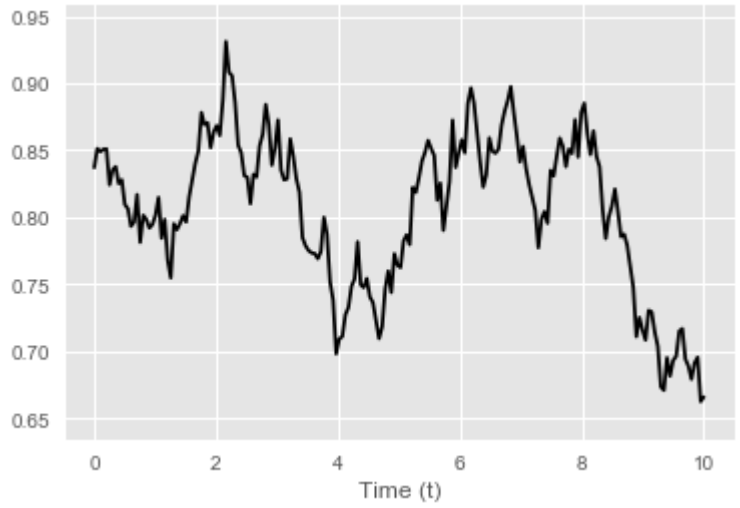
Handouts 25 & 26 - Gaussian processes and Brownian motion

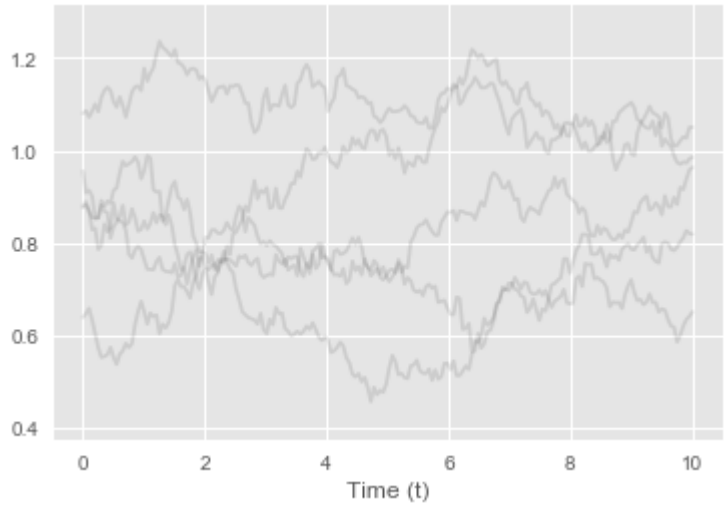
In [1]:
```
from symbulate import *
%matplotlib inline
```

Exercise 25.1

In [8]:
```
P = GaussianProcess(mean_fn = lambda t: 0.9, cov_fn = lambda t, s: 0.04 * exp(
-0.1 * abs(t - s)))
X = RandomProcess(P, TimeIndex(fs=inf))
X.sim(1).plot(alpha=1)
```

