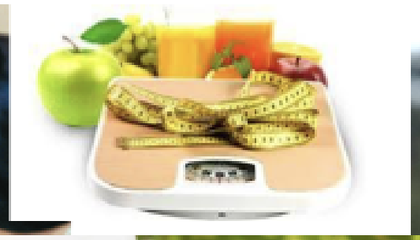
Normal distributions

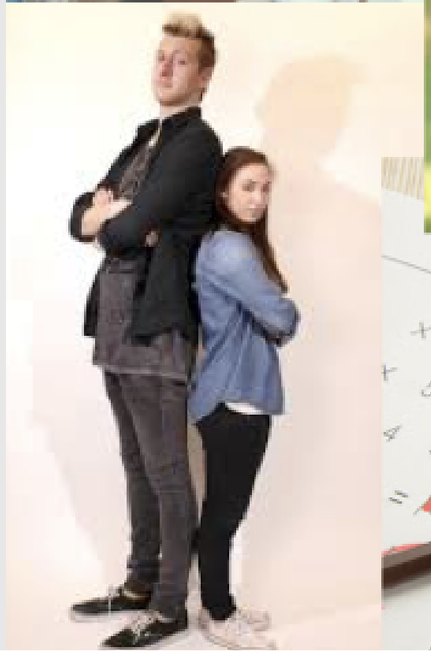
FOUNDATIONS OF PROBABILITY IN PYTHON

Alexander A. Ramírez M.
CEO @ Synergy Vision

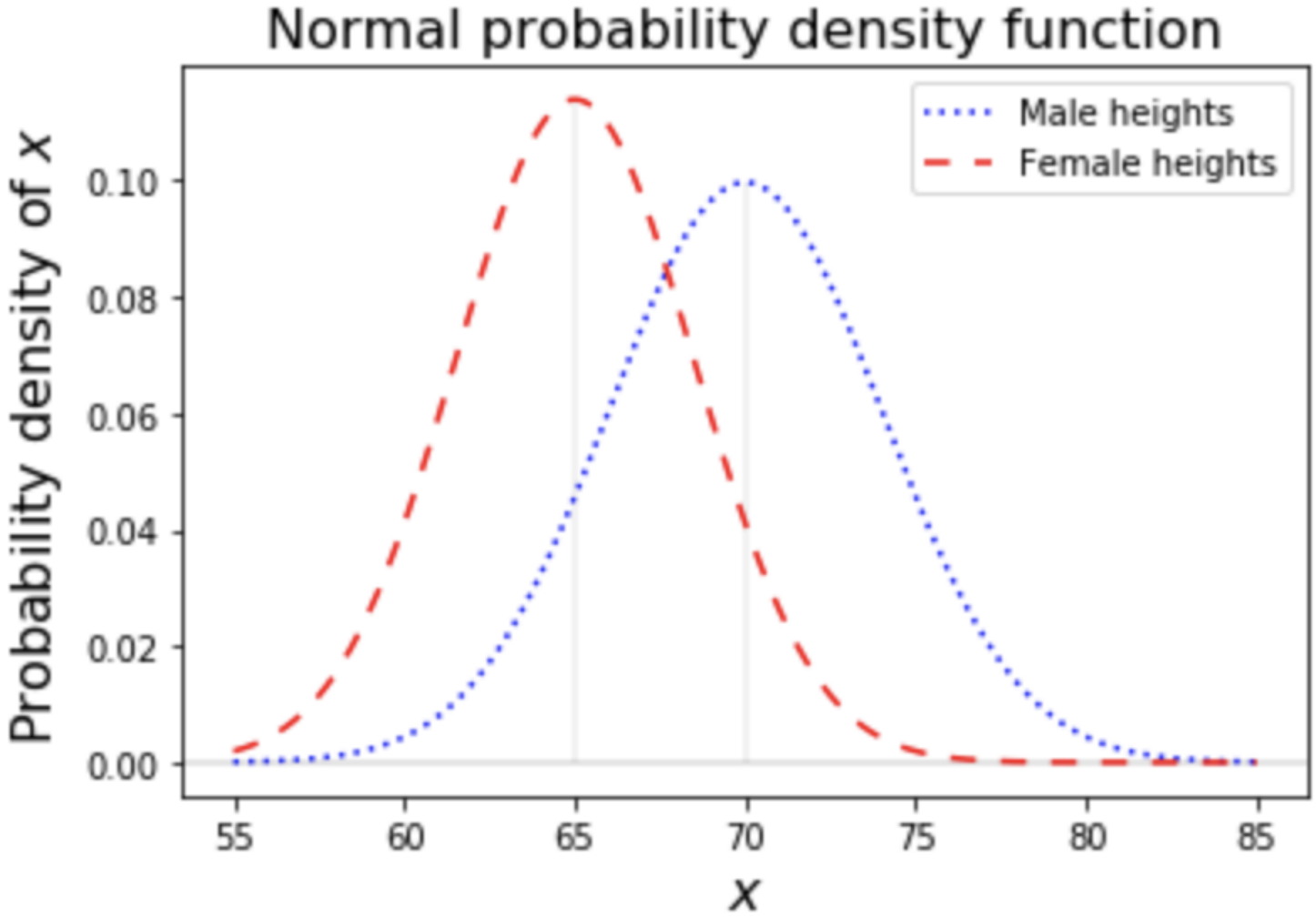
datacamp

Modeling for measures

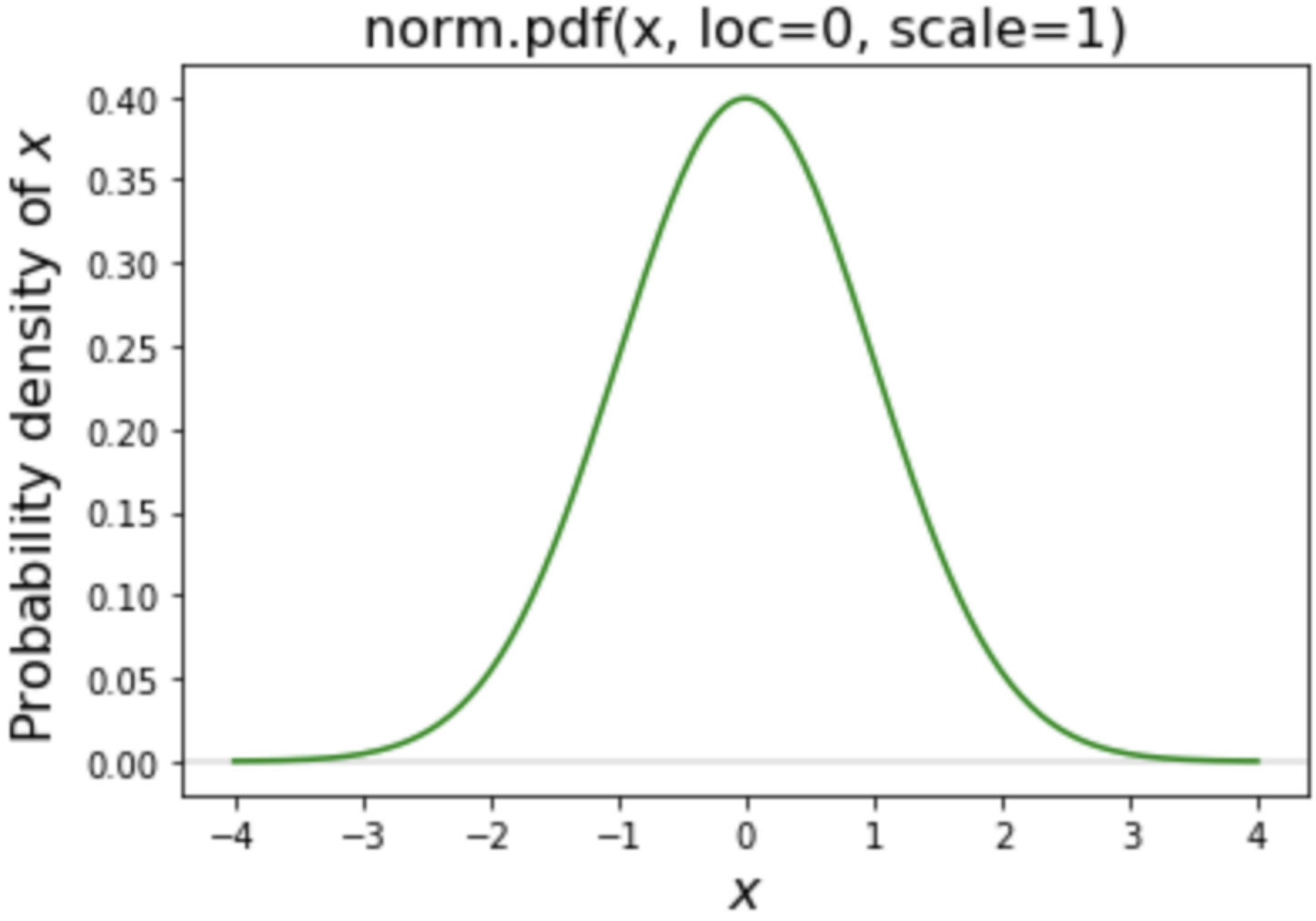




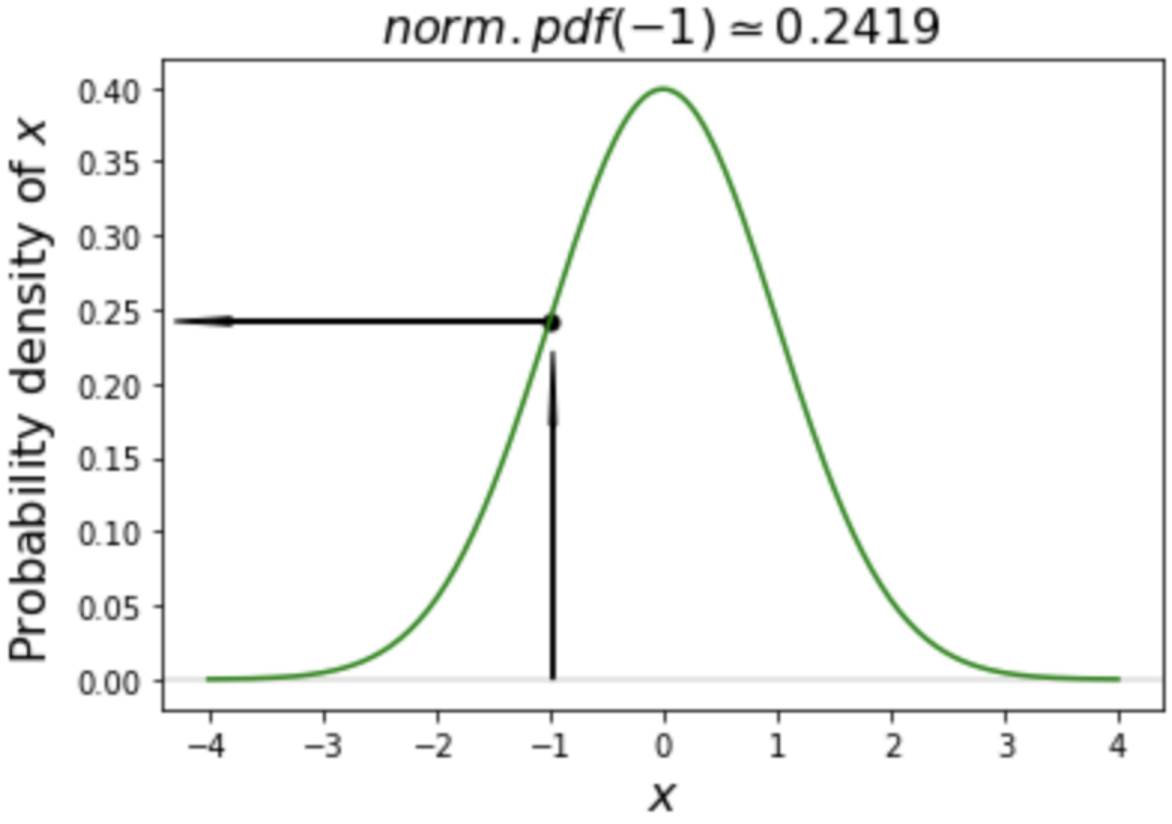
Adults' heights example

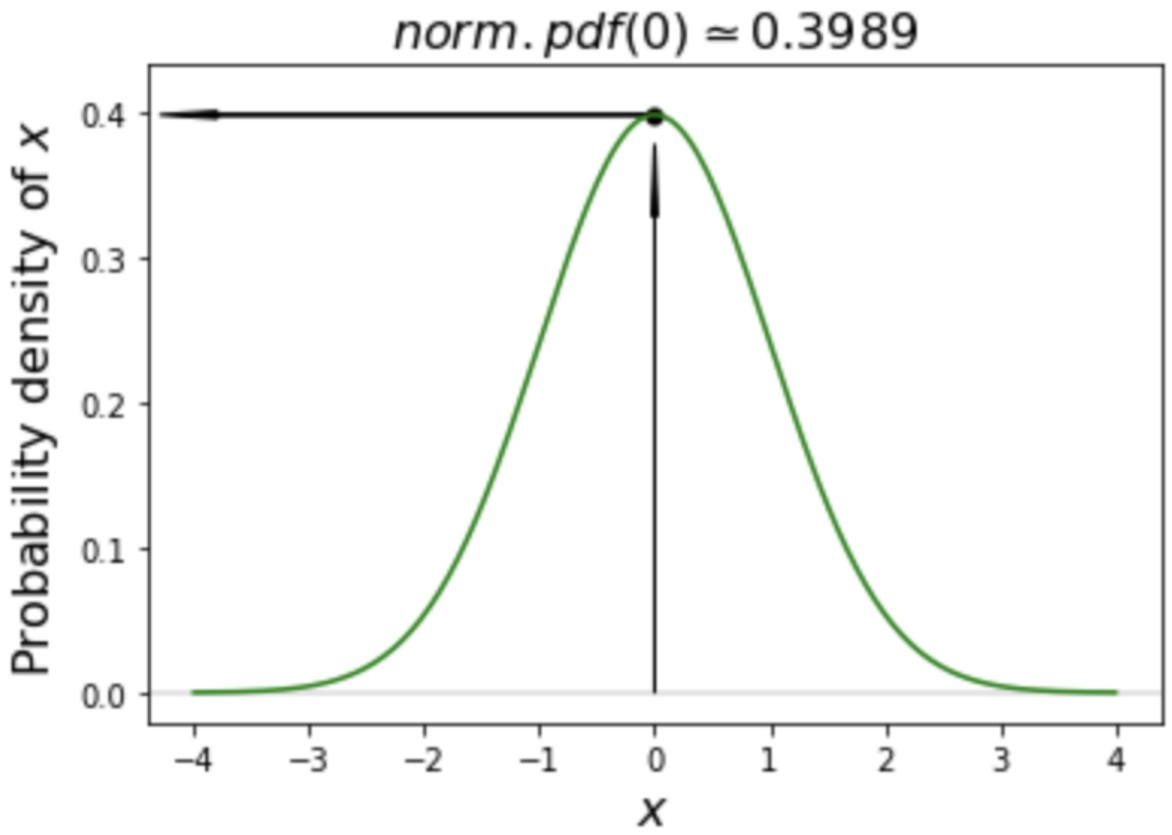


Probability density

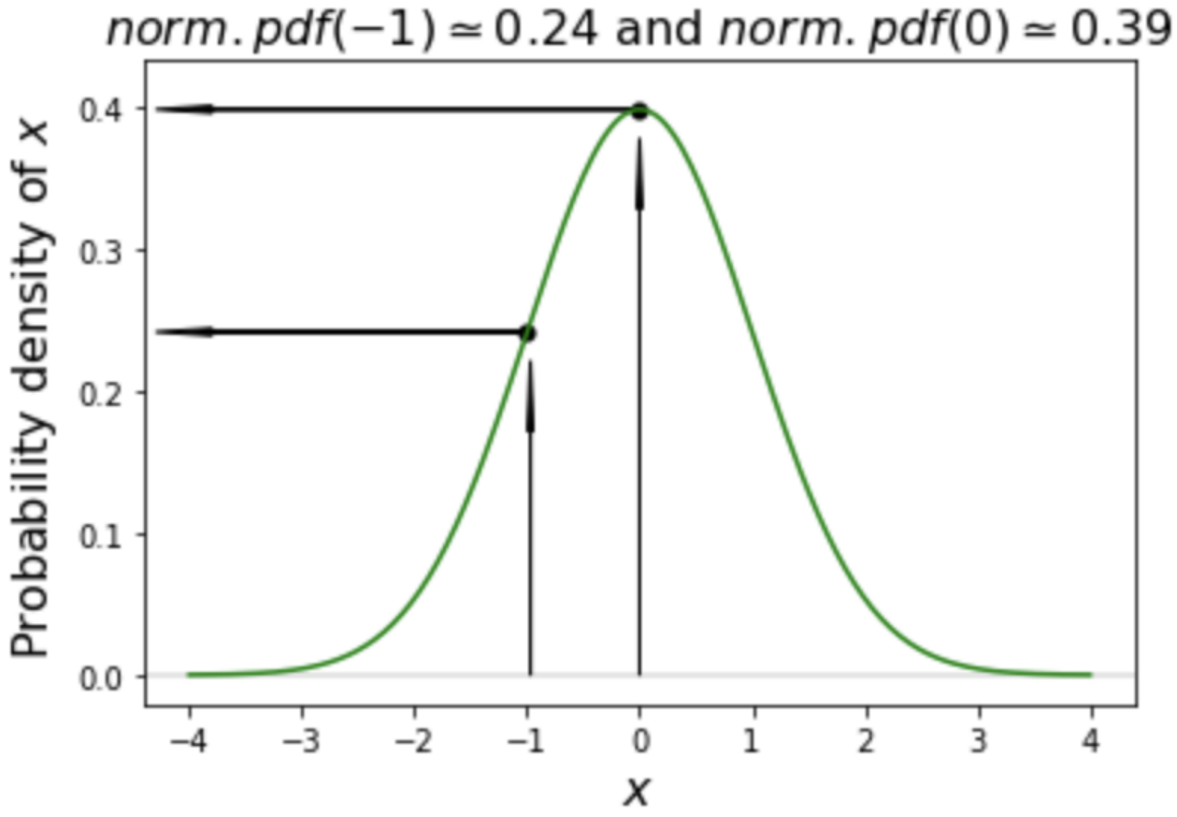


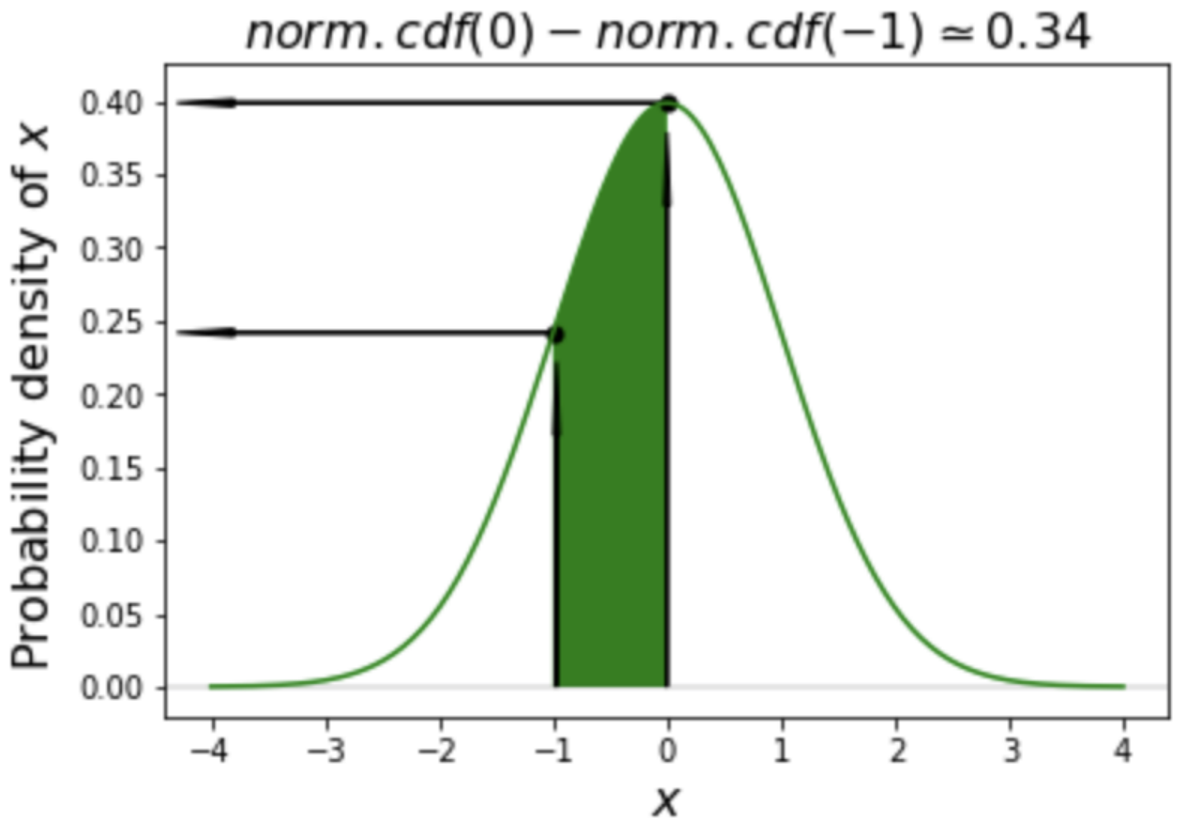
Probability density examples



