A note on grading this assignment. Due to the overwhelming confusion in interpreting the questions in this assignment on Piazza, I have created two versions of this assignment. The primary difference is in the interpretation of what probability we are calculating in problem 2. You will see two sets of plots, the ones with my second version will have “ANSWER 2” in the title, the first version will have no such designation. Under each question answer, there will be two answers depending on how the question applies to the interpretation of calculating the probability in problem 2.

Please keep the above in mind as you are grading and pick the one that fits your particular interpretation of the assignment since there was not a consensus among the course staff at the time of submission. Thanks!

```
import random
import numpy as np
import matplotlib.pyplot as plt

"""Problem 0, Generate regions between [-10, 10]"""
def generateRegions(numRegions, low, high):
    regions = [sorted((np.random.uniform(low, high), np.random.uniform(low, high))) for n
in range(numRegions)]
    return regions

regionList = generateRegions(10000, -10, 10)

"""Problem 1"""
def contains(region, point):
    """Takes a region as a 2 element list or tuple and returns if point is in the region"""
    return (region[0] <= point) and (region[1] >= point)

def size(region):
    return region[1] - region[0]

"""Problem 2:  What is the probability of getting x=1 for regions containing x=0?
    ANSWER: 0.10613579023117636 for TA interpretation conditionalProb function
    ANSWER: 0.7749117017530301 for TA interpretation conditionalProbTwo function"""

def conditionalProb(regions, x, y):
    """Takes a list of regions as intervals and returns P(y ∈ r | x ∈ r). This
        is one interpretation of what the TAs said P2 should be"""

    xRegions = 0
    totalProb = 0
    intervalProb = []

    for r in regions:
        if contains(r, x):
            xRegions += 1

            if contains(r, y):
                intervalProb += [1 / size(r)]

    if intervalProb and xRegions:
        condProb = 1 / xRegions
        intervalProb = [abs(condProb * p) for p in intervalProb]
        totalProb = sum(intervalProb)

        return totalProb
    else:
        return 0
```

```
def conditionalProbTwo(regions, x, y):
    """This is another interpretation of what the TAs said P2 should be"""
    xRegions = 0
    totalProb = 0
    xyRegions = 0

    for r in regions:
        if contains(r, x):
            xRegions += 1 / size(r)

            if contains(r, y):
                xyRegions += 1 / size(r)

    if xRegions and xyRegions:
        totalProb = xyRegions / xRegions
    return totalProb

# print(conditionalProb(regionList, 0, 1))
# print(conditionalProbTwo(regionList, 0, 1))

# regionList = generateRegions(10000, -10, 10)

# print(conditionalProbTwo(regionList, 0, 2))

"""Problem 3: Plot the probability of getting x for x ranging from 0 to 10, for regions
    containing x=0. What does this function look like? Write a sentence explaining why
    intuitively.

    Given the plethora of TA interpretations on Piazza, I have answered and coded both
of
    the most popular ones. Please see the title of the graph for ANSWER 2 plots.

    ANSWER 1: The maximum probability is the chance of choosing x=0 given x=0 ∈ r at
about
    0.14 and then the graph decays somewhat exponentially though it doesn't look
perfect.
    All regions with x=0 have a chance of getting x=0 but the other are not guaranteed to
    contain x=1-10. The farther out from x=0 you go, the less likely it is that the region
    will contain that number.

    ANSWER 2: The maximum probability is the chance of choosing x=0 given x=0 ∈ r is
1.0
    and then the graph decays somewhat exponentially though it doesn't look perfect.
    All regions with x=0 are guaranteed to have x=0 (probability of 1) and less likely to
    contain x=1-10 and x=0. The farther out from x=0 you go, the less likely it is that the
region
```

```
    will contain that number. There is likely some average width for a region given a
    uniform distribution, numbers in that average width are more likely with diminishing
    probability as you go towards the bounds of the average width and beyond. For
example,
    if it is given x=0 is in the region and the average width is 2, -1 to 1 is more likely
    than -2 and 2 even though there will be some wider regions that include -2 to 2. """

def plotProb(z, start, end, regions, saveName=None, scale='linear', color='blue',
leg=None, legTitle='Enter Title', func=conditionalProb):
    probList = []
    xAxis = []

    for i in range(start, end + 1):
        xAxis += [i]
        probList += [func(regions, z, i)]

    x, y = xAxis, probList

    plt.plot(x, y, c=color, label=leg)
    plt.yscale(scale)
    plt.title('P(x ∈ r I x=' + str(z) + ' ∈ r); ANSWER 2')
    plt.xlabel('X Range')
    plt.ylabel('Probability')

    if leg:
        plt.legend(loc=0, title=legTitle)

    if saveName:
        plt.savefig(saveName + '.pdf')

    plt.show()

# region10k = generateRegions(10000, -10, 10)
# # plotProb(0, 0, 10, region20k, saveName='a6_p3_20k', legTitle='10000 Regions')
# plotProb(0, 0, 10, region10k,  leg='Answer 2', color='red', legTitle='10000 Regions',
func=conditionalProbTwo)

"""Problem 4: One way to check if the curve has an exponential decrease is to plot a
    logarithmic y axis and look for a straight line. Why does this check if the curve
    is exponential?

    ANSWER: Plotting y values on a log scale gives all intervals of y the same 'tick'
    distance. So an interval of 1 to 10 has the same y distance as 10 to 100; log scale
    gives all y values a constant ratio between ticks. Assuming the x and y axis have
    the same tick distances and number of intervals, an exponential decrease should
    appear as a straight line. """
# plotProb(0, 0, 10, regionList, saveName='a6_p4_ans2', scale='log',
```

```
func=conditionalProbTwo)

"""Problem 5: Plot Q3 with a logarithmic y axis for x ranging from -5 to 5, and x
    ranging from -10 to 10. What do these two plots show? How do you interpret them?
    Explain in a few sentences.

    ANSWER: For both versions of generating the probability, the plots are both
approximately straight lines
    (matplotlib doesn't like to give equal scale to linear and log axis) on both sides of 0
which has the maximum
    probability. This means that the probabilities exhibit (approximately) exponential
decay.
    The y-axis scale is shorter for the -5 to 5 plot as you are more likely to get each
number than you
    are for the -10 to 10 plot. The plots overlap each other when plotted together showing
that the probabilities
    for -5 to 5 are the same even if the range we check is -10 to 10."""

# regionList = generateRegions(10000, -10, 10)
# plotProb(0, -10, 10, regionList, saveName='a6p5_-10to10_ans2', scale='log',
color='red', leg='[-10, 10]', func=conditionalProbTwo, legTitle='X Range')
# plotProb(0, -5, 5, regionList, saveName='a6p5_-5to5_ans2', scale='log', leg='[-5, 5]',
func=conditionalProbTwo, legTitle='X Range')

"""Problem 6: In previous questions, we've been assuming that people implement the
law
    perfectly and we have been trying to approximate their behavior using 10,000
regions.
    However, people themselves have limited resources. What if people themselves only
used
    a few consequential regions in order to compute generalizations? Re-plot Question 3
    using only 10, 100, and 1000 consequential regions. What patterns do you see?

    ANSWER: The curves are not as smooth going from 10 to 1000 as when there were
10000
    regions and the probabilities from 0 to 10 should all be the same but they aren't;
some
    x higher/lower for a given distance from 0. As the number of regions approaches +inf,
the
    law of large numbers takes effect and the curve begins to reflect the true probability
of
    the dataset. For a small number of regions, the probabilities are very inaccurate.

    With fewer regions to choose from, people are more likely to generalize to a region
    that doesn't fit the subject; they will have less accuracy. We are using a uniform
    distribution meaning that all numbers should be equally as likely in the random
sample.
```

