Feb 15, 2020 Sympy Tutorial

February 15, 2020

1 Real Example

- Moment Condition with Beta distribution case

[83]:
```
from sympy import Symbol, symbols
m1, m2= symbols('m1 m2')
beta = (m1**2/(m2-m1**2)) * ((1-m1)**2)/(m1**2+m1*(1-m1)) - (m1*(1-m1))/(m1**2␣
↪+ m1*(1-m1))
```

[84]:
```
beta
```

[84]:
$$\frac{m_1^2 (1-m_1)^2}{\left(-m_1^2+m_2\right)\left(m_1^2+m_1(1-m_1)\right)} - \frac{m_1(1-m_1)}{m_1^2+m_1(1-m_1)}$$

[95]:
```
from sympy import simplify
beta = simplify(beta)
beta
```

[95]:
$$\frac{m_1^2 - m_1 m_2 - m_1 + m_2}{m_1^2 - m_2}$$

[97]:
```
from sympy import expand
expand(beta)
```

[97]:
$$\frac{m_1^2}{m_1^2-m_2} - \frac{m_1 m_2}{m_1^2-m_2} - \frac{m_1}{m_1^2-m_2} + \frac{m_2}{m_1^2-m_2}$$

---

2 Tutorial Start

- `1. Rational values`
- `2. SymPy pprint`
- `3. Square root`
- `4. SymPy symbols`
- `5. SymPy canonical form of expression`
- `6. SymPy expanding algebraic expressions`

- 7. SymPy simplify an expression
- 8. SymPy comparig expression
- 9. SymPy evaluating expression
- 10. SymPy solving equations
- 11. SymPy sequence
- 12. SymPy limit
- 13. SymPy matrixes
- 14. SymPy plotting

[2]:
```python
import sympy
from sympy import Rational
```

[4]:
```python
r1 = Rational(1/10)
r2 = Rational(1/10)
r3 = Rational(1/10)
```

[8]:
```python
val = (r1 + r2 + r3) * 3
print(val.evalf()) # The expression is in the symbolic form; we evaluate it␣
↪with evalf() method.
```

```
0.900000000000000
```

[9]:
```python
val2 = (1/10 + 1/10 + 1/10) * 3
print(val2)
```

```
0.9000000000000001
```

[10]:
```python
from sympy import pprint, Symbol, exp, sqrt
from sympy import init_printing
init_printing(use_unicode=True) # For some characters we need to enable unicode␣
↪support.
```

[17]:
```python
x = Symbol('x')

a = sqrt(2)
print(a)
print("------------------------")
pprint(a)
print("------------------------")
print("------------------------")
```

```
sqrt(2)
------------------------
√2
```

```
------------------------
------------------------
```

[18]:
```python
c = (exp(x) ** 2)/2
print(c)
print("------------------------")
pprint(c)
print("------------------------")
print("------------------------")
```

```
exp(2*x)/2
------------------------
 2 x


 2
------------------------
------------------------
```

[19]:
```python
from sympy import sqrt, pprint, Mul
```

[20]:
```python
x = sqrt(2)
y = sqrt(2)
```

[22]:
```python
pprint(Mul(x,  y, evaluate=False))
print("------------------------")
pprint(Mul(x,  y, evaluate=True))
print("------------------------")
print('equals to ')
print(x * y)
```

```
√2 √2
------------------------
2
------------------------
equals to
2
```

[23]:
```python
from sympy import Symbol, symbols
from sympy.abc import x, y
```

[24]:
```python
expr = 2*x + 5*y
print(expr)
```

```
2*x + 5*y
```

[28]:
```python
# We can define symbolic variable with Symbol
a = Symbol('a')
```

```
b = Symbol('b')
expr2 = a*b + a - b
print(expr2)
```

```
a*b + a - b
```

[29]:
```
# Multiple symbols can be defined with symbols() method.
i, j = symbols('i j')
expr3 = 2*i*j + i*j
print(expr3)
```

```
3*i*j
```

[31]:
```
from sympy.abc import a, b
expr = b*a + -4*a + b + a*b + 4*a + (a + b)*3
print(expr)
```

```
2*a*b + 3*a + 4*b
```