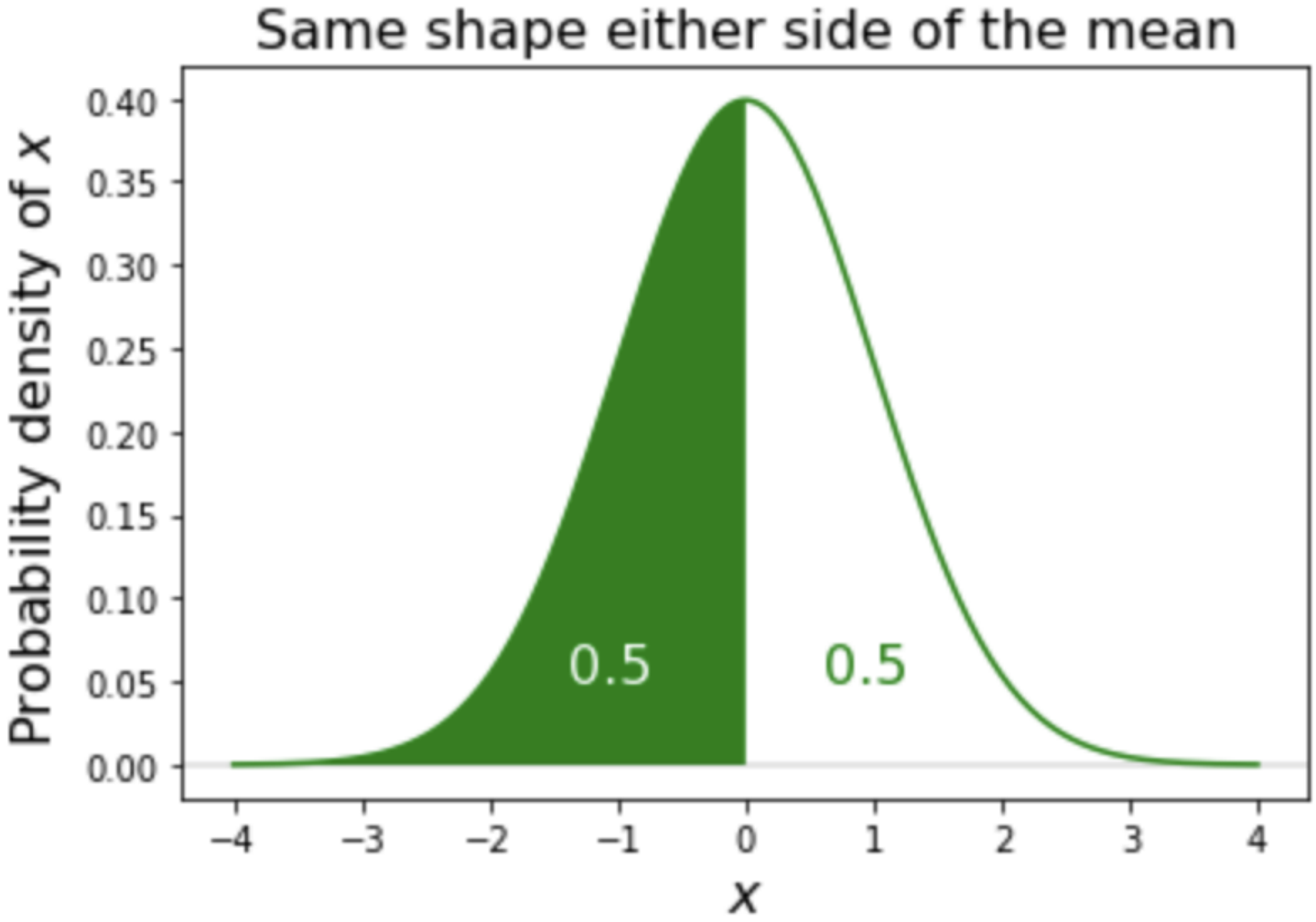


Probability density and probability

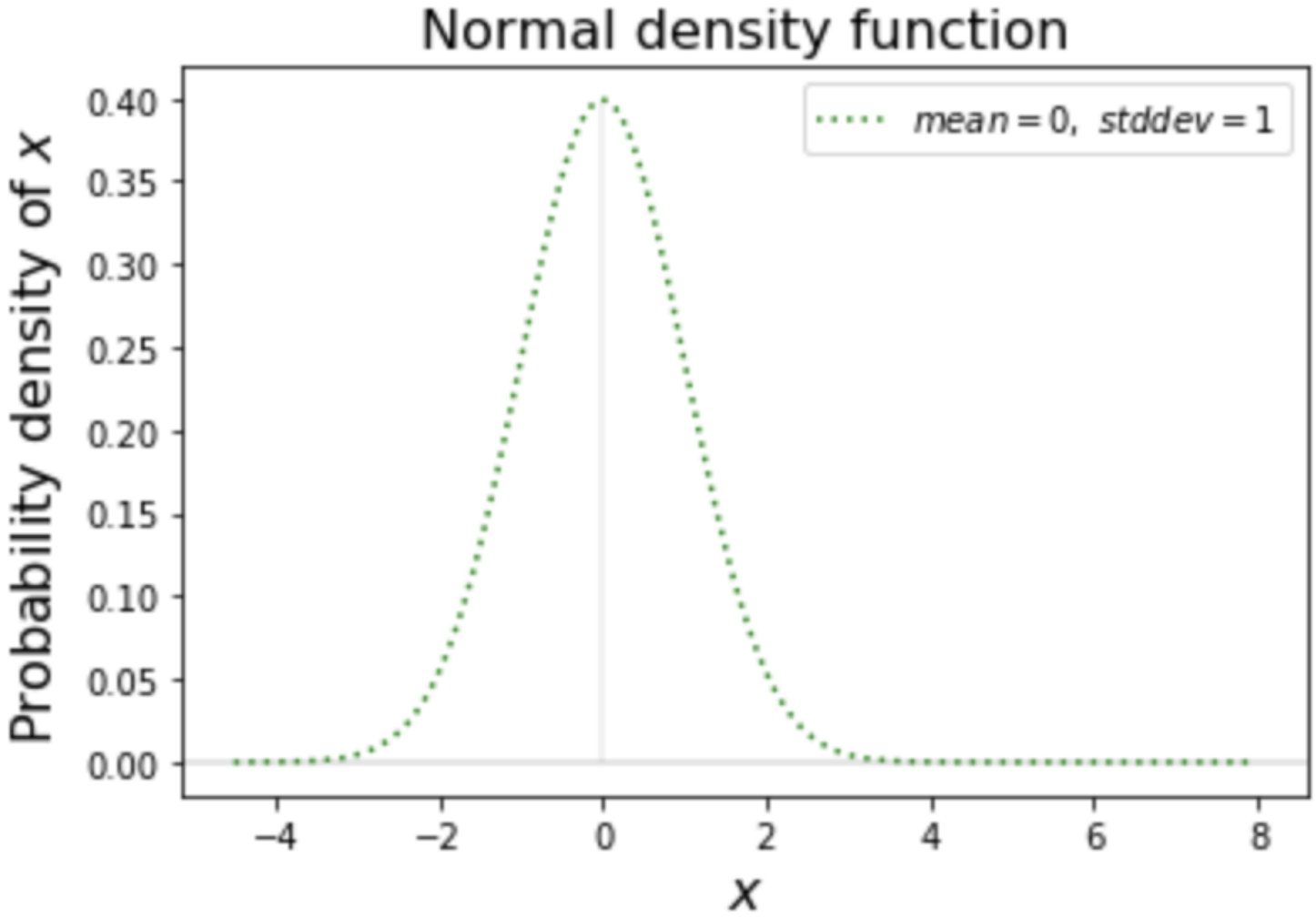




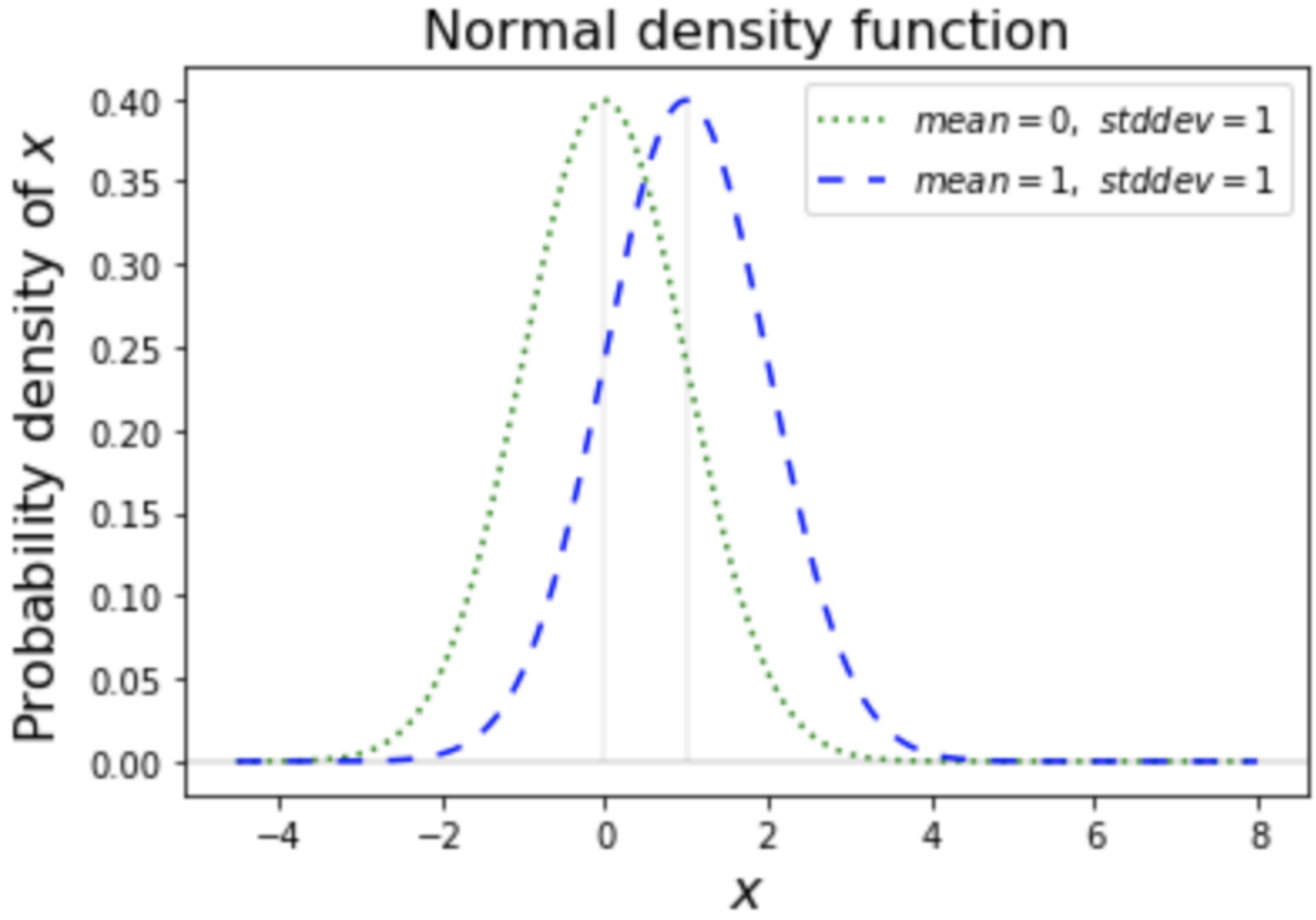
Symmetry


Same shape either side of the mean
Probability density of x
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.5
0.5
−4
−3
−2
−1
0
1
2
3
4
x

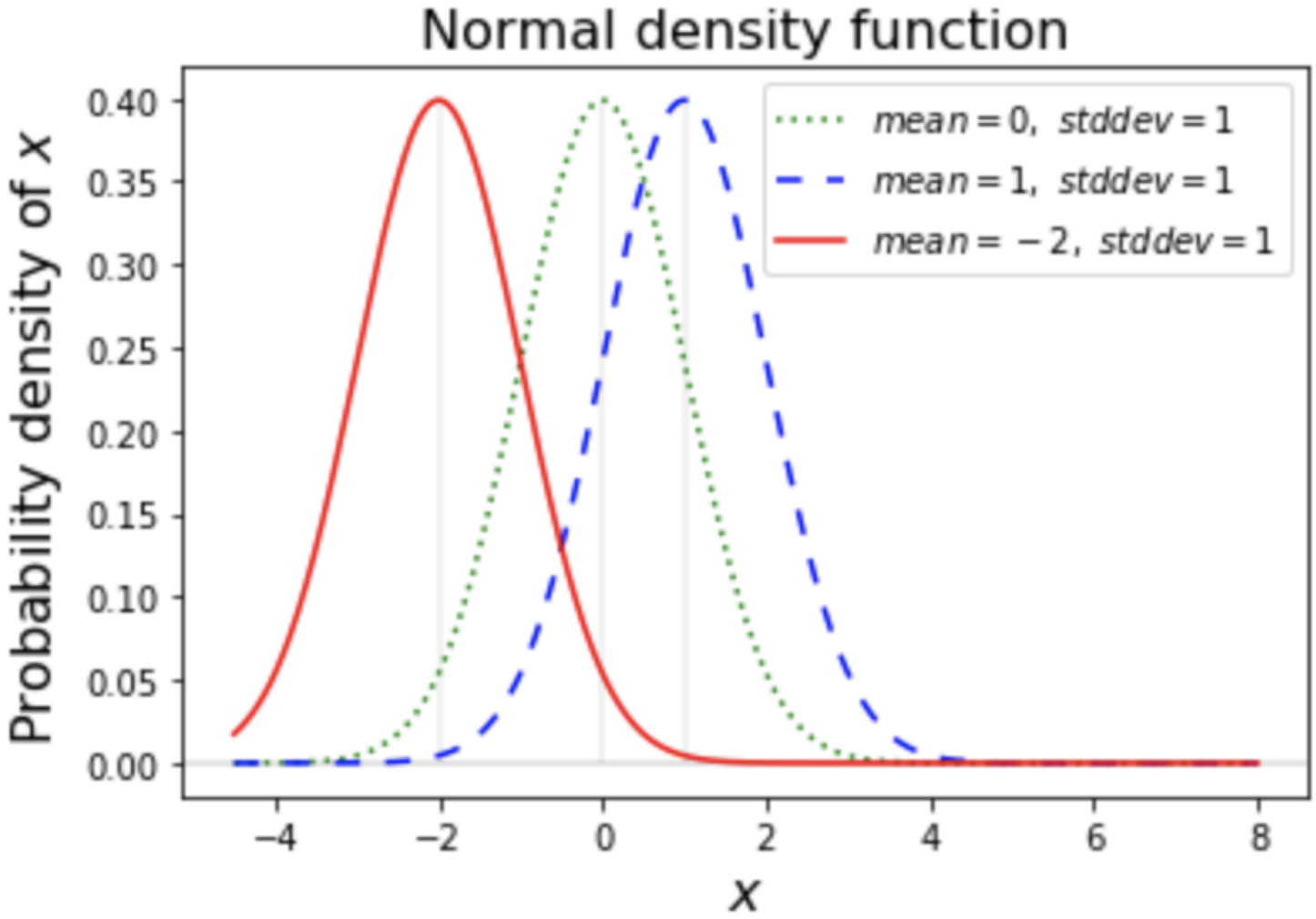
Mean


Normal density function
mean = 0, stddev = 1
Probability density of x
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−2
0
2
4
6
8
x

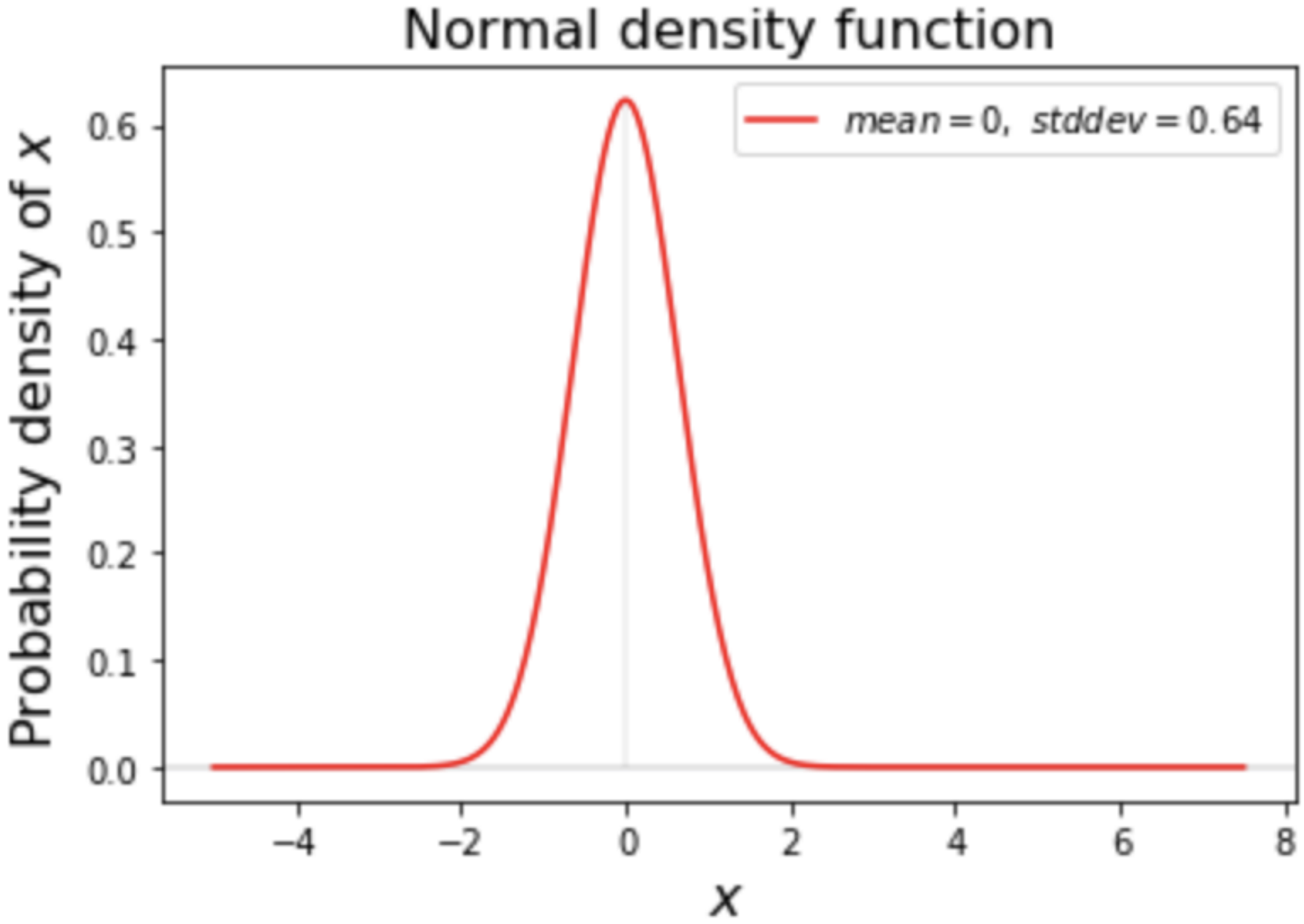
Mean (Cont.)