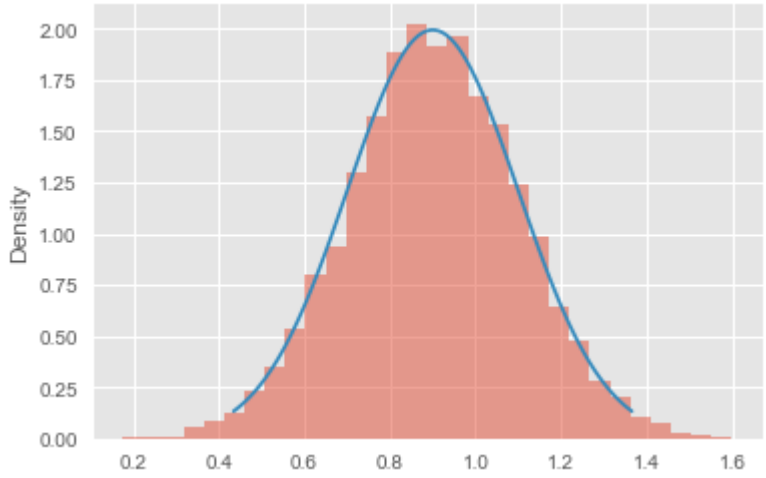


In [9]:
```
X.sim(5).plot()
```

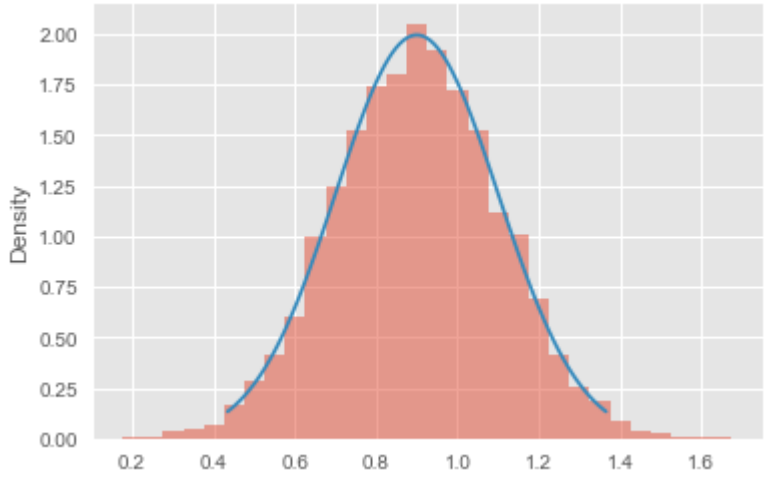


In [10]:
```
x = X[3].sim(10000)
x.plot()
Normal(mean=0.9, sd=0.2).plot()
x.count_gt(.3) / 10000
```

Out[10]: 0.9991

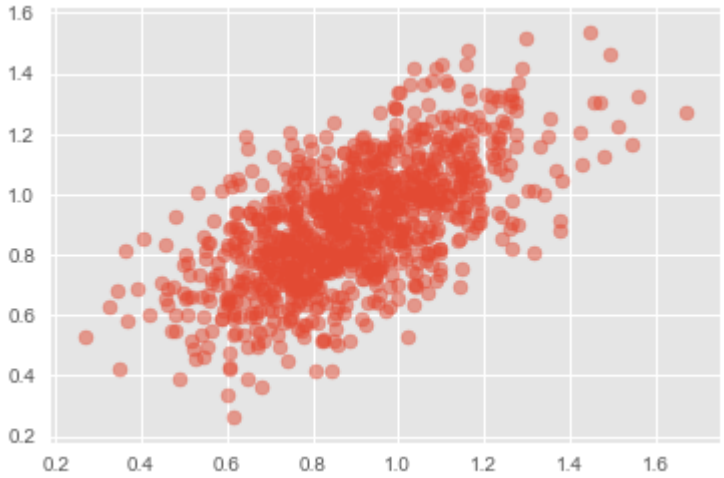


In [12]:
```
X[5.3].sim(10000).plot()
Normal(mean=0.9, sd=0.2).plot()
```



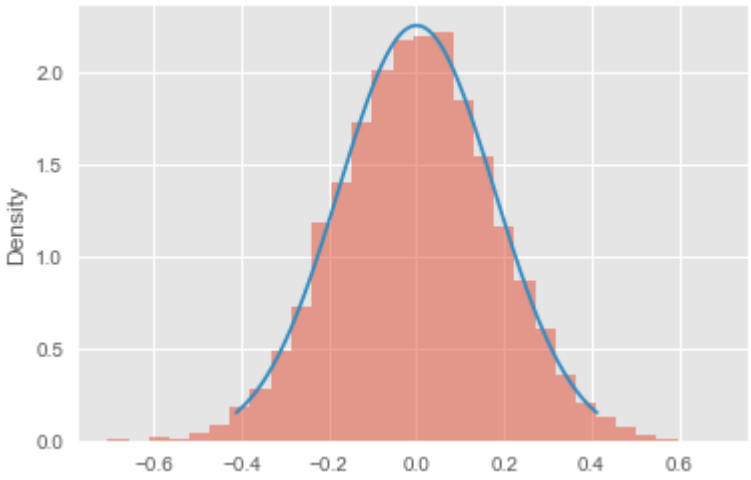
In [14]:
```
xy = (X[3] & X[8]).sim(1000)
xy.plot()
xy.cov(), xy.corr()
```

Out[14]: (0.025907993264414179, 0.6231730043275453)



In [16]:
```
Y = X[3] - X[8]
y = Y.sim(10000)
y.plot()
Normal(0, var=0.04 + 0.04 - 2 * 0.04 * exp(-0.1 * abs(3 - 8))).plot()
y.count_gt(0.5) / 10000
```