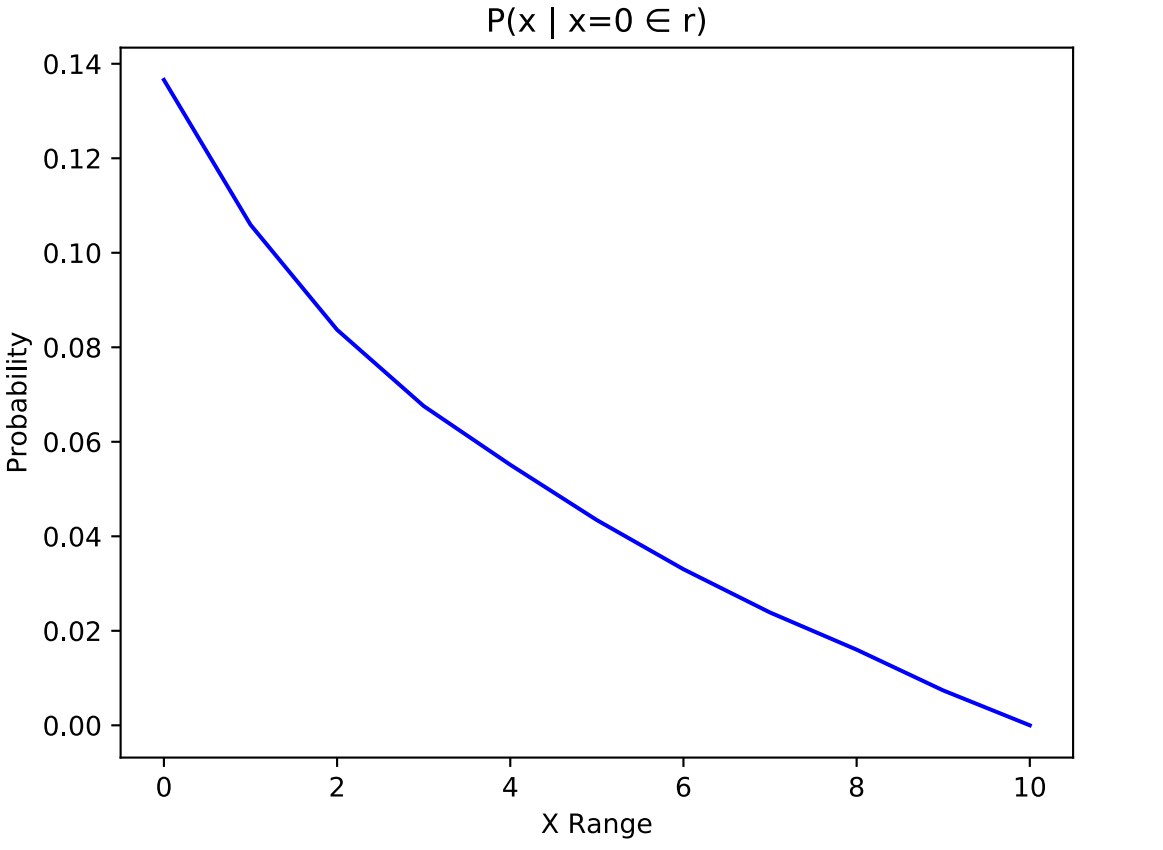
Generating more regions increases the likelihood that we will see the true probability distribution; ie generalize to the correct category.

