GLM_Probability_functions

March 19, 2019

19 March 2019

1 Probability functions for GLM regression

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
```

```
In [2]: from scipy import stats
        from statsmodels.distributions.empirical_distribution import ECDF
```

```
In [3]: # Set random seed
        #np.random.seed(280857)
```

1.1 Normal (Gaussian) distribution

```
In [4]: # Parameters
        mu = 0
        sd = 1

        # Statistical properties
        print("Mean =", np.round(mu), "; std =", np.round(sd))

Mean = 0 ; std = 1
```

```
In [5]: # Theoretical CDF
        xlim0 = stats.norm.ppf(0.01, loc=mu, scale=sd)
        xlim1 = stats.norm.ppf(0.99, loc=mu, scale=sd)
        x0 = np.linspace(xlim0, xlim1, 100)
        y0 = stats.norm.cdf(x0, loc=mu, scale=sd)
```

```
In [6]: # Sample size
        N = 100
```

```
In [7]: # Random sample (numpy)
        y = np.random.normal(loc=mu, scale=sd, size=N)
        print("Mean=", np.mean(y).round(2), "; sd=", np.std(y).round(2))
```

```
# Empirical CDF
ecdf = ECDF(y)
x1 = ecdf.x
y1 = ecdf.y
```

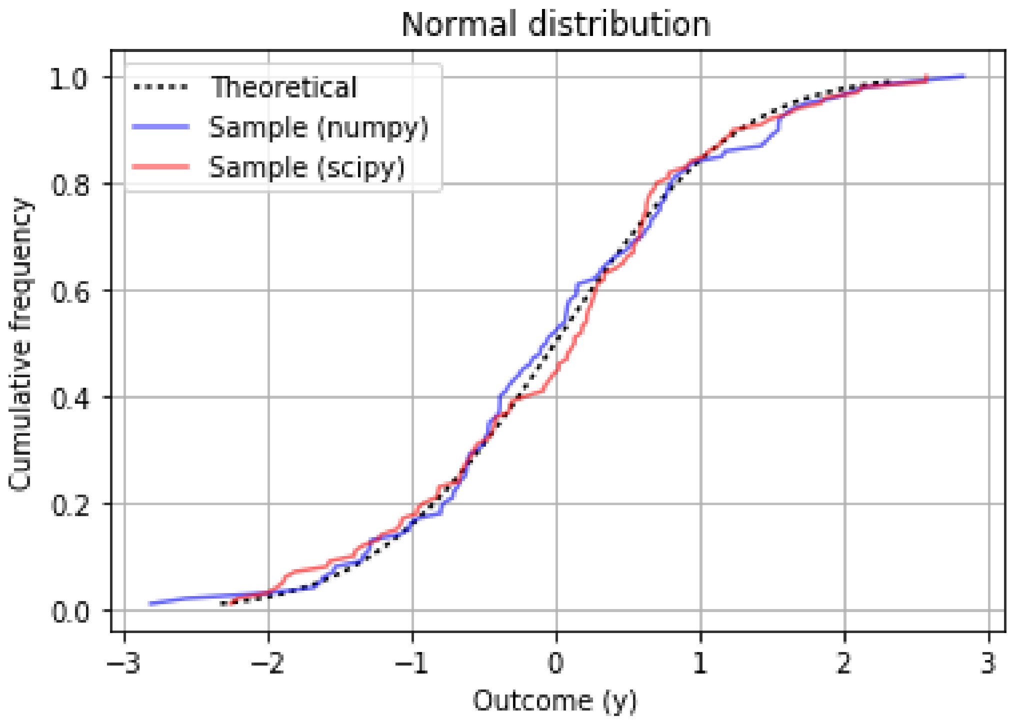
```
Mean= 0.02 ; sd= 1.07
```

```
In [8]: # Random sample (scipy)
        y = np.random.normal(loc=mu, scale=sd, size=N)
        print("Mean=", np.mean(y).round(2), "; sd=", np.std(y).round(2))

        # Empirical CDF
        ecdf = ECDF(y)
        x2 = ecdf.x
        y2 = ecdf.y
```

```
Mean= 0.03 ; sd= 1.05
```

```
In [9]: # Plot
        fig, ax = plt.subplots(1, 1)
        _ = ax.plot(x0, y0, marker="", ls=":", c="k", alpha=1.0, label="Theoretical
        _ = ax.plot(x1, y1, marker="", ls="-", c="b", alpha=0.6, label="Sample (num
        _ = ax.plot(x2, y2, marker="", ls="-", c="r", alpha=0.6, label="Sample (sci
        ax.set_xlabel("Outcome (y)")
        ax.set_ylabel("Cumulative frequency")
        ax.set_title("Normal distribution")
        ax.legend()
        ax.grid(b=True)
        plt.show()
```



1.2 Exponential distribution

