 0 0 0 0 1 1 0 0 0 2 0 0 1 0 0 1]
 [0 1 0 0 1 0 1 0 0 0 1 0 0 3 1 0 0 1 0]
 [0 0 0 1 0 0 0 0 0 1 0 0 0 1 4 2 0 0 0]
 [0 0 1 0 0 0 0 1 1 0 0 0 1 0 2 5 3 1 1]
 [0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 3 6 4 2]
 [0 1 0 0 1 0 2 0 0 0 1 0 0 1 0 1 4 7 5]
 [0 0 0 0 0 2 0 1 0 0 0 2 0 0 0 0 2 5 6]
 [0 1 0 0 1 0 3 1 0 0 1 0 1 1 0 0 0 3 4]
 [0 0 2 0 0 0 1 4 2 0 0 0 1 0 0 1 0 1 4]
 [0 0 1 1 0 0 0 2 5 3 1 0 1 0 0 1 0 0 2]
 [0 0 1 0 0 0 0 1 3 4 2 0 1 0 0 1 0 0 1]
 [0 0 0 2 0 0 0 0 1 4 3 1 0 0 1 0 0 0 0]
 [0 1 0 0 3 1 1 0 0 2 5 3 1 1 0 0 0 1 0]
 [0 0 2 0 1 2 0 2 1 0 3 4 4 2 0 1 0 0 2]
 [0 0 0 1 0 2 1 0 1 0 1 4 3 3 1 0 2 0 0]
 [0 0 0 1 0 0 1 0 0 2 0 2 3 2 4 2 0 1 0]
 [0 0 0 0 0 1 0 0 0 0 1 1 1 2 2 3 3 1 0]
 [0 1 0 0 1 0 2 0 0 0 1 0 0 2 1 1 2 4 2]
 [0 0 2 0 0 0 0 3 1 0 0 0 1 0 1 2 0 2 5]]
seq1 =  gcatcga , seq2 =  gcatcga , score = 7
```

C_2	T_3	A_4	G_5	A_6	C_7	C_8	T_9	A_{10}	G_{11}	C_{12}	A_{13}	T_{14}	C_{15}	G_{16}	A_{17}	C_{18}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
1	0	0	0	0	1	1	0	0	0	2	0	0	1	0	0	1
0	0	1	0	1	0	0	0	1	0	0	3	1	0	0	1	0
0	1	0	0	0	0	0	1	0	0	0	1	4	2	0	0	0
1	0	0	0	0	1	1	0	0	0	1	0	2	5	3	1	1
0	0	0	1	0	0	0	0	0	1	0	0	0	3	6	4	2
0	0	1	0	2	0	0	0	1	0	0	1	0	1	4	7	5
0	0	0	2	0	1	0	0	0	2	0	0	0	0	2	5	6
0	0	1	0	3	1	0	0	1	0	1	1	0	0	0	3	4
2	0	0	0	1	4	2	0	0	0	1	0	0	1	0	1	4
1	1	0	0	0	2	5	3	1	0	1	0	0	1	0	0	2
1	0	0	0	0	1	3	4	2	0	1	0	0	1	0	0	1
0	2	0	0	0	0	1	4	3	1	0	0	1	0	0	0	0
0	0	3	1	1	0	0	2	5	3	1	1	0	0	0	1	0
2	0	1	2	0	2	1	0	3	4	4	2	0	1	0	0	2
0	1	0	2	1	0	1	0	1	4	3	3	1	0	2	0	0
0	1	0	0	1	0	0	2	0	2	3	2	4	2	0	1	0
0	0	0	1	0	0	0	0	1	1	1	2	2	3	3	1	0
0	0	1	0	2	0	0	0	1	0	0	2	1	1	2	4	2
2	0	0	0	0	3	1	0	0	0	1	0	1	2	0	2	5

Website:




GCATCGA
****
GCATCGA

In [61]:
```
import urllib.request
import requests
import math
from numpy import *
from numpy.linalg import norm
a = 0.4
b = 0.2
r = float(requests.get(f"http://ramcdougal.com/cgi-bin/error_function.py?a={a}&b={b}", headers={"User-Ag
```

In [62]:
```
def f(a, b):
    r = float(requests.get(f"http://ramcdougal.com/cgi-bin/error_function.py?a={a}&b={b}", headers={"Use
    return r
```

In [63]:
```
f(0.4,0.2)
```

Out[63]:
```
1.294915
```

In [68]:
```
def dfda(a,b,h):
    return (f(a+h,b)-f(a,b))/h #Based on the alagorithem
def dfdb(a,b,h):
    return (f(a,b+h)-f(a,b))/h
def gradf(a,b,h):
    return array([dfda(a,b,h), dfdb(a,b,h)])
```

In [69]:
```
#https://www.cs.toronto.edu/~guerzhoy/411/lec/W02/python/graddescent2d.html
def grad_descent2(f, gradf, init_t, alpha): #t = initial guess, alpha = gamma(step size)
    EPS = 1e-8 #EPS:Epsilon is a very small number to prevent any division by zero
    #in the implementation (e.g. 10E-8).
    h = 1e-3
    prev_t = init_t-10*EPS
    t = init_t.copy()

    max_iter = 1000#In practice, the Maximum Number of Steps = 1,000 or greater.
    #So I set max_steps = 1000
    iter = 0
    while norm(t - prev_t) > EPS and iter < max_iter: #Stopping rule:
        #If there have been more than Maximum Number of Steps,Gradient Descent would stop.
        prev_t = t.copy()
        t -= alpha*gradf(t[0], t[1],h)
        print (t, f(t[0], t[1]), gradf(t[0], t[1],h))
        iter += 1

    return t
```

In [66]:
```
f(0.5,0.5)
```

Out[66]: 1.216377

In [70]:
```
grad_descent2(f, gradf, array([0.01, 0.01]), 0.1) #step size=0.1
#so it can descent quickly in the beginning.
```

```
[0.0511 0.1457] 1.4223669 [-0.3288 -1.0856]
[0.08398 0.25426] 1.306428148 [-0.26304 -0.86848]
[0.110284 0.341108] 1.23220471632 [-0.210432 -0.694784]
[0.1313272 0.4105864] 1.18468361572 [-0.1683456 -0.5558272]
[0.14816176 0.46616912] 1.15425562789 [-0.13467648 -0.44466176]
[0.16162941 0.5106353 ] 1.13477012891 [-0.10774119 -0.35572941]
[0.17240353 0.54620824] 1.12229014004 [-0.08619295 -0.28458353]
[0.18102282 0.57466659] 1.11429553162 [-0.06895435 -0.22766682]
[0.18791826 0.59743327] 1.10917304997 [-0.05516349 -0.18213346]
[0.19343461 0.61564662] 1.10588991566 [-0.04413079 -0.14570677]
[0.19784769 0.6302173 ] 1.10378491295 [-0.03530463 -0.11656541]
[0.20137815 0.64187384] 1.10243467389 [-0.0282437  -0.09325233]
[0.20420252 0.65119907] 1.10156809097 [-0.02259497 -0.07460187]
[0.20646202 0.65865926] 1.1010115339 [-0.01807597 -0.05968148]
[0.20826961 0.6646274 ] 1.10065378233 [-0.01446077 -0.04774519]
[0.20971569 0.66940192] 1.10042357719 [-0.01156863 -0.03819616]
[0.21087255 0.67322154] 1.10027525055 [-0.00925489 -0.03055692]
[0.21179804 0.67627723] 1.10017952531 [-0.00740392 -0.02444554]
[0.21253843 0.67872179] 1.10011762415 [-0.00592313 -0.01955643]
```

In [ ]:
```
#In practice, the Maximum Number of Steps = 1,000 or greater
#If there have been more than Maximum Number of Steps,
#Gradient Descent would stop.
#step size = slop * learning rate
#New intercept = old intercept - step size
```

In [71]:
```
grad_descent2(f, gradf, array([0.001, 0.001]), 0.1)
```

```
[0.0439 0.1385] 1.43266866 [-0.3432 -1.1   ]
[0.07822 0.2485 ] 1.3130235784 [-0.27456 -0.88   ]
[0.105676 0.3365  ] 1.23642763498 [-0.219648 -0.704   ]
[0.1276408 0.4069   ] 1.18738775822 [-0.1757184 -0.5632   ]
[0.14521264 0.46322   ] 1.15598745874 [-0.14057472 -0.45056   ]
[0.15927011 0.508276  ] 1.13587944437 [-0.11245978 -0.360448  ]
[0.17051609 0.5443208 ] 1.12300085698 [-0.08996782 -0.2883584 ]
[0.17951287 0.57315664] 1.11475099457 [-0.07197426 -0.23068672]
[0.1867103  0.59622531] 1.10946502938 [-0.05757941 -0.18454938]
[0.19246824 0.61468025] 1.10607716902 [-0.04606353 -0.1476395 ]
[0.19707459 0.6294442 ] 1.10390506438 [-0.03685081 -0.1181116 ]
[0.20075967 0.64125536] 1.10251181822 [-0.02948066 -0.09448928]
[0.20370774 0.65070429] 1.10161766124 [-0.02358452 -0.07559143]
[0.20606619 0.65826343] 1.10104341724 [-0.01886762 -0.06047314]
[0.20795295 0.66431074] 1.10067431428 [-0.0150941  -0.04837851]
[0.20946236 0.6691486 ] 1.10043681894 [-0.01207528 -0.03870281]
[0.21066989 0.67301888] 1.10028380635 [-0.00966021 -0.03096224]
[0.21163591 0.6761151 ] 1.10018506589 [-0.00772818 -0.0247698 ]
[0.21240873 0.67859208] 1.100121222 [-0.00618254 -0.01981584]
[0.21302698 0.68057366] 1.10007984195 [-0.00494604 -0.01585267]
[0.21352159 0.68215893] 1.10005294274 [-0.00395683 -0.01268214]
[0.21391727 0.68342715] 1.10003539446 [-0.00316546 -0.01014571]
[0.21423382 0.68444172] 1.10002389735 [-0.00253237 -0.00811657]
[0.21448705 0.68525337] 1.10001632621 [-0.00202589 -0.00649325]
[0.21468964 0.6859027 ] 1.10001131031 [-0.00162072 -0.0051946 ]
[0.21485172 0.68642216] 1.10000796382 [-0.00129657 -0.00415568]
[0.21498137 0.68683773] 1.10000571303 [-0.00103726 -0.00332455]
[0.2150851  0.68717018] 1.10000418528 [-0.0008298  -0.00265964]
[0.21516808 0.68743615] 1.10000313773 [-0.00066384 -0.0021277 ]
[0.21523446 0.68764892] 1.10000241148 [-0.00053108 -0.00170217]
[0.21528757 0.68781913] 1.100001902 [-0.00042486 -0.00136173]
[0.21533006 0.68795531] 1.10000154021 [-0.00033989 -0.00108939]
[0.21536405 0.68806425] 1.10000128008 [-0.00027191 -0.00087151]
[0.21539124 0.6881514 ] 1.10000109072 [-0.00021753 -0.00069721]
[0.21541299 0.68822112] 1.10000095124 [-0.00017402 -0.00055777]
[0.21543039 0.68827689] 1.10000084734 [-0.00013922 -0.00044622]
[0.21544431 0.68832152] 1.10000076913 [-0.00011138 -0.00035697]
[0.21545545 0.68835721] 1.10000070971 [-8.9100e-05 -2.8558e-04]
[0.21546436 0.68838577] 1.10000066419 [-7.12799999e-05 -2.28460000e-04]
[0.21547149 0.68840862] 1.10000062906 [-5.70200001e-05 -1.82770000e-04]
[0.21547719 0.68842689] 1.10000060178 [-4.5620e-05 -1.4621e-04]
```