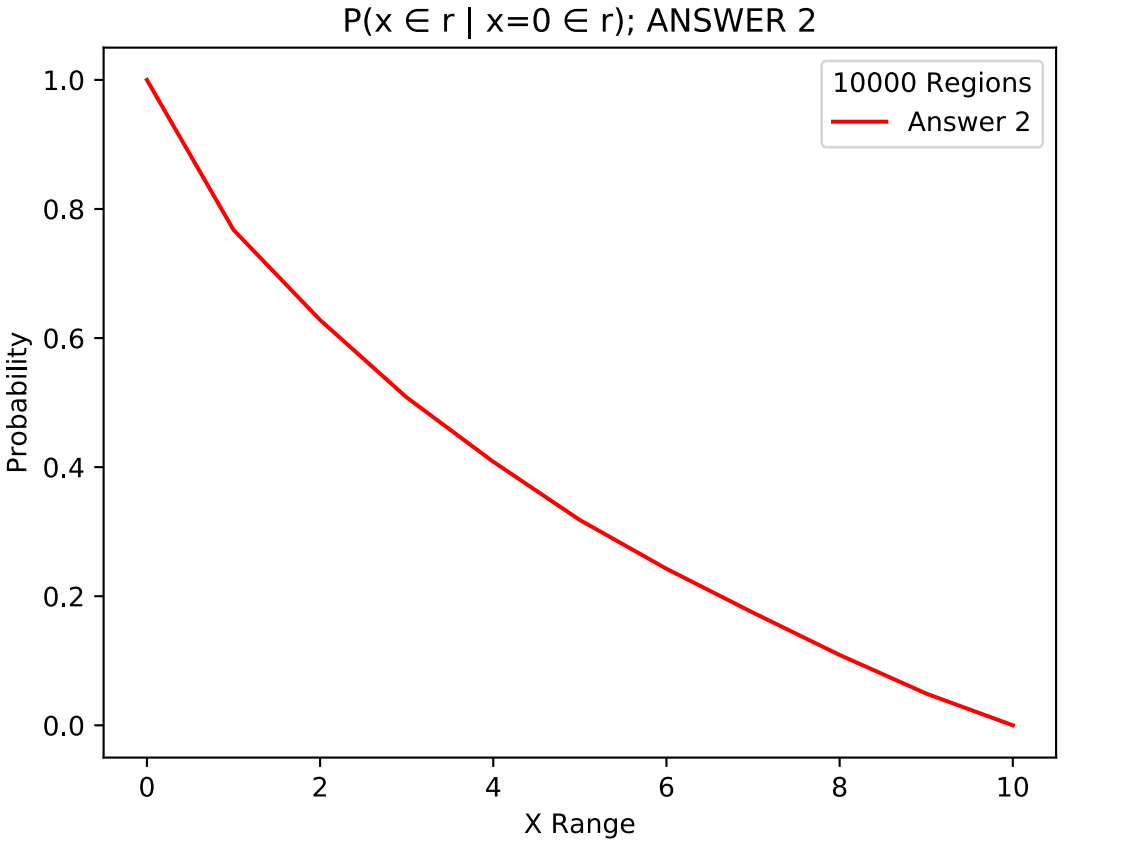
An extreme example would be if people only had one consequential region for animals.
Every animal would fit into that one category. Cats and people would be the same as would
cows, dogs, birds, etc.
"""