```
In [10]: # Parameter
         llambda = 1

         # Statistical properties
         mu = 1 / llambda
         sd = 1 / llambda
         print("mean=", np.round(mu, 2), "; std=", np.round(sd, 2))
```

```
mean= 1.0 ; std= 1.0
```

```
In [11]: # Theoretical CDF
         xlim0 = stats.expon.ppf(0.01, loc=0, scale=sd)
         xlim1 = stats.expon.ppf(0.99, loc=0, scale=sd)
         x0 = np.linspace(xlim0, xlim1, 100)
         y0 = stats.expon.cdf(x0, loc=0, scale=sd)
```

```
In [12]: # Sample size
         N = 100
```

```
In [13]: # Random sample (numpy)
         y = np.random.exponential(scale=sd, size=N)
```

```
print("mean=", np.mean(y).round(2), "; std=", np.std(y).round(2))

# Empirical CDF
ecdf = ECDF(y)
x1 = ecdf.x
y1 = ecdf.y
```

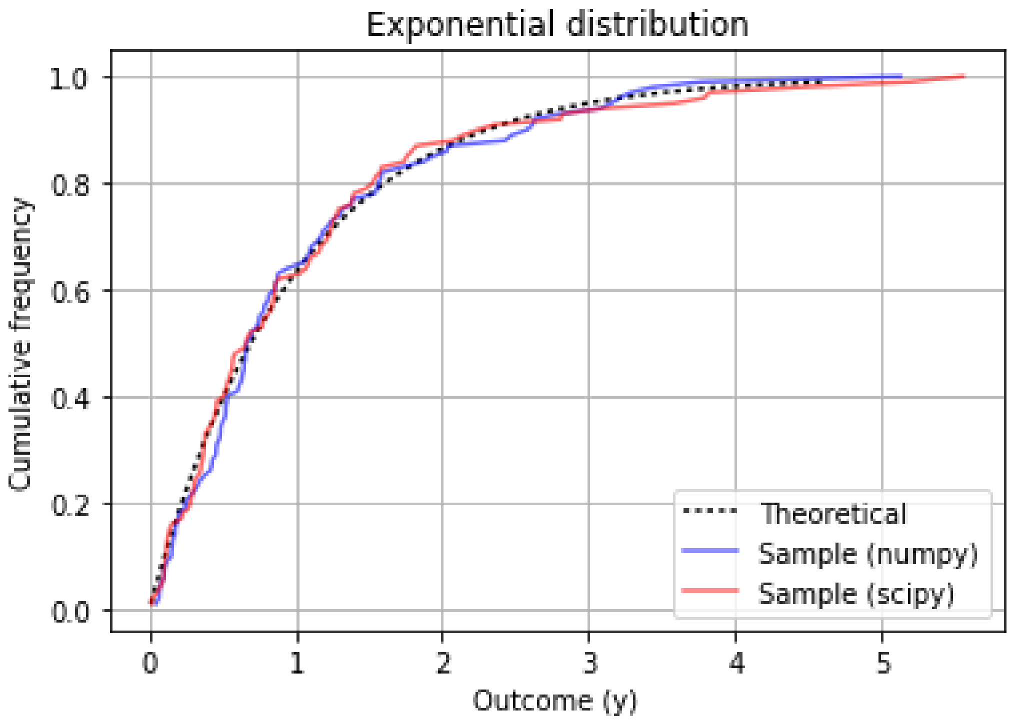
```
mean= 1.03 ; std= 0.97
```

```
In [14]: # Random sample (scipy)
         y = stats.expon.rvs(loc=0, scale=sd, size=N)
         print("mean=", np.mean(y).round(2), "; std=", np.std(y).round(2))

         # Empirical CDF
         ecdf = ECDF(y)
         x2 = ecdf.x
         y2 = ecdf.y
```

```
mean= 1.04 ; std= 1.09
```

```
In [15]: # Plot
         fig, ax = plt.subplots(1, 1)
         _ = ax.plot(x0, y0, marker="", ls=":", c="k", alpha=1.0, label="Theoretica
         _ = ax.plot(x1, y1, marker="", ls="-", c="b", alpha=0.6, label="Sample (nu
         _ = ax.plot(x2, y2, marker="", ls="-", c="r", alpha=0.6, label="Sample (sc
         ax.set_xlabel("Outcome (y)")
         ax.set_ylabel("Cumulative frequency")
         ax.set_title("Exponential distribution")
         ax.legend()
         ax.grid(b=True)
         plt.show()
```



1.3 Gamma distribution

