Assignment 1 - Probability, Linear Algebra, Programming, and Git

Yifei Wang

Netid: *yw323*

Probability and Statistics Theory

1

Let $f(x) = \begin{cases} 0 & x < 0 \\ \alpha x^2 & 0 \leq x \leq 2 \\ 0 & 2 < x \end{cases}$

For what value of α is $f(x)$ a valid probability density function?

Note: for all assignments, write out all equations and math for all assignments using markdown and LaTeX (https://tobi.oetiker.ch/lshort/lshort.pdf) and show all work

ANSWER

For a valid probability density function, $f(x)$ should satisfy that $\int_{-\infty}^{+\infty} f(x) = 1$. Thus we have,

$$\begin{aligned} 1 &= \int_{-\infty}^{+\infty} f(x)\mathrm{d}x \\ &= \int_{-\infty}^{0} 0\mathrm{d}x + \int_{0}^{2} \alpha x^2\,\mathrm{d}x + \int_{2}^{\infty} 0\mathrm{d}x \\ &= \frac{8}{3}\alpha \end{aligned}$$

Therefore, we have that $\alpha = \frac{3}{8}$

This also satisfies the requirement of $f(x) > 0$ for a probability density function.

2

What is the cumulative distribution function (CDF) that corresponds to the following probability distribution function? Please state the value of the CDF for all possible values of x.

$$f(x) = \begin{cases} \frac{1}{3} & 0 < x < 3 \\ 0 & \text{otherwise} \end{cases}$$

ANSWER

Denote the CDF of $f(x)$ as $P(X)$. For any $0 < X < 3$ we have,

$$\begin{aligned} P(X) &= \int_{-\infty}^{X} f(x)\mathrm{d}x \\ &= \int_{-\infty}^{0} 0\mathrm{d}x + \int_{0}^{X} \frac{1}{3}\mathrm{d}x \\ &= \frac{X}{3} \end{aligned}$$

Therefore for any $X \in \mathbb{R}$ we have,

$$P(X) = \begin{cases} 0 & X \leq 0 \\ \frac{X}{3} & 0 < X \leq 3 \\ 1 & X > 3 \end{cases}$$

3

For the probability distribution function for the random variable X,

$$f(x) = \begin{cases} \frac{1}{3} & 0 < x < 3 \\ 0 & \text{otherwise} \end{cases}$$

what is the (a) expected value and (b) variance of X. *Show all work*.

ANSWER

(a) Expected value:

$$\begin{aligned}\mathrm{E}[X] &= \int_{-\infty}^{\infty} xf(x)\mathrm{d}x \\ &= \int_{0}^{3} \frac{x}{3}\mathrm{d}x \\ &= \frac{3}{2}\end{aligned}$$

(b) Variance:

Denote μ as $\mathrm{E}[X]$, then we have,

$$\begin{aligned}\mathrm{Var}[X] &= \int_{-\infty}^{\infty} (x-\mu)^2 f(x)\mathrm{d}x \\ &= \int_{-\infty}^{\infty} x^2 f(x)\mathrm{d}x - 2\mu \int_{-\infty}^{\infty} xf(x)\mathrm{d}x + \mu^2 \int_{-\infty}^{\infty} f(x)\mathrm{d}x \\ &= \int_{-\infty}^{\infty} x^2 f(x)\mathrm{d}x - \mu^2 \\ &= \int_{0}^{3} \frac{x^2}{3}\mathrm{d}x - (\frac{3}{2})^2 \\ &= 3 - \frac{9}{4} \\ &= \frac{3}{4}\end{aligned}$$

4

Consider the following table of data that provides the values of a discrete data vector $\mathbf{x}$ of samples from the random variable X, where each entry in $\mathbf{x}$ is given as x_i.

Table 1. Dataset N=5 observations

n	x_0	x_1	x_2	x_3	x_4
$\mathbf{x}$	2	3	10	-1	-1

What is the (a) mean, (b) variance, and the of the data?

Show all work. Your answer should include the definition of mean, median, and variance in the context of discrete data.

ANSWER

(a) Mean

$$\begin{aligned}\mathrm{E}[X] &= \frac{1}{N}\sum_{0}^{N} x_i \\ &= \frac{1}{5} \times (2 + 3 + 10 + (-1) + (-1)) \\ &= 2.6\end{aligned}$$

(b) Variance

$$\begin{aligned}\mathrm{Var}[X] &= \frac{1}{N}\sum_{0}^{N} (x_i - \mathrm{E}[X])^2 \\ &= \frac{1}{5} \times ((2 - 2.6)^2 + (3 - 2.6)^2 + (10 - 2.6)^2 + (-1 - 2.6)^2 + (-1 - 2.6)^2) \\ &= 16.24\end{aligned}$$

(c) Median

The median is the middle value that separates the higher half from the lower half of a data sample.

Therefore, the median of this data is 2.

5

Review of counting from probability theory.

(a) How many different 7-place license plates are possible if the first 3 places only contain letters and the last 4 only contain numbers?

(b) How many different batting orders are possible for a baseball team with 9 players?

(c) How many batting orders of 5 players are possible for a team with 9 players total?

(d) Let's assume this class has 26 students and we want to form project teams. How many unique teams of 3 are possible?

Hint: For each problem, determine if order matters, and if it should be calculated with or without replacement.

ANSWER

(a)

There are 26 different letters (a-z) and 10 different numbers (0-9) in all. Also, order matters for this problem, and it should be calculated with replacement because repeated characters are allowed. Thus there are $26 \times 26 \times 26 \times 10 \times 10 \times 10 \times 10 = 175760000$ different 2-place license plates.

(b)

Order matters for this problem, and it should be calculated without replacement because one player could only take one position. This is a permutation of 9 players. Thus there are $9! = 362880$ different possible batting orders for these 9 players.

(c)

Order matters for this problem, and it should be calculated without replacement because one player could only take one position. This is the permutation of 5 players from 9 players. Thus there are $9 \times 8 \times 7 \times 6 \times 5 = 15120$ different possible batting orders for choosing 5 players from 9 players.

(d)

Order does not matter for this problem, and it should be calculated without replacement. This is a combination problem, so there are $\binom{26}{3} = 2600$ possible unique teams of 3.

Linear Algebra

6

Matrix manipulations and multiplication. Machine learning involves working with many matrices, so this exercise will provide you with the opportunity to practice those skills.

Let $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} -1 \\ 3 \\ 8 \end{bmatrix}$, $\mathbf{c} = \begin{bmatrix} 4 \\ -3 \\ 6 \end{bmatrix}$, and $\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Compute the following or indicate that it cannot be computed:

1. $\mathbf{AA}$
2. $\mathbf{AA}^T$
3. $\mathbf{Ab}$
4. $\mathbf{Ab}^T$
5. $\mathbf{bA}$
6. $\mathbf{b}^T\mathbf{A}$
7. $\mathbf{bb}$
8. $\mathbf{b}^T\mathbf{b}$
9. $\mathbf{bb}^T$
10. $\mathbf{b} + \mathbf{c}^T$
11. $\mathbf{b}^T\mathbf{b}^T$
12. $\mathbf{A}^{-1}\mathbf{b}$
13. $\mathbf{A} \circ \mathbf{A}$
14. $\mathbf{b} \circ \mathbf{c}$

Note: The element-wise (or Hadamard) product is the product of each element in one matrix with the corresponding element in another matrix, and is represented by the symbol "$\circ$".