Normal density function
Probability density of x
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−2
0
2
4
6
8
x
mean = 0, stddev = 1
mean = 1, stddev = 1

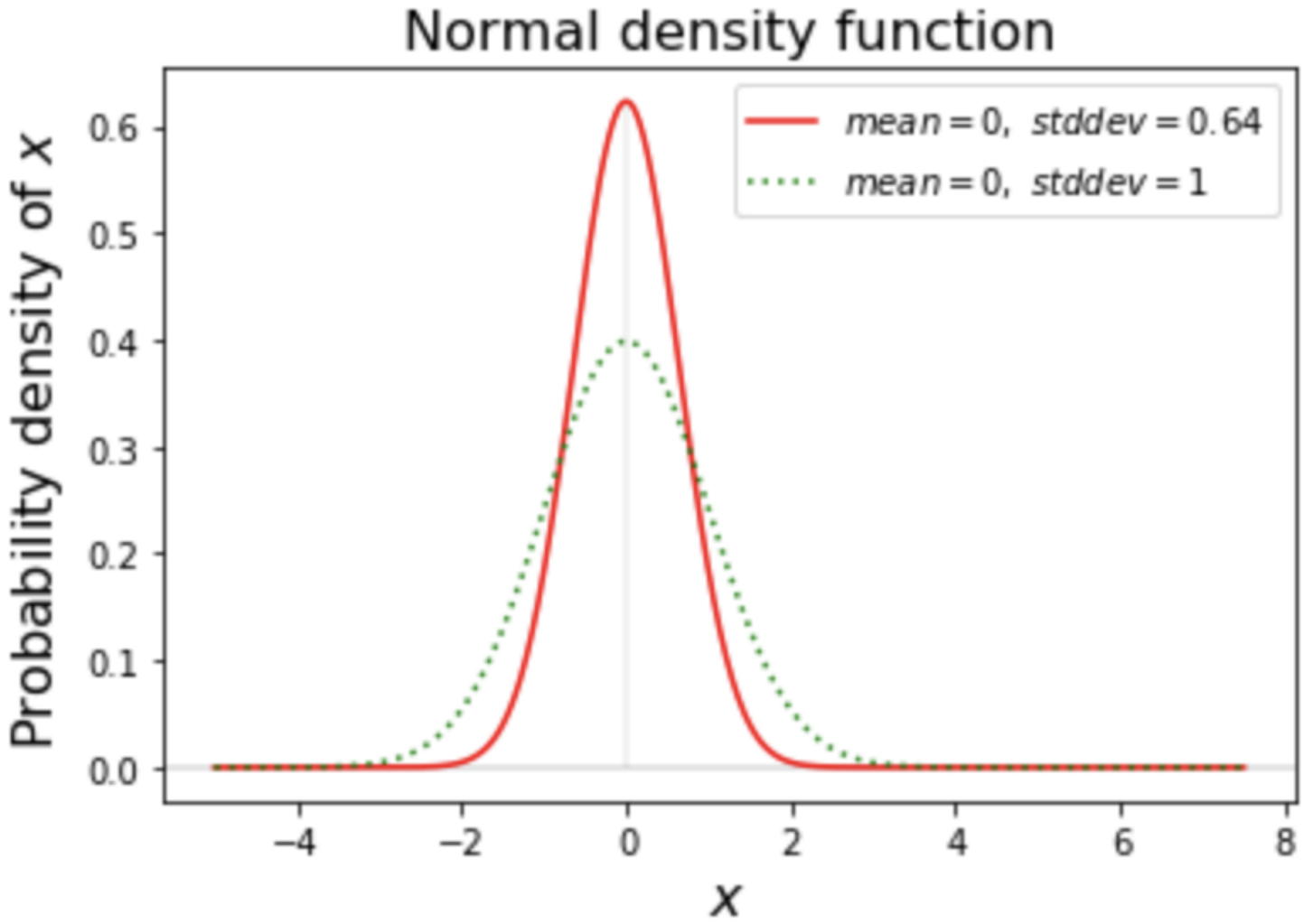
Mean (Cont.)


Normal density function
Probability density of x
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−2
0
2
4
6
8
x
mean = 0, stddev = 1
mean = 1, stddev = 1
mean = −2, stddev = 1

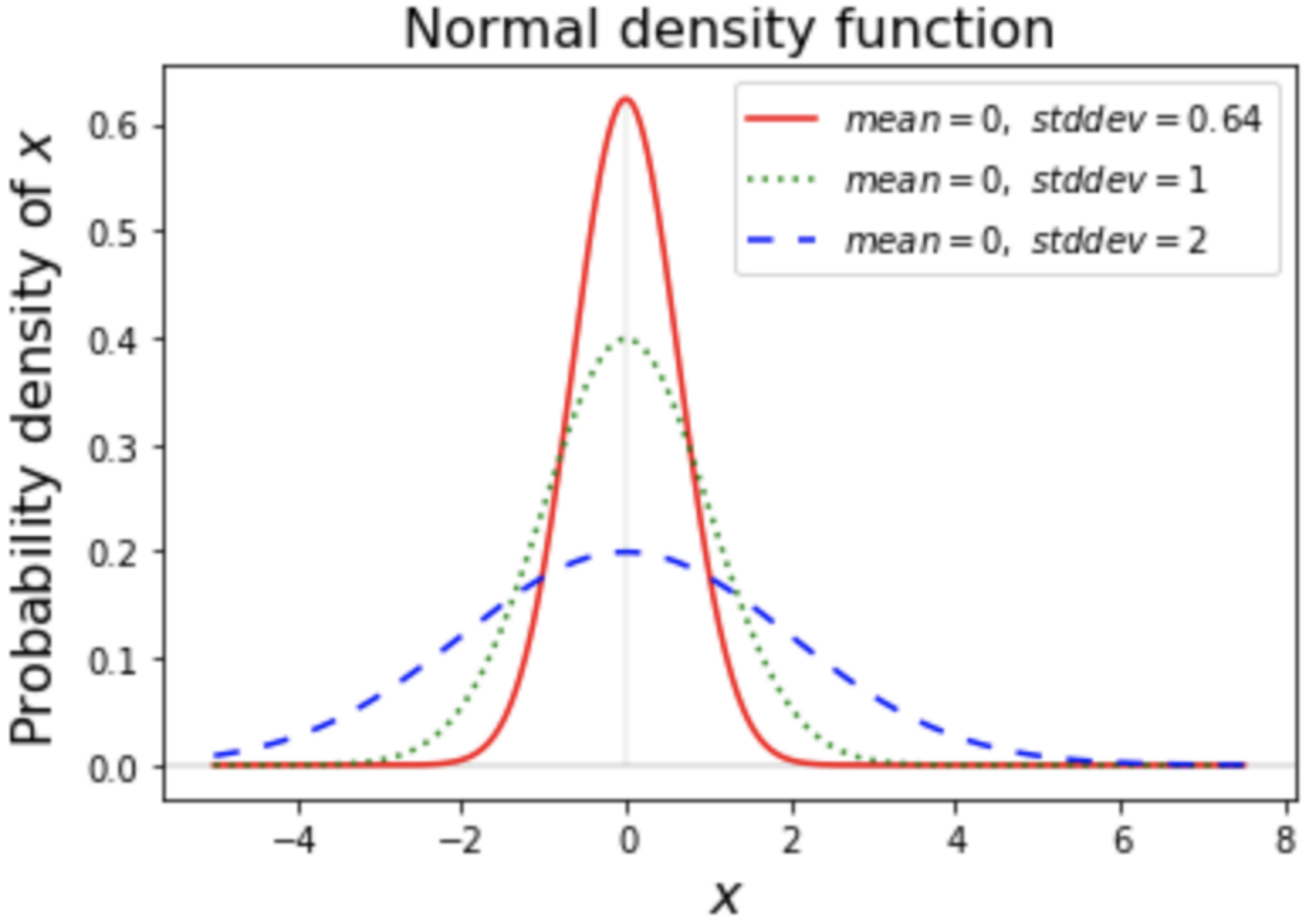
Standard deviation



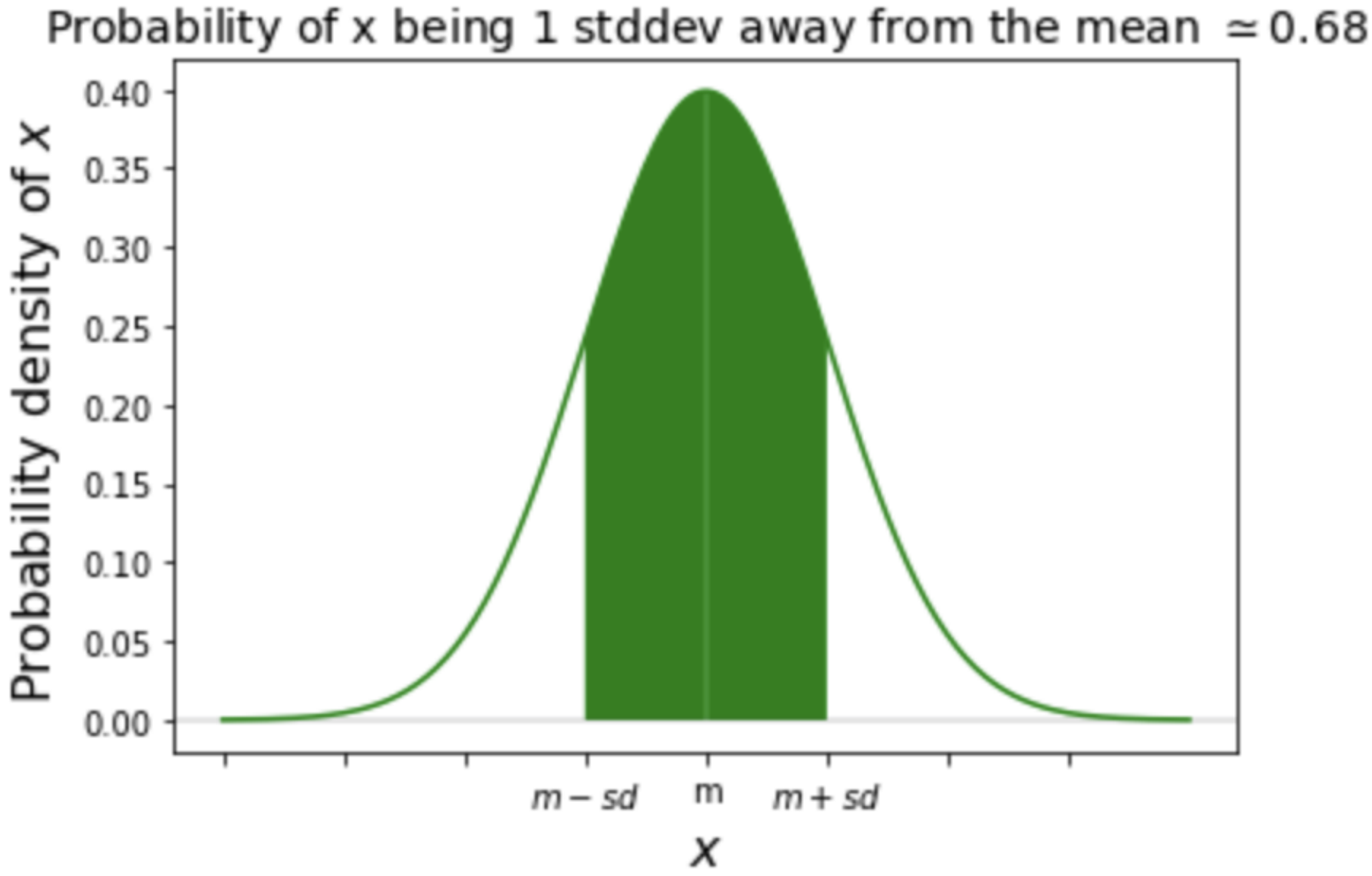
Standard deviation (Cont.)



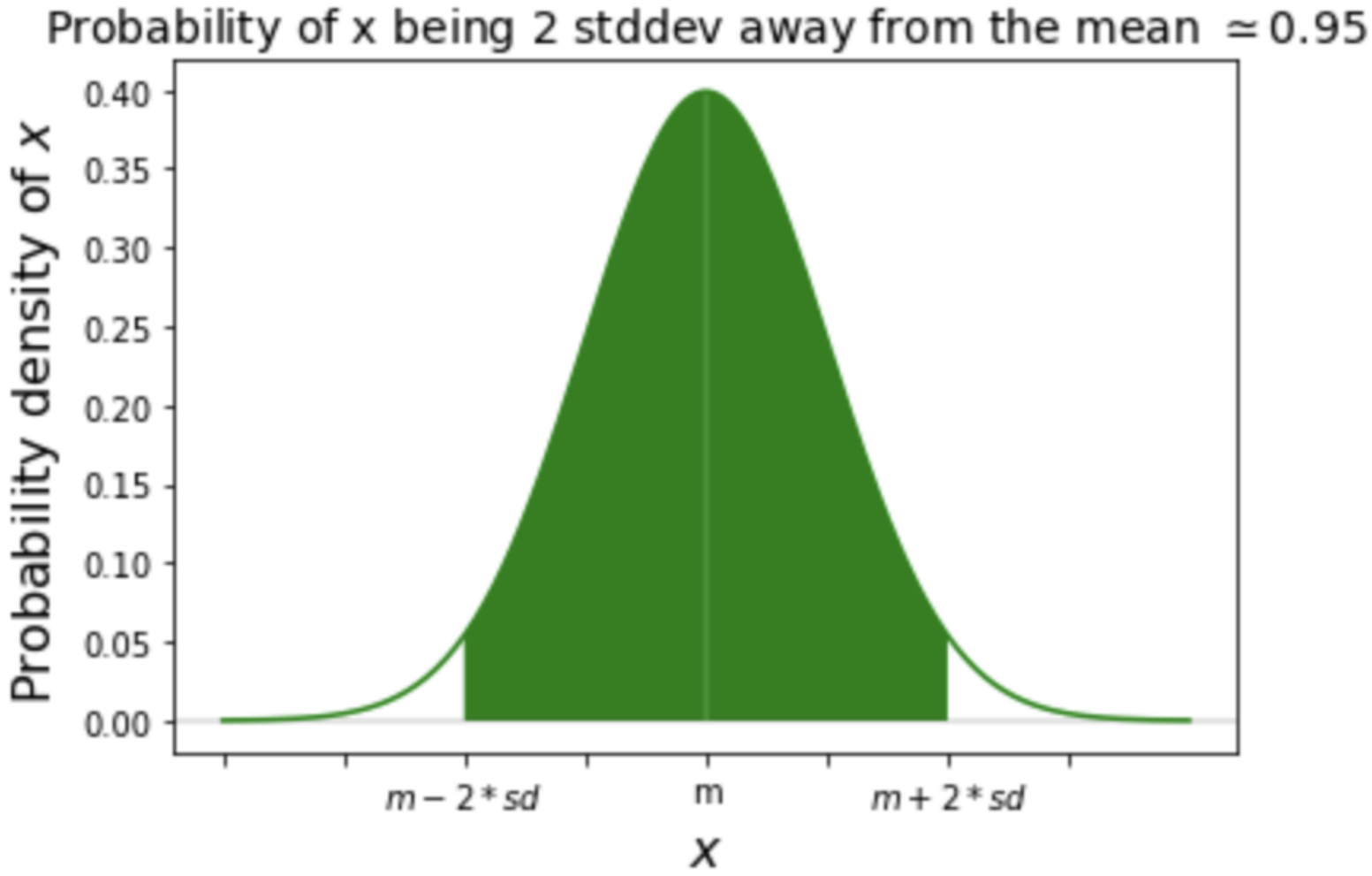
Standard deviation (Cont.)