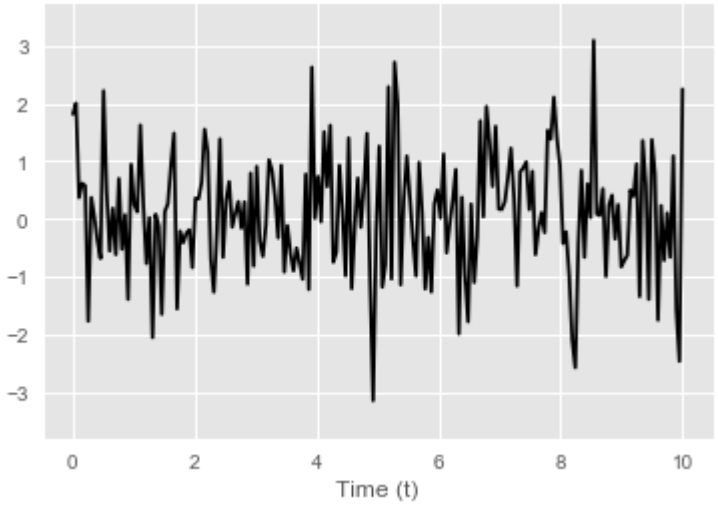
Out[16]: 0.0022



Gaussian white noise

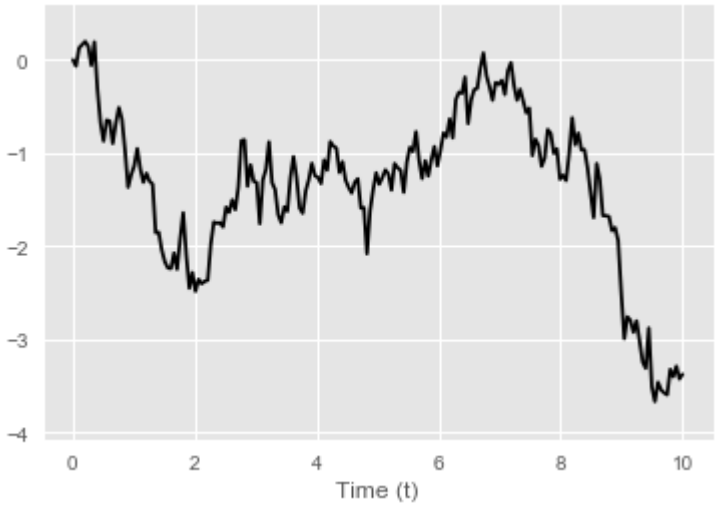
In [17]:
```
def delta(x):
    if x == 0:
        return 1
    else:
        return 0

P = GaussianProcess(mean_fn = lambda t: 0, cov_fn = lambda t, s: delta(abs(t-s
)))
X = RandomProcess(P, TimeIndex(fs=inf))
X.sim(1).plot(alpha=1)
```

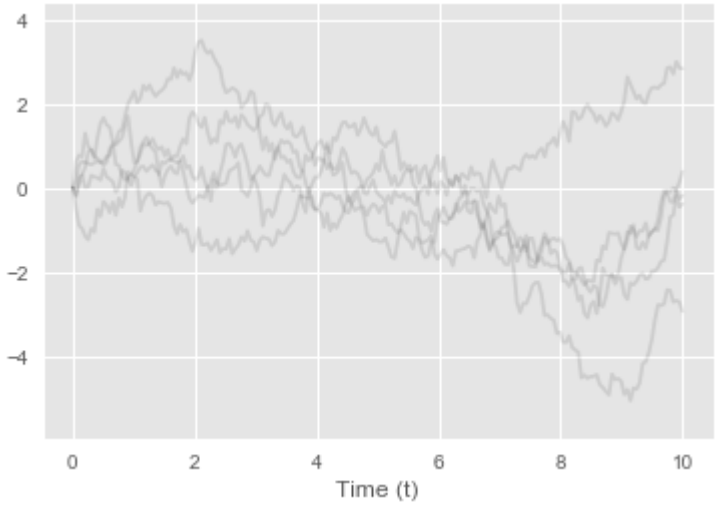


Exer 26.1: Brownian motion

In [18]:
```
P = GaussianProcess(mean_fn = lambda t: 0, cov_fn = lambda t, s: min(s, t))
W = RandomProcess(P, TimeIndex(fs=inf))
W.sim(1).plot(alpha=1)
```