```
In [16]: # Parameters
         k = 2
         theta = 2

         # Statistical properties
         mu = k * theta
         sd = np.sqrt(k) * theta
         print("mean=", np.round(mu, 2), "; std=", np.round(sd, 2))
```

```
mean= 4 ; std= 2.83
```

```
In [17]: # Theoretical CDF
         xlim0 = stats.gamma.ppf(0.01, a=k, loc=0, scale=theta)
         xlim1 = stats.gamma.ppf(0.99, a=k, loc=0, scale=theta)
         x0 = np.linspace(xlim0, xlim1, 100)
         y0 = stats.gamma.cdf(x0, a=k, loc=0, scale=theta)
```

```
In [18]: # Sample size
         N = 100
```

```
In [19]: y = np.random.gamma(shape=k, scale=theta, size=N)
         print("mean=", np.mean(y).round(2), "; std=", np.std(y).round(2))
```

```
# Empirical CDF
ecdf = ECDF(y)
x1 = ecdf.x
y1 = ecdf.y
```

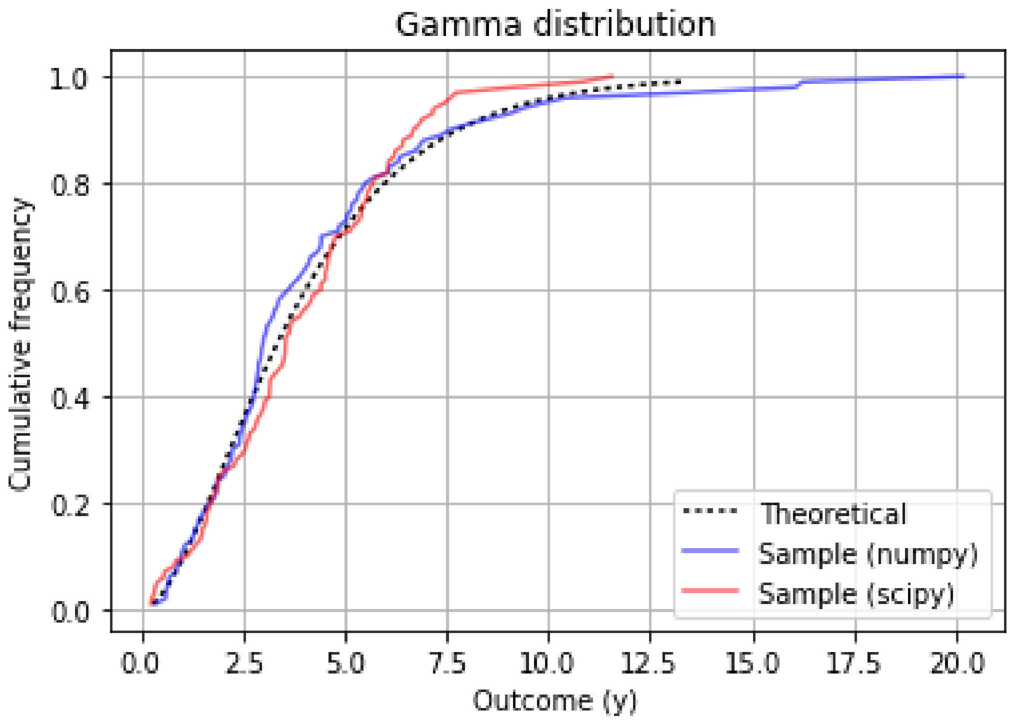
```
mean= 4.09 ; std= 3.39
```

```
In [20]: y = stats.gamma.rvs(a=k, loc=0, scale=theta, size=N)
         print("mean=", np.mean(y).round(2), "; std=", np.std(y).round(2))

         # Empirical CDF
         ecdf = ECDF(y)
         x2 = ecdf.x
         y2 = ecdf.y
```

```
mean= 3.89 ; std= 2.27
```

```
In [21]: # Plot
         fig, ax = plt.subplots(1, 1)
         _ = ax.plot(x0, y0, marker="", ls=":", c="k", alpha=1.0, label="Theoretica
         _ = ax.plot(x1, y1, marker="", ls="-", c="b", alpha=0.6, label="Sample (nu
         _ = ax.plot(x2, y2, marker="", ls="-", c="r", alpha=0.6, label="Sample (sc
         ax.set_xlabel("Outcome (y)")
         ax.set_ylabel("Cumulative frequency")
         ax.set_title("Gamma distribution")
         ax.legend()
         ax.grid(b=True)
         plt.show()
```



1.4 Inverse Gaussian (Wald) distribution

References: **numpy** **scipy**

Treatment of scale appears inconsistent; use lambda = 1.