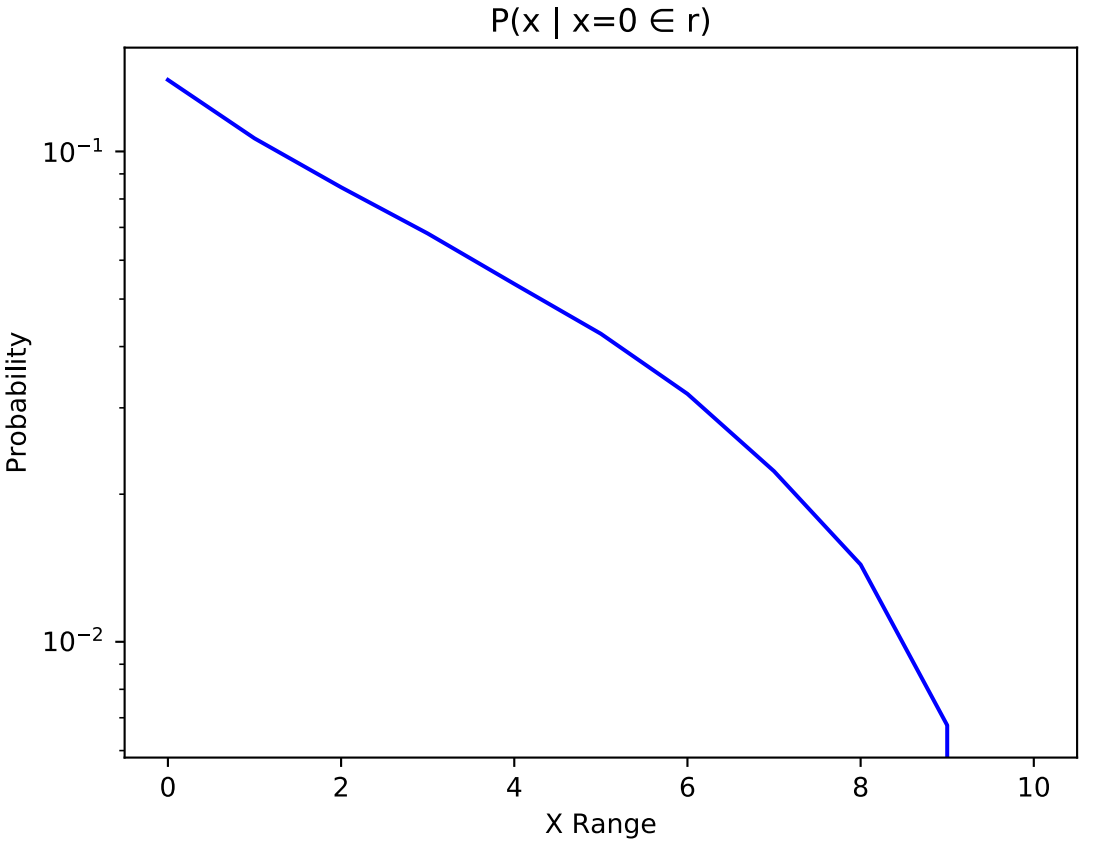
```
# region10 = generateRegions(10, -10, 10)
# region100 = generateRegions(100, -10, 10)
# region1000 = generateRegions(1000, -10, 10)
# plotProb(0, 0, 10, region10, color='red', leg='10 Regions', legTitle='Number of
Regions', func=conditionalProbTwo)
# plotProb(0, 0, 10, region100, color='green', leg='100 Regions', legTitle='Number of
Regions', func=conditionalProbTwo)
# plotProb(0, 0, 10, region1000, saveName='a6p6_ans2',color='blue',leg='1000
Regions', legTitle='Number of Regions', func=conditionalProbTwo)
# plt.show()
```

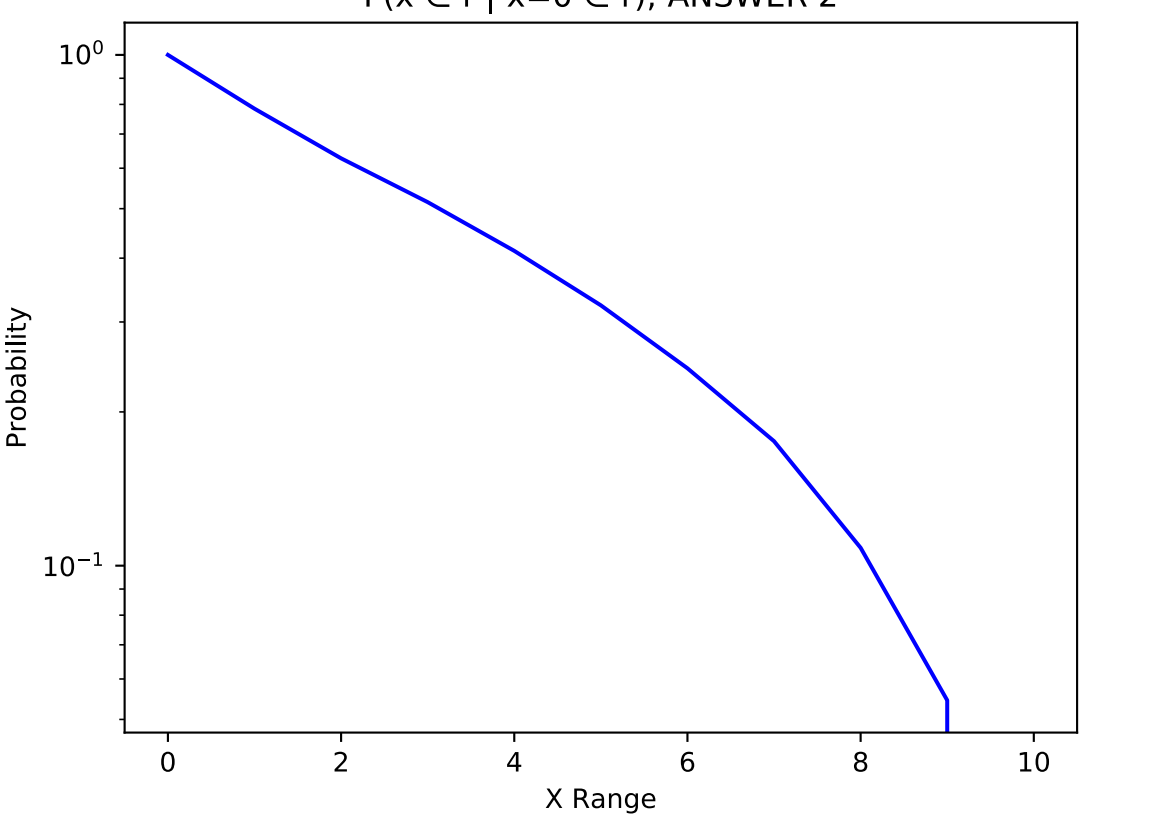
"""Problem 7: Describe a way you could test how many consequential regions people actually
made use of in this kind of generalization. Could you tell the difference between 10 and
10,000? Could you tell the difference between 10,000 and 20,000, why or why not?