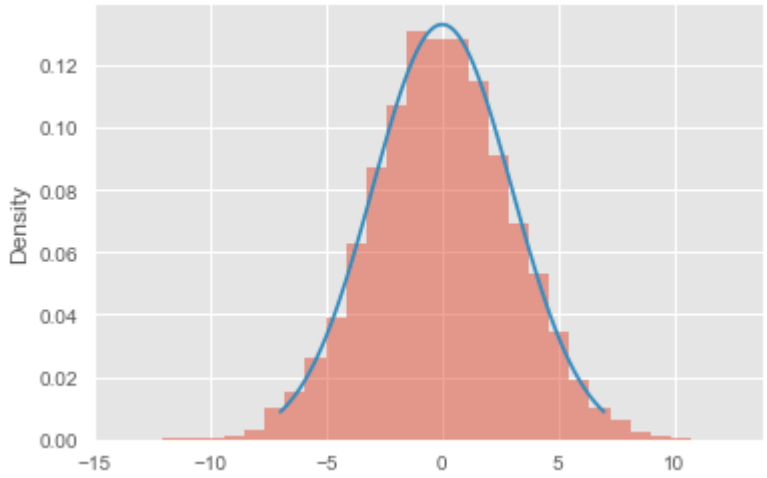


In [19]:
```
W.sim(5).plot()
```



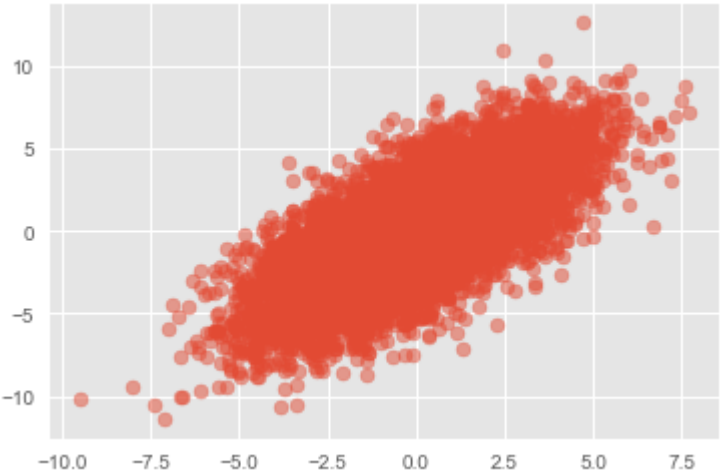
In [22]:
```
w = W[9].sim(10000)
w.plot()
Normal(0, var=9).plot()
(w.count_gt(6) + w.count_lt(-6)) / 10000
```

Out[22]: 0.0481



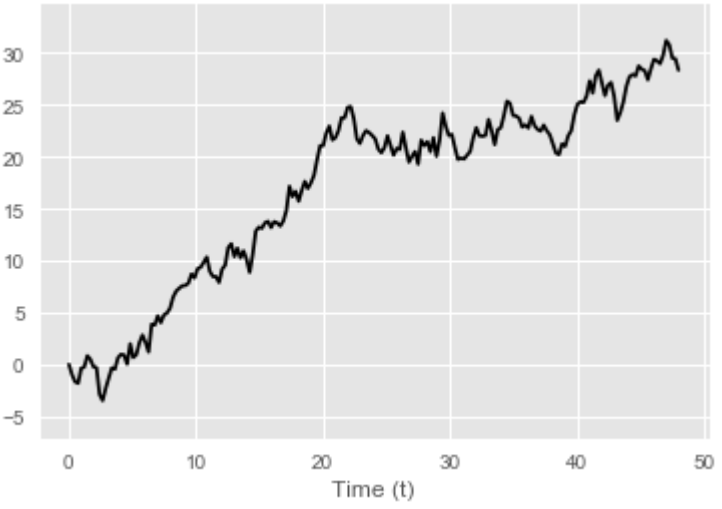
In [21]:
```
xy = (W[4.5] & W[9]).sim(10000)
xy.plot()
xy.cov(), xy.corr()
```

Out[21]: (4.5667522527749265, 0.71180999203872497)