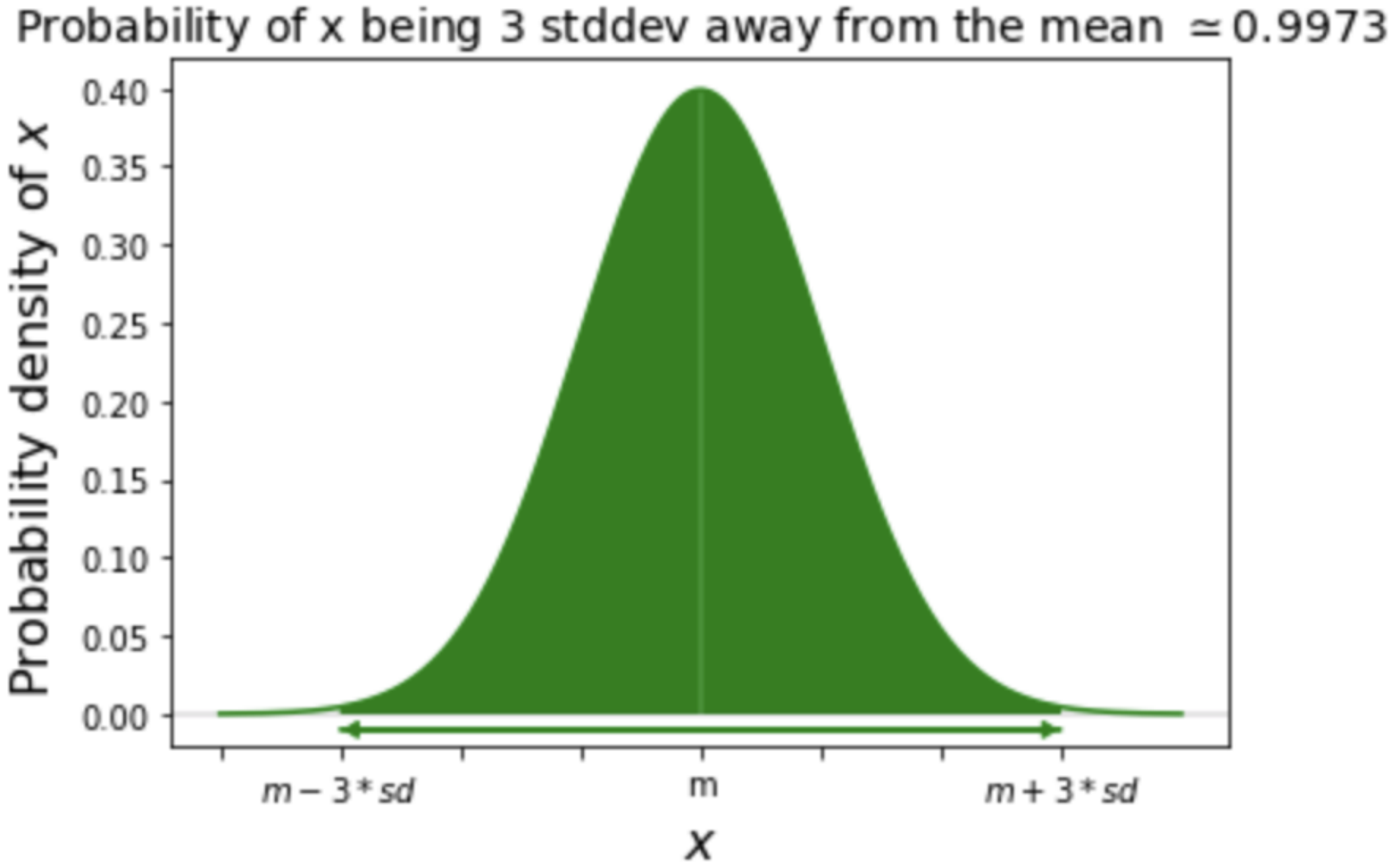
One standard deviation



Two standard deviations



Three standard deviations



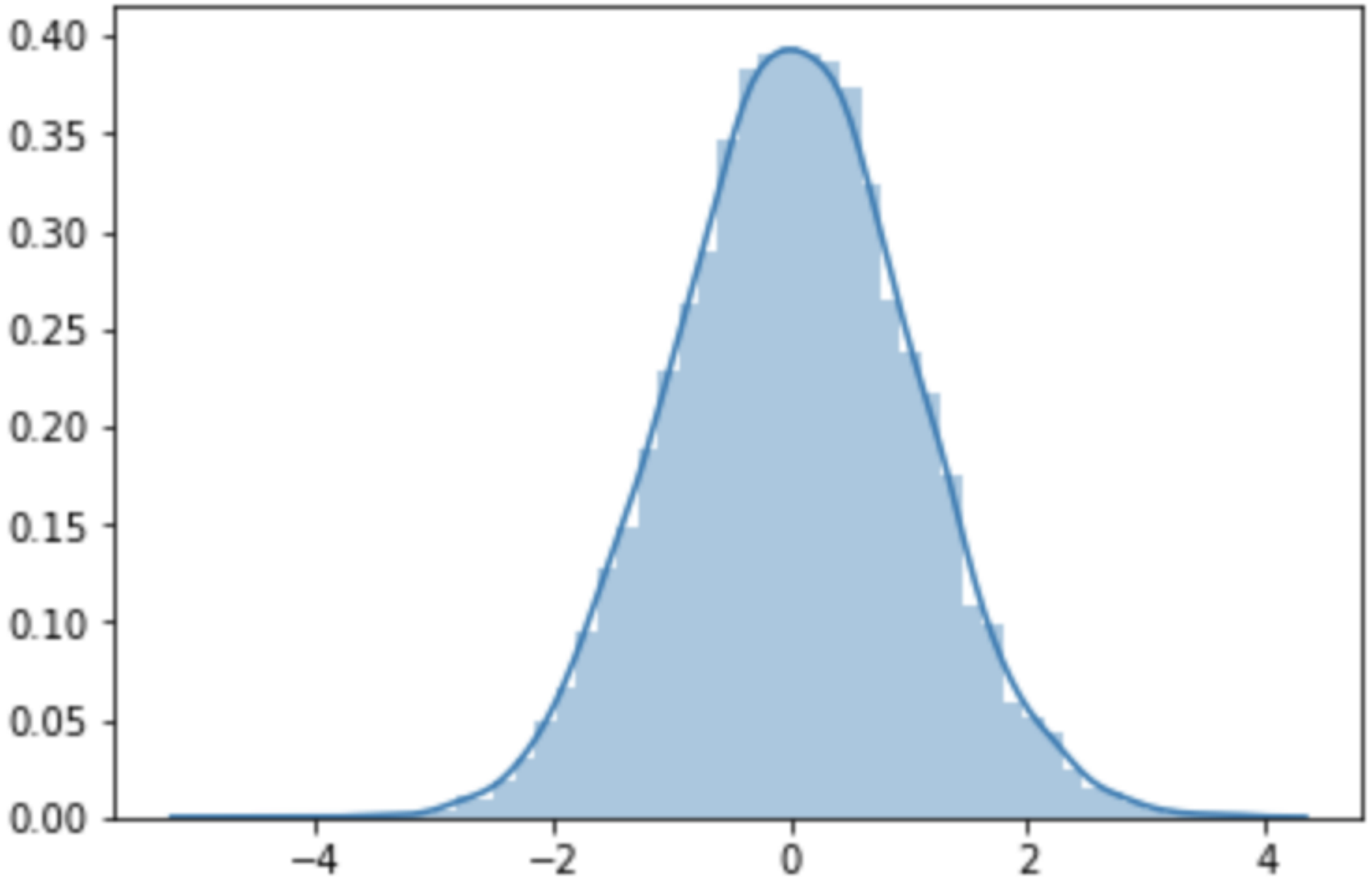
Normal sampling

```
# Import norm, matplotlib.pyplot, and seaborn
from scipy.stats import norm
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Create the sample using norm.rvs()
sample = norm.rvs(loc=0, scale=1, size=10000, random_state=13)
```

```
# Plot the sample
sns.distplot(sample)
plt.show()
```

Normal sampling (Cont.)



Let's do some exercises with normal distributions

FOUNDATIONS OF PROBABILITY IN PYTHON

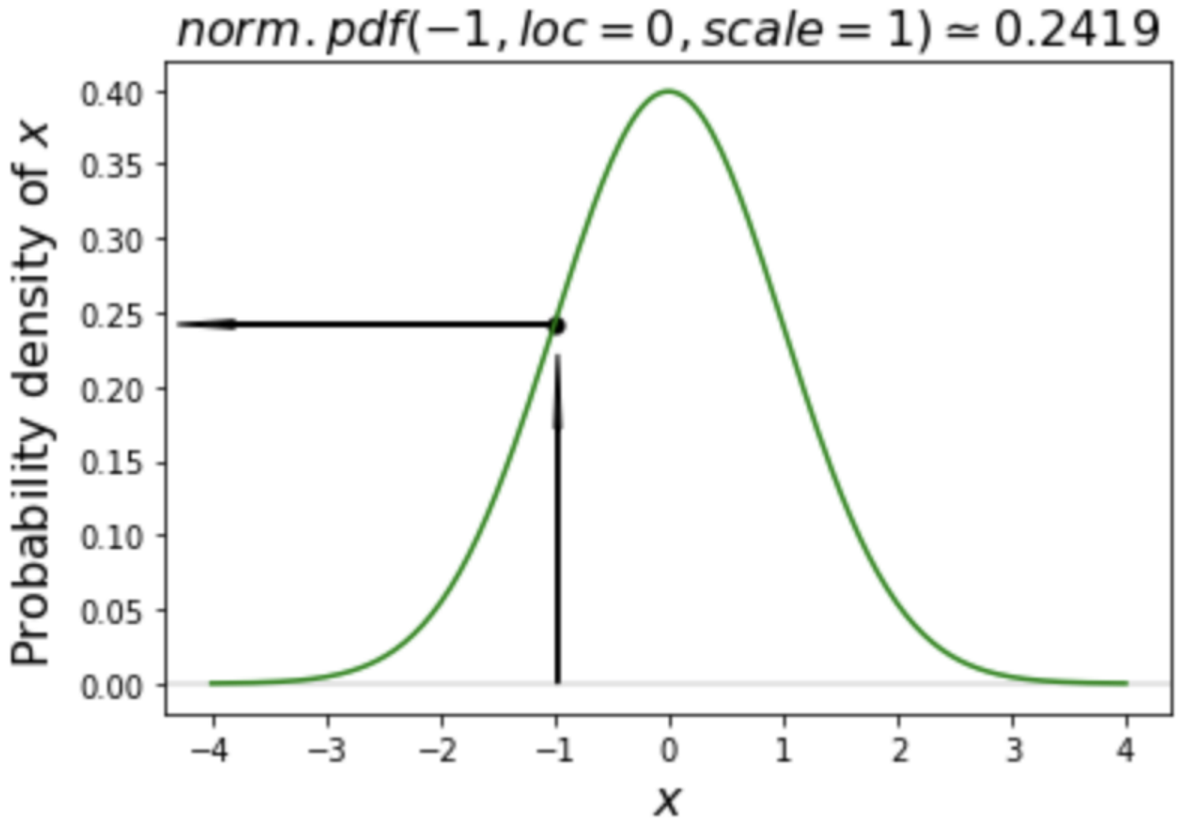
Normal probabilities

FOUNDATIONS OF PROBABILITY IN PYTHON

Alexander A. Ramírez M.
CEO @ Synergy Vision

datacamp

Probability density



In Python this can be done in a couple of lines:

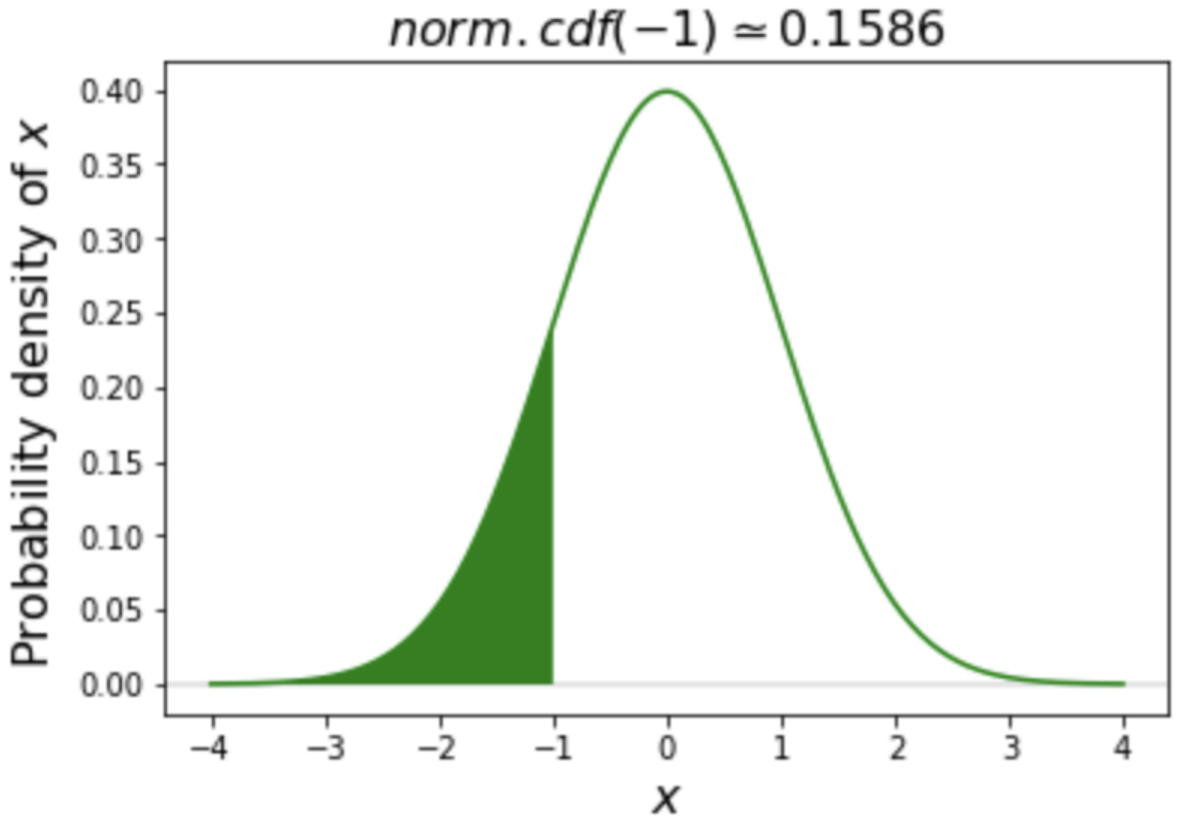
```
# Import norm
from scipy.stats import norm
```

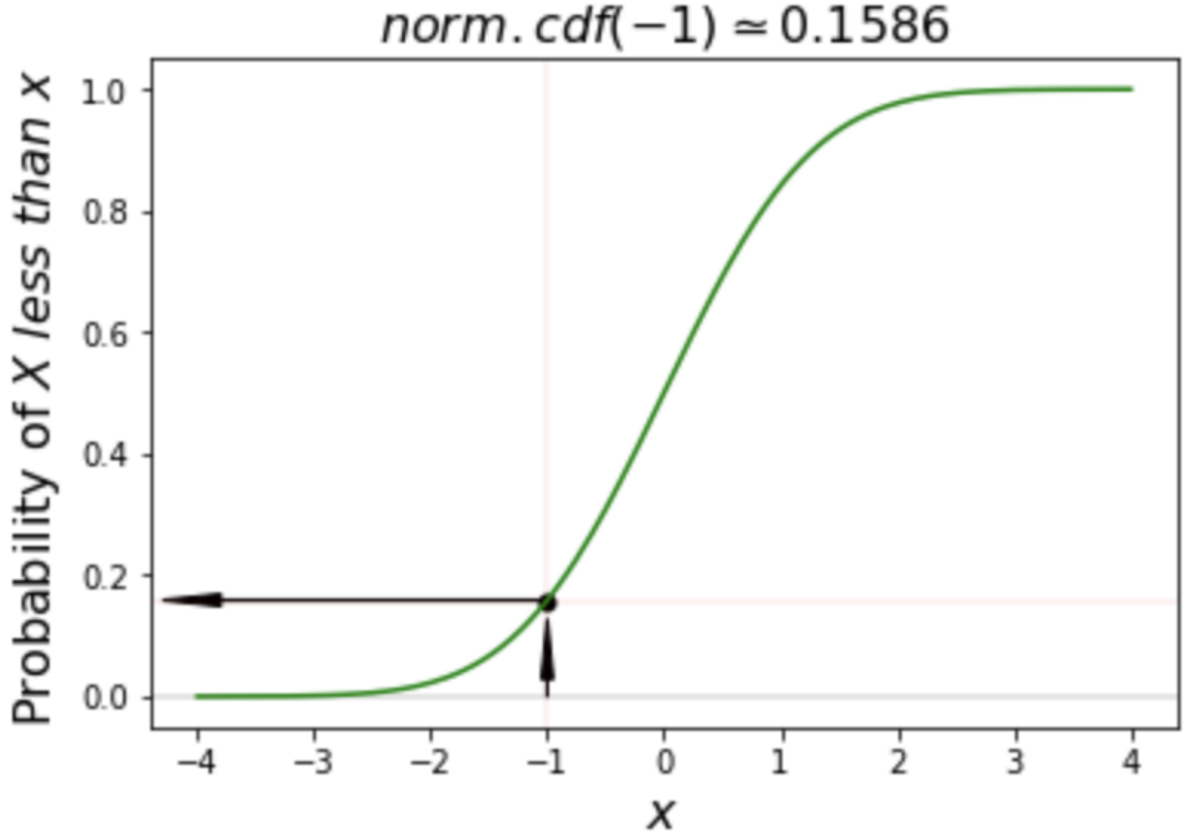
```
# Calculate the probability density
# with pdf
norm.pdf(-1, loc=0, scale=1)
```

```
0.24197072451914337
```

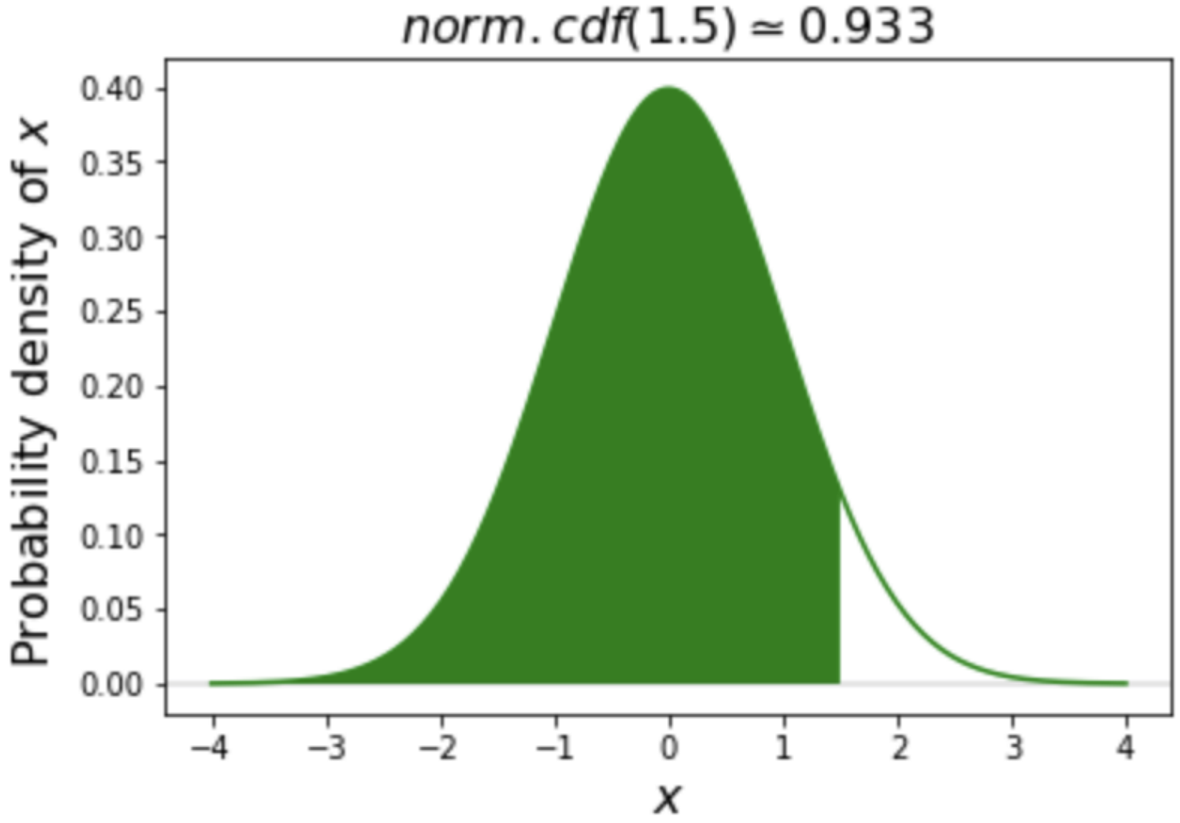
`loc` parameter specifies the mean and `scale` parameter specifies the standard deviation.

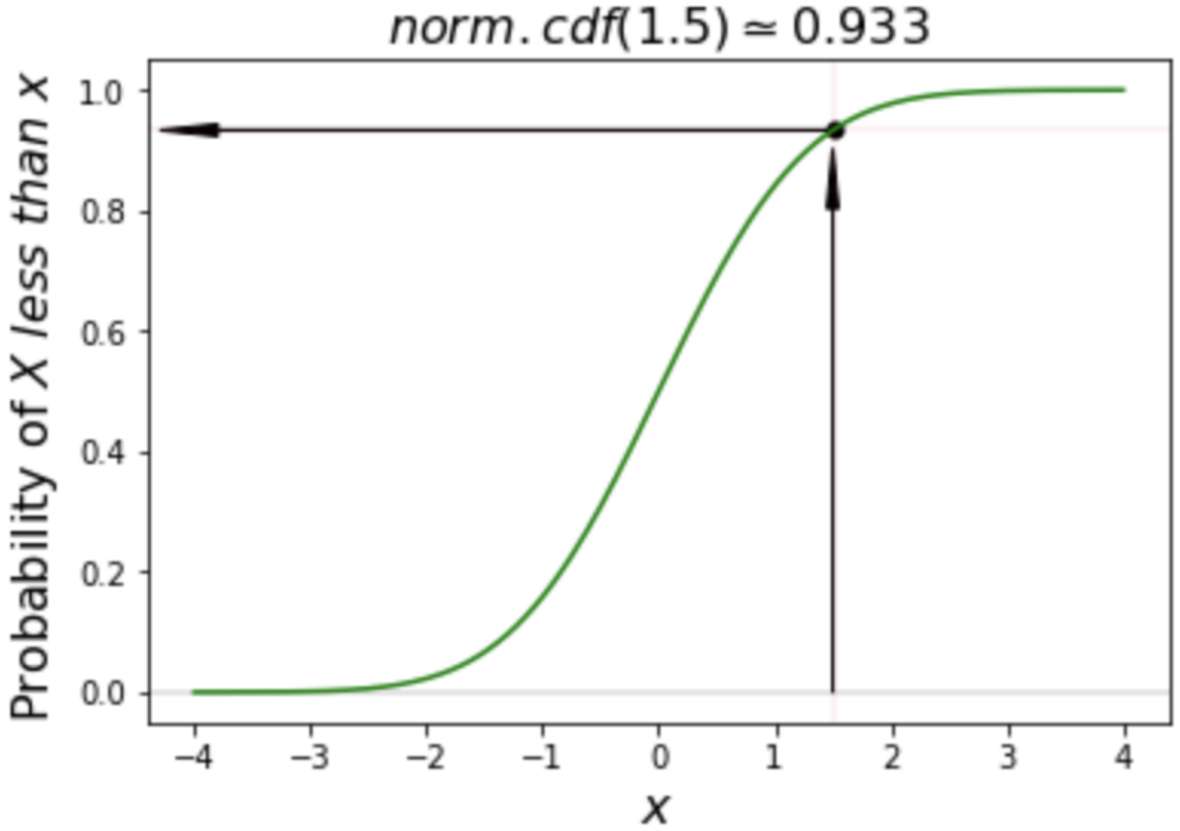
pdf() vs. cdf()


norm.cdf(−1) ≃ 0.1586
Probability density of x
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−3
−2
−1
0
1
2
3
4
x