```
In [22]: # Parameters
         mu = 0.5
         llambda = 1

         # Statistical properties
         sd = mu / np.sqrt(llambda)
         print("mean=", np.round(mu, 2), "; std=", np.round(sd, 2))

mean= 0.5 ; std= 0.5
```

```
In [23]: # Theoretical CDF
         xlim0 = stats.invgauss.ppf(0.01, mu, scale=llambda)
         xlim1 = stats.invgauss.ppf(0.99, mu, scale=llambda)
         x0 = np.linspace(xlim0, xlim1, 100)
         y0 = stats.invgauss.cdf(x0, mu, scale=llambda)
```

```
In [24]: # Sample size
         N = 100
```

```
In [25]: # Random sample (numpy)
         y = np.random.wald(mean=mu, scale=llambda, size=N)
         print("mean=", np.mean(y).round(2), "; std=", np.std(y).round(2))

         # Empirical CDF
         ecdf = ECDF(y)
         x1 = ecdf.x
         y1 = ecdf.y
```

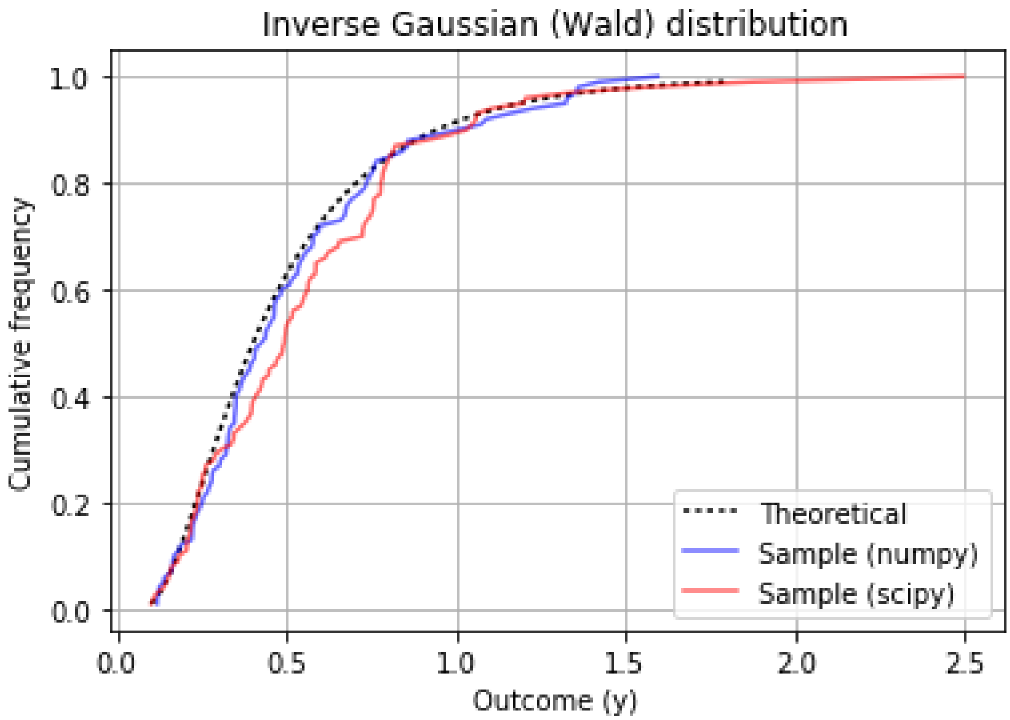
```
mean= 0.52 ; std= 0.33
```

```
In [26]: # Random sample (scipy)
         y = stats.invgauss.rvs(mu, loc=0, scale=llambda, size=N)
         print("mean=", np.mean(y).round(2), "; std=", np.std(y).round(2))

         # Empirical CDF
         ecdf = ECDF(y)
         x2 = ecdf.x
         y2 = ecdf.y
```

```
mean= 0.56 ; std= 0.38
```

```
In [27]: # Plot
         fig, ax = plt.subplots(1, 1)
         _ = ax.plot(x0, y0, marker="", ls=":", c="k", alpha=1.0, label="Theoretica
         _ = ax.plot(x1, y1, marker="", ls="-", c="b", alpha=0.6, label="Sample (nu
         _ = ax.plot(x2, y2, marker="", ls="-", c="r", alpha=0.6, label="Sample (sc
         ax.set_xlabel("Outcome (y)")
         ax.set_ylabel("Cumulative frequency")
         ax.set_title("Inverse Gaussian (Wald) distribution")
         ax.legend()
         ax.grid(b=True)
         plt.show()
```



1.5 Poisson distribution