ANSWER

1.

$$\mathbf{AA} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 1\times1+2\times2+3\times3 & 1\times2+2\times4+3\times5 & 1\times3+2\times5+ \\ 2\times1+4\times2+5\times3 & 2\times2+4\times4+5\times5 & 2\times3+4\times5+ \\ 3\times1+5\times2+6\times3 & 3\times2+5\times4+6\times5 & 3\times3+5\times5+ \end{bmatrix}$$

2.

$$\mathbf{AA}^T = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 1\times1+2\times2+3\times3 & 1\times2+2\times4+3\times5 & 1\times3+2\times5+ \\ 2\times1+4\times2+5\times3 & 2\times2+4\times4+5\times5 & 2\times3+4\times5+ \\ 3\times1+5\times2+6\times3 & 3\times2+5\times4+6\times5 & 3\times3+5\times5+ \end{bmatrix}$$

3. $\mathbf{Ab} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} -1 \\ 3 \\ 8 \end{bmatrix} = \begin{bmatrix} 1 \times (-1) + 2 \times 3 + 3 \times 8 \\ 2 \times (-1) + 4 \times 3 + 5 \times 8 \\ 3 \times (-1) + 5 \times 3 + 6 \times 8 \end{bmatrix} = \begin{bmatrix} 29 \\ 50 \\ 60 \end{bmatrix}$

4. $\mathbf{Ab}^T$ is $(3 \times 3) \times (1 \times 3)$ matrix, which cannot to be calculated.

5. $\mathbf{bA}$ is $(3 \times 1) \times (3 \times 3)$ matrix, which cannot to be calculated.

6.

$\mathbf{b}^T\mathbf{A} = \begin{bmatrix} -1 & 3 & 8 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} = \begin{bmatrix} -1 \times 1 + 3 \times 2 + 8 \times 3 & -1 \times 2 + 3 \times 4 + 8 \times 5 & -1 \times 3 + 3 \end{bmatrix}$

7. $\mathbf{bb}$ is $(3 \times 1) \times (3 \times 1)$ matrix, which cannot to be calculated.

8. $\mathbf{b}^T\mathbf{b} = \begin{bmatrix} -1 & 3 & 8 \end{bmatrix} \times \begin{bmatrix} -1 \\ 3 \\ 8 \end{bmatrix} = \begin{bmatrix} -1 \times (-1) + 3 \times 3 + 8 \times 8 \end{bmatrix} = 74$

9. $\mathbf{bb}^T = \begin{bmatrix} -1 \\ 3 \\ 8 \end{bmatrix} \times \begin{bmatrix} -1 & 3 & 8 \end{bmatrix} = \begin{bmatrix} -1 \times (-1) & -1 \times 3 & -1 \times 8 \\ 3 \times (-1) & 3 \times 3 & 3 \times 8 \\ 8 \times (-1) & 8 \times 3 & 8 \times 8 \end{bmatrix} = \begin{bmatrix} 1 & -3 & -8 \\ -3 & 9 & 24 \\ -8 & 24 & 64 \end{bmatrix}$

10. $\mathbf{b} + \mathbf{c}^T$ cannot to be calculated, because $\mathbf{b}$ is 3×1 and $\mathbf{c}^T$ is 1×3.

11. $\mathbf{b}^T\mathbf{b}^T$ cannot to be calculated, because it's $(1 \times 3) \times (1 \times 3)$.

12. First we try to find $\mathbf{A}^{-1}$.

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 2 & 4 & 5 & 0 & 1 & 0 \\ 3 & 5 & 6 & 0 & 0 & 1 \end{array}\right] \xrightarrow[R_3 \leftarrow R_3+(-3)\times R_1]{R_2 \leftarrow R_2+(-2)\times R_1} \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 0 & -1 & -2 & 1 & 0 \\ 0 & -1 & -3 & -3 & 0 & 1 \end{array}\right] \xrightarrow[R_2 \leftarrow (-R_2),\, R_3 \leftarrow (-R_3)]{R_2 \rightleftharpoons R_3} \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & 3 & 3 & 0 & -1 \\ 0 & 0 & 1 & 2 & -1 & 0 \end{array}\right]$$

$$\xrightarrow[R_2 \leftarrow R_2+(-3)\times R_3]{R_1 \leftarrow R_1+(-3)\times R_3} \left[\begin{array}{ccc|ccc} 1 & 2 & 0 & -5 & -3 & 0 \\ 0 & 1 & 0 & -3 & 3 & -1 \\ 0 & 0 & 1 & 2 & -1 & 0 \end{array}\right] \xrightarrow{R_1 \leftarrow R_1+(-2)\times R_2} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & -3 & 2 \\ 0 & 1 & 0 & -3 & 3 & -1 \\ 0 & 0 & 1 & 2 & -1 & 0 \end{array}\right]$$

Thus we have $\mathbf{A}^{-1} = \begin{bmatrix} 1 & -3 & 2 \\ -3 & 3 & -1 \\ 2 & -1 & 0 \end{bmatrix}$. Therefore,

$$\mathbf{A}^{-1}\mathbf{b} = \begin{bmatrix} 1 & -3 & 2 \\ -3 & 3 & -1 \\ 2 & -1 & 0 \end{bmatrix} \times \begin{bmatrix} -1 \\ 3 \\ 8 \end{bmatrix} = \begin{bmatrix} 1\times(-1)+(-3)\times 3+2\times 8 \\ -3\times(-1)+3\times 3+(-1)\times 8 \\ 2\times(-1)+(-1)\times 3+0\times 8 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \\ -5 \end{bmatrix}$$

13. $\mathbf{A} \circ \mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} \circ \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 1\times 1 & 2\times 2 & 3\times 3 \\ 2\times 2 & 4\times 4 & 5\times 5 \\ 3\times 3 & 5\times 5 & 6\times 6 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 9 \\ 4 & 16 & 25 \\ 9 & 25 & 36 \end{bmatrix}$

14. $\mathbf{b} \circ \mathbf{c} = \begin{bmatrix} -1 \\ 3 \\ 8 \end{bmatrix} \circ \begin{bmatrix} 4 \\ -3 \\ 6 \end{bmatrix} = \begin{bmatrix} -1\times 4 \\ 3\times(-3) \\ 8\times 6 \end{bmatrix} = \begin{bmatrix} -4 \\ -9 \\ 48 \end{bmatrix}$