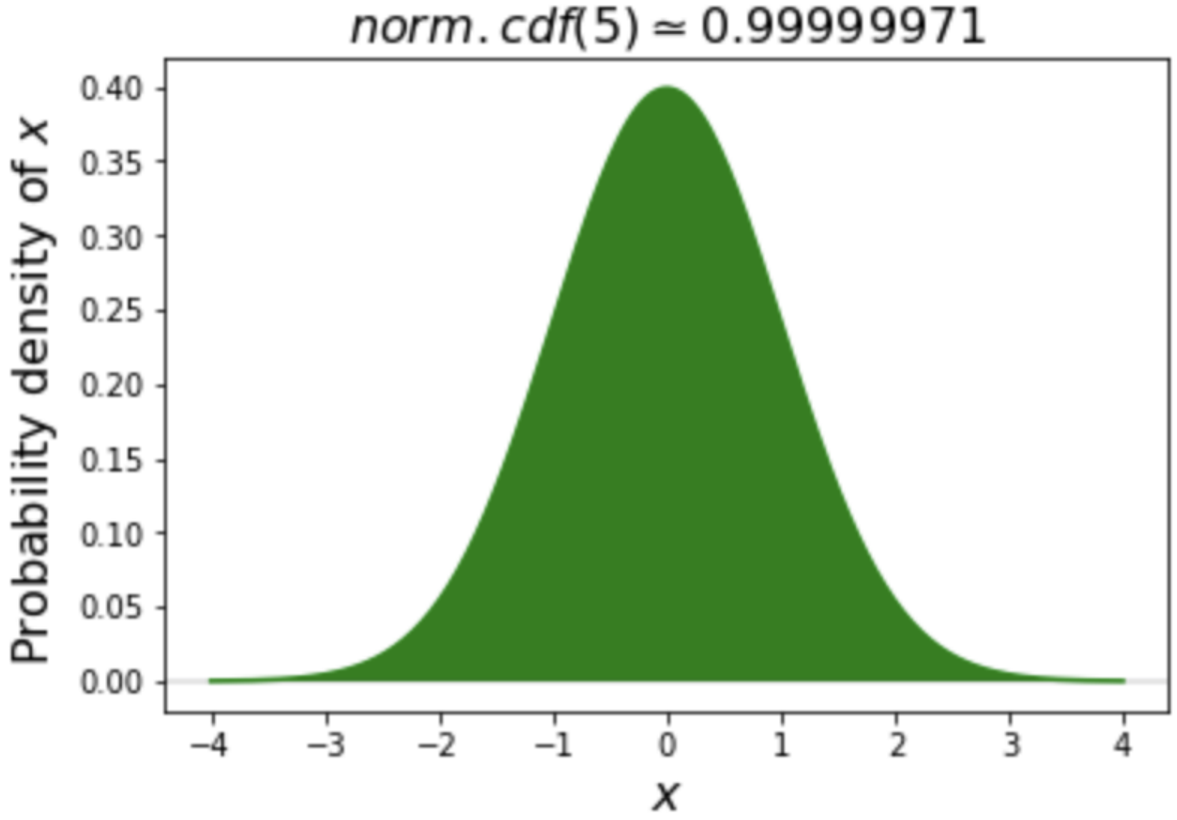

norm.cdf(−1) ≃ 0.1586
Probability of X less than x
1.0
0.8
0.6
0.4
0.2
0.0
−4
−3
−2
−1
0
1
2
3
4
x

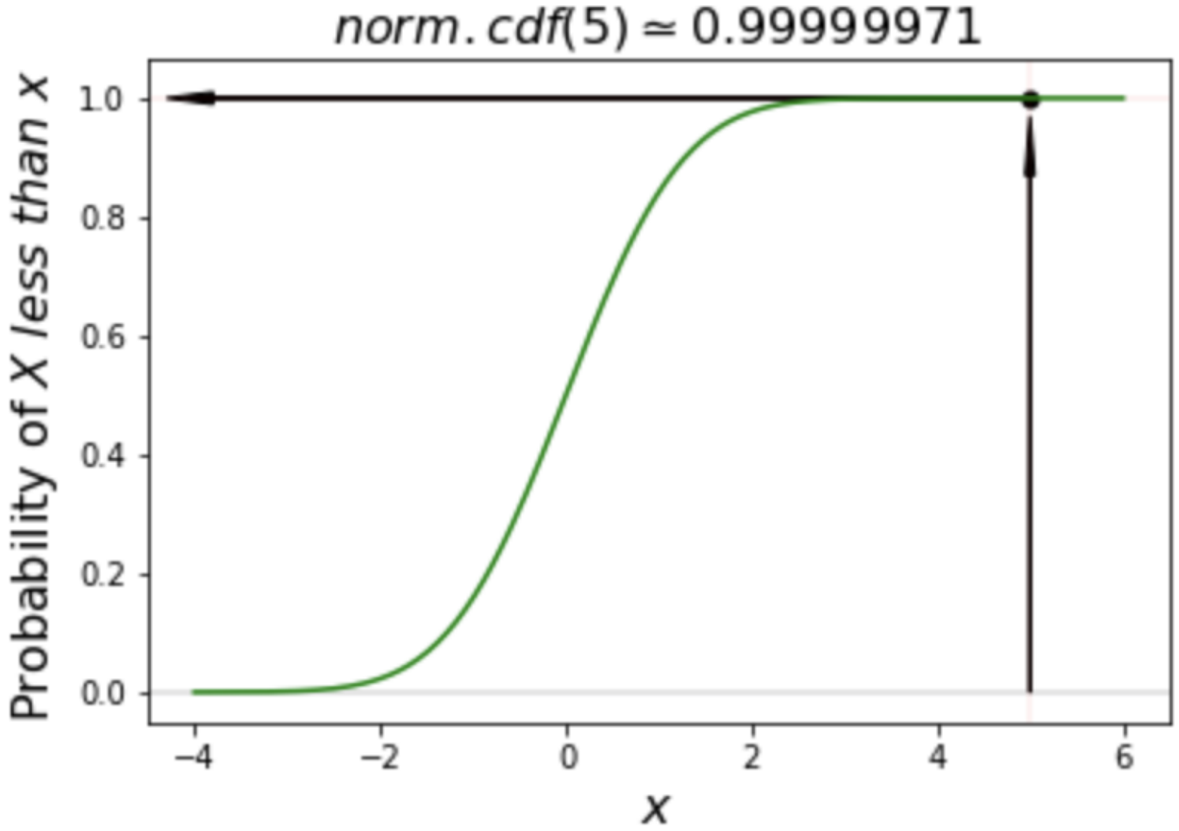
pdf() vs. cdf() (Cont.)



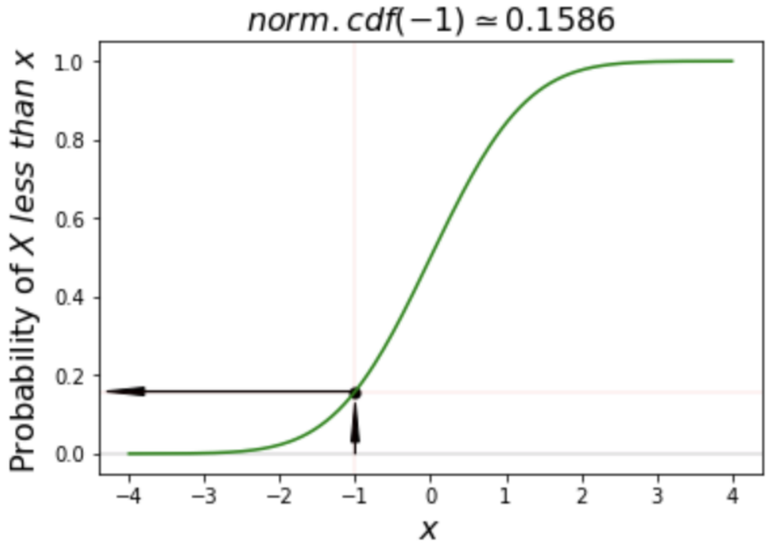


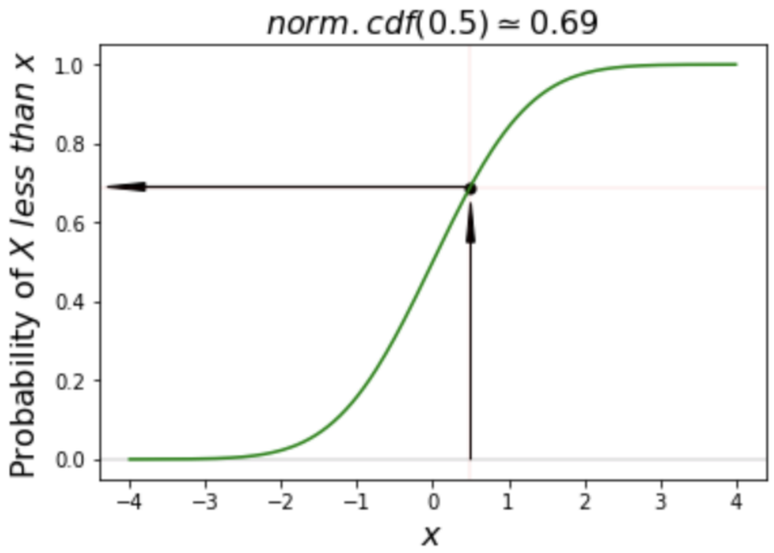
pdf() vs. cdf() (Cont.)





Cumulative distribution function examples





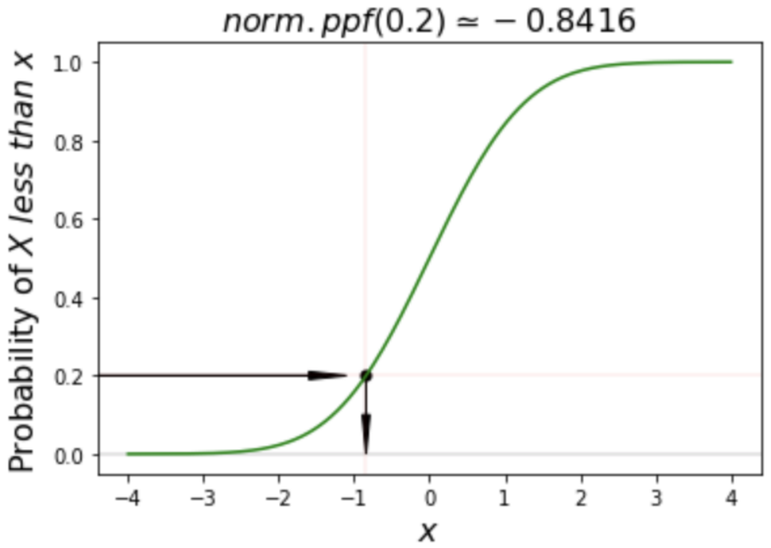

```
# Calculate cdf of -1
norm.cdf(-1)
```

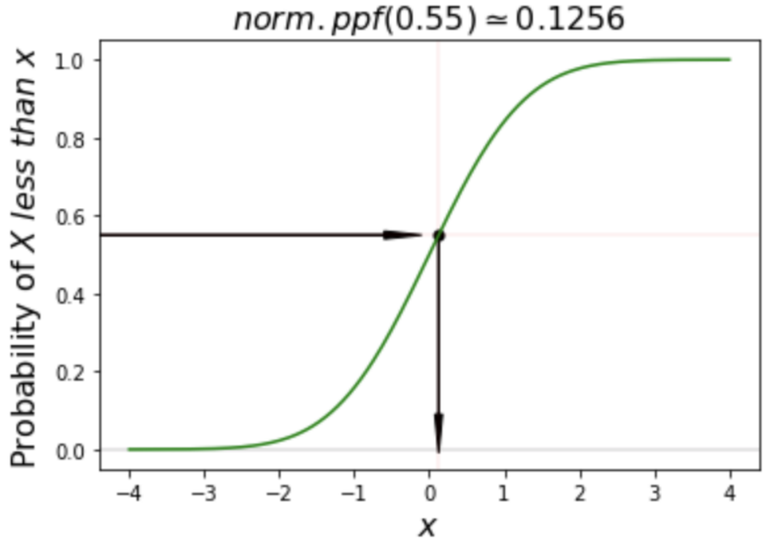
```
0.15865525393145707
```

```
# Calculate cdf of 0.5
norm.cdf(0.5)
```

```
0.6914624612740131
```

The percent point function (ppf)





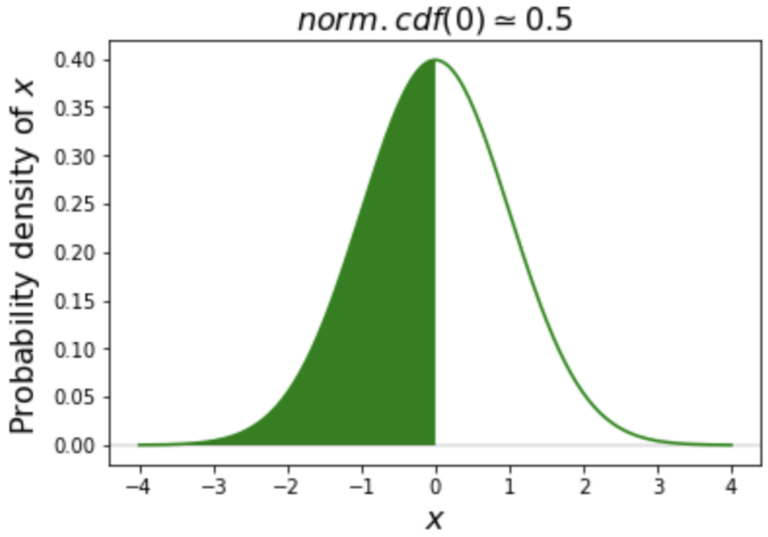

```
# Calculate ppf of 0.2
norm.ppf(0.2)
```

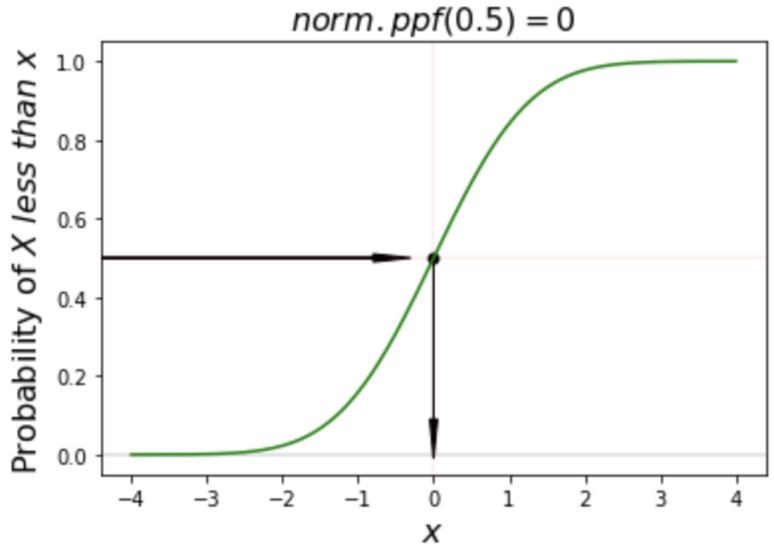
```
-0.8416212335729142
```

```
# Calculate ppf of 55%
norm.ppf(0.55)
```

```
0.1256613468550741 6
```

ppf() is the inverse of cdf()





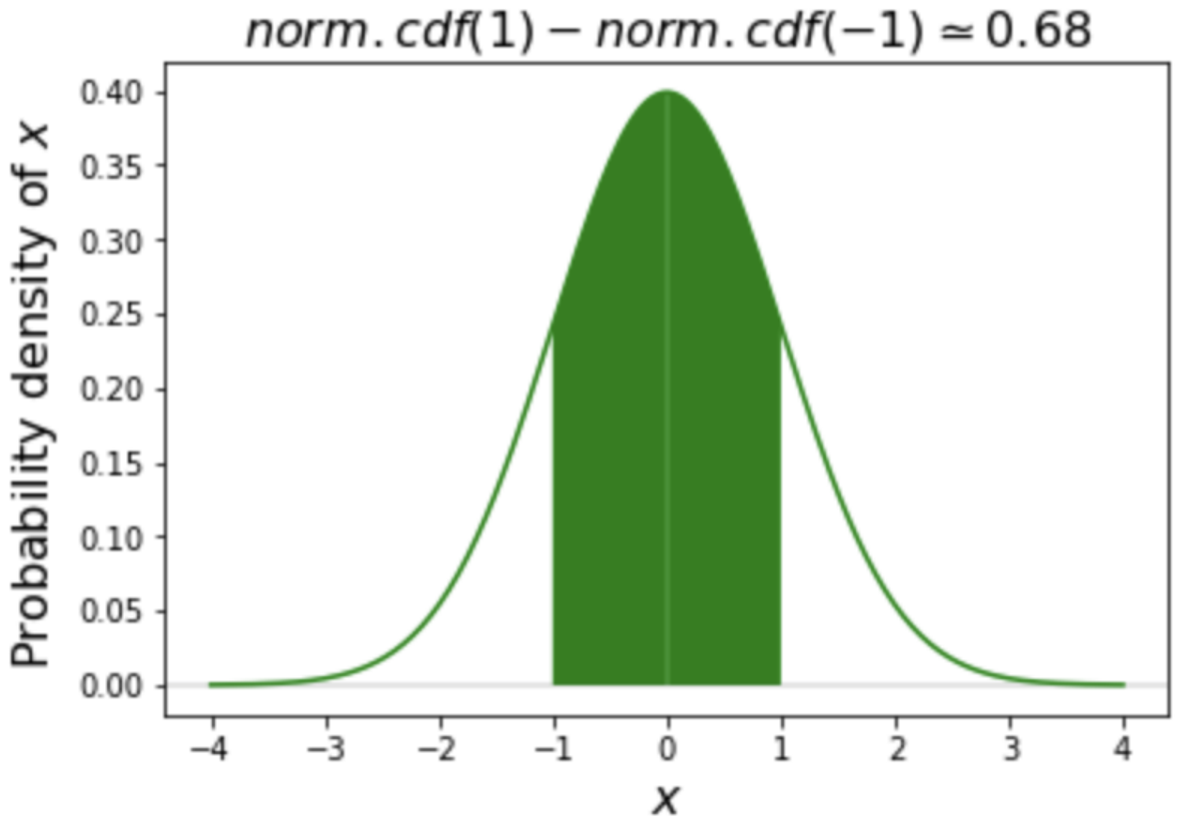

```
# Calculate cdf of value 0
norm.cdf(0)
```

```
0.5
```

```
# Calculate ppf of probability 50%
norm.ppf(0.5)
```

```
0
```

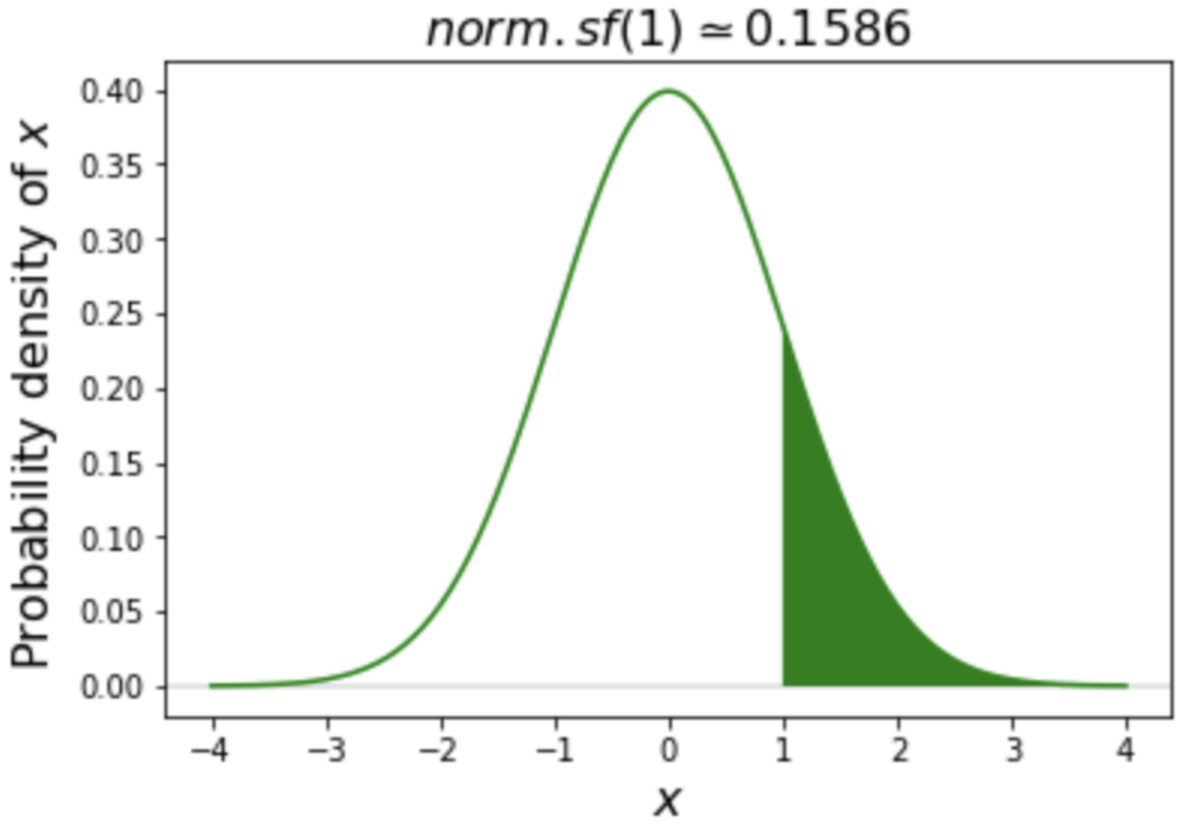
Probability between two values




```
# Create our variables
a = -1
b = 1
# Calculate the probability between
# two values, subtracting
norm.cdf(b) - norm.cdf(a)
```

```
0.6826894921370859
```