```
[0.21548175 0.68844152] 1.10000058049 [-3.64999999e-05 -1.16970000e-04]
[0.2154854  0.68845321] 1.10000056379 [-2.92000002e-05 -9.35800002e-05]
[0.21548832 0.68846257] 1.10000055064 [-2.33499999e-05 -7.48600000e-05]
[0.21549066 0.68847006] 1.10000054027 [-1.86800000e-05 -5.98899998e-05]
[0.21549253 0.68847605] 1.10000053206 [-1.49499999e-05 -4.79100000e-05]
[0.21549402 0.68848084] 1.10000052555 [-1.19600001e-05 -3.83300001e-05]
[0.21549522 0.68848467] 1.10000052037 [-9.56000012e-06 -3.06600001e-05]
[0.21549617 0.68848774] 1.10000051626 [-7.65999997e-06 -2.45300000e-05]
[0.21549694 0.68849019] 1.10000051298 [-6.11999984e-06 -1.96299998e-05]
[0.21549755 0.68849215] 1.10000051037 [-4.89999996e-06 -1.57000000e-05]
[0.21549804 0.68849372] 1.10000050828 [-3.91999988e-06 -1.25599999e-05]
[0.21549843 0.68849498] 1.10000050662 [-3.14000004e-06 -1.00499999e-05]
[0.21549875 0.68849598] 1.10000050529 [-2.50999999e-06 -8.04000000e-06]
[0.215499   0.68849679] 1.10000050423 [-2.01000017e-06 -6.43000009e-06]
[0.2154992  0.68849743] 1.10000050338 [-1.59999991e-06 -5.13999998e-06]
[0.21549936 0.68849794] 1.1000005027 [-1.27999988e-06 -4.10999990e-06]
[0.21549949 0.68849835] 1.10000050216 [-1.01999986e-06 -3.28999983e-06]
[0.21549959 0.68849868] 1.10000050173 [-8.20000068e-07 -2.63000000e-06]
[0.21549967 0.68849895] 1.10000050138 [-6.50000054e-07 -2.09999995e-06]
[0.21549974 0.68849916] 1.10000050111 [-5.30000044e-07 -1.68999992e-06]
[0.21549979 0.68849932] 1.10000050089 [-4.30000036e-07 -1.35000011e-06]
[0.21549983 0.68849946] 1.10000050071 [-3.40000028e-07 -1.08000009e-06]
[0.21549987 0.68849957] 1.10000050057 [-2.70000022e-07 -8.70000072e-07]
[0.21549989 0.68849965] 1.10000050045 [-2.10000017e-07 -6.90000057e-07]
[0.21549991 0.68849972] 1.10000050036 [-1.70000014e-07 -5.50000046e-07]
[0.21549993 0.68849978] 1.10000050029 [-1.40000012e-07 -4.40000036e-07]
[0.21549995 0.68849982] 1.10000050023 [-1.10000009e-07 -3.50000029e-07]
[0.21549996 0.68849986] 1.10000050019 [-9.00000074e-08 -2.90000024e-07]
[0.21549997 0.68849989] 1.10000050015 [-7.00000058e-08 -2.30000019e-07]
[0.21549997 0.68849991] 1.10000050012 [-6.00000050e-08 -1.80000015e-07]
[0.21549998 0.68849993] 1.10000050009 [-4.00000033e-08 -1.40000012e-07]
[0.21549998 0.68849994] 1.10000050008 [-4.00000033e-08 -1.20000010e-07]
[0.21549999 0.68849995] 1.10000050006 [-3.00000025e-08 -9.00000074e-08]
[0.21549999 0.68849996] 1.10000050005 [-2.00000017e-08 -8.00000066e-08]
```

Out[71]: `array([0.21549999, 0.68849996])`

In [72]:
```
grad_descent2(f, gradf, array([0.99, 0.01]), 0.1)
```

```
[0.8229 0.1051] 1.04914606 [ 0.6684 -0.3804]
[0.75606 0.14314] 1.0078300696 [ 0.26736 -0.15216]
[0.729324 0.158356] 1.00124024714 [ 0.106944 -0.060864]
[0.7186296 0.1644424] 1.00019416994 [ 0.0427776 -0.0243456]
[0.71435184 0.16687696] 1.00003011535 [ 0.01711104 -0.00973824]
[0.71264074 0.16785078] 1.00000519372 [ 0.00684442 -0.0038953 ]
[0.71195629 0.16824031] 1.0000017371 [ 0.00273776 -0.00155812]
[0.71168252 0.16839613] 1.00000139638 [ 0.0010951  -0.00062325]
[0.71157301 0.16845845] 1.00000142679 [ 0.00043805 -0.00024929]
[0.7115292  0.16848338] 1.00000146564 [ 1.7522e-04 -9.9720e-05]
[0.71151168 0.16849335] 1.00000148544 [ 7.009e-05 -3.989e-05]
[0.71150467 0.16849734] 1.00000149405 [ 2.80300001e-05 -1.59600000e-05]
[0.71150187 0.16849894] 1.0000014976 [ 1.12100000e-05 -6.38000008e-06]
[0.71150075 0.16849958] 1.00000149903 [ 4.49000015e-06 -2.54999999e-06]
[0.7115003  0.16849983] 1.00000149961 [ 1.80000015e-06 -1.01999986e-06]
[0.71150012 0.16849993] 1.00000149985 [ 7.10000059e-07 -4.10000034e-07]
[0.71150005 0.16849997] 1.00000149994 [ 2.90000024e-07 -1.60000013e-07]
[0.71150002 0.16849999] 1.00000149998 [ 1.10000009e-07 -7.00000058e-08]
[0.71150001 0.1685    ] 1.00000149999 [ 5.00000041e-08 -3.00000025e-08]
[0.7115 0.1685] 1.00000149999 [2.00000017e-08 0.00000000e+00]
```

Out[72]:
```
array([0.7115, 0.1685])
```

In [73]:
```
grad_descent2(f, gradf, array([0.01, 0.99]), 0.1)
```

```
[0.0511 0.9297] 1.1851285 [-0.3288  0.4824]
[0.08398 0.88146] 1.154470132 [-0.26304  0.38592]
[0.110284 0.842868] 1.13485123408 [-0.210432  0.308736]
[0.1313272 0.8119944] 1.12229710549 [-0.1683456  0.2469888]
[0.14816176 0.78729552] 1.11426403606 [-0.13467648  0.19759104]
[0.16162941 0.76753642] 1.10912412991 [-0.10774118  0.15807283]
[0.17240353 0.75172913] 1.10583559667 [-0.08619295  0.12645827]
[0.18102282 0.73908331] 1.10373174059 [-0.06895436  0.10116661]
[0.18791826 0.72896665] 1.102385917 [-0.05516348  0.08093329]
[0.19343461 0.72087332] 1.10152510532 [-0.04413079  0.06474664]
[0.19784768 0.71439865] 1.1009745981 [-0.03530463  0.0517973 ]
[0.20137815 0.70921892] 1.10062260339 [-0.0282437   0.04143785]
[0.20420252 0.70507514] 1.10039759063 [-0.02259496  0.03315028]
[0.20646201 0.70176011] 1.10025379358 [-0.01807598  0.02652022]
[0.20826961 0.69910809] 1.10016193234 [-0.01446078  0.02121617]
[0.20971569 0.69698647] 1.10010327627 [-0.01156862  0.01697294]
[0.21087255 0.69528918] 1.10006584447 [-0.0092549   0.01357835]
[0.21179804 0.69393134] 1.10004197458 [-0.00740391  0.01086269]
[0.21253843 0.69284507] 1.10002676703 [-0.00592313  0.00869015]
[0.21313075 0.69197606] 1.10001708954 [-0.00473851  0.00695211]
[0.2136046  0.69128085] 1.10001094022 [-0.00379081  0.00556169]
[0.21398368 0.69072468] 1.10000704007 [-0.00303265  0.00444935]
[0.21428694 0.69027974] 1.10000457231 [-0.00242612  0.00355948]
[0.21452955 0.68992379] 1.10000301561 [-0.0019409   0.00284758]
[0.21472364 0.68963904] 1.10000203745 [-0.00155271  0.00227807]
[0.21487892 0.68941123] 1.10000142594 [-0.00124217  0.00182246]
[0.21500313 0.68922898] 1.10000104618 [-0.00099374  0.00145797]
[0.21510251 0.68908319] 1.10000081242 [-0.00079499  0.00116637]
[0.21518201 0.68896655] 1.10000067023 [-0.00063599  0.0009331 ]
[0.2152456  0.68887324] 1.10000058518 [-0.00050879  0.00074648]
[0.21529648 0.68879859] 1.1000005355 [-0.00040703  0.00059718]
[0.21533719 0.68873887] 1.10000050751 [-0.00032563  0.00047775]
[0.21536975 0.6886911 ] 1.10000049264 [-0.00026051  0.00038219]
[0.2153958  0.68865288] 1.10000048555 [-0.0002084   0.00030576]
[0.21541664 0.6886223 ] 1.10000048296 [-0.00016672  0.00024461]
[0.21543331 0.68859784] 1.10000048287 [-0.00013338  0.00019568]
[0.21544665 0.68857827] 1.10000048405 [-0.0001067   0.00015655]
[0.21545732 0.68856262] 1.1000004858 [-8.53600002e-05  1.25240000e-04]
[0.21546586 0.6885501 ] 1.10000048772 [-6.82800001e-05  1.00190000e-04]
[0.21547268 0.68854008] 1.10000048959 [-5.46299999e-05  8.01500000e-05]
[0.21547815 0.68853206] 1.1000004913 [-4.37100001e-05  6.41200000e-05]
```