In [1]:
```
import numpy as np

A = np.array([
    [1,2,3],
    [2,4,5],
    [3,5,6]
])
b = np.array([-1,3,8]).reshape((3,1))
c = np.array([4,-3,6]).reshape((3,1))
I = np.eye(3)

# 1
print("1. A @ A = \n",np.matmul(A, A), end = '\n\n')

# 2
print("2. A @ A.T = \n",np.matmul(A, A.T), end = '\n\n')

# 3
print("3. A @ b = \n",np.matmul(A, b), end = '\n\n')

# 4
try:
    print("4. A @ b.T = \n",np.matmul(A, b.T), end = '\n\n')
except ValueError as VE:
    print("4.\n", VE, end = '\n\n')

# 5
try:
    print("5. b @ A = \n",np.matmul(b, A), end = '\n\n')
except ValueError as VE:
    print("5.\n", VE, end = '\n\n')

# 6
print("6. b.T @ A = \n",np.matmul(b.T, A), end = '\n\n')

# 7
try:
    print("7. b @ b = \n",np.matmul(b, b), end = '\n\n')
except ValueError as VE:
    print("7.\n", VE, end = '\n\n')

# 8
print("8. b.T @ b = \n",np.matmul(b.T, b), end = '\n\n')

# 9
print("9. b @ b.T = \n",np.matmul(b, b.T), end = '\n\n')

# 10
print("10.\n shapes (3,1) and (1,3) not aligned \n")

# 11
try:
    print("11. b.T @ b.T = \n",np.matmul(b.T, b.T), end = '\n\n')
except ValueError as VE:
    print("11.\n", VE, end = '\n\n')
```

```
# 12
print("12. A^(-1) @ b = \n", np.matmul(np.linalg.inv(A), b), end = '\n\n
')

# 13
print("13. A * A = \n", A * A, end = '\n\n')

# 14
print("14. b * c = \n", b * c, end = '\n\n')
```

```
1. A @ A =
 [[14 25 31]
 [25 45 56]
 [31 56 70]]

2. A @ A.T =
 [[14 25 31]
 [25 45 56]
 [31 56 70]]

3. A @ b =
 [[29]
 [50]
 [60]]

4.
 shapes (3,3) and (1,3) not aligned: 3 (dim 1) != 1 (dim 0)

5.
 shapes (3,1) and (3,3) not aligned: 1 (dim 1) != 3 (dim 0)

6. b.T @ A =
 [[29 50 60]]

7.
 shapes (3,1) and (3,1) not aligned: 1 (dim 1) != 3 (dim 0)

8. b.T @ b =
 [[74]]

9. b @ b.T =
 [[ 1 -3 -8]
 [-3  9 24]
 [-8 24 64]]

10.
 shapes (3,1) and (1,3) not aligned

11.
 shapes (1,3) and (1,3) not aligned: 3 (dim 1) != 1 (dim 0)

12. A^(-1) @ b =
 [[ 6.]
 [ 4.]
 [-5.]]

13. A * A =
 [[ 1  4  9]
 [ 4 16 25]
 [ 9 25 36]]

14. b * c =
 [[-4]
 [-9]
 [48]]
```

6

Eigenvectors and eigenvalues. Eigenvectors and eigenvalues are useful for some machine learning algorithms, but the concepts take time to solidly grasp. For an intuitive review of these concepts, explore this interactive website at Setosa.io (http://setosa.io/ev/eigenvectors-and-eigenvalues/). Also, the series of linear algebra videos by Grant Sanderson of 3Brown1Blue are excellent and can be viewed on youtube here (https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab).

1. Calculate the eigenvalues and corresponding eigenvectors of matrix $\mathbf{A}$ above, from the last question.
2. Choose one of the eigenvector/eigenvalue pairs, $\mathbf{v}$ and λ, and show that $\mathbf{Av} = \lambda\mathbf{v}$. Also show that this relationship extends to higher orders: $\mathbf{AAv} = \lambda^2\mathbf{v}$
3. Show that the eigenvectors are orthogonal to one another (e.g. their inner product is zero). This is true for real, symmetric matrices.

ANSWER

1. Calculate the eigenvalues and eigenvectors using numpy.

In [2]:
```
import numpy as np

A = np.array(np.mat('1 2 3;2 4 5;3 5 6'))
w, v = np.linalg.eig(A)

for i in range(3):
    print("eigenvalue%a: %a\neigenvector%a:\n%a\n" % (i, w[i], i, v[:,i]
.reshape((3,1))))
```

```
eigenvalue0: 11.344814282762082
eigenvector0:
array([[-0.32798528],
       [-0.59100905],
       [-0.73697623]])

eigenvalue1: -0.5157294715892574
eigenvector1:
array([[-0.73697623],
       [-0.32798528],
       [ 0.59100905]])

eigenvalue2: 0.1709151888271788
eigenvector2:
array([[ 0.59100905],
       [-0.73697623],
       [ 0.32798528]])
```

2. Show that $\mathbf{Av} = \lambda\mathbf{v}$ and $\mathbf{AAv} = \lambda^2\mathbf{v}$

For $\mathbf{Av} = \lambda\mathbf{v}$

In [3]:
```
for i in range(3):
    e_w = w[i]; e_v = v[:, i]
    print(np.dot(A, e_v))
    print(np.dot(e_w, e_v), end = '\n\n')
```

```
[-3.72093206 -6.70488789 -8.36085845]
[-3.72093206 -6.70488789 -8.36085845]

[ 0.38008036  0.16915167 -0.30480078]
[ 0.38008036  0.16915167 -0.30480078]

[ 0.10101242 -0.12596043  0.05605767]
[ 0.10101242 -0.12596043  0.05605767]
```

For $\mathbf{AAv} = \lambda^2\mathbf{v}$

In [4]:
```
for i in range(3):
    e_w = w[0]; e_v = v[:, 0]
    print(np.dot(np.dot(A, A), e_v))
    print(np.dot(np.dot(e_w, e_w), e_v), end = '\n\n')
```

```
[-42.2132832  -76.06570795 -94.85238636]
[-42.2132832  -76.06570795 -94.85238636]

[-42.2132832  -76.06570795 -94.85238636]
[-42.2132832  -76.06570795 -94.85238636]

[-42.2132832  -76.06570795 -94.85238636]
[-42.2132832  -76.06570795 -94.85238636]
```

3. Show that all the eigenvectors are orthogonal to each other.

In [5]:
```
for i in range(3):
    for j in range(i+1, 3):
        print(np.dot(v[:, i], v[:, j]))
```

```
-2.220446049250313e-16
-4.440892098500626e-16
-1.0547118733938987e-15
```

All the dot products between each pairs of them are zeros, so all the eigenvectors are orthogonal to each other.

Numerical Programming

7

Speed comparison between vectorized and non-vectorized code. Begin by creating an array of 10 million random numbers using the numpy random.randn module. Compute the sum of the squares first in a for loop, then using Numpy's `dot` module. Time how long it takes to compute each and report the results and report the output. How many times faster is the vectorized code than the for loop approach?

*Note: all code should be well commented, properly formatted, and your answers should be output using the `print()` function as follows (where the # represents your answers, to a reasonable precision):

```
Time [sec] (non-vectorized): ######

Time [sec] (vectorized):     ######

The vectorized code is ##### times faster than the vectorized code
```

ANSWER

In [6]:
```
import numpy as np
import time

# Generate the random samples
sam = np.random.randn(10000000)

# Compute the sum of squares the non-vectorized way (using a for loop)
start_1 = time.time()
cum_sum = 0
for i in sam:
    cum_sum += i**2
end_1 = time.time()

# Compute the sum of squares the vectorized way (using numpy)
start_2 = time.time()
cum_sum2 = np.dot(sam, sam)
end_2 = time.time()

# Print the results
print("Time [sec] (non-vectorized): %.4f" % (end_1 - start_1))
print("Time [sec] (vectorized):     %.4f" % (end_2 - start_2))
print("The vectorized code is %3.1f times faster than the vectorized cod
e" % ((end_1 - start_1)/(end_2 - start_2)))
```

```
Time [sec] (non-vectorized): 3.5946
Time [sec] (vectorized):     0.0043
The vectorized code is 844.8 times faster than the vectorized code
```

8

One popular Agile development framework is Scrum (a paradigm recommended for data science projects). It emphasizes the continual evolution of code for projects, becoming progressively better, but starting with a quickly developed minimum viable product. This often means that code written early on is not optimized, and that's a good thing - it's best to get it to work first before optimizing. Imagine that you wrote the following code during a sprint towards getting an end-to-end system working. Vectorize the following code and show the difference in speed between the current implementation and a vectorized version.