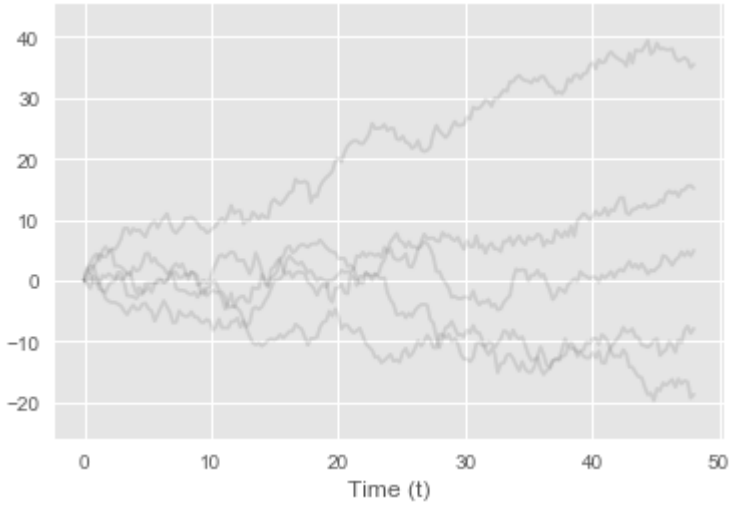


Exer 26.2 - Brownian motion with drift

In [29]:
```
P = GaussianProcess(mean_fn = lambda t: 0.1 * t, cov_fn = lambda t, s: 2**2 *
min(s, t))
B = RandomProcess(P, TimeIndex(fs=inf))
B.sim(1).plot(alpha=1, tmax=48)
```

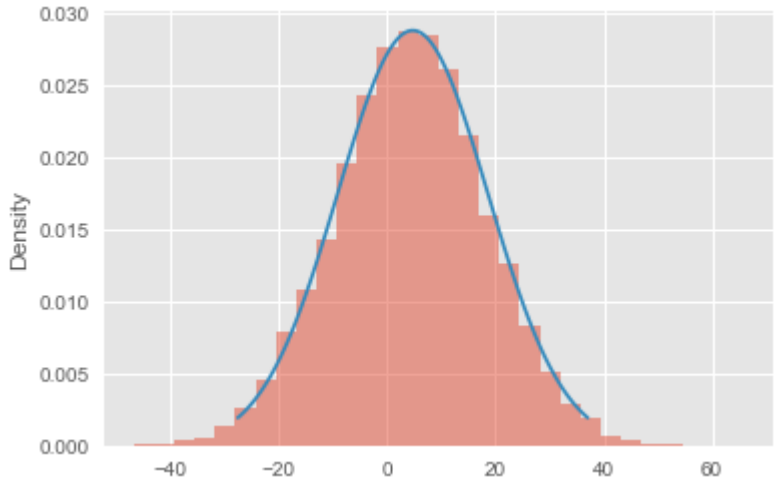


In [30]:
```
B.sim(5).plot(tmax=48)
```



In [31]:
```
b = B[48].sim(10000)
b.plot()
Normal(mean = 0 + 0.1 * 48, var = 2**2 * 48).plot()
b.count_gt(0) / 10000
```

Out[31]: 0.6282

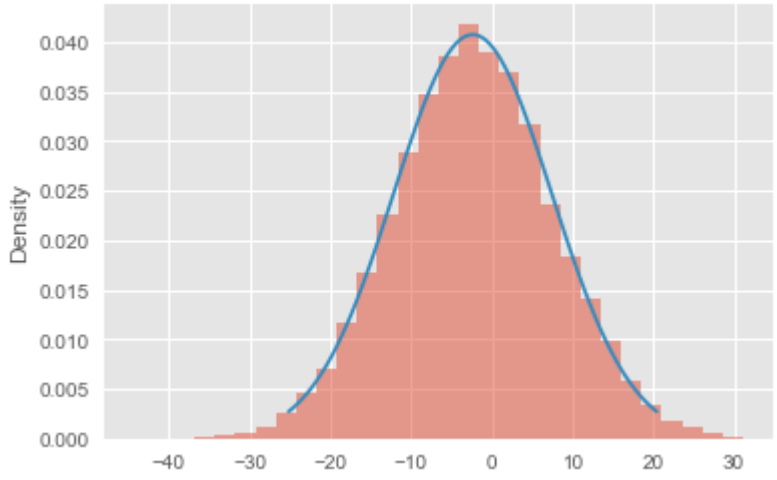


In [32]:
```
xy = (B[24] & B[48]).sim(10000)
xy.plot()
xy.cov(), xy.corr()
```

Out[32]: (94.997867270394607, 0.70306733866342441)

In [35]:
```
Y = B[24] - B[48]
y = Y.sim(10000)
y.plot()
Normal(mean = 0.1 * (24 - 48), var = 2**2 * 24 + 2**2 * 48 - 2 * 2**2 * 24).pl
ot()
y.count_gt(0) / 10000
```

Out[35]: 0.4049



In [ ]:
```
```