```
[0.21548252 0.68852565] 1.1000004928 [-3.497e-05  5.129e-05]
[0.21548602 0.68852052] 1.10000049408 [-2.79699999e-05  4.10400001e-05]
[0.21548881 0.68851642] 1.10000049517 [-2.23800001e-05  3.28300001e-05]
[0.21549105 0.68851313] 1.10000049607 [-1.79000001e-05  2.62700000e-05]
[0.21549284 0.68851051] 1.10000049682 [-1.43199999e-05  2.10100002e-05]
[0.21549427 0.68850841] 1.10000049743 [-1.14599998e-05  1.68100001e-05]
[0.21549542 0.68850672] 1.10000049792 [-9.16000009e-06  1.34500000e-05]
[0.21549633 0.68850538] 1.10000049833 [-7.32999994e-06  1.07600000e-05]
[0.21549707 0.6885043 ] 1.10000049866 [-5.87000004e-06  8.60000005e-06]
[0.21549765 0.68850344] 1.10000049892 [-4.69000017e-06  6.88999990e-06]
[0.21549812 0.68850275] 1.10000049913 [-3.75000009e-06  5.51000001e-06]
[0.2154985 0.6885022] 1.10000049931 [-3.01000003e-06  4.39999992e-06]
[0.2154988  0.68850176] 1.10000049944 [-2.39999998e-06  3.53000007e-06]
[0.21549904 0.68850141] 1.10000049955 [-1.91999994e-06  2.82000001e-06]
[0.21549923 0.68850113] 1.10000049964 [-1.53999991e-06  2.26000019e-06]
[0.21549939 0.6885009 ] 1.10000049971 [-1.22999988e-06  1.81000015e-06]
[0.21549951 0.68850072] 1.10000049977 [-9.79999859e-07  1.44000012e-06]
[0.21549961 0.68850058] 1.10000049982 [-7.89999843e-07  1.15000010e-06]
[0.21549969 0.68850046] 1.10000049985 [-6.29999830e-07  9.30000077e-07]
[0.21549975 0.68850037] 1.10000049988 [-4.99999819e-07  7.40000061e-07]
[0.2154998 0.6885003] 1.10000049991 [-4.10000034e-07  5.90000049e-07]
[0.21549984 0.68850024] 1.10000049993 [-3.30000027e-07  4.70000039e-07]
[0.21549987 0.68850019] 1.10000049994 [-2.60000022e-07  3.80000031e-07]
[0.2154999  0.68850015] 1.10000049995 [-2.00000017e-07  3.00000025e-07]
[0.21549992 0.68850012] 1.10000049996 [-1.60000013e-07  2.40000020e-07]
[0.21549993 0.6885001 ] 1.10000049997 [-1.30000011e-07  1.90000016e-07]
[0.21549995 0.68850008] 1.10000049998 [-1.10000009e-07  1.50000012e-07]
[0.21549996 0.68850006] 1.10000049998 [-8.00000066e-08  1.30000011e-07]
[0.21549997 0.68850005] 1.10000049998 [-6.00000050e-08  1.00000008e-07]
[0.21549997 0.68850004] 1.10000049999 [-6.00000050e-08  8.00000066e-08]
[0.21549998 0.68850003] 1.10000049999 [-4.00000033e-08  6.00000050e-08]
[0.21549998 0.68850003] 1.10000049999 [-3.00000025e-08  5.00000041e-08]
```

Out[73]: array([0.21549998, 0.68850003])

In [74]:
```
grad_descent2(f, gradf, array([0.5, 0.5]), 0.1)
```

```
[0.4431 0.5377] 1.1744661 [ 0.4552 -0.3016]
[0.39758 0.56786] 1.147646196 [ 0.36416 -0.24128]
[0.361164 0.591988] 1.13048391504 [ 0.291328 -0.193024]
[0.3320312 0.6112904] 1.11950202131 [ 0.2330624 -0.1544192]
[0.30872496 0.62673232] 1.11247518218 [ 0.18644992 -0.12353536]
[0.29007997 0.63908586] 1.10797926343 [ 0.14915994 -0.09882829]
[0.27516397 0.64896869] 1.105102882 [ 0.11932795 -0.07906263]
[0.26323118 0.65687495] 1.10326280324 [ 0.09546235 -0.06325011]
[0.25368494 0.66319996] 1.10208579712 [ 0.07636989 -0.05060008]
[0.24604796 0.66825997] 1.10133302857 [ 0.06109591 -0.04048007]
[0.23993836 0.67230797] 1.100851669 [ 0.04887673 -0.03238405]
[0.23505069 0.67554638] 1.10054392875 [ 0.03910138 -0.02590725]
[0.23114055 0.6781371 ] 1.10034723885 [ 0.03128111 -0.02072579]
[0.22801244 0.68020968] 1.10022156844 [ 0.02502488 -0.01658064]
[0.22550995 0.68186775] 1.10014130826 [ 0.02001991 -0.01326451]
[0.22350796 0.6831942 ] 1.10009007685 [ 0.01601592 -0.01061161]
[0.22190637 0.68425536] 1.10005739684 [ 0.01281274 -0.00848929]
[0.2206251  0.68510429] 1.10003656809 [ 0.0102502  -0.00679142]
[0.21960008 0.68578343] 1.10002330688 [ 0.00820015 -0.00543314]
[0.21878006 0.68632674] 1.10001487504 [ 0.00656013 -0.00434651]
[0.21812405 0.6867614 ] 1.10000952294 [ 0.00524809 -0.00347721]
[0.21759924 0.68710912] 1.10000613301 [ 0.00419848 -0.00278177]
[0.21717939 0.68738729] 1.10000399179 [ 0.00335878 -0.00222541]
[0.21684351 0.68760983] 1.10000264408 [ 0.00268703 -0.00178033]
[0.21657481 0.68778787] 1.10000179967 [ 0.00214963 -0.00142426]
[0.21635985 0.68793029] 1.10000127376 [ 0.0017197  -0.00113941]
[0.21618788 0.68804423] 1.10000094879 [ 0.00137575 -0.00091154]
[0.2160503  0.68813539] 1.10000075008 [ 0.00110061 -0.00072922]
[0.21594024 0.68820831] 1.10000063034 [ 0.00088049 -0.00058338]
[0.21585219 0.68826665] 1.10000055965 [ 0.00070439 -0.0004667 ]
[0.21578175 0.68831332] 1.10000051916 [ 0.00056351 -0.00037336]
[0.2157254  0.68835065] 1.10000049705 [ 0.00045081 -0.00029869]
[0.21568032 0.68838052] 1.10000048595 [ 0.00036064 -0.00023896]
[0.21564426 0.68840442] 1.10000048127 [ 0.00028852 -0.00019116]
[0.21561541 0.68842353] 1.10000048022 [ 0.00023082 -0.00015293]
[0.21559232 0.68843883] 1.10000048111 [ 0.00018465 -0.00012234]
[0.21557386 0.68845106] 1.10000048293 [ 1.47720000e-04 -9.78800001e-05]
[0.21555909 0.68846085] 1.10000048509 [ 1.1817e-04 -7.8300e-05]
[0.21554727 0.68846868] 1.10000048727 [ 9.45400001e-05 -6.26400001e-05]
[0.21553782 0.68847494] 1.1000004893 [ 7.56300000e-05 -5.01099999e-05]
[0.21553025 0.68847995] 1.10000049111 [ 6.05099999e-05 -4.00900000e-05]
```

```
[0.2155242  0.68848396] 1.10000049268 [ 4.840e-05 -3.208e-05]
[0.21551936 0.68848717] 1.10000049401 [ 3.87199999e-05 -2.56600001e-05]
[0.21551549 0.68848974] 1.10000049512 [ 3.09800001e-05 -2.05299999e-05]
[0.21551239 0.68849179] 1.10000049604 [ 2.47800001e-05 -1.64200000e-05]
[0.21550991 0.68849343] 1.10000049679 [ 1.98300001e-05 -1.31300000e-05]
[0.21550793 0.68849475] 1.10000049741 [ 1.587e-05 -1.050e-05]
[0.21550634 0.6884958 ] 1.10000049792 [ 1.26899999e-05 -8.41000003e-06]
[0.21550507 0.68849664] 1.10000049833 [ 1.01500000e-05 -6.73000011e-06]
[0.21550406 0.68849731] 1.10000049865 [ 8.12000001e-06 -5.38000000e-06]
[0.21550325 0.68849785] 1.10000049892 [ 6.49999987e-06 -4.30000013e-06]
[0.2155026  0.68849828] 1.10000049913 [ 5.19999999e-06 -3.44000006e-06]
[0.21550208 0.68849862] 1.10000049931 [ 4.14999990e-06 -2.76000001e-06]
[0.21550166 0.6884989 ] 1.10000049944 [ 3.33000005e-06 -2.19999996e-06]
[0.21550133 0.68849912] 1.10000049955 [ 2.66000000e-06 -1.75999992e-06]
[0.21550106 0.68849929] 1.10000049964 [ 2.13000018e-06 -1.40999989e-06]
[0.21550085 0.68849943] 1.10000049972 [ 1.70000014e-06 -1.12999987e-06]
[0.21550068 0.68849955] 1.10000049977 [ 1.36000011e-06 -8.99999852e-07]
[0.21550054 0.68849964] 1.10000049982 [ 1.09000009e-06 -7.29999838e-07]
[0.21550044 0.68849971] 1.10000049985 [ 8.80000073e-07 -5.69999825e-07]
[0.21550035 0.68849977] 1.10000049988 [ 7.00000058e-07 -4.59999816e-07]
[0.21550028 0.68849981] 1.10000049991 [ 5.50000046e-07 -3.70000031e-07]
[0.21550022 0.68849985] 1.10000049993 [ 4.40000036e-07 -3.00000025e-07]
[0.21550018 0.68849988] 1.10000049994 [ 3.6000003e-07 -2.4000002e-07]
[0.21550014 0.6884999 ] 1.10000049995 [ 2.90000024e-07 -1.90000016e-07]
[0.21550011 0.68849992] 1.10000049996 [ 2.30000019e-07 -1.50000012e-07]
[0.21550009 0.68849994] 1.10000049997 [ 1.80000015e-07 -1.20000010e-07]
[0.21550007 0.68849995] 1.10000049998 [ 1.40000012e-07 -1.00000008e-07]
[0.21550006 0.68849996] 1.10000049998 [ 1.20000010e-07 -8.00000066e-08]
[0.21550005 0.68849997] 1.10000049998 [ 1.00000008e-07 -6.00000050e-08]
[0.21550004 0.68849997] 1.10000049999 [ 7.00000058e-08 -5.00000041e-08]
[0.21550003 0.68849998] 1.10000049999 [ 6.00000050e-08 -4.00000033e-08]
```

Out[74]: array([0.21550003, 0.68849998])

In [75]:
```
grad_descent2(f, gradf, array([0.99, 0.99]), 0.1)
```