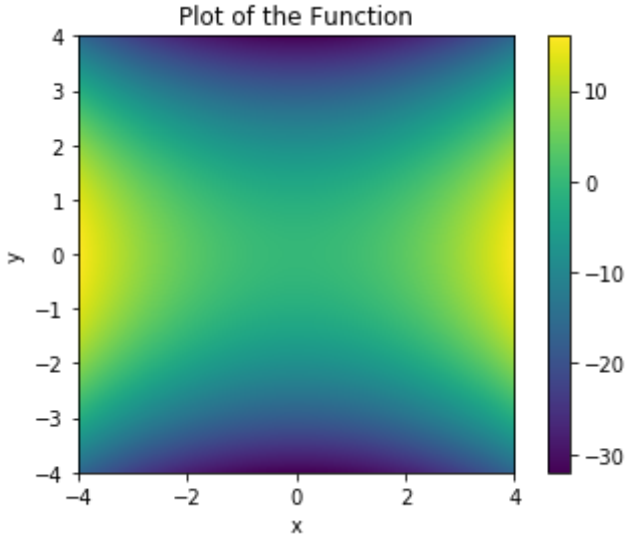
The function below computes the function $f(x, y) = x^2 - 2y^2$ and determines whether this quantity is above or below a given threshold, `thresh=0`. This is done for $x, y \in \{-4, 4\}$, over a 2,000-by-2,000 grid covering that domain.

(a) Vectorize this code and demonstrate (as in the last exercise) the speed increase through vectorization and (b) plot the resulting data - both the function $f(x, y)$ and the thresholded output - using `imshow` (https://matplotlib.org/api/_as_gen/matplotlib.pyplot.imshow.html?highlight=matplotlib%20pyplot%20imshow#matplotlib.pyplot.imshow) from `matplotlib`.

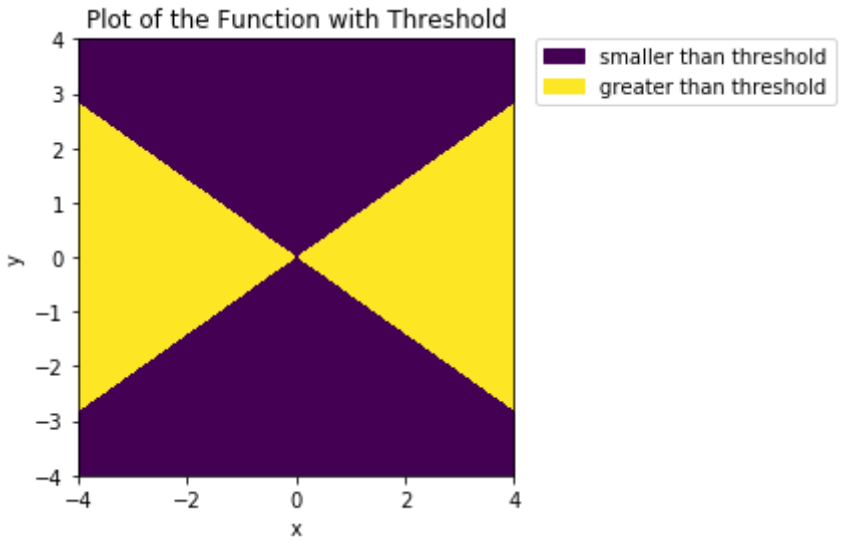
Hint: look at the `numpy` `meshgrid` (https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.meshgrid.html) documentation

In [7]:
```
import numpy as np
import time
import matplotlib.pyplot as plt
%matplotlib inline

# Initialize variables for this exerise
thresh = 0
n = 2000
x = np.linspace(-4, 4, 2000)
y = np.linspace(-4, 4, 2000)
X, Y = np.meshgrid(x, y)
Z = np.zeros_like(X)

# Nonvectorized implementation
start1 = time.time()
for i, xi in enumerate(x):
    for j, yj in enumerate(y):
        Z[j, i] += xi**2 - 2*yj**2
end1 = time.time()

# Vectorized implementation
start2 = time.time()
Z2 = X**2 - 2*Y**2
end2 = time.time()

# Print the time for each and the speed increase
print("Time [sec] (non-vectorized): %.4f" % (end1 - start1))
print("Time [sec] (vectorized):     %.4f" % (end2 - start2))
print("The vectorized code is %3.1f times faster than the vectorized cod
e" % ((end1 - start1)/(end2 - start2)))

# Plot the result
plt.imshow(Z, extent = [-4, 4, -4, 4])
plt.xlabel("x")
plt.ylabel("y")
plt.title("Plot of the Function")
plt.colorbar()
plt.show()
```

```
Time [sec] (non-vectorized): 4.0216
Time [sec] (vectorized):     0.0321
The vectorized code is 125.1 times faster than the vectorized code
```



In [8]:
```
# plot the thresholded output
'''Z3 = np.zeros_like(Z)
Z3[Z > thresh] = 1
plt.imshow(Z3, extent = [-4, 4, -4, 4])
plt.xlabel("x")
plt.ylabel("y")
plt.title("Plot of the Function with Threshold")
plt.colorbar()
plt.show()'''

Z3 = np.zeros_like(Z)
Z3[Z > thresh] = 1

import matplotlib.patches as mp
im = plt.imshow(Z3, extent = [-4, 4, -4, 4])
values = np.unique(Z3.ravel())
colors = [im.cmap(im.norm(value)) for value in values]
patches = [mp.Patch(color=colors[0], label="smaller than threshold"),
          mp.Patch(color=colors[1], label="greater than threshold")]

plt.legend(handles = patches, bbox_to_anchor = (1.05, 1), loc = 2, borde
raxespad = 0. )
plt.xlabel("x")
plt.ylabel("y")
plt.title("Plot of the Function with Threshold")
plt.show()
```



9

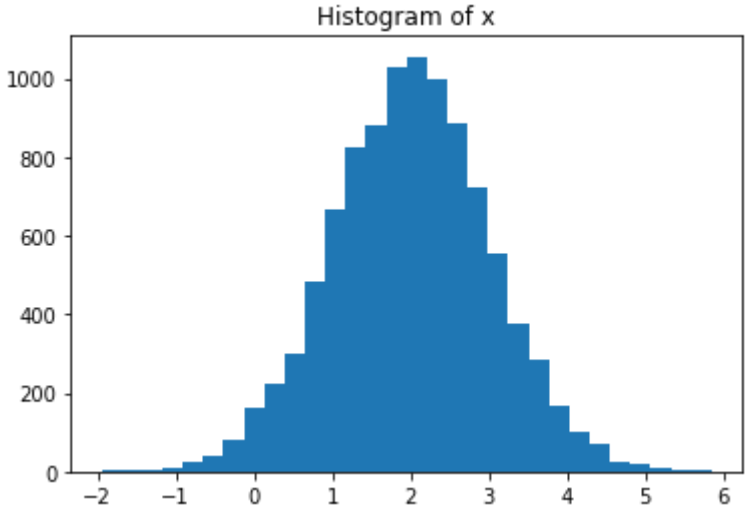
This exercise will walk through some basic numerical programming exercises.