Tail probability

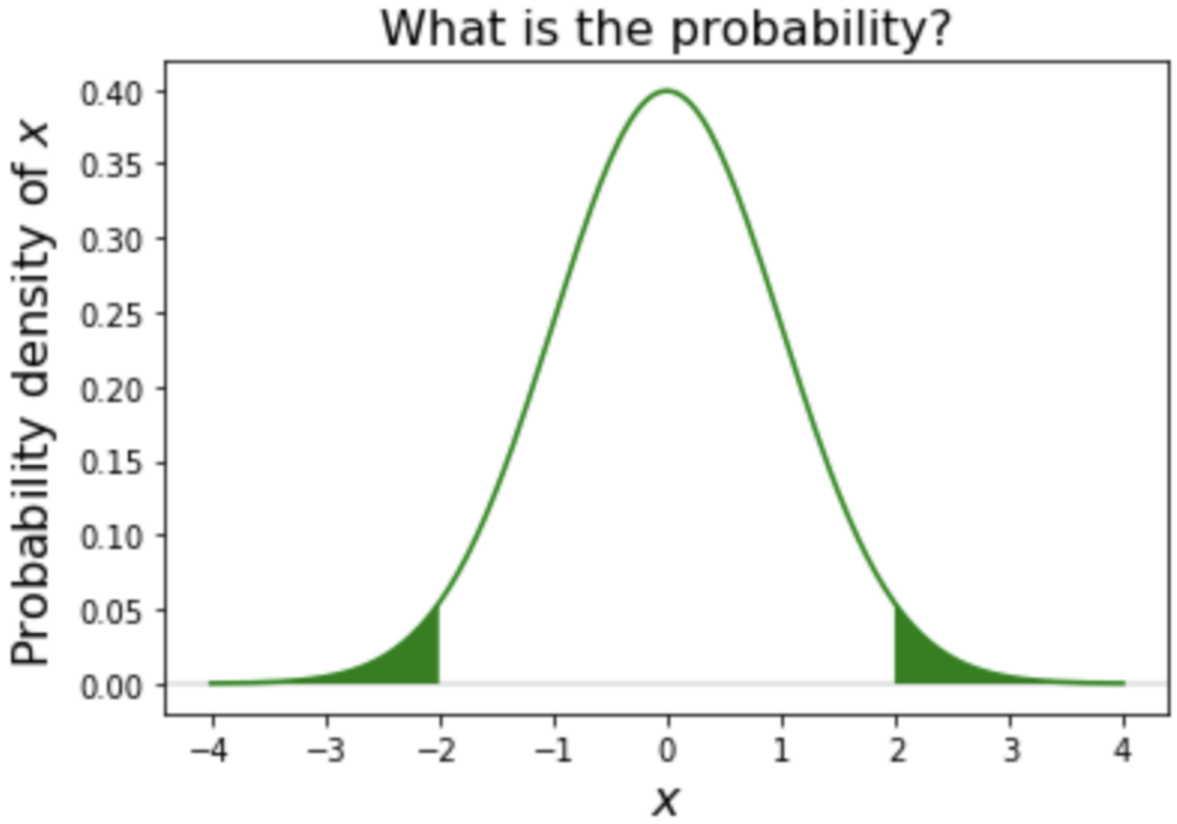



```
# Create our variable
a = 1

# Calculate the complement
# of cdf() using sf()
norm.sf(a)
```

```
0.1586525393145707
```

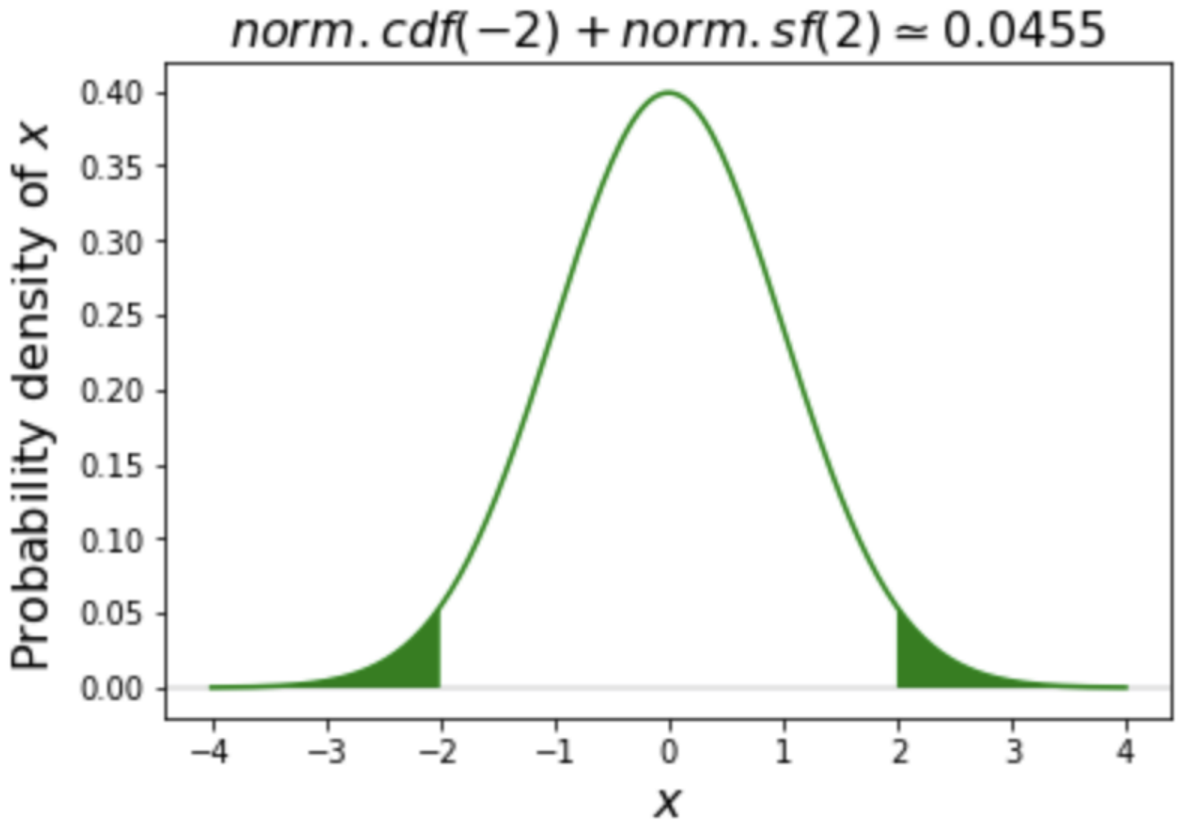
Tails




```
# Create our variables
a = -2
b = 2
# Calculate tail probability
# by adding each tail
norm.cdf(a) + norm.sf(b)
```

```
0.04550026389635839
```

Tails (Cont.)

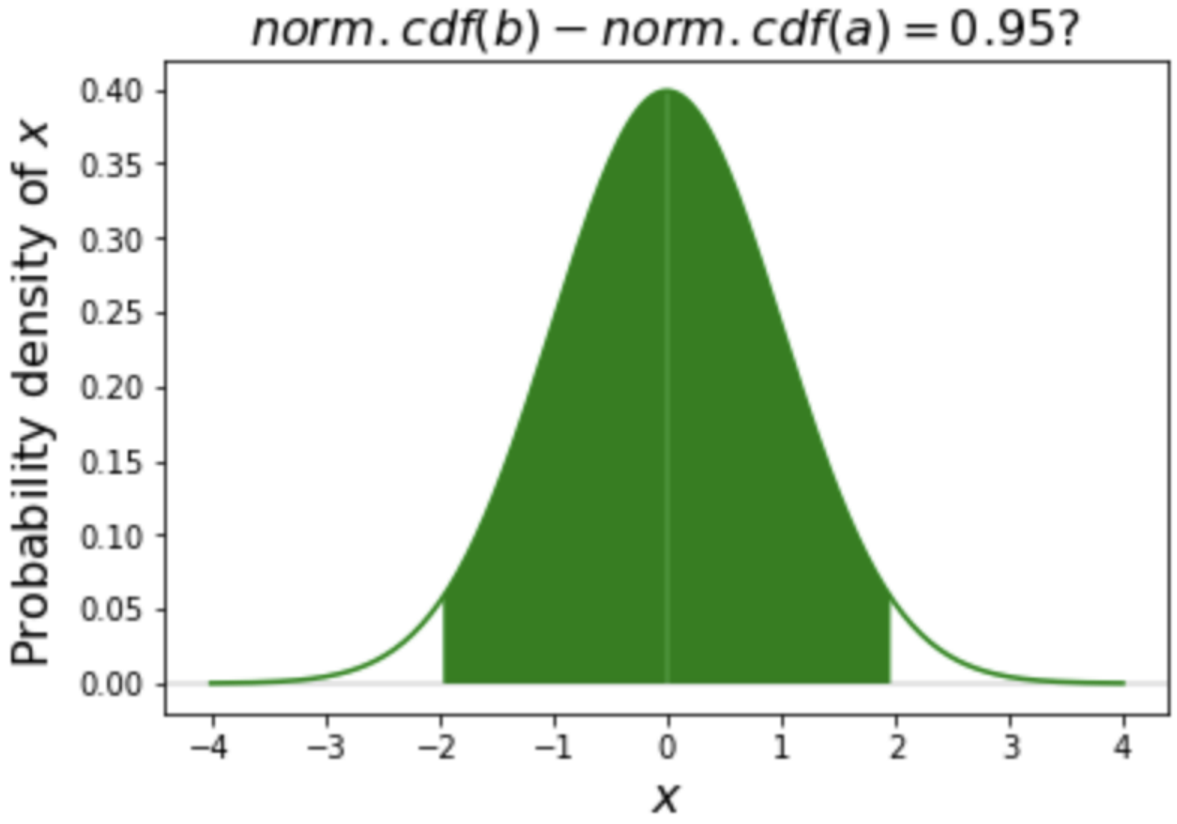



```
# Create our variables
a = -2
b = 2
```

```
# Calculate tail probability
# by adding each tail
norm.cdf(a) + norm.sf(b)
```

```
0.04550026389635839
```

Intervals




```
# Create our variable
alpha = 0.95

# Calculate the interval
norm.interval(alpha)
```

```
(-1.959963984540054, 1.959963984540054)
```

On to some practice!

FOUNDATIONS OF PROBABILITY IN PYTHON

Poisson distributions

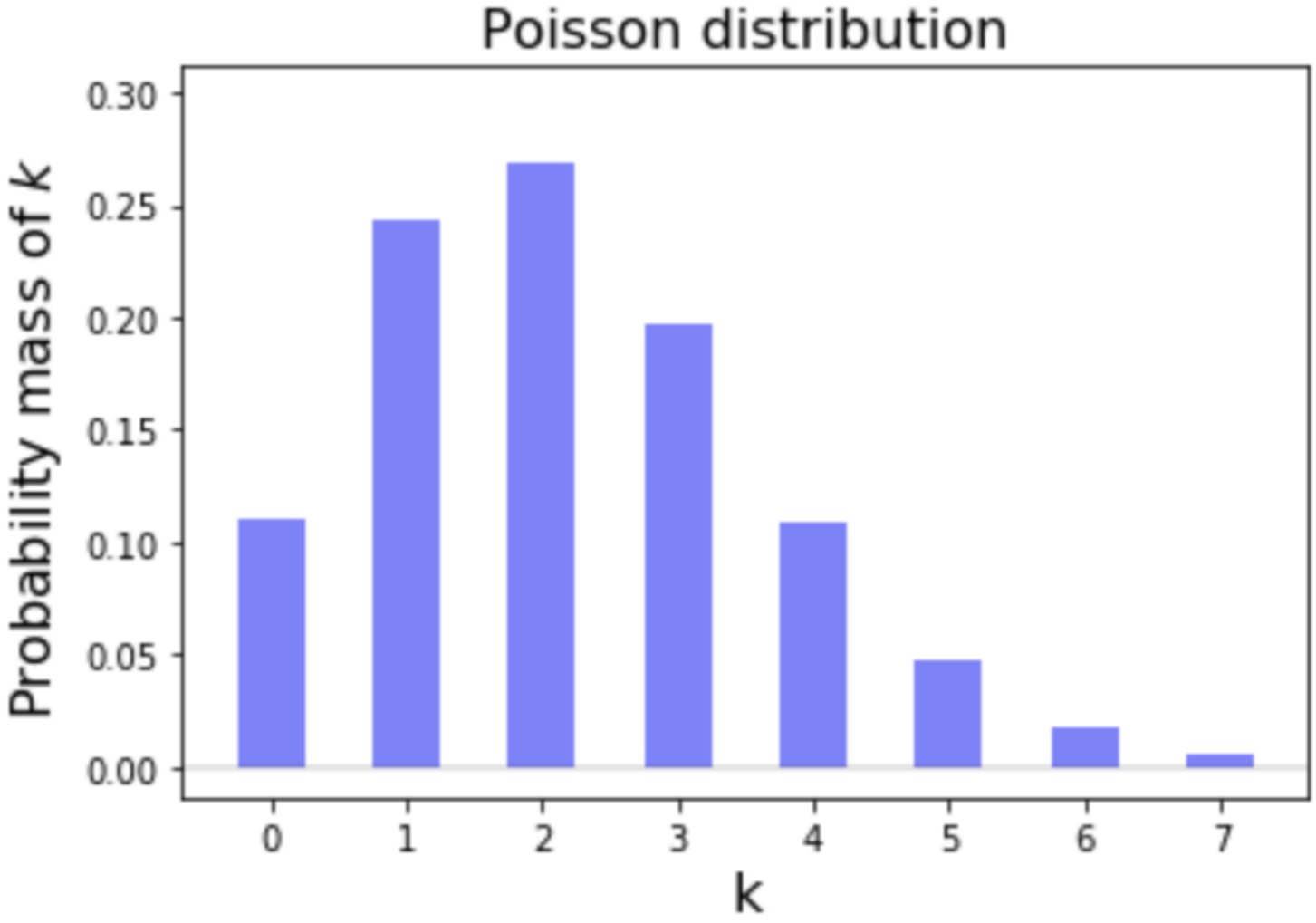
FOUNDATIONS OF PROBABILITY IN PYTHON

Alexander A. Ramírez M.
CEO @ Synergy Vision

datacamp

Poisson modeling

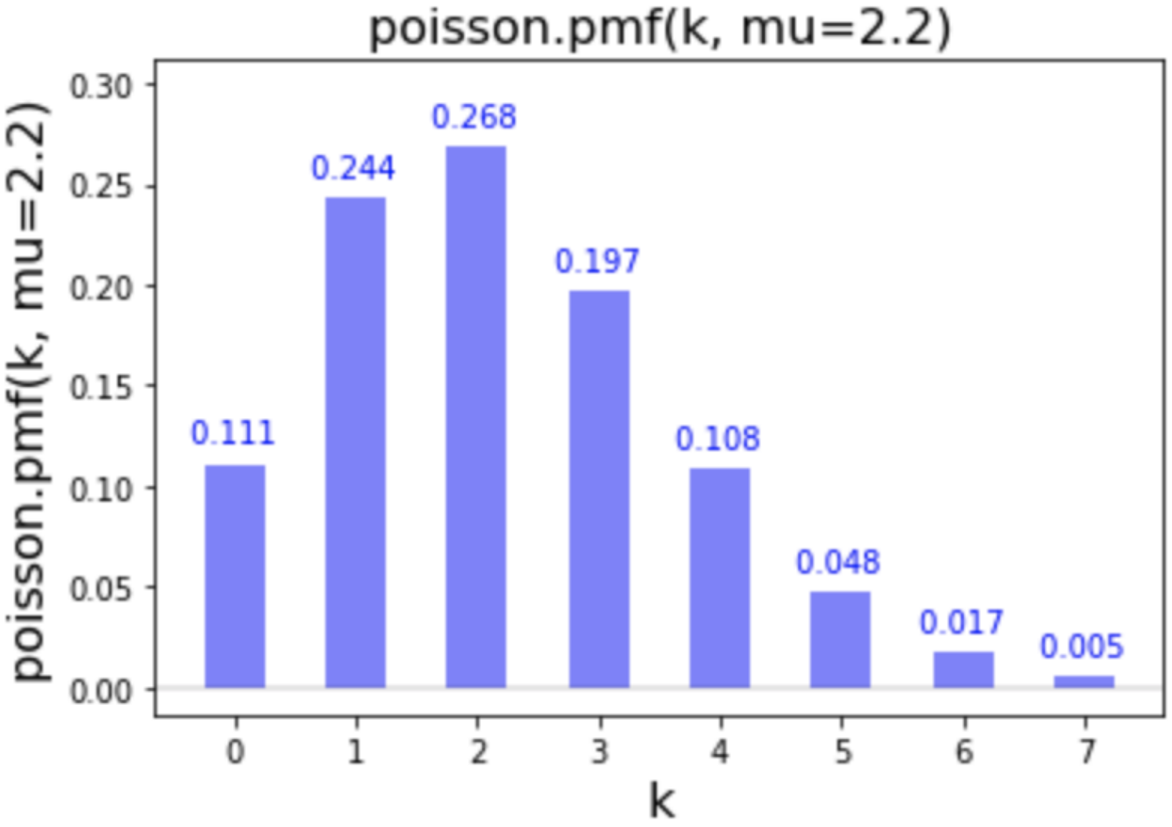
Poisson distribution properties



Probability mass function (pmf)

Imagine you have 2.2 calls per minute.

Probability mass function (pmf) (Cont.)