```
[0.8351 0.9297] 1.5412213 [1.2392 0.4824]
[0.71118 0.88146] 1.382244084 [0.99136 0.38592]
[0.612044 0.842868] 1.28052621136 [0.793088 0.308736]
[0.5327352 0.8119944] 1.21544880935 [0.6344704 0.2469888]
[0.46928816 0.78729552] 1.17381690125 [0.50757632 0.19759104]
[0.41853053 0.76753642] 1.14718658341 [0.40606106 0.15807283]
[0.37792442 0.75172913] 1.13015446257 [0.32484884 0.12645826]
[0.34543954 0.73908331] 1.11926293164 [0.25987907 0.10116661]
[0.31945163 0.72896665] 1.11229957275 [0.20790326 0.08093329]
[0.2986613  0.72087332] 1.10784879968 [0.16632261 0.06474663]
[0.28202904 0.71439865] 1.10500492628 [0.13305808 0.0517973 ]
[0.26872324 0.70921892] 1.1031885445 [0.10644647 0.04143785]
[0.25807859 0.70507514] 1.10202901775 [0.08515717 0.03315027]
[0.24956287 0.70176011] 1.10128928684 [0.06812574 0.02652022]
[0.2427503  0.69910809] 1.10081775193 [0.05450059 0.02121618]
[0.23730024 0.69698647] 1.10051748392 [0.04360047 0.01697294]
[0.23294019 0.69528918] 1.10032652386 [0.03488039 0.01357836]
[0.22945215 0.69393134] 1.10020527855 [0.02790431 0.01086269]
[0.22666172 0.69284507] 1.1001284569 [0.02232345 0.00869015]
[0.22442938 0.69197606] 1.10007991132 [0.01785875 0.00695211]
[0.2226435  0.69128085] 1.10004933838 [0.01428701 0.0055617 ]
[0.2212148  0.69072468] 1.10003016866 [0.0114296  0.00444935]
[0.22007184 0.69027974] 1.10001821763 [0.00914368 0.00355948]
[0.21915747 0.68992379] 1.10001082303 [0.00731495 0.00284759]
[0.21842598 0.68963903] 1.10000629373 [0.00585196 0.00227807]
[0.21784078 0.68941123] 1.10000355759 [0.00468156 0.00182245]
[0.21737263 0.68922898] 1.10000193654 [0.00374525 0.00145796]
[0.2169981  0.68908319] 1.10000100313 [0.0029962  0.00116637]
[0.21669848 0.68896655] 1.10000048899 [0.00239697 0.0009331 ]
[0.21645878 0.68887324] 1.10000022655 [0.00191757 0.00074648]
[0.21626703 0.68879859] 1.10000011187 [0.00153405 0.00059718]
[0.21611362 0.68873887] 1.1000000811 [0.00122724 0.00047774]
[0.2159909 0.6886911] 1.1000000955 [0.0009818 0.0003822]
[0.21589272 0.68865288] 1.100000132 [0.00078544 0.00030576]
[0.21581417 0.6886223 ] 1.10000017719 [0.00062834 0.0002446 ]
[0.21575134 0.68859784] 1.10000022356 [0.00050268 0.00019569]
[0.21570107 0.68857827] 1.10000026721 [0.00040214 0.00015655]
[0.21566086 0.68856262] 1.10000030632 [0.00032171 0.00012524]
[0.21562869 0.6885501 ] 1.10000034029 [0.00025737 0.00010019]
[0.21560295 0.68854008] 1.10000036918 [2.059e-04 8.015e-05]
[0.21558236 0.68853206] 1.10000039339 [1.6472e-04 6.4120e-05]
```

```
[0.21556589 0.68852565] 1.10000041346 [1.31780000e-04 5.12999998e-05]
[0.21555271 0.68852052] 1.10000042997 [1.05420000e-04 4.10400001e-05]
[0.21554217 0.68851642] 1.10000044346 [8.43400001e-05 3.28400001e-05]
[0.21553373 0.68851313] 1.10000045444 [6.747e-05 2.627e-05]
[0.21552699 0.6885105 ] 1.10000046335 [5.397e-05 2.101e-05]
[0.21552159 0.6885084 ] 1.10000047054 [4.31800000e-05 1.68100001e-05]
[0.21551727 0.68850672] 1.10000047635 [3.454e-05 1.344e-05]
[0.21551382 0.68850538] 1.10000048102 [2.76400001e-05 1.07600000e-05]
[0.21551105 0.6885043 ] 1.10000048478 [2.21100001e-05 8.61000005e-06]
[0.21550884 0.68850344] 1.10000048781 [1.76799999e-05 6.87999990e-06]
[0.21550707 0.68850275] 1.10000049023 [1.41500001e-05 5.51000001e-06]
[0.21550566 0.6885022 ] 1.10000049217 [1.13200000e-05 4.41000014e-06]
[0.21550453 0.68850176] 1.10000049373 [9.05999986e-06 3.52999985e-06]
[0.21550362 0.68850141] 1.10000049498 [7.24999993e-06 2.82000001e-06]
[0.2155029  0.68850113] 1.10000049599 [5.79000003e-06 2.24999996e-06]
[0.21550232 0.6885009 ] 1.10000049679 [4.63000016e-06 1.80000015e-06]
[0.21550185 0.68850072] 1.10000049743 [3.71000008e-06 1.44000012e-06]
[0.21550148 0.68850058] 1.10000049794 [2.97000002e-06 1.16000010e-06]
[0.21550119 0.68850046] 1.10000049835 [2.37999997e-06 9.30000077e-07]
[0.21550095 0.68850037] 1.10000049868 [1.89999994e-06 7.40000061e-07]
[0.21550076 0.68850029] 1.10000049895 [1.50999990e-06 5.89999827e-07]
[0.21550061 0.68850024] 1.10000049916 [1.20999988e-06 4.69999817e-07]
[0.21550049 0.68850019] 1.10000049932 [9.79999859e-07 3.79999809e-07]
[0.21550039 0.68850015] 1.10000049946 [7.80000065e-07 3.00000025e-07]
[0.21550031 0.68850012] 1.10000049957 [6.20000051e-07 2.40000020e-07]
[0.21550025 0.6885001 ] 1.10000049965 [5.00000041e-07 2.00000017e-07]
[0.2155002  0.68850008] 1.10000049972 [4.00000033e-07 1.60000013e-07]
[0.21550016 0.68850006] 1.10000049978 [3.20000026e-07 1.20000010e-07]
[0.21550013 0.68850005] 1.10000049982 [2.60000022e-07 1.00000008e-07]
[0.2155001  0.68850004] 1.10000049986 [2.00000017e-07 8.00000066e-08]
[0.21550008 0.68850003] 1.10000049989 [1.60000013e-07 6.00000050e-08]
[0.21550006 0.68850002] 1.10000049991 [1.30000011e-07 5.00000041e-08]
[0.21550005 0.68850002] 1.10000049993 [1.00000008e-07 4.00000033e-08]
[0.21550004 0.68850002] 1.10000049994 [9.00000074e-08 3.00000025e-08]
[0.21550003 0.68850001] 1.10000049995 [7.00000058e-08 3.00000025e-08]
```

Out[75]:
```
array([0.21550003, 0.68850001])
```

Implement a two-dimensional version of the gradient descent algorithm to find optimal choices of a and b. (7 points) Let the error function be L (loss function) and parameter The goal is to find an optimal set of θ parameters to minimize L, as follows:

In [6]:
```
pip install pandas plotnine
```

```
Requirement already satisfied: pandas in /opt/anaconda3/lib/python3.8/site-packages (1.2.4)
Collecting plotnine
  Downloading plotnine-0.8.0-py3-none-any.whl (4.7 MB)
     |████████████████████████████████████████████████████████████████| 4.7 MB 3.4 MB/s eta 0:00:01
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/anaconda3/lib/python3.8/site-packages (f
rom pandas) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in /opt/anaconda3/lib/python3.8/site-packages (from panda
s) (2021.1)
Requirement already satisfied: numpy>=1.16.5 in /opt/anaconda3/lib/python3.8/site-packages (from panda
s) (1.20.1)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.8/site-packages (from python-dat
eutil>=2.7.3->pandas) (1.15.0)
Collecting descartes>=1.1.0
  Downloading descartes-1.1.0-py3-none-any.whl (5.8 kB)
Requirement already satisfied: statsmodels>=0.12.1 in /opt/anaconda3/lib/python3.8/site-packages (from
plotnine) (0.12.2)
Requirement already satisfied: matplotlib>=3.1.1 in /opt/anaconda3/lib/python3.8/site-packages (from p
lotnine) (3.3.4)
Collecting mizani>=0.7.3
  Downloading mizani-0.7.3-py3-none-any.whl (63 kB)
     |████████████████████████████████████████████████████████████████| 63 kB 12.1 MB/s eta 0:00:01
Requirement already satisfied: scipy>=1.5.0 in /opt/anaconda3/lib/python3.8/site-packages (from plotni
ne) (1.6.2)
Requirement already satisfied: patsy>=0.5.1 in /opt/anaconda3/lib/python3.8/site-packages (from plotni
ne) (0.5.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/anaconda3/lib/python3.8/site-packages (from m
atplotlib>=3.1.1->plotnine) (1.3.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /opt/anaconda3/lib/python3.
8/site-packages (from matplotlib>=3.1.1->plotnine) (2.4.7)
Requirement already satisfied: pillow>=6.2.0 in /opt/anaconda3/lib/python3.8/site-packages (from matpl
otlib>=3.1.1->plotnine) (8.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/anaconda3/lib/python3.8/site-packages (from matplo
tlib>=3.1.1->plotnine) (0.10.0)
Collecting palettable
  Downloading palettable-3.3.0-py2.py3-none-any.whl (111 kB)
     |████████████████████████████████████████████████████████████████| 111 kB 49.7 MB/s eta 0:00:01
Installing collected packages: palettable, mizani, descartes, plotnine
Successfully installed descartes-1.1.0 mizani-0.7.3 palettable-3.3.0 plotnine-0.8.0
Note: you may need to restart the kernel to use updated packages.
```

In [134]:
```
conda install -c conda-forge cartopy
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/anaconda3

  added / updated specs:
    - cartopy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    cartopy-0.18.0             |   py38hf1ba7ce_1         1.7 MB
    shapely-1.7.1              |   py38h9250791_0         376 KB
    ------------------------------------------------------------
                                           Total:         2.0 MB

The following NEW packages will be INSTALLED:

  cartopy            pkgs/main/osx-64::cartopy-0.18.0-py38hf1ba7ce_1
  shapely            pkgs/main/osx-64::shapely-1.7.1-py38h9250791_0



Downloading and Extracting Packages
cartopy-0.18.0       | 1.7 MB    | ##################################### | 100%
shapely-1.7.1        | 376 KB    | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