[33]:
```
from sympy import expand, pprint
from sympy.abc import x

expr = (x + 1) ** 2

pprint(expr)

print('-----------------------')
print('-----------------------')

expr = expand(expr)
pprint(expr)
```

```
       2
(x + 1)
-----------------------
-----------------------
 2
x  + 2⋅x + 1
```

[35]:
```
from sympy import sin, cos, simplify, pprint
from sympy.abc import x

expr = sin(x) / cos(x)

pprint(expr)

print('-----------------------')
```

```
expr = simplify(expr)
pprint(expr)
```

```
sin(x)

cos(x)
-----------------------
tan(x)
```

[38]:
```
from sympy import pprint, Symbol, sin, cos

x = Symbol('x')

a = cos(x)**2 - sin(x)**2
b = cos(2*x)

print(a.equals(b))
```

```
True
```

[39]:
```
# we cannot use == operator
print(a == b)
```

```
False
```

[42]:
```
from sympy import pi

print(pi.evalf(3))  # The example evaluates a pi value to three places.
print(pi.evalf(30)) # The example evaluates a pi value to thirty places.
```

```
3.14
3.14159265358979323846264338328
```

[44]:
```
from sympy.abc import a, b
from sympy import pprint

f = b*a + -4*a + b + a*b + 4*a + (a + b)*3
pprint(f)
print(f.subs([(a, 3), (b, 2)]))
print(f.subs([(a, 1), (b, 1)]))
```

```
2 a b + 3 a + 4 b
29
9
```

[49]:
```
from sympy import Symbol, solve

x = Symbol('x')
```

```
sol = solve(x**2 - x, x)

print(sol)
print('-------------')

f = x**2 - x
pprint(f)

sol2 = solve(f, x)
print(sol2)
```

```
[0, 1]
-------------
 2
x  - x
[0, 1]
```

[51]:
```
# * solveset(), we find a solution for the given interval.
from sympy.solvers import solveset
from sympy import Symbol, Interval, pprint

x = Symbol('x')

sol = solve(x**2 - 1, x)
print(sol)
print('---')

sol2 = solveset(x**2 - 1, x, Interval(0, 100))
print(sol2)
```

```
[-1, 1]
---
{1}
```

[ ]:
```
---
## SymPy sequence
* A sequence can be finite or infinite.
* The number of elements is called the length of the sequence.
*
```

[52]:
```
from sympy import summation, sequence, pprint
from sympy.abc import x
```

[53]:
```
s = sequence(x, (x, 1, 10))
print(s)
```

```
SeqFormula(x, (x, 1, 10))
```

[54]:
```
pprint(s)
```

```
[1, 2, 3, 4, …]
```

[55]:
```
print(list(s))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

[56]:
```
print(s.length)
```

```
10
```

[57]:
```
print(summation(s.formula, (x, s.start, s.stop)))
```

```
55
```

[58]:
```
print(sum(list(s)))
```

```
55
```

[ ]:
```
---
# SymPy limit
* A limit is the value that a function (or sequence) "approaches" as the input␣
↪(or index) "approaches" some value.
---
```

[62]:
```
from sympy import sin, limit, oo
from sympy.abc import x

l1 = limit(1/x, x, oo)
print(l1)
```

```
0
```

[63]:
```
l2 = limit(1/x, x, 0)
print(l2)
```

```
oo
```

[69]:
```
from sympy import Matrix, pprint

M = Matrix([[1, 2], [3, 4], [0, 3]])
print('M matrix shape: ', M.shape)
```

```
M matrix shape:  (3, 2)
```

[70]:
```
print(M)
```

```
Matrix([[1, 2], [3, 4], [0, 3]])
```

[71]:
```
pprint(M)
```

```
1  2

3  4

0  3
```

[72]:
```
N = Matrix([2, 2])
print('N matrix shape: ', N.shape)
pprint(N)
```

```
N matrix shape:  (2, 1)
 2

 2
```

[73]:
```
print("M * N")
pprint(M*N)
```

```
M * N
 6

 14

 6
```

[75]:
```
import sympy
from sympy.abc import x
from sympy.plotting import plot

plot(1/x)
```

```
<Figure size 640x480 with 1 Axes>
```

[75]: `<sympy.plotting.plot.Plot at 0x2b1d1602ee90>`