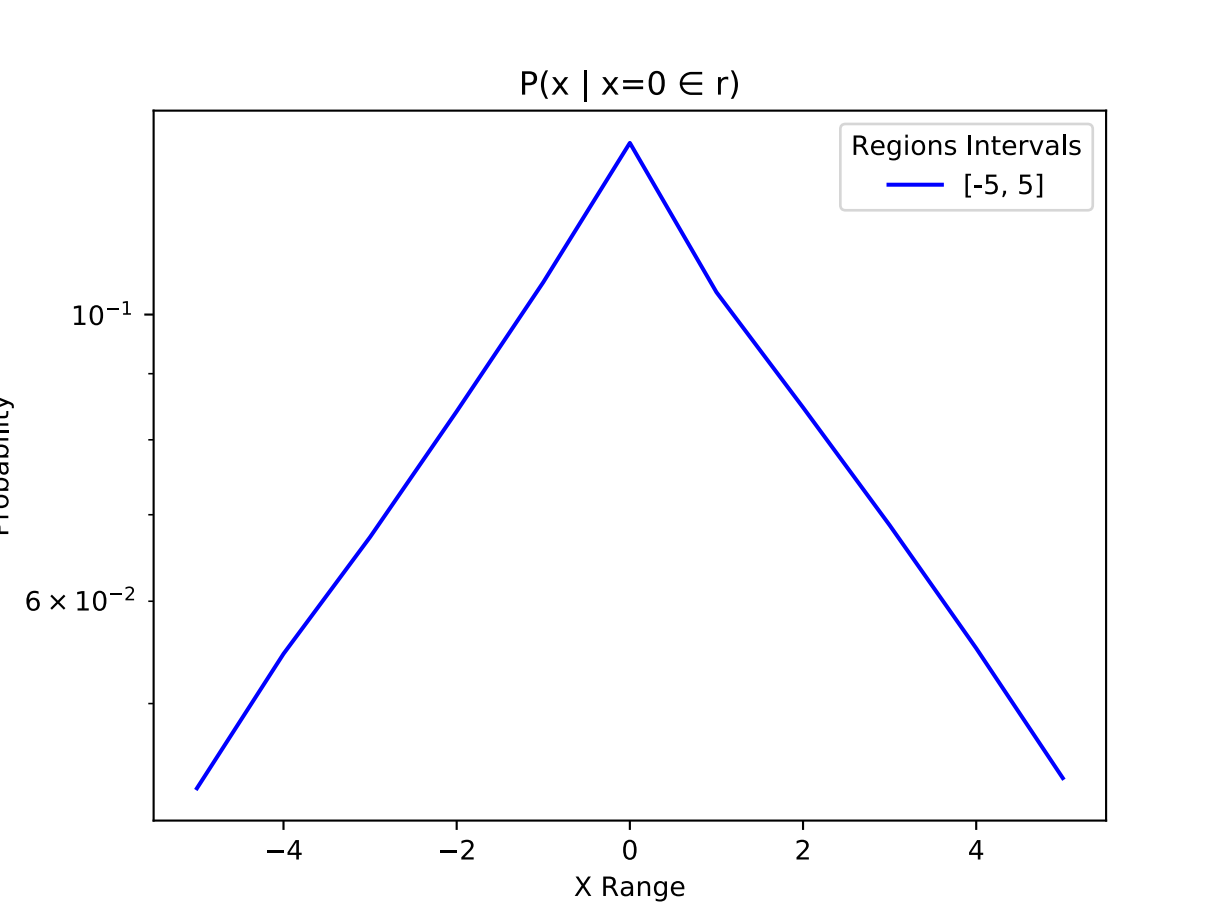
ANSWER: You could make a standard plot of regions ranging from 10 to 20000 or more
and then plot the data from people doing generalization. The plot of people data
could then be compared to other known plots to see how close they were. Telling the
difference between 10,000 and 20,000 would be difficult visually. I actually plotted
20,000 and compared it to my 10,000 plot from problem 3 and I could not tell a difference
just by looking. I imagine that there is a fancier way to do the comparison on a data
level that could check exponential decay at many points on the 10,000 curve
and compare it to the points on the 20,000 curve. For exponential decaying graphs/ data,
there should be a fraction that, when multiplied by the y value at some x value,
equals the y value at x + 1. The closer the y is to the next y using this method,
the closer to having exponential decay a curve is. This could be used to compare known
data with different numbers of regions to people data to approximate the number of
regions people are using. """