In [226]:
```
conda install -c conda-forge basemap-data-hires
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/anaconda3

  added / updated specs:
    - basemap-data-hires


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    basemap-data-hires-1.2.2   |                 0       105.4 MB  conda-forge
    ------------------------------------------------------------
                                           Total:       105.4 MB

The following NEW packages will be INSTALLED:

  basemap-data-hires conda-forge/osx-64::basemap-data-hires-1.2.2-0



Downloading and Extracting Packages
basemap-data-hires-1 | 105.4 MB  | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

In [285]:
```
import numpy as np
import pandas as pd
from plotnine import *
import random
```

In [304]:
```
from mpl_toolkits.basemap import Basemap
```

In [305]:
```
data = pd.read_csv("/Users/ycliu/Desktop/worldcities.csv")
data
```

Out[305]:

	city	city_ascii	lat	lng	country	iso2	iso3	admin_name	capital	population	id
0	Tokyo	Tokyo	35.6897	139.6922	Japan	JP	JPN	Tōkyō	primary	37977000.0	1392685764
1	Jakarta	Jakarta	-6.2146	106.8451	Indonesia	ID	IDN	Jakarta	primary	34540000.0	1360771077
2	Delhi	Delhi	28.6600	77.2300	India	IN	IND	Delhi	admin	29617000.0	1356872604
3	Mumbai	Mumbai	18.9667	72.8333	India	IN	IND	Mahārāshtra	admin	23355000.0	1356226629
4	Manila	Manila	14.6000	120.9833	Philippines	PH	PHL	Manila	primary	23088000.0	1608618140
...	...	...	...	...	...	...	...	...	...	...	...
40996	Tukchi	Tukchi	57.3670	139.5000	Russia	RU	RUS	Khabarovskiy Kray	NaN	10.0	1643472801
40997	Numto	Numto	63.6667	71.3333	Russia	RU	RUS	Khanty-Mansiyskiy Avtonomnyy Okrug-Yugra	NaN	10.0	1643985006
40998	Nord	Nord	81.7166	-17.8000	Greenland	GL	GRL	Sermersooq	NaN	10.0	1304217709
40999	Timmiarmiut	Timmiarmiut	62.5333	-42.2167	Greenland	GL	GRL	Kujalleq	NaN	10.0	1304206491
41000	Nordvik	Nordvik	74.0165	111.5100	Russia	RU	RUS	Krasnoyarskiy Kray	NaN	0.0	1643587468

41001 rows × 11 columns

In [306]:
```
#Keep the data I need.
df = data.copy()
df = data[['lat', 'lng']]
df.head()
```

Out[306]:

	lat	lng
0	35.6897	139.6922
1	-6.2146	106.8451
2	28.6600	77.2300
3	18.9667	72.8333
4	14.6000	120.9833

In [307]:
```
lat=[]
for i in range(len(df.values)):
    lat.append(df.values[i][0])
    i+=1
```

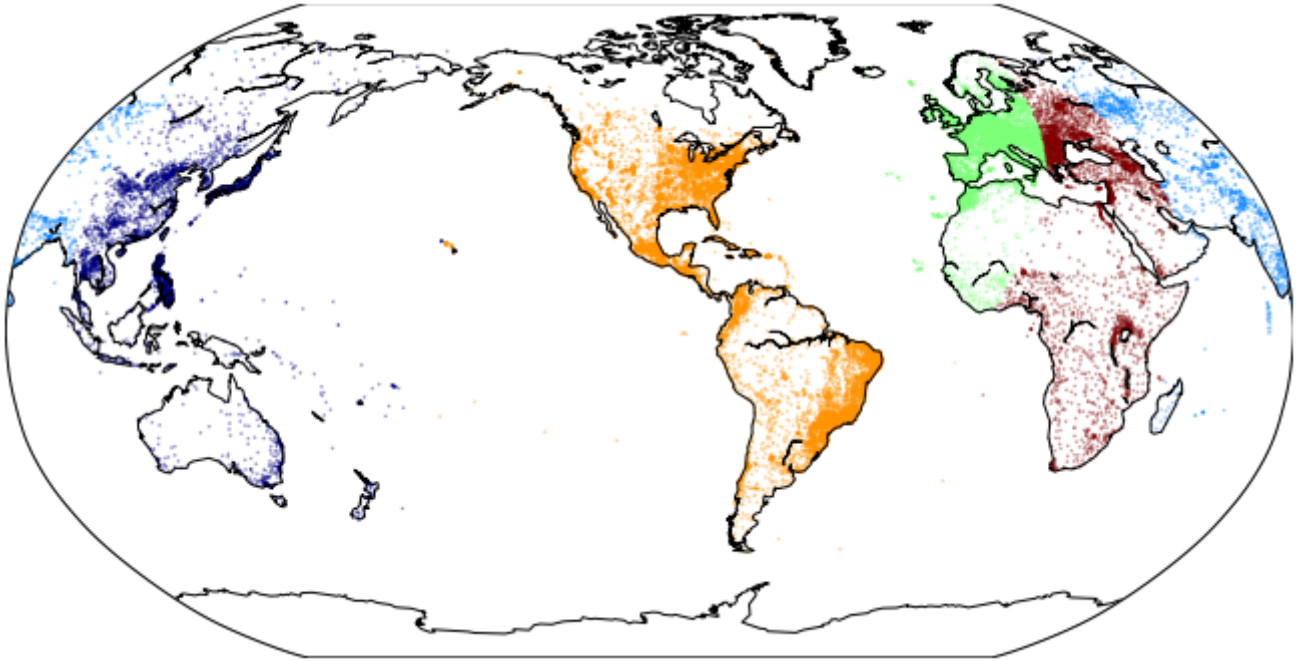
In [308]:
```
lng = []
for i in range(len(df.values)):
    lng.append(df.values[i][1])
    i+=1
```

In [309]:
```
#https://stackoverflow.com/questions/4913349/haversine-formula-in-python-bearing-and-distance-between-tw

from math import radians, cos, sin, asin, sqrt

def haversine(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance in kilometers between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])

    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 6371 # Radius of earth in kilometers. Use 3956 for miles. Determines return value units.
    return c * r
```

In [311]:
```
def centers(df,k):

    pts = [np.array(pt) for pt in zip(lat,lng)]
    centers = random.sample(pts, k)

    old_cluster_ids, cluster_ids = None, []

    while cluster_ids != old_cluster_ids:
        old_cluster_ids = list(cluster_ids)
        cluster_ids = []
        for pt in pts:
            min_cluster = -1
            min_dist = float('inf')
            for i, c in enumerate(centers):
                dist = haversine(pt[0],pt[1],c[0],c[1]) #My modification here
                if dist<min_dist:
                    min_cluster = i
                    min_dist = dist
            cluster_ids.append(min_cluster)
        df['cluster']= cluster_ids
        cluster_pts = [[pt for pt, cluster in zip(pts, cluster_ids) if cluster == match]
                       for match in range(k)]
        centers = [sum(pts)/len(pts) for pts in cluster_pts]
        #https://matplotlib.org/basemap/users/robin.html
    plt.figure(figsize=(12,6))
    map = Basemap(projection='robin',lat_0=0, lon_0=-100,resolution='c')
    map.drawmapboundary(fill_color='white') #The blue color in the HW instruction is hard to see the poi
    x,y = map(lng,lat)
    cluster = df['cluster'].tolist()
    map.scatter(x, y, c = cluster,cmap = 'jet',s = 0.6, alpha = 0.3)
    map.drawcoastlines()
    plt.show()
```

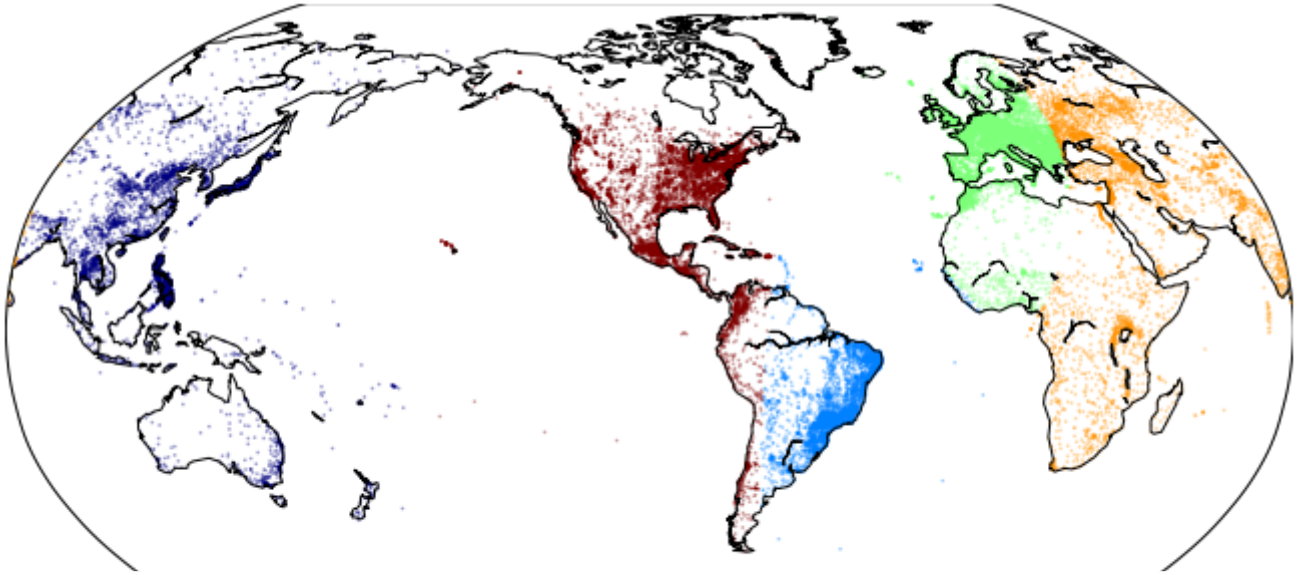
In [312]:
```
centers(df,5)#map1
```

```
<ipython-input-311-b23bdc214fbc>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```