In Python we do the following:

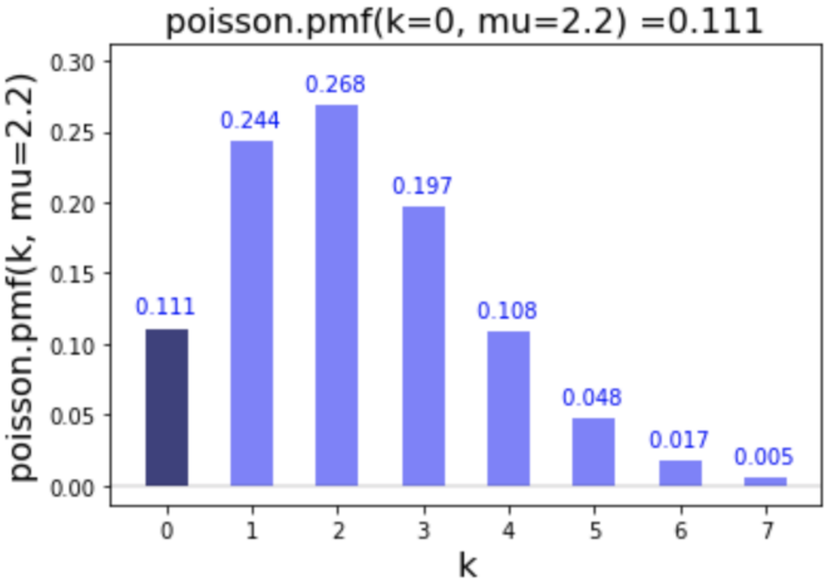
```
# Import poisson
from scipy.stats import poisson
```

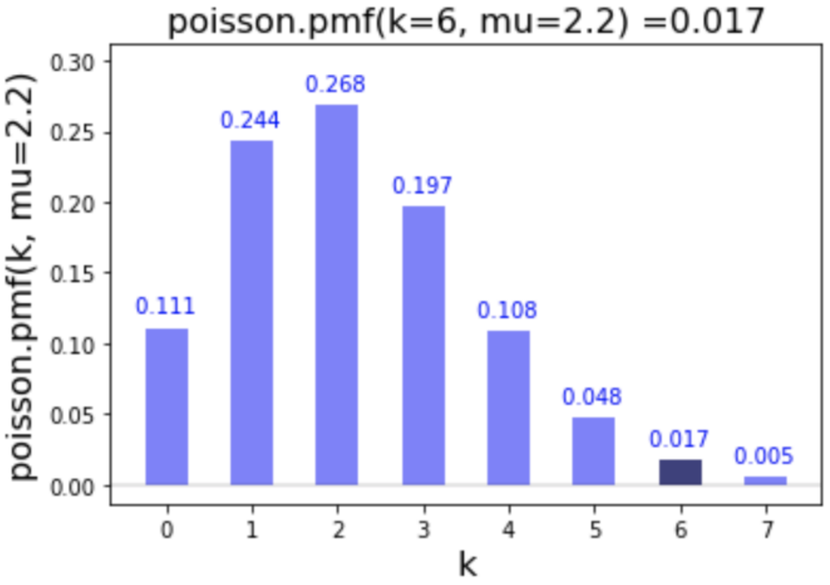
```
# Calculate the probability mass
# with pmf
poisson.pmf(k=3, mu=2.2)
```

```
0.19663867170702193
```

`mu` parameter specifies the mean of successful events.

pmf examples





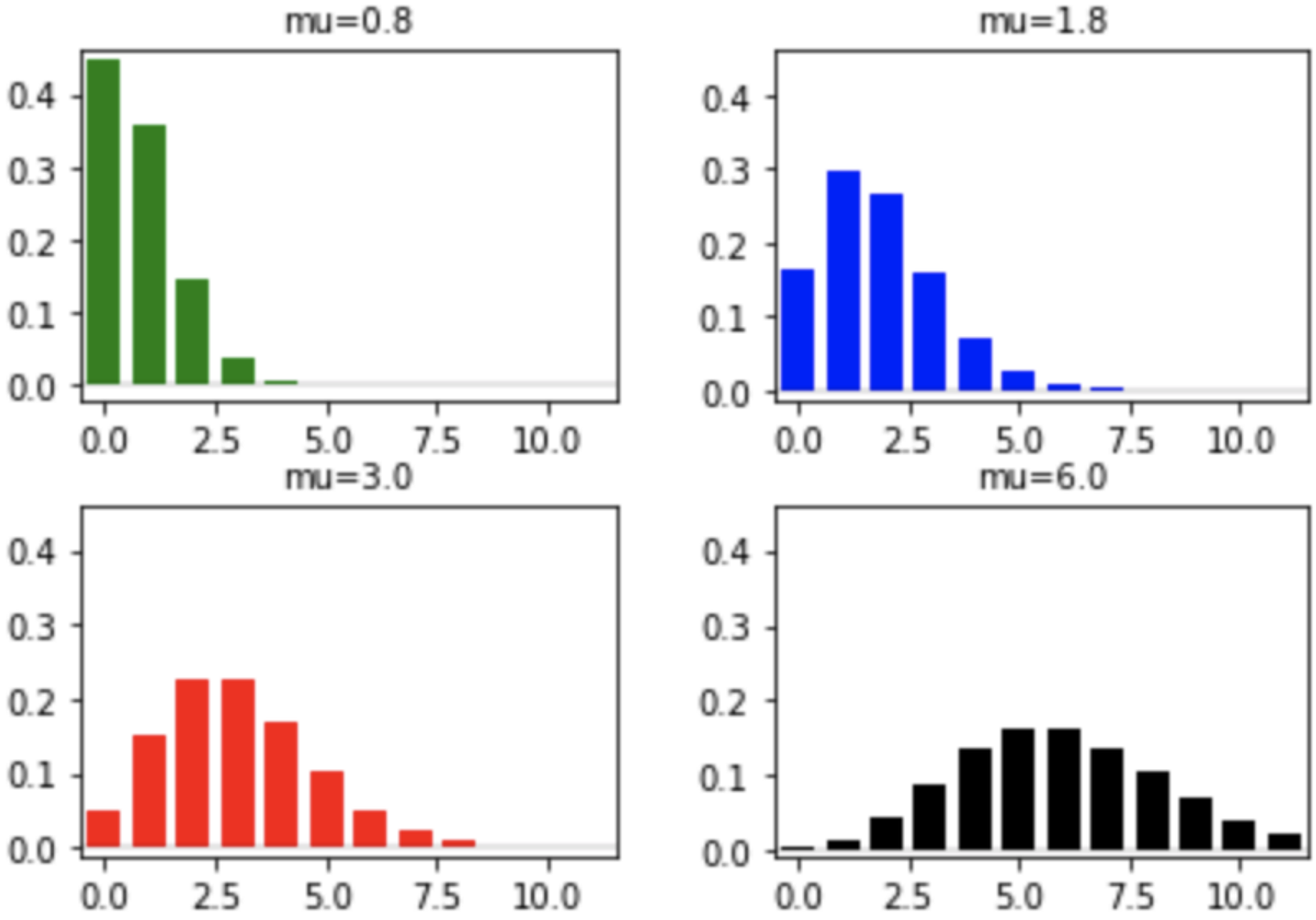

```
# Calculate pmf of 0
poisson.pmf(k=0, mu=2.2)
```

```
0.11080315836233387
```

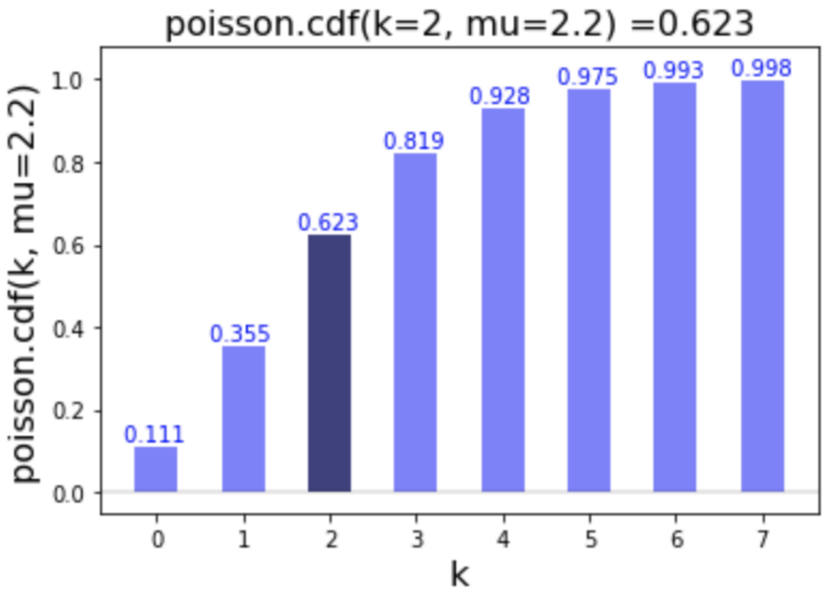
```
# Calculate pmf of 6
poisson.pmf(k=6, mu=2.2)
```

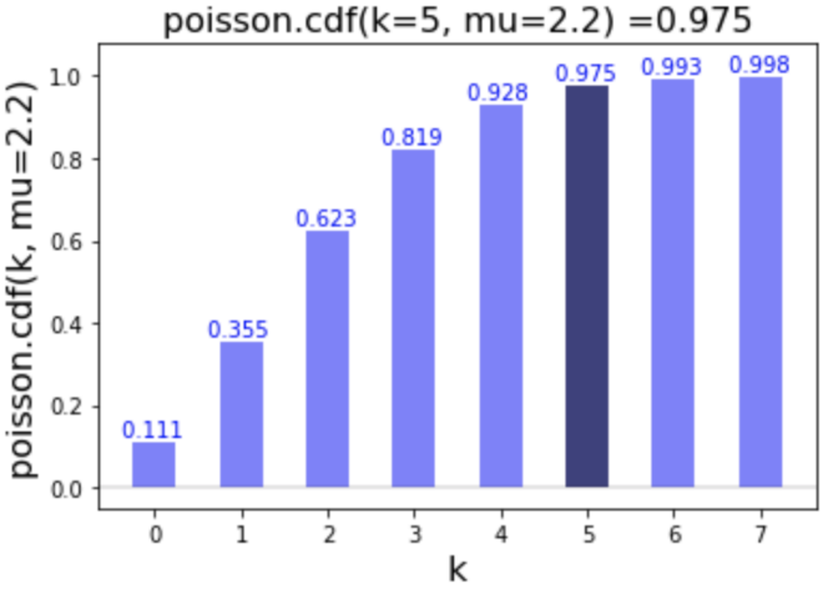
```
0.0174840480280308
```

Different means



Cumulative distribution function (cdf)





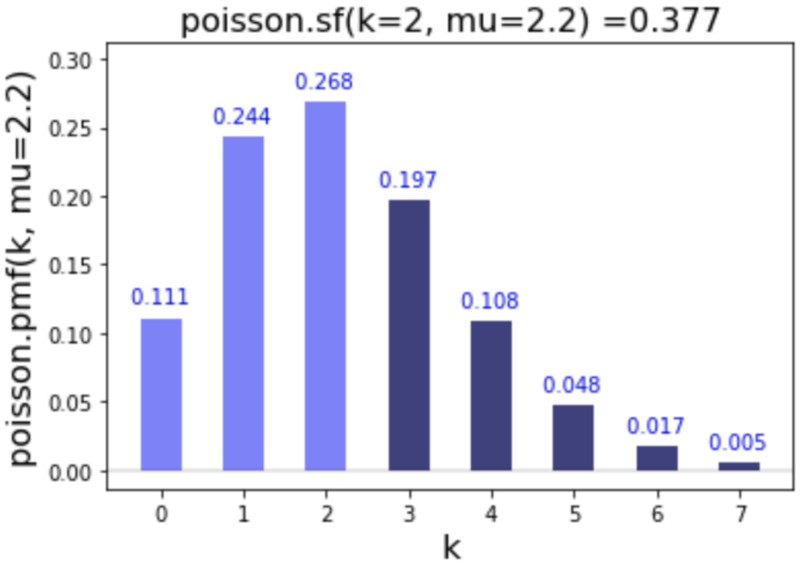

```
# Calculate cdf of 2
poisson.cdf(k=2, mu=2.2)
```

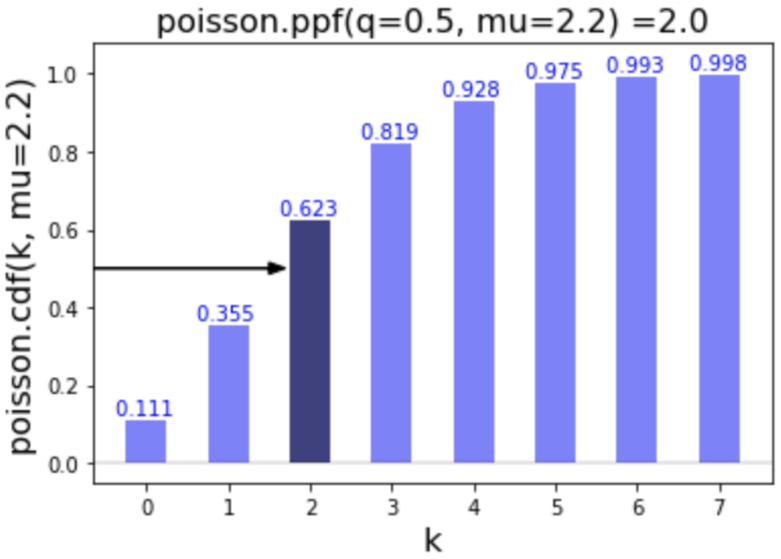
```
0.6227137499963162
```

```
# Calculate cdf of 5
poisson.cdf(k=5, mu=2.2)
```

```
0.9750902496952996
```

Survival function and percent point function (ppf)






```
# Calculate sf of 2
poisson.sf(k=2, mu=2.2)
```

```
0.37728625000036838
```

```
# Calculate ppf of 0.5
poisson.ppf(q=0.5, mu=2.2)
```

```
2.0
```

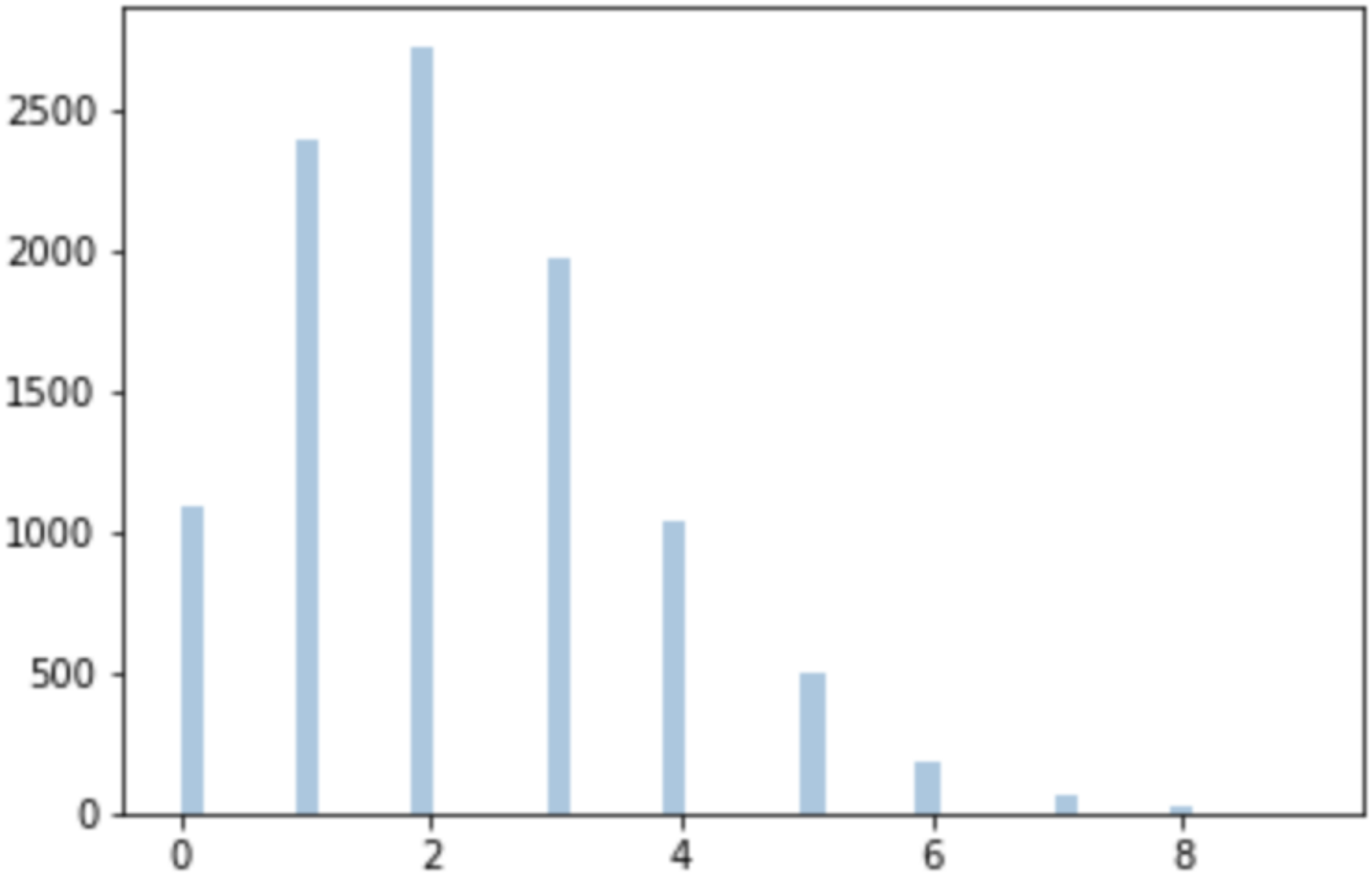
Sample generation (rvs)

```
# Import poisson, matplotlib.pyplot, and seaborn
from scipy.stats import poisson
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Create the sample using poisson.rvs()
sample = poisson.rvs(mu=2.2, size=10000, random_state=13)
```

```
# Plot the sample
sns.distplot(sample, kde=False)
plt.show()
```

Sample generation (Cont.)