1. Synthesize $n = 10^4$ normally distributed data points with mean $\mu = 2$ and a standard deviation of $\sigma = 1$. Call these observations from a random variable X, and call the vector of observations that you generate, $\mathbf{x}$.
2. Calculate the mean and standard deviation of $\mathbf{x}$ to validate (1) and provide the result to a precision of four significant figures.
3. Plot a histogram of the data in $\mathbf{x}$ with 30 bins
4. What is the 90th percentile of $\mathbf{x}$? The 90th percentile is the value below which 90% of observations can be found.
5. What is the 99th percentile of $\mathbf{x}$?
6. Now synthesize $n = 10^4$ normally distributed data points with mean $\mu = 0$ and a standard deviation of $\sigma = 3$. Call these observations from a random variable Y, and call the vector of observations that you generate, $\mathbf{y}$.
7. Plot the histogram of the data in $\mathbf{y}$ on a (new) plot with the histogram of $\mathbf{x}$, so that both histograms can be seen and compared.
8. Using the observations from $\mathbf{x}$ and $\mathbf{y}$, estimate $E[XY]$

ANSWER

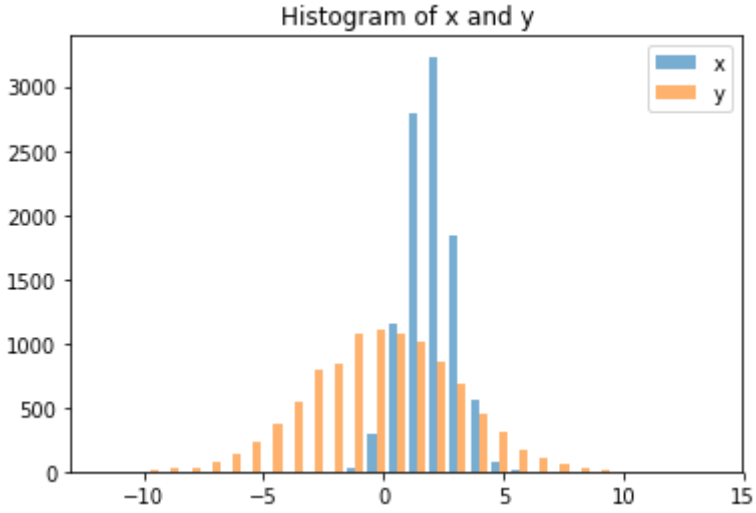
In [9]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
np.random.seed(234)

# 1
x = np.random.normal(2, 1, 10**4)

# 2
print("mean of x:\t %1.3f" % np.mean(x))
print("sd of x:\t %1.3f" % np.std(x))

# 3
plt.figure()
plt.title("Histogram of x")
plt.hist(x, bins = 30)
plt.show()
```

```
mean of x:	 2.008
sd of x:	 1.004
```



In [10]:
```
# 4
a_90 = np.percentile(x, 90)
print('90th percentile of x:\t', a_90)

# 5
a_99 = np.percentile(x, 99)
print('99th percentile of x:\t', a_99)

# 6
y = np.random.normal(0, 3, 10**4)

# 7
plt.figure()
plt.title("Histogram of x and y")
#plt.hist(y, bins = 30, alpha = 0.6, color = 'orange', label = 'y')
#plt.hist(x, bins = 30, alpha = 0.6, label = 'x')
plt.hist([x, y], bins = 30, alpha = 0.6, label = ['x', 'y'])
plt.legend()
plt.show()
```

```
90th percentile of x:    3.2803181105195205
99th percentile of x:    4.367114368317831
```



1. We know that

$$\mathrm{E}[XY] = \int xyP(x,y)\mathrm{d}(xy)$$

In [11]:
```
# 8
print("Estimate of the estimated value of XY is", np.mean(x*y))
```

```
Estimate of the estimated value of XY is -0.08534242758828127
```

10

Estimate the integral of the function $f(x)$ on the interval $0 \leq x < 2.5$ assuming we only know the following points from f:

Table 1. Dataset containing n=5 observations

x_i	**0.0**	**0.5**	**1.0**	**1.5**	**2.0**
y_i	6	7	8	4	1

ANSWER

Using rectangle rule to solve this proble.

In [12]:
```
xi = np.array([0.5]*5)
yi = np.array([6, 7, 8, 4, 1])