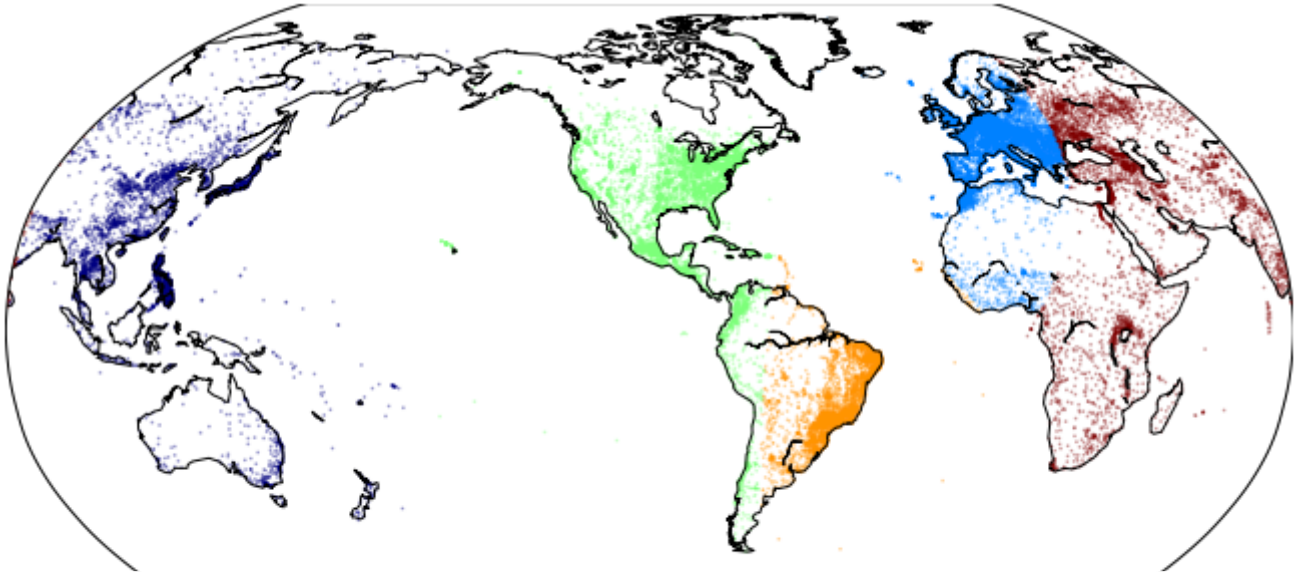
In [315]:
```
centers(df,5)#map2
```

g.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy)



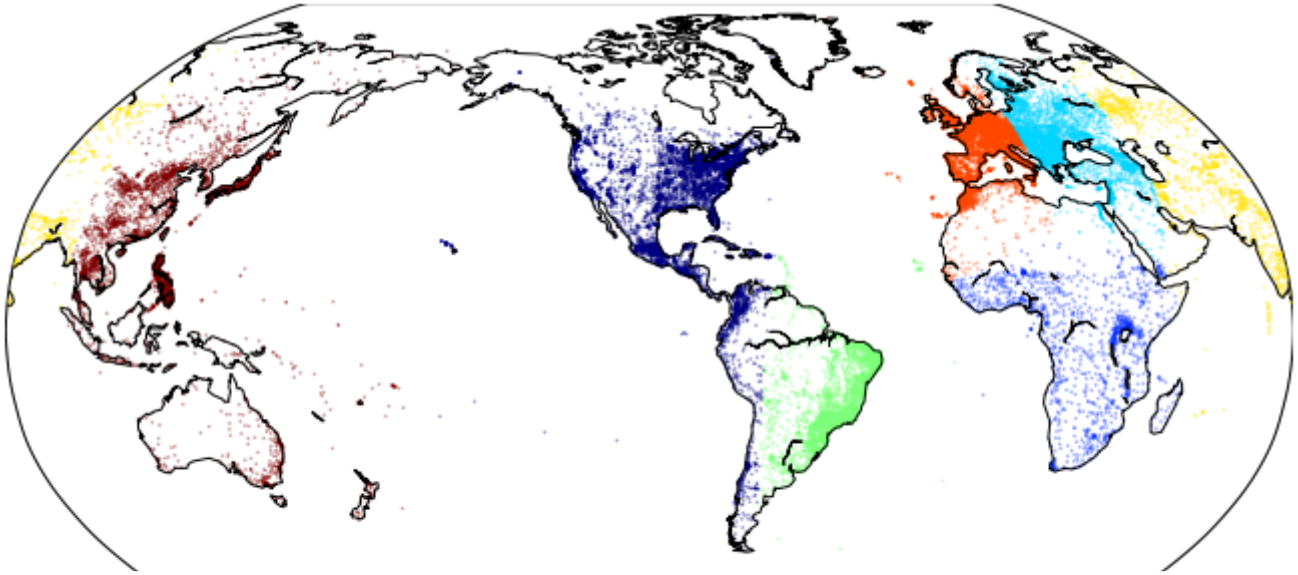
In [317]:
```
centers(df,5)#map3
```

```
g.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy)
```



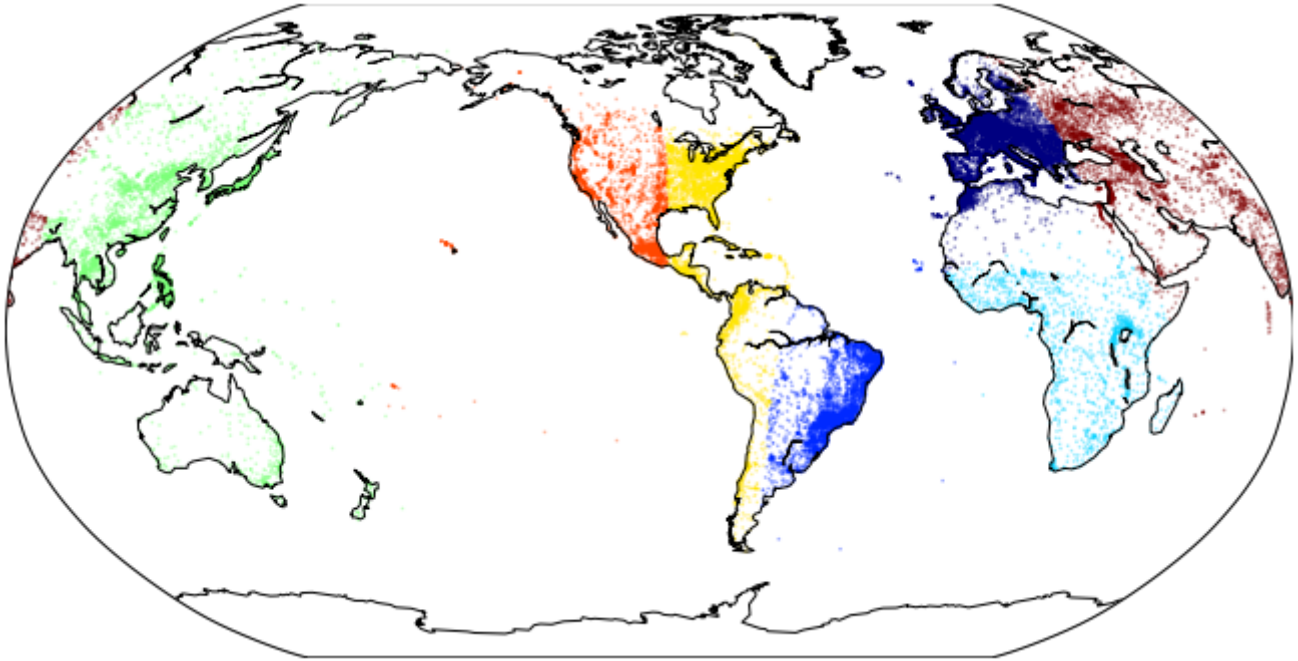
In [316]:
```
centers(df,7)#map4
```

```
g.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy)
```



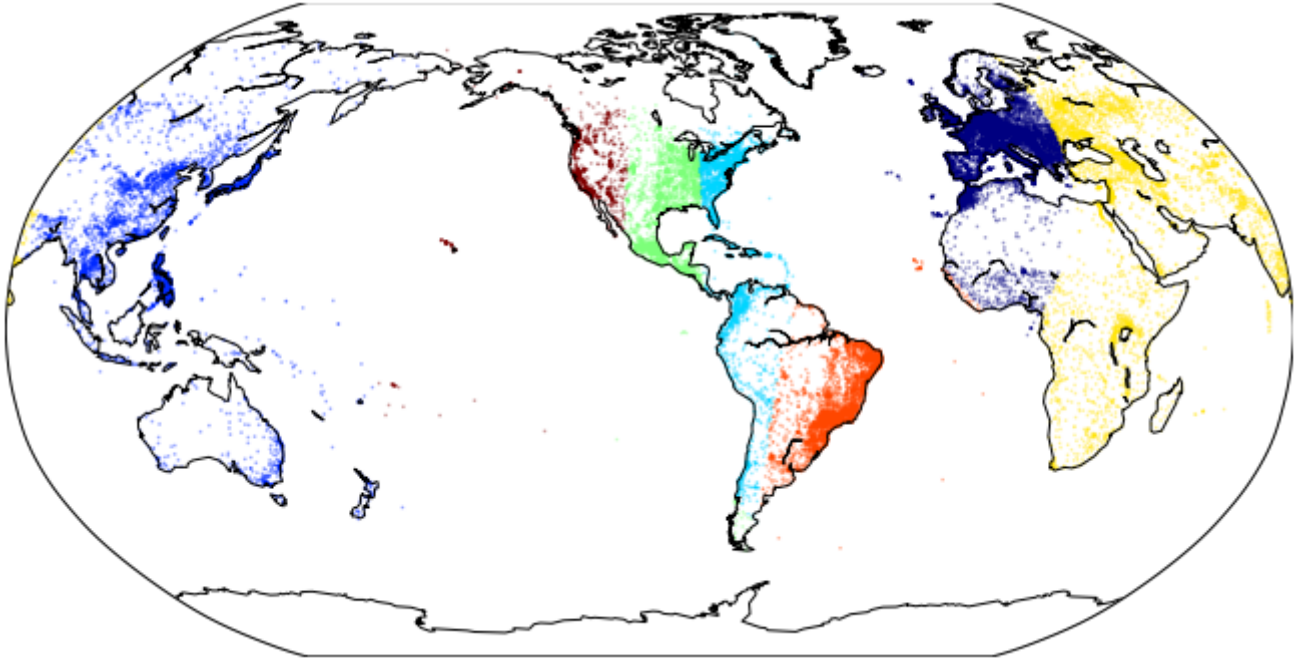
In [318]:
```
centers(df,7)#map5
```

```
<ipython-input-311-b23bdc214fbc>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```



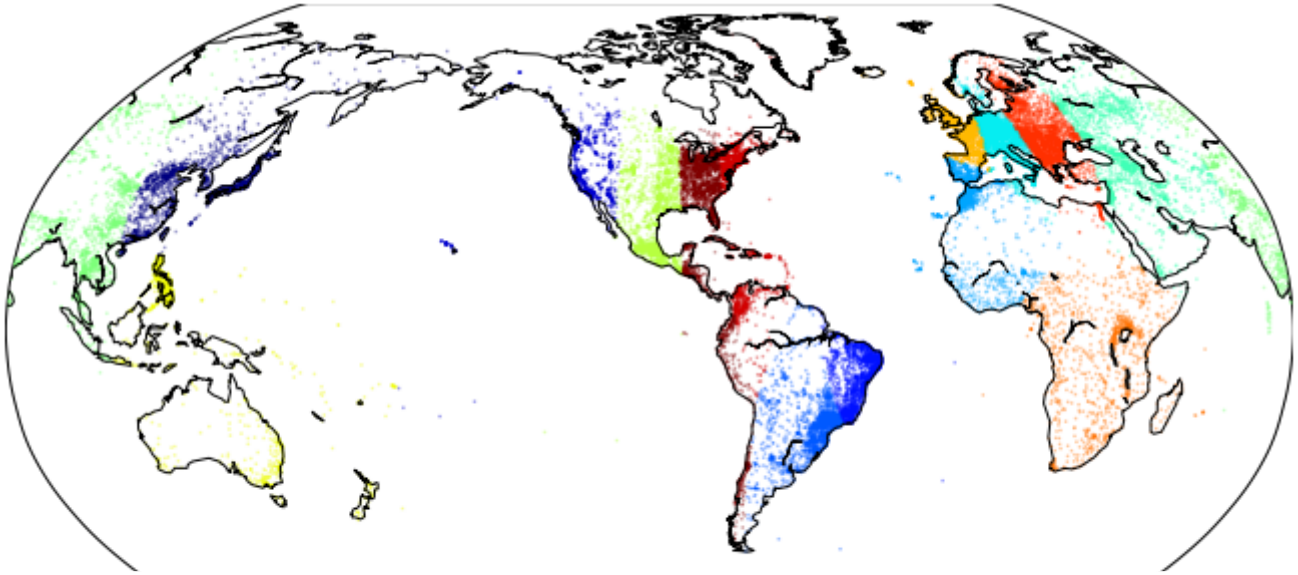
In [319]:
```
centers(df,7)#map6
```