Let's practice with Poisson

FOUNDATIONS OF PROBABILITY IN PYTHON

Geometric distributions

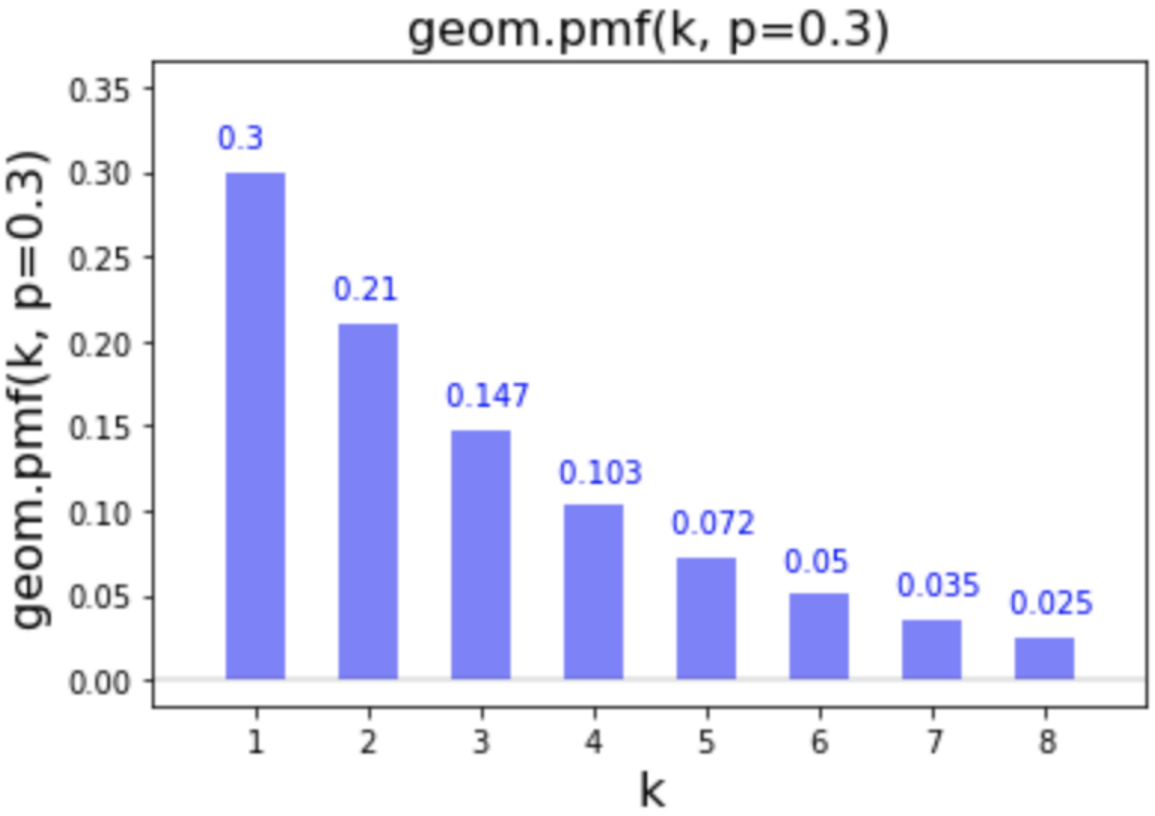
FOUNDATIONS OF PROBABILITY IN PYTHON

Alexander A. Ramírez M.
CEO @ Synergy Vision

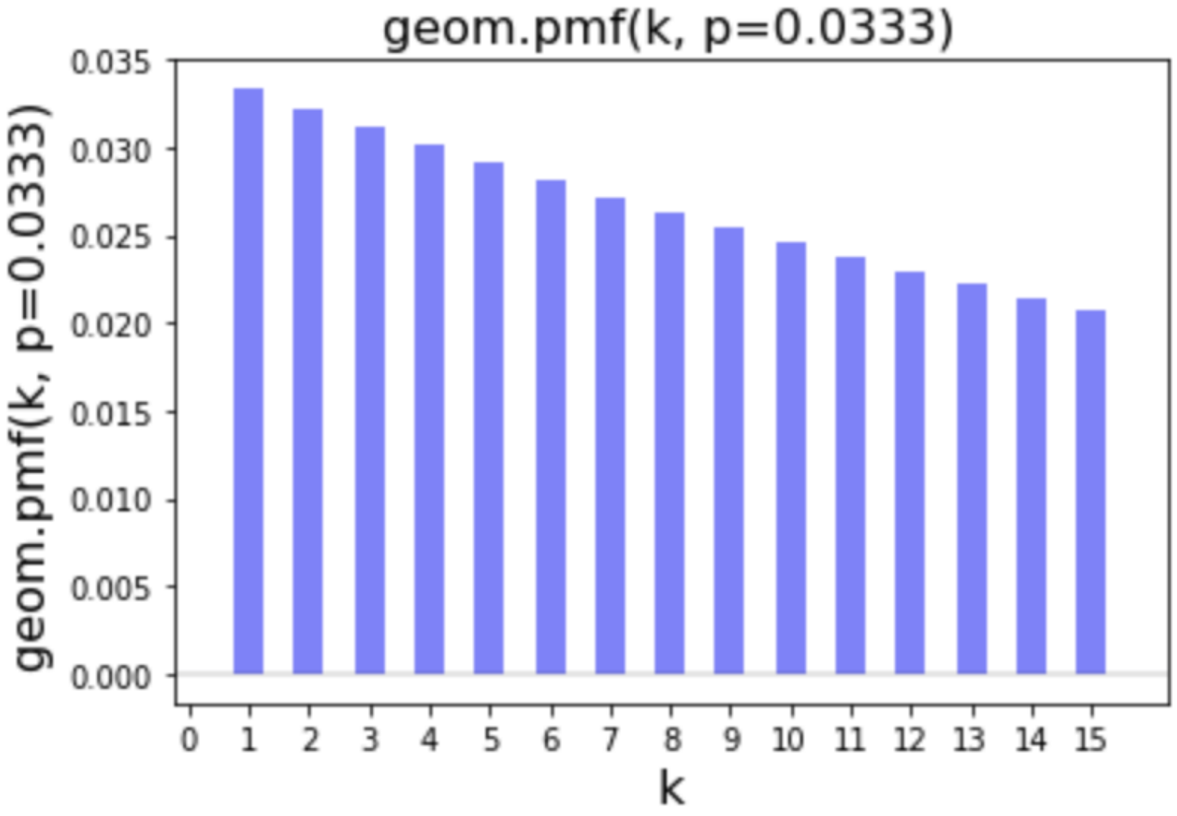
datacamp

Geometric modeling

Geometric parameter

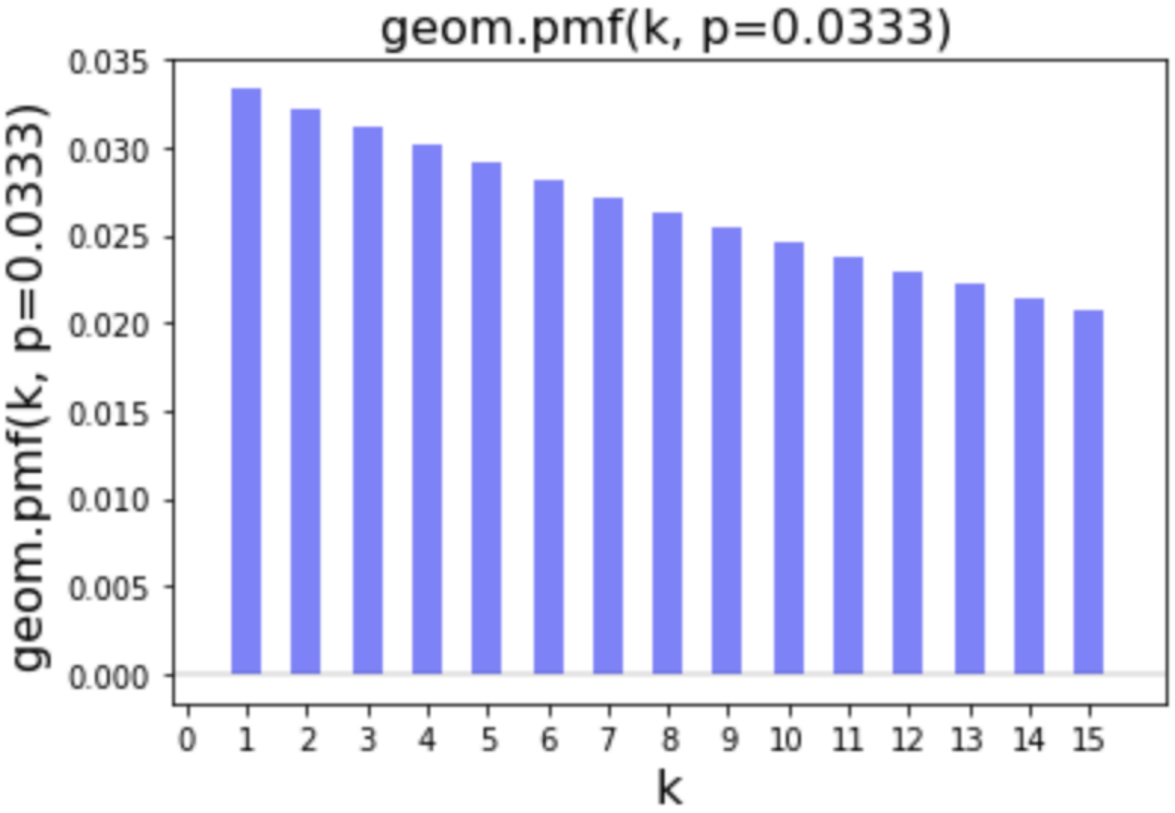


Model for a basketball player with probability 0.3 of scoring.



We can model a grizzly bear that has a 0.033 probability of catching a salmon.

Probability mass function (pmf)



In Python we code this as follows:

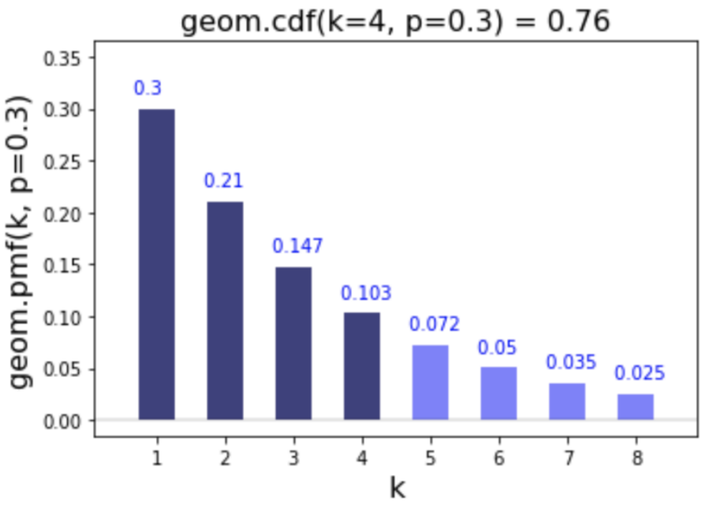
```
# Import geom
from scipy.stats import geom
```

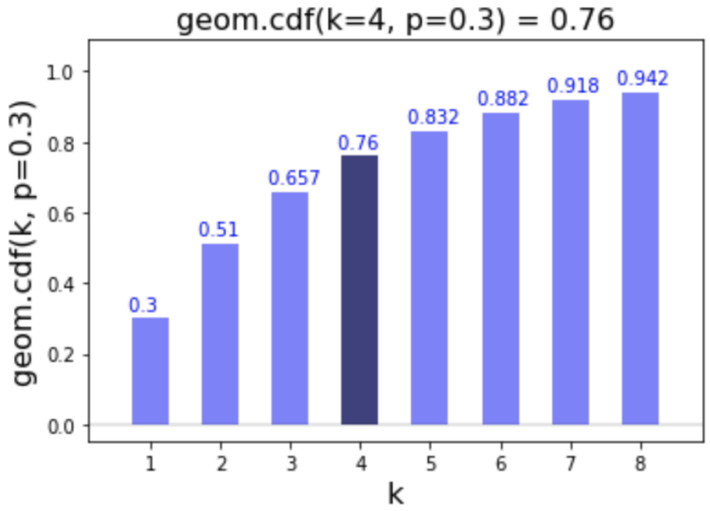
```
# Calculate the probability mass
# with pmf
geom.pmf(k=30, p=0.0333)
```

```
0.024551029087396l2
```

`p` parameter specifies probability of success.

Cumulative distribution function (cdf)

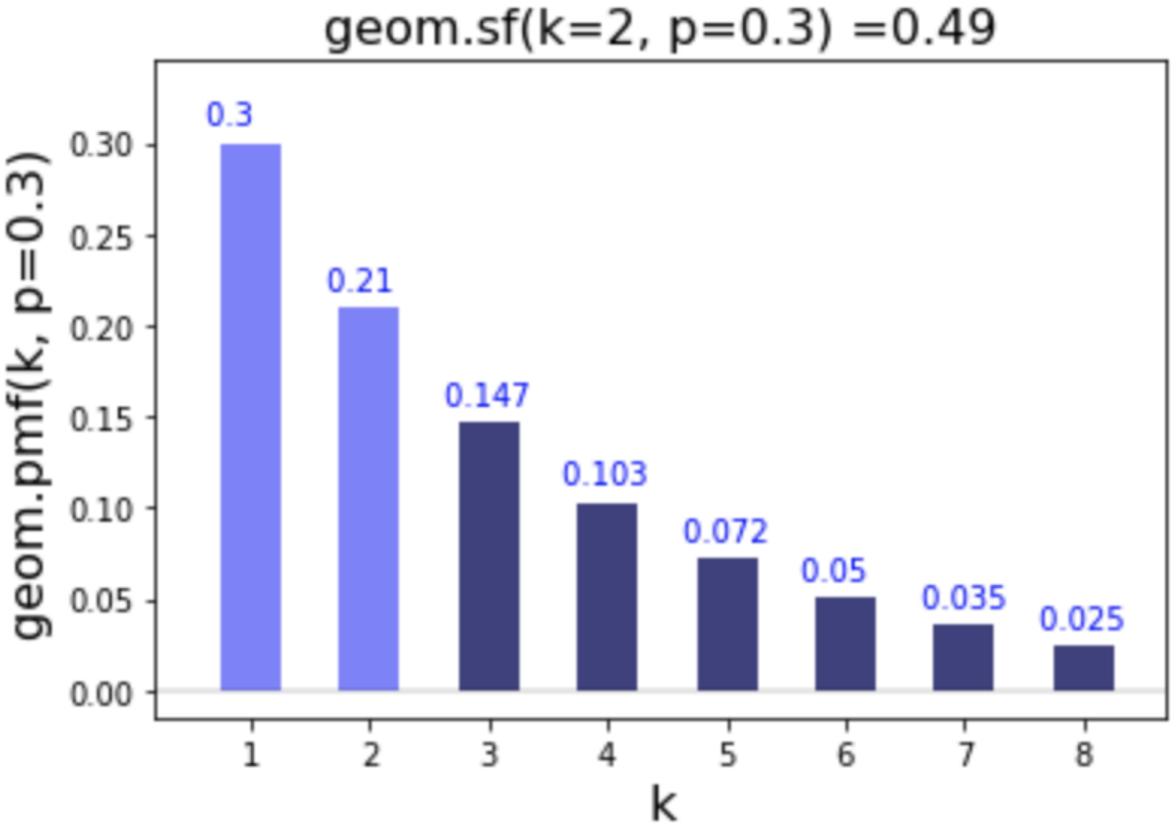





```
# Calculate cdf of 4
geom.cdf(k=4, p=0.3)
```

```
0.7599999999999999
```

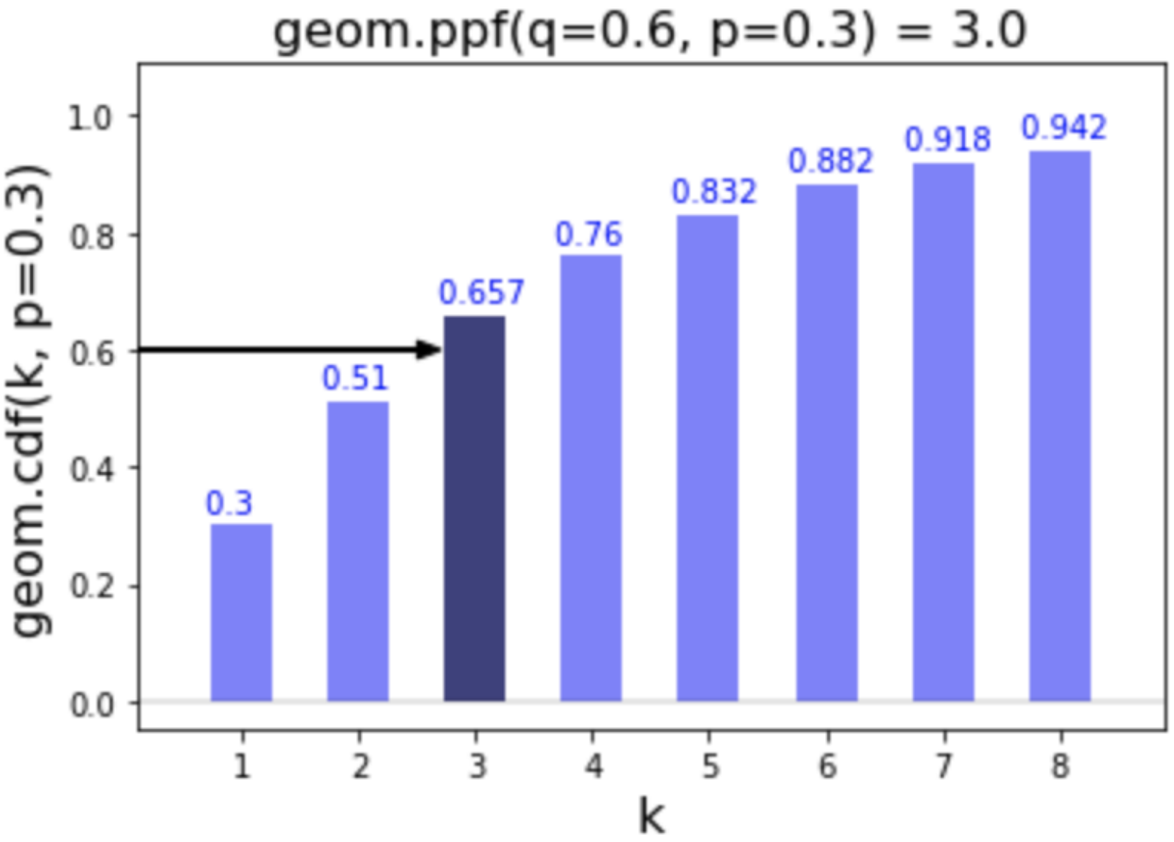
Survival function (sf)




```
# Calculate sf of 2
geom.sf(k=2, p=0.3)
```

```
0.49000000000000005
```

Percent point function (ppf)




```
# Calculate ppf of 0.6
geom.ppf(q=0.6, p=0.3)
```

```
3.0
```

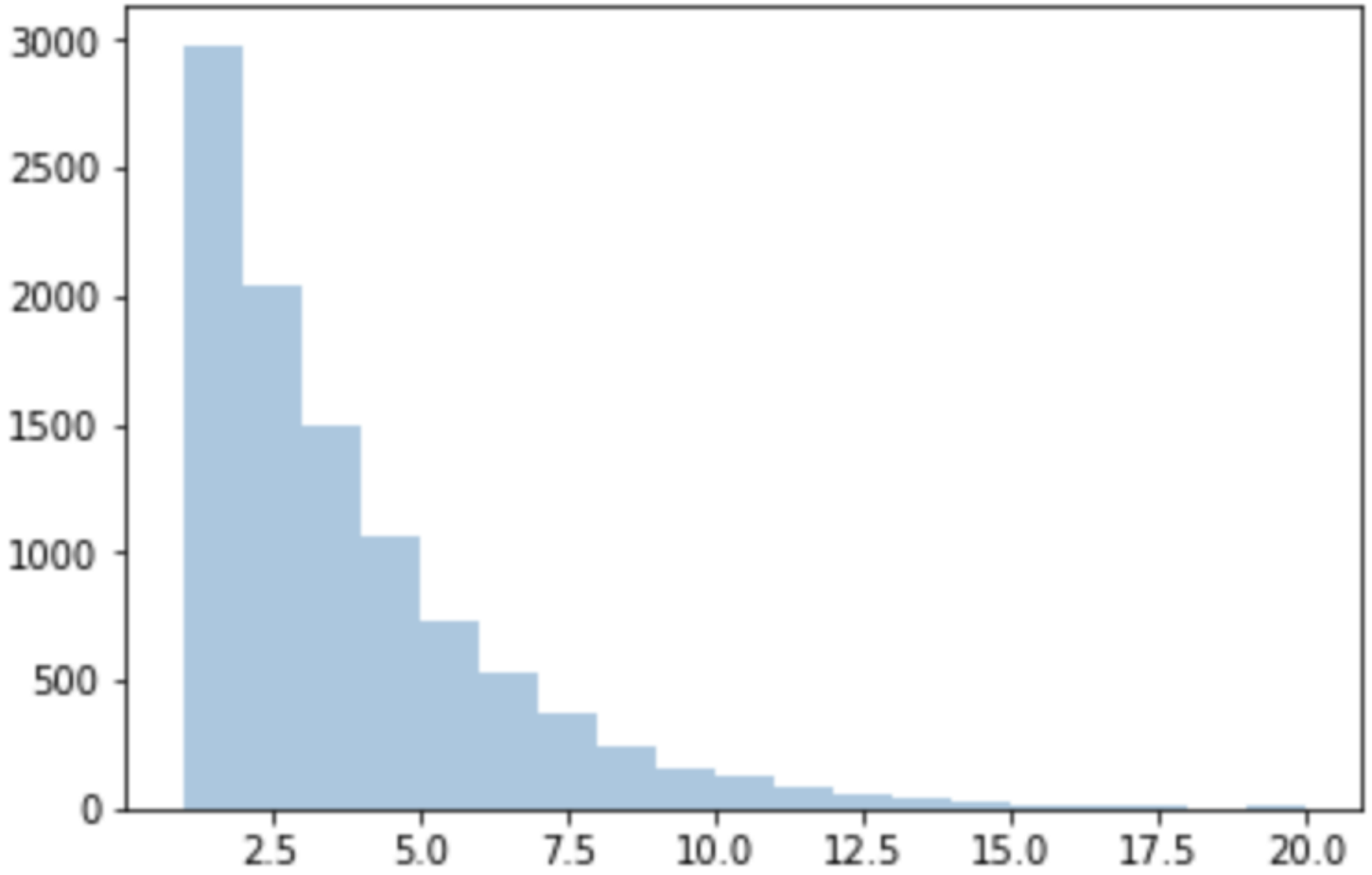
Sample generation (rvs)

```
# Import poisson, matplotlib.pyplot, and seaborn
from scipy.stats import geom
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Create the sample using geom.rvs()
sample = geom.rvs(p=0.3, size=10000, random_state=13)
```

```
# Plot the sample
sns.distplot(sample, bins = np.linspace(0,20,21), kde=False)
plt.show()
```

Sample generation (rvs) (Cont.)



Let's go try until we succeed!

FOUNDATIONS OF PROBABILITY IN PYTHON