print("Estimate Integral: ", sum(xi*yi))
```

```
Estimate Integral:  13.0
```

Version Control via Git

11

Complete the Atlassian Git tutorial (https://www.atlassian.com/git/tutorials/what-is-version-control), specifically the following sections. Try each concept that's presented. For this tutorial, instead of using BitBucket, use Github. Create a github account here if you don't already have one: https://github.com/ (https://github.com/)

1. What is version control (https://www.atlassian.com/git/tutorials/what-is-version-control)
2. What is Git (https://www.atlassian.com/git/tutorials/what-is-git)
3. Install Git (https://www.atlassian.com/git/tutorials/install-git)
4. Setting up a repository (https://www.atlassian.com/git/tutorials/install-git)
5. Saving changes (https://www.atlassian.com/git/tutorials/saving-changes)
6. Inspecting a repository (https://www.atlassian.com/git/tutorials/inspecting-a-repository)
7. Undoing changes (https://www.atlassian.com/git/tutorials/undoing-changes)
8. Rewriting history (https://www.atlassian.com/git/tutorials/rewriting-history)
9. Syncing (https://www.atlassian.com/git/tutorials/syncing)
10. Making a pull request (https://www.atlassian.com/git/tutorials/making-a-pull-request)
11. Using branches (https://www.atlassian.com/git/tutorials/using-branches)
12. Comparing workflows (https://www.atlassian.com/git/tutorials/comparing-workflows)

For your answer, affirm that you either completed the tutorial or have previous experience with all of the concepts above. Do this by typing your name below and selecting the situation that applies from the two options in brackets.

ANSWER

*I, [**Yifei Wang**], affirm that I have [**completed the above tutorial / I have previous experience that covers all the content in this tutorial**]*

12

Using Github to create a static HTML website:

1. Create a branch in your `machine-learning-course` repo called "gh-pages" and checkout that branch (this will provide an example of how to create a simple static website using Github Pages (https://pages.github.com/))
2. Create a file called "index.html" with the contents "Hello World" and add, commit, and push it to that branch.
3. Submit the following: (a) a link to your github repository and (b) a link to your new "Hello World" website. The latter should be at the address https://[USERNAME].github.io/ECE590-assignment0 (https://[USERNAME].github.io/ECE590-assignment0) (where [USERNAME] is your github username).

ANSWER

(a). https://github.com/ywang512/ids705 (https://github.com/ywang512/ids705)

(b). https://ywang512.github.io/ids705/ (https://ywang512.github.io/ids705/)

Exploratory Data Analysis

13

Here you'll bring together some of the individual skills that you demonstrated above and create a Jupyter notebook based blog post on data analysis.

1. Find a dataset that interests you and relates to a question or problem that you find intriguing
2. Using a Jupyter notebook, describe the dataset, the source of the data, and the reason the dataset was of interest.
3. Check the data and see if they need to be cleaned: are there missing values? Are there clearly erroneous values? Do two tables need to be merged together? Clean the data so it can be visualized.
4. Plot the data, demonstrating interesting features that you discover. Are there any relationships between variables that were surprising or patterns that emerged? Please exercise creativity and curiosity in your plots.
5. What insights are you able to take away from exploring the data? Is there a reason why analyzing the dataset you chose is particularly interesting or important? Summarize this as if your target audience was the readership of a major news organization - boil down your findings in a way that is accessible, but still accurate.
6. Create a public repository on your github account titled "machine-learning-course". In it, create a readme file that contains the heading "ECE590: Introductory Machine Learning for Data Science". Add, commit, and push that Jupyter notebook to the master branch. Provide the link to the that post here.

ANSWER

https://github.com/ywang512/ids705 (https://github.com/ywang512/ids705)

Visualization of Suicide Data

Content

This compiled dataset was pulled from Kaggle - Suicide Rates Overview 1985 to 2016 (https://www.kaggle.com/russellyates88/suicide-rates-overview-1985-to-2016), and was built to find signals correlated to increased suicide rates among different cohorts globally, across the socio-economic spectrum.

References

United Nations Development Program. (2018). Human development index (HDI). Retrieved from http://hdr.undp.org/en/indicators/137506 (http://hdr.undp.org/en/indicators/137506)

World Bank. (2018). World development indicators: GDP (current US$) by country:1985 to 2016. Retrieved from http://databank.worldbank.org/data/source/world-development-indicators# (http://databank.worldbank.org/data/source/world-development-indicators#)

[Szamil]. (2017). Suicide in the Twenty-First Century [dataset]. Retrieved from https://www.kaggle.com/szamil/suicide-in-the-twenty-first-century/notebook (https://www.kaggle.com/szamil/suicide-in-the-twenty-first-century/notebook)

World Health Organization. (2018). Suicide prevention. Retrieved from http://www.who.int/mental_health/suicide-prevention/en/ (http://www.who.int/mental_health/suicide-prevention/en/)

Inspiration

Suicide Prevention.

Variables

- **country**: 101 countries in all
- **year**: from 1985 to 2016
- **sex**: male or female
- **age**: grouped age
- **suicides_no**: suicides count
- **population**
- **suicides/100k pop**: suicides count rescale
- **country-year**: country with year combine
- **HDI for year**: Human Development Index, much missing
- **gdp_for_year**: Gross Domestic Product
- **gdp_per_capita**: Gross Domestic Product rescale
- **generation**

Data Preprocessing

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

df_raw = pd.read_csv("../projects/hw1_suicide_rates/suicide_rates_hw1.csv")
print("Raw data has %d rows and %d columns\n" % df_raw.shape)


### Mising values
print("Count of missing values for each variables:")
print(np.sum(df_raw.isna(), axis = 0))
```

```
Raw data has 27820 rows and 12 columns

Count of missing values for each variables:
country                   0
year                      0
sex                       0
age                       0
suicides_no               0
population                0
suicides/100k pop         0
country-year              0
HDI for year          19456
 gdp_for_year ($)         0
gdp_per_capita ($)        0
generation                0
dtype: int64
```

In [14]:
```
suicide_country = df_raw[['country', 'suicides_no', 'population']].group
by('country').agg('sum')
suicide_country = suicide_country.assign(suicide_avg=100000*suicide_coun
try.suicides_no/suicide_country.population)
suicide_country.sort_values(by = ('suicide_avg'), ascending = False).hea
d(10)
```

Out[14]:

	suicides_no	population	suicide_avg
country			
Lithuania	28039	68085210	41.182219
Russian Federation	1209742	3690802620	32.777207
Sri Lanka	55641	182525626	30.483939
Belarus	59892	197372292	30.344685
Hungary	73891	248644256	29.717558
Latvia	12770	44852640	28.471011
Kazakhstan	101546	377513869	26.898614
Slovenia	10615	40268619	26.360477
Estonia	7034	27090810	25.964525
Ukraine	319950	1286469184	24.870398

In [15]:
```
suicide_country.sort_values(by = ('suicide_avg')).head(10)
```

Out[15]:

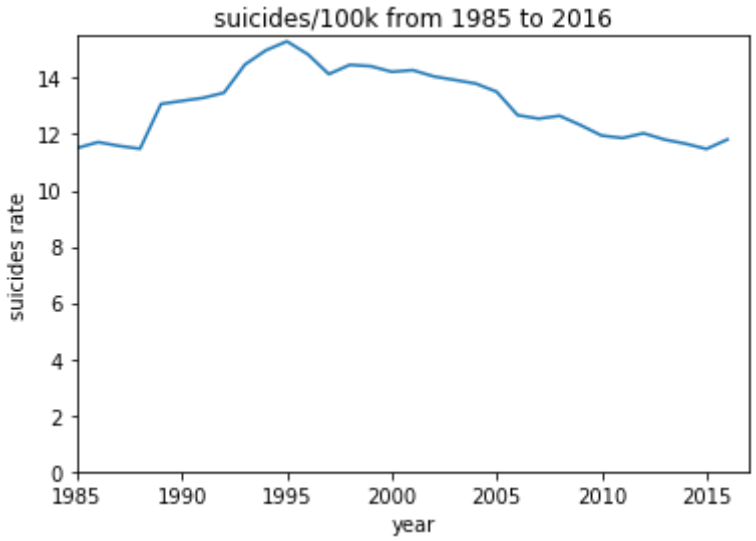
	suicides_no	population	suicide_avg
country			
Dominica	0	66400	0.000000
Saint Kitts and Nevis	0	117300	0.000000
Oman	33	8987087	0.367194
Jamaica	184	39481817	0.466037
Antigua and Barbuda	11	1990228	0.552700
Maldives	20	2900246	0.689597
South Africa	7321	873130762	0.838477
Bahamas	93	6557048	1.418321
Azerbaijan	1656	111790300	1.481345
Grenada	38	2347286	1.618891

In [16]:
```
df_raw.loc[df_raw['country'] == 'Dominica']
```

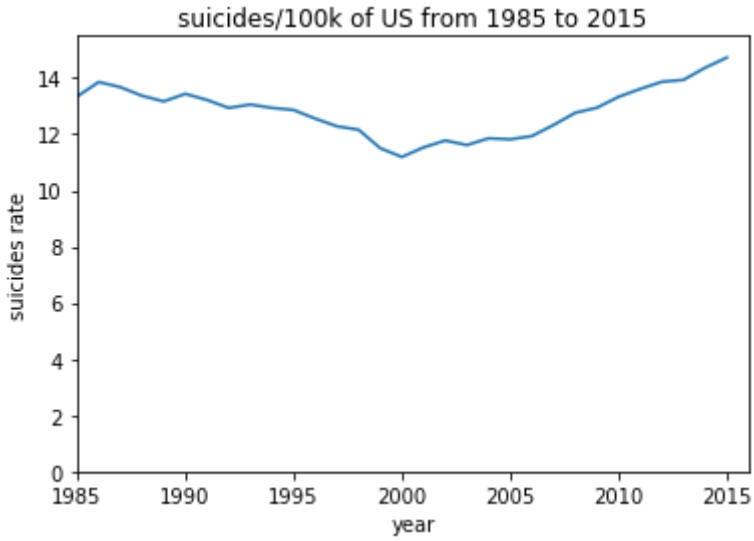
Out[16]:

	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year
7682	Dominica	1985	female	15-24 years	0	8900	0.0	Dominica1985
7683	Dominica	1985	female	25-34 years	0	5100	0.0	Dominica1985
7684	Dominica	1985	female	35-54 years	0	5400	0.0	Dominica1985
7685	Dominica	1985	female	5-14 years	0	8600	0.0	Dominica1985
7686	Dominica	1985	female	55-74 years	0	3800	0.0	Dominica1985
7687	Dominica	1985	female	75+ years	0	1200	0.0	Dominica1985
7688	Dominica	1985	male	15-24 years	0	9500	0.0	Dominica1985
7689	Dominica	1985	male	25-34 years	0	5800	0.0	Dominica1985
7690	Dominica	1985	male	35-54 years	0	4900	0.0	Dominica1985
7691	Dominica	1985	male	5-14 years	0	9300	0.0	Dominica1985
7692	Dominica	1985	male	55-74 years	0	3200	0.0	Dominica1985
7693	Dominica	1985	male	75+ years	0	700	0.0	Dominica1985

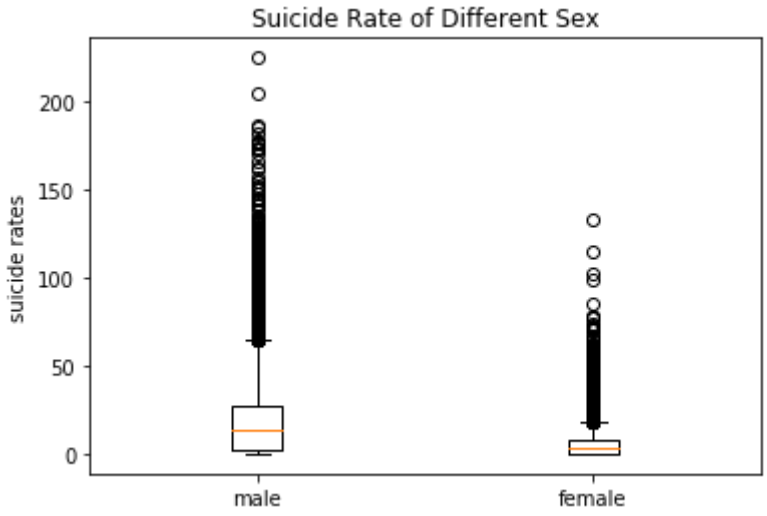
In [17]:
```
## year

suicide_year = df_raw[['year', 'suicides_no', 'population']].groupby('ye
ar').agg('sum')
suicide_year = suicide_year.assign(suicide_avg=100000*suicide_year.suici
des_no/suicide_year.population)

plt.plot(range(1985, 2017), suicide_year['suicide_avg'])
plt.axis([1985, 2017, 0, 15.5])
plt.title("suicides/100k from 1985 to 2016")
plt.ylabel("suicides rate")
plt.xlabel("year")
plt.show()
```



In [18]:
```
df_us = df_raw[df_raw['country'] == 'United States']
df_us_year = df_us[['year', 'suicides_no', 'population']].groupby('year'
).agg('sum')
df_us_year = df_us_year.assign(suicide_avg=100000*df_us_year.suicides_no
/df_us_year.population)

plt.plot(range(1985, 2016), df_us_year['suicide_avg'])
plt.axis([1985, 2016, 0, 15.5])
plt.title("suicides/100k of US from 1985 to 2015")
plt.ylabel("suicides rate")
plt.xlabel("year")
plt.show()
```

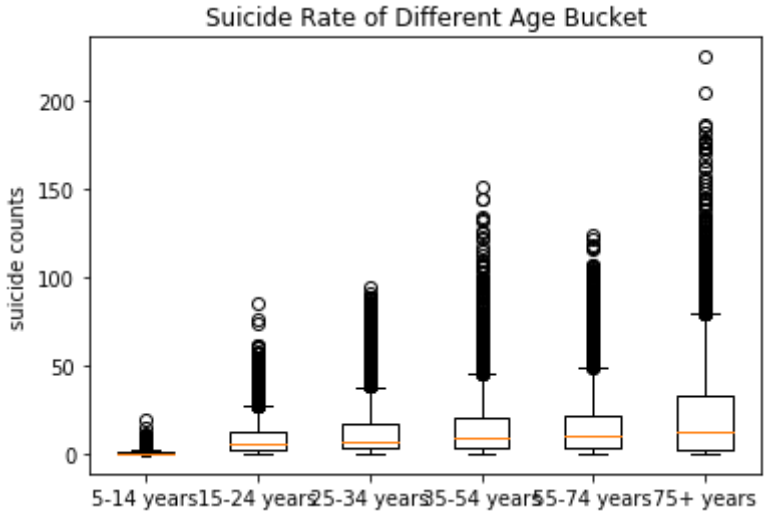


In [19]:
```
### sex

suicide_male = df_raw[df_raw['sex'] == 'male'][['suicides/100k pop']]
suicide_female = df_raw[df_raw['sex'] == 'female'][['suicides/100k pop'
]]

plt.boxplot([suicide_male['suicides/100k pop'], suicide_female['suicide
s/100k pop']], labels = ['male', 'female'])
plt.title("Suicide Rate of Different Sex")
plt.ylabel("suicide rates")
plt.show()
```

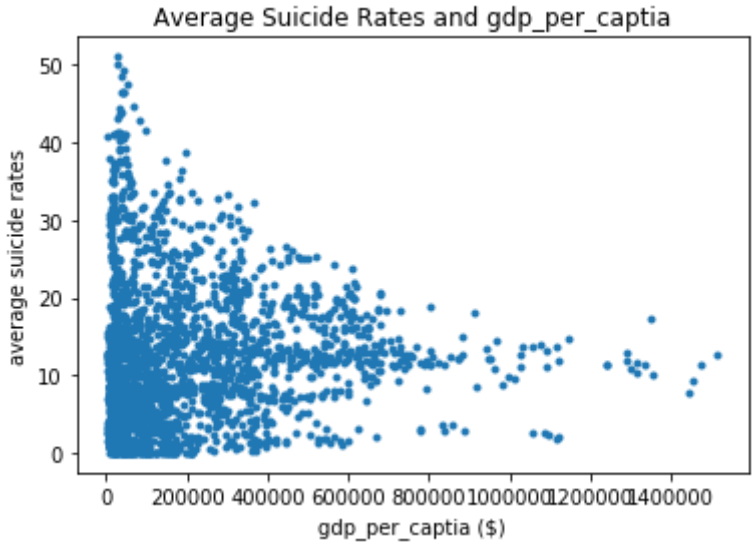


In [20]:
```
### age bucket

print("6 unique age buckets:\n", df_raw['age'].unique(), end = '\n\n')
label_age = ['5-14 years', '15-24 years', '25-34 years', '35-54 years',
'55-74 years', '75+ years']

suicide_age1 = df_raw.loc[df_raw['age'] == label_age[0]][['suicides/100k
 pop']]
suicide_age2 = df_raw.loc[df_raw['age'] == label_age[1]][['suicides/100k
 pop']]
suicide_age3 = df_raw.loc[df_raw['age'] == label_age[2]][['suicides/100k
 pop']]
suicide_age4 = df_raw.loc[df_raw['age'] == label_age[3]][['suicides/100k
 pop']]
suicide_age5 = df_raw.loc[df_raw['age'] == label_age[4]][['suicides/100k
 pop']]
suicide_age6 = df_raw.loc[df_raw['age'] == label_age[5]][['suicides/100k
 pop']]

plt.boxplot([suicide_age1['suicides/100k pop'], suicide_age2['suicides/1
00k pop'],
             suicide_age3['suicides/100k pop'], suicide_age4['suicides/1
00k pop'],
             suicide_age5['suicides/100k pop'], suicide_age6['suicides/1
00k pop']], labels = label_age)
plt.title("Suicide Rate of Different Age Bucket")
plt.ylabel("suicide counts")
plt.show()
```