```
<ipython-input-311-b23bdc214fbc>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```



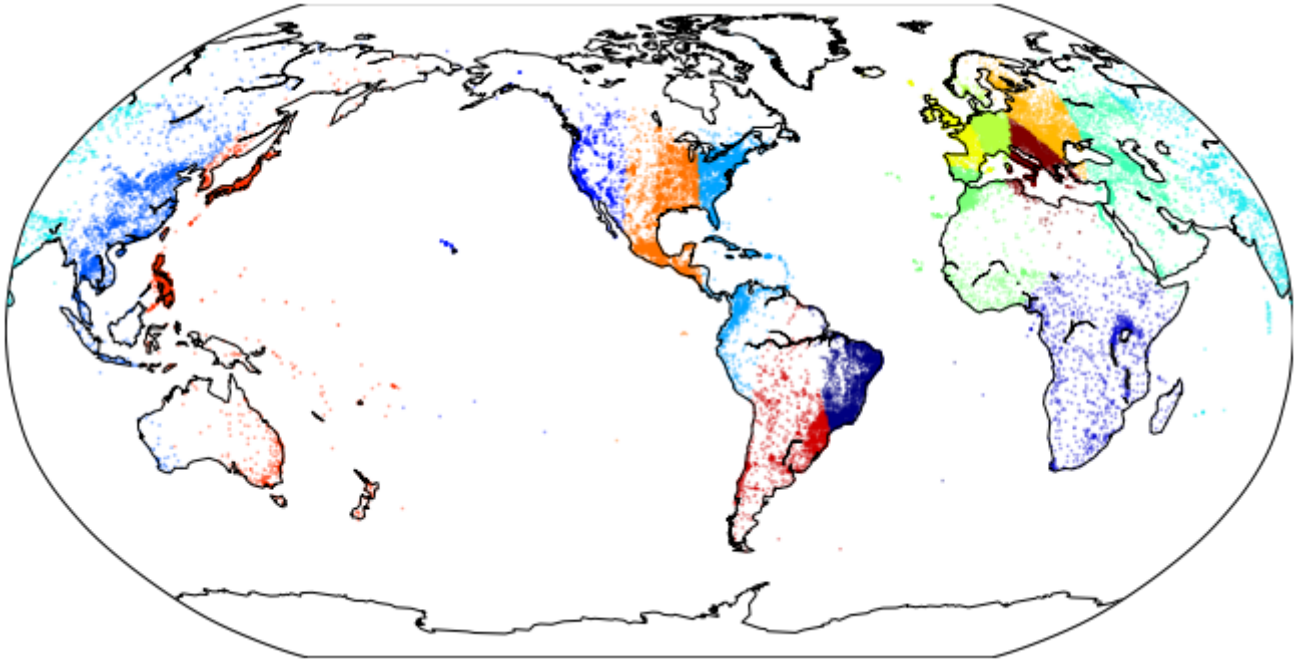
In [320]:
```
centers(df,15)#map7
```

```
g.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy)
```



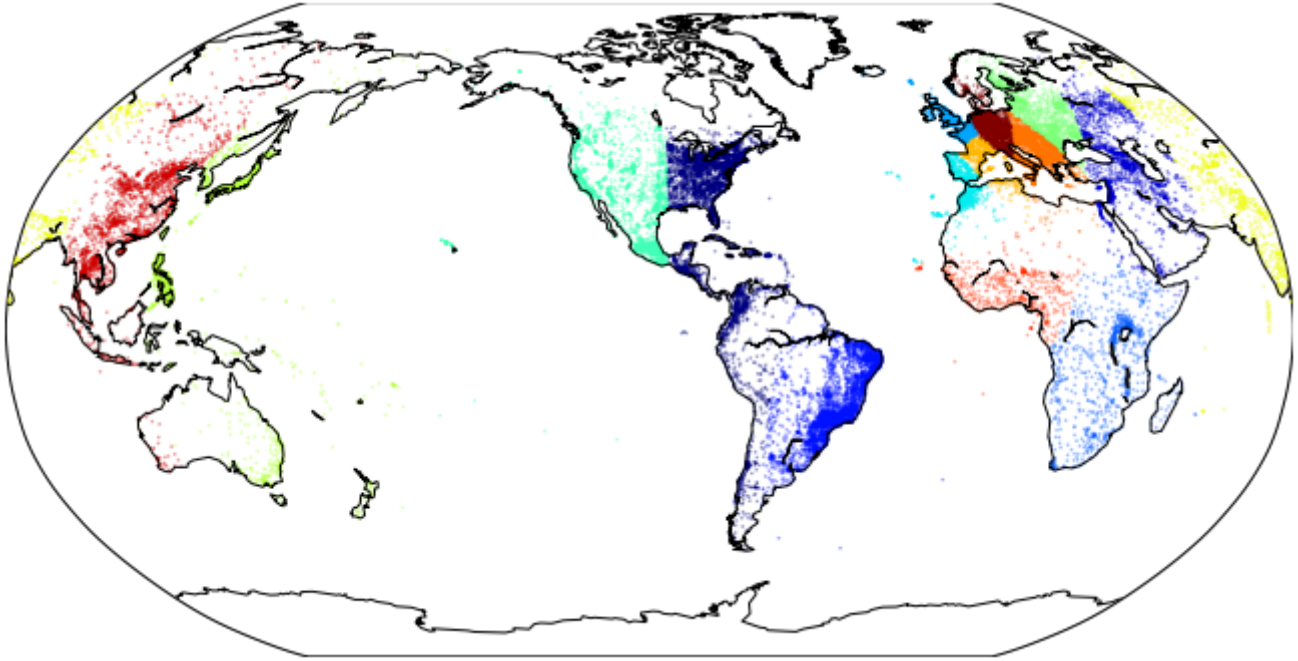
In [322]:
```
centers(df,15)#map8
```

```
<ipython-input-311-b23bdc214fbc>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```



In [323]:
```
centers(df,15)#map9
```

```
<ipython-input-311-b23bdc214fbc>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```



Run it several times to get a sense of the variation of clusters for each k (share your plots) (5 points); comment briefly on the diversity of results for each k. (5 points)

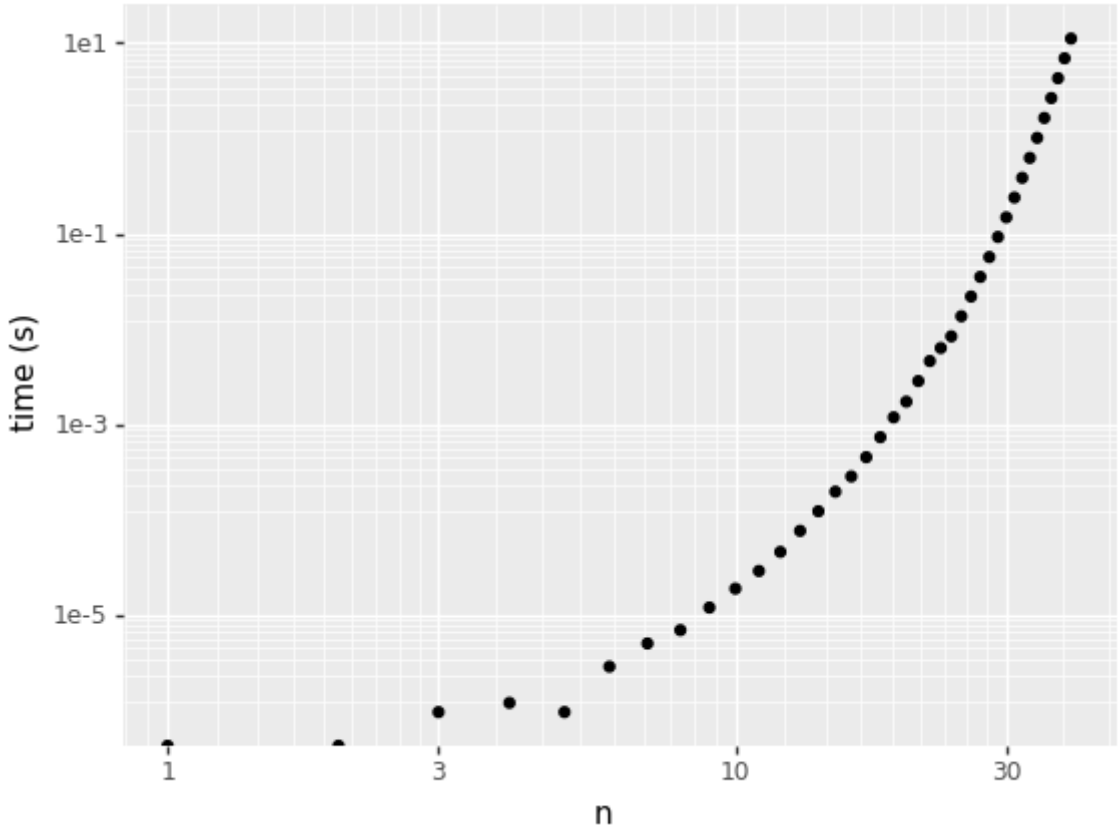
1) I run k = 5 three times. As showing in the map, the division of cities in Asia and Europe change a lot(map 1 has four divisions in this two areas; map 2 and 3 has three divisions). Besides, South America and North America are separated in the map 2&3. 2) I run k =7 three times. As showing in the map, the clusters changing is focusing on the divisions of Europe and North America(map4 has different