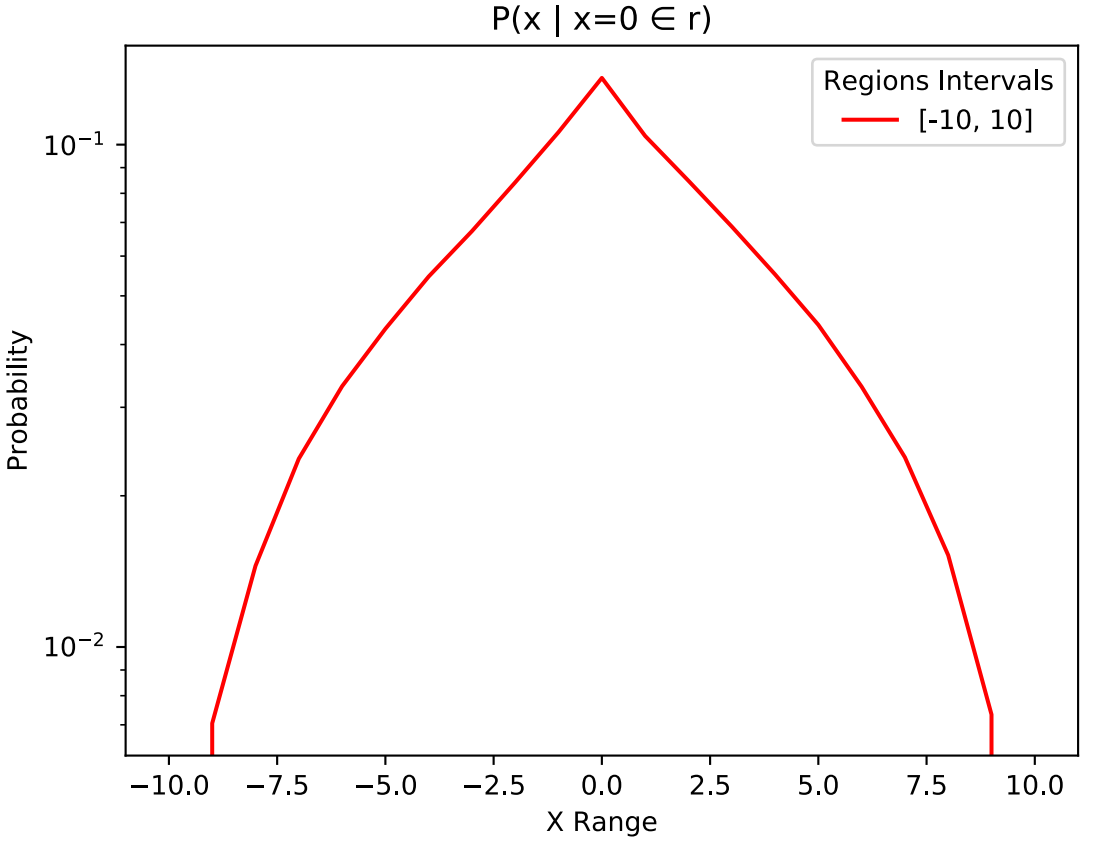

P(x | x=0 ∈ r)
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Probability
0
2
4
6
8
10
X Range

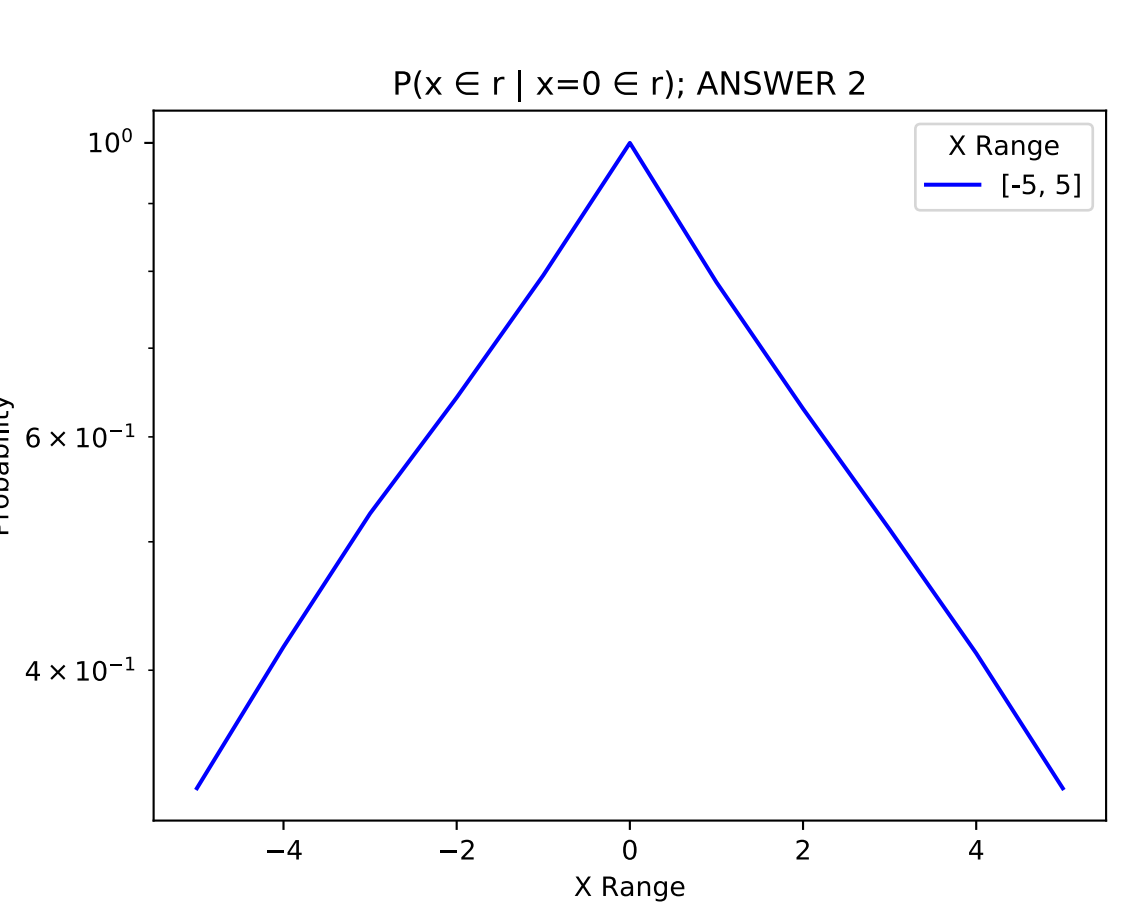

P(x ∈ r | x=0 ∈ r); ANSWER 2
10000 Regions
Answer 2
Probability
X Range
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10