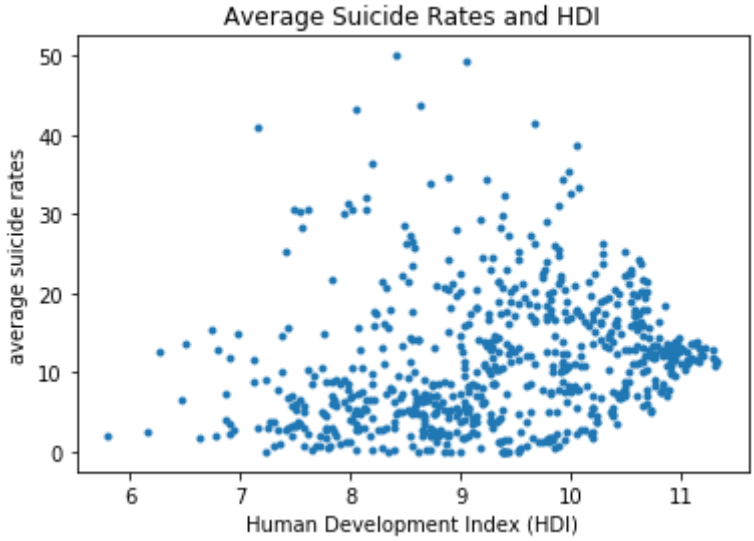
```
6 unique age buckets:
 ['15-24 years' '35-54 years' '75+ years' '25-34 years' '55-74 years'
 '5-14 years']
```



In [21]:
```
### gdp per capita

suicide_gdp = df_raw[['country-year', 'gdp_per_capita ($)',
                      'suicides_no', 'population']].groupby('country-yea
r').agg('sum')
suicide_gdp = suicide_gdp.assign(suicide_avg=100000*suicide_gdp.suicides
_no/suicide_gdp.population)

plt.plot(suicide_gdp['gdp_per_capita ($)'], suicide_gdp['suicide_avg'],
'.')
plt.title("Average Suicide Rates and gdp_per_captia")
plt.xlabel("gdp_per_captia ($)")
plt.ylabel("average suicide rates")
plt.show()
```

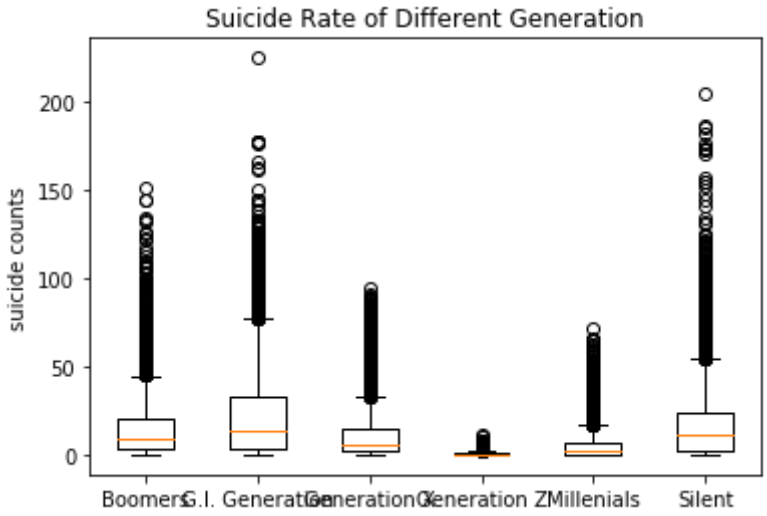


In [22]:
```
### HDI

df_clean = df_raw[df_raw['HDI for year'].notna()]
df_hdi = df_clean[['country-year', 'HDI for year', 'suicides_no', 'popul
ation']].groupby('country-year').agg('sum')
df_hdi = df_hdi.assign(suicide_avg=100000*df_hdi.suicides_no/df_hdi.popu
lation)

plt.plot(df_hdi['HDI for year'], df_hdi['suicide_avg'], '.')
plt.title("Average Suicide Rates and HDI")
plt.xlabel("Human Development Index (HDI)")
plt.ylabel("average suicide rates")
plt.show()
```



In [23]:
```
### generation
label_gen = sorted(df_raw['generation'].unique())
print("6 unique age buckets:\n", label_gen, end = '\n\n')

suicide_gen1 = df_raw.loc[df_raw['generation'] == label_gen[0]][['suicid
es/100k pop']]
suicide_gen2 = df_raw.loc[df_raw['generation'] == label_gen[1]][['suicid
es/100k pop']]
suicide_gen3 = df_raw.loc[df_raw['generation'] == label_gen[2]][['suicid
es/100k pop']]
suicide_gen4 = df_raw.loc[df_raw['generation'] == label_gen[3]][['suicid
es/100k pop']]
suicide_gen5 = df_raw.loc[df_raw['generation'] == label_gen[4]][['suicid
es/100k pop']]
suicide_gen6 = df_raw.loc[df_raw['generation'] == label_gen[5]][['suicid
es/100k pop']]

plt.boxplot([suicide_gen1['suicides/100k pop'], suicide_gen2['suicides/1
00k pop'],
             suicide_gen3['suicides/100k pop'], suicide_gen4['suicides/1
00k pop'],
             suicide_gen5['suicides/100k pop'], suicide_gen6['suicides/1
00k pop']], labels = label_gen)
plt.title("Suicide Rate of Different Generation")
plt.ylabel("suicide counts")
plt.show()
```

```
6 unique age buckets:
 ['Boomers', 'G.I. Generation', 'Generation X', 'Generation Z', 'Millen
ials', 'Silent']
```



Visual Analysis

1. The suicide rate around the world goes up and then down, while the united states goes down and then up.
2. Men has a higher suicide rate than women.
3. Elders are more likely to commit suicide.
4. The higher the GDP per captia, the lower the suicide rates.
5. Need more research to find the relationship between HDI, generation and suicide rates

Future Research

1. These plot here only show the marginal relationship between one independent variable and the dependent variable. This relationship might be biased by other independent variables and we could not see them here. In order to seperate these relationships, we can apply a linear regression model on this data. In this way, we might be able to find more interesting stuffs.
2. Moreover, there might be some interactions between these variables, which also need to be addressed if more researches would be carried out.
3. United States have a different pattern of suicide rates against year when compared to the whole world. Is this only the pattern of United States? Or is this a pattern of developed countries? We need further investigation to this dataset in order to answer this questions.
4. Notice that there exists some points with extrem high suicide rates. We need to pay attention to these points in the future. Also, we could form a hypothesis for the phenomenon and find ways to test it. This might help us better understand the possible causes of high or low suicide rates so that governments could make policies accordingly to decrease the suicide rates.