In [90]:
```
import time
import plotnine as p9
import pandas as pd
from tqdm import tqdm
```

In [ ]:
```
def timeit(function, *args, n=3):
    times = []
    for i in range(n):
        start = time.time()
        function(*args)
        end = time.time()
        times.append(time.time() - start)
    return min(times)
```

In [92]:
```
def r(n):
     # preconditions: n an integer >= 1
    if n in (1, 2):
        return 1
    return r(n - 1) + r(n - 2)

ns = range(1, 40)
times = [timeit(r, n) for n in tqdm(ns)]
(p9.ggplot(pd.DataFrame({'n': ns, 'time (s)': times}),
           p9.aes(x='n', y='time (s)'))
+ p9.geom_point()
+ p9.scale_x_continuous(trans='log10')
+ p9.scale_y_continuous(trans='log10'))
```

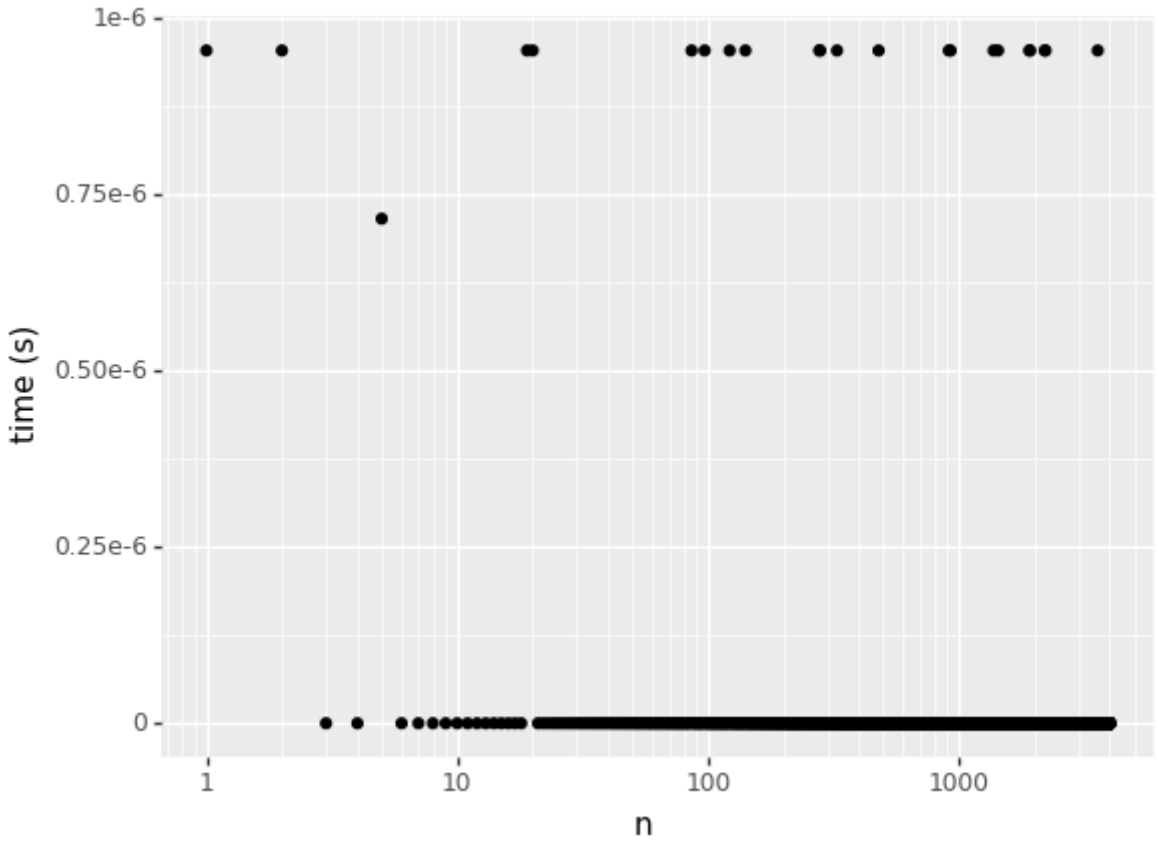
```
  0%|          | 0/39 [00:00<?, ?it/s]

 64%|██████████     | 25/39 [00:00<00:00, 196.50it/s]

 64%|██████████     | 25/39 [00:15<00:00, 196.50it/s]

 92%|██████████████▋  | 36/39 [00:20<00:02,  1.37it/s]

 95%|███████████████▏ | 37/39 [00:33<00:02,  1.31s/it]

 97%|███████████████▌ | 38/39 [00:54<00:02,  2.55s/it]

100%|███████████████▉| 39/39 [01:27<00:00,  2.25s/it]
  0%|          | 0/49 [1:30:48<?, ?it/s]
/opt/anaconda3/lib/python3.8/site-packages/pandas/core/arraylike.py:358: RuntimeWarning: divide by zer
o encountered in log10
  0%|          | 0/49 [1:31:35<?, ?it/s]
```



Out[92]: <ggplot: (8786140390988)>

In [101]:
```
from functools import lru_cache

@lru_cache()
def r(n):
    # preconditions: n an integer >= 1
    if n in (1, 2):
        return 1
    return r(n - 1) + r(n - 2)
ns = range(1, 4000)
times = [timeit(r, n) for n in tqdm(ns)]
(p9.ggplot(pd.DataFrame({'n': ns, 'time (s)': times}),
           p9.aes(x='n', y='time (s)'))
+ p9.geom_point()
+ p9.scale_x_continuous(trans='log10'))
```

```
100%|██████████| 3999/3999 [00:00<00:00, 192899.78it/s]
```



In [232]:
```
import numpy as np

a = 'tgcatcgagaccctacgtgac'
b = 'actagacctagcatcgac'
match = 1
mismatch = 1
gap = 1
def best_matrix(a,b,match,mismatch,gap): #a:sequence1, b:sequence2
    bm = np.zeros((len(a)+1,len(b)+1),int)#best_matrix
    max_score = 0
    max_pos = 0
    for i in range(len(a)+1):
        for j in range(len(b)+1):
            if i == 0 or j == 0:
                bm[i][j] = 0  #Set the gap in the matrix
            else:
                match_score = bm[i - 1, j - 1] + (match if a[i - 1] == b[j - 1] else - mismatch)
                gap1 = bm[i - 1, j] - gap
                gap2 = bm[i, j - 1] - gap
                score = max(match_score, gap1, gap2, 0)
                bm[i, j] = score
                if score > max_score:
                    max_score = score
                    max_pos = (i, j) #max_position
    print(bm)
    return(bm,max_pos)

#https://tiefenauer.github.io/blog/smith-waterman/

# test: http://rna.informatik.uni-freiburg.de/Teaching/index.jsp?toolName=Smith-Waterman
```

In [242]:
```
#diagonal: match
#up: gap in b
#left: gap in a
#https://stackoverflow.com/questions/23400317/smith-waterman-algorithm-to-generate-matrix-in-python
def go(bm, x, y): #x,y: position in matrix
    diagonal = bm[x-1][y-1]
    up = bm[x-1][y]
    left = bm[x][y-1]
    if diagonal >= up and diagonal >= left:
        if diagonal != 0:
            return "diagonal"
        else:
            return 0
    elif up > diagonal and up >= left:
        if up != 0:
            return "up"
        else:
            return 0
    elif left > diagonal and left > up:
        if left != 0:
            return "left"
        else:
            return 0
    else:
        print("WRONG")
```

In [254]:
```
#https://stackoverflow.com/questions/12666494/how-do-i-decide-which-way-to-backtrack-in-the-smith-waterr
def traceback(a, b, bm, start_pos):
    aligned_seq1 = []
    aligned_seq2 = []
    x, y = start_pos
    move = go(bm, x, y)
    while move != 0:
        if move == "diagonal":
            aligned_seq1.append(a[x-1])
            aligned_seq2.append(b[y-1])
            x = x - 1
            y = y - 1
        elif move == "up":
            aligned_seq1.append(a[x-1])
            aligned_seq2.append('-')
            x = x - 1
        else: #move left
            aligned_seq1.append('-')
            aligned_seq2.append(b[y-1])
            y = y - 1

        move = go(bm, x, y)

    aligned_seq1.append(a[x - 1])
    aligned_seq2.append(b[y - 1])

    return ''.join(reversed(aligned_seq1)), ''.join(reversed(aligned_seq2))
```

In [255]:
```
def align(a,b, match,mismatch,gap):
    bm, max_pos = best_matrix(a,b, match,mismatch,gap)
    score = bm[max_pos[0]][max_pos[1]]
    a, b = traceback(a, b, bm, max_pos)
    print ("seq1 = ", a, ", seq2 = ", b, ", score =" , score)
```

In [256]:
```
align('tgcatcgagaccctacgtgac', 'actagacctagcatcgac', 1, 1, 1)
```

```
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0]
 [0 0 1 0 0 0 0 1 1 0 0 0 2 1 0 1 0 0 1 1]
 [0 1 0 0 1 0 1 0 0 0 1 0 1 3 2 1 0 1 0 1]
 [0 0 0 1 0 0 0 0 0 1 0 0 0 2 4 3 2 1 0 1]
 [0 0 1 0 0 0 0 1 1 0 0 0 1 1 3 5 4 3 2 2]
 [0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 2 4 6 5 4]
 [0 1 0 0 1 0 2 1 0 0 1 0 0 1 1 3 5 7 6 5]
 [0 0 0 0 0 2 1 1 0 0 0 2 1 0 0 2 4 6 6 5]
 [0 1 0 0 1 1 3 2 1 0 1 1 1 2 1 1 3 5 5 6]
 [0 0 2 1 0 0 2 4 3 2 1 0 2 1 1 2 2 4 6 5]
 [0 0 1 1 0 0 1 3 5 4 3 2 1 1 0 2 1 3 5 4]
 [0 0 1 0 0 0 0 2 4 4 3 2 3 2 1 1 1 2 4 3]
 [0 0 0 2 1 0 0 1 3 5 4 3 2 2 3 2 1 1 3 2]
 [0 1 0 1 3 2 1 0 2 4 6 5 4 3 2 2 1 2 2 3]
 [0 0 2 1 2 2 1 2 1 3 5 5 6 5 4 3 2 1 3 2]
 [0 0 1 1 1 3 2 1 1 2 4 6 5 5 4 3 4 3 2 3]
 [0 0 0 2 1 2 2 1 0 2 3 5 5 4 6 5 4 3 2 3]
 [0 0 0 1 1 2 1 1 0 1 2 4 4 4 5 5 6 5 4 3]
 [0 1 0 0 2 1 3 2 1 0 2 3 3 5 4 4 5 7 6 5]
 [0 0 2 1 1 1 2 4 3 2 1 2 4 4 4 5 4 6 8 7]]
seq1 =  agacccta-cgt-gac , seq2 =  agacc-tagcatcgac , score = 8
```