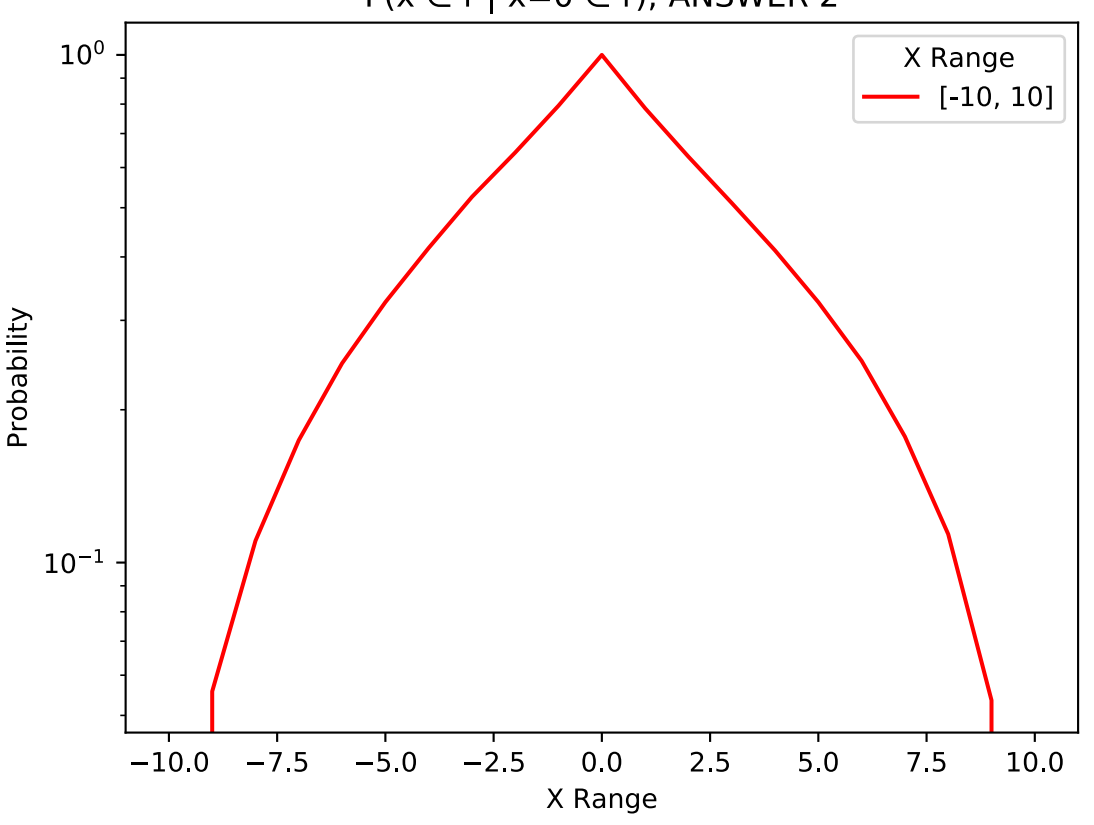

P(x | x=0 ∈ r)
Probability
10^{-1}
10^{-2}
0
2
4
6
8
10
X Range

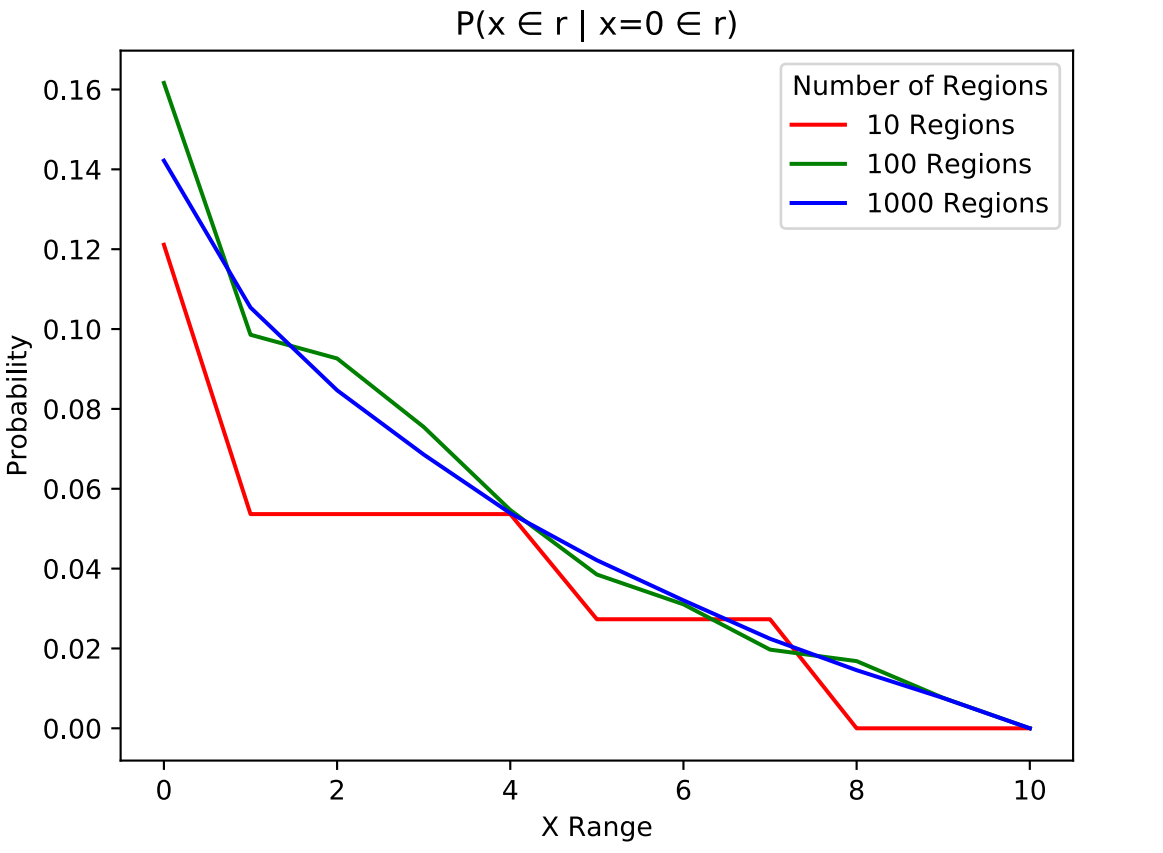

P(x ∈ r | x=0 ∈ r); ANSWER 2
Probability
10^0
10^{-1}
0
2
4
6
8
10
X Range

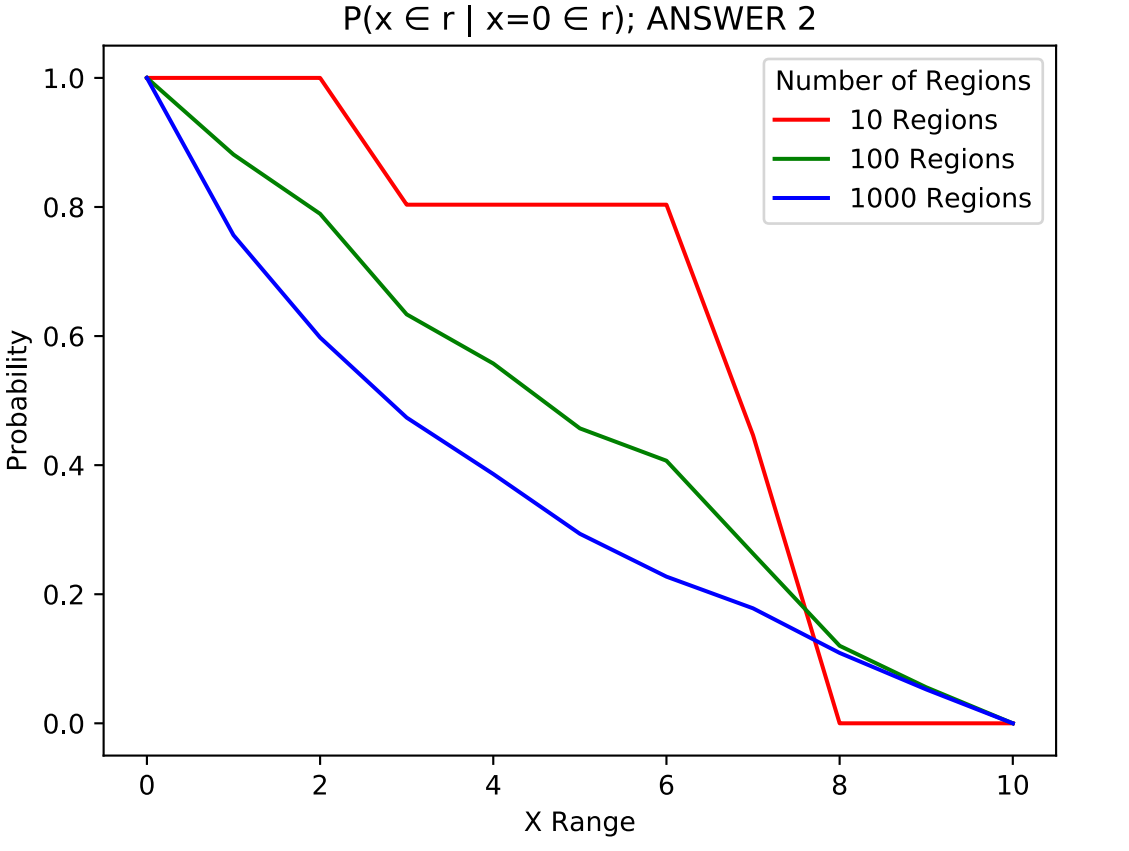

P(x | x=0 ∈ r)
Regions Intervals
[-5, 5]
10^{-1}
6×10^{-2}
Probability
−4
−2
0
2
4
X Range


P(x | x=0 ∈ r)
Regions Intervals
[-10, 10]
Probability
10^{-1}
10^{-2}
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
X Range


P(x ∈ r | x=0 ∈ r); ANSWER 2
X Range
[-5, 5]
10^0
6×10^{-1}
4×10^{-1}
Probability
−4
−2
0
2
4
X Range


P(x ∈ r | x=0 ∈ r); ANSWER 2
X Range
[-10, 10]
10^0
10^{-1}
Probability
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
X Range


P(x ∈ r | x=0 ∈ r)
Number of Regions
10 Regions
100 Regions
1000 Regions
Probability
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8
10
X Range


P(x ∈ r | x=0 ∈ r); ANSWER 2
Number of Regions
10 Regions
100 Regions
1000 Regions
Probability
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
X Range