```
In [28]: # Parameters
         llambda = 4

         # Statistical properties
         mu = llambda
         sd = np.sqrt(llambda)
         print("mean=", np.round(mu, 2), ";", "std=", np.round(sd, 2))
```

```
mean= 4 ; std= 2.0
```

```
In [29]: # Theoretical CDF
         xlim0 = stats.poisson.ppf(0.01, mu, loc=0)
         xlim1 = stats.poisson.ppf(0.99, mu, loc=0)
         x0 = np.linspace(xlim0, xlim1, 100)
         y0 = stats.poisson.cdf(x0, mu, loc=0)
```

```
In [30]: # Sample size
         N = 100
```

```
In [31]: # Random sample (numpy)
         y = np.random.poisson(lam=llambda, size=N)
```

```
print("mean=", np.mean(y).round(2), ";", "std=", np.std(y).round(2))

# Empirical CDF
ecdf = ECDF(y)
x1 = ecdf.x
y1 = ecdf.y
```

```
mean= 4.15 ; std= 2.09
```

In [32]:

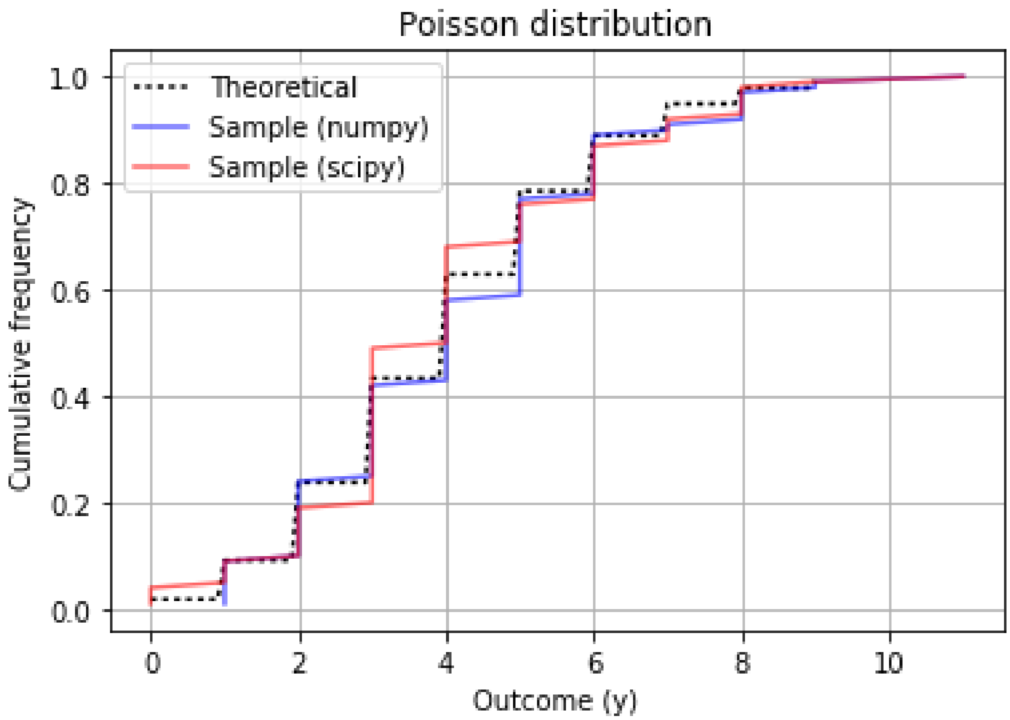
```
# Random sample (scipy)
y = stats.poisson.rvs(mu, loc=0, size=N)
print("mean=", np.mean(y).round(2), ";", "std=", np.std(y).round(2))

# Empirical CDF
ecdf = ECDF(y)
x2 = ecdf.x
y2 = ecdf.y
```

```
mean= 4.0 ; std= 2.11
```

In [33]:

```
# Plot
fig, ax = plt.subplots(1, 1)
_ = ax.plot(x0, y0, marker="", ls=":", c="k", alpha=1.0, label="Theoretica
_ = ax.plot(x1, y1, marker="", ls="-", c="b", alpha=0.6, label="Sample (nu
_ = ax.plot(x2, y2, marker="", ls="-", c="r", alpha=0.6, label="Sample (sc
ax.set_xlabel("Outcome (y)")
ax.set_ylabel("Cumulative frequency")
ax.set_title("Poisson distribution")
ax.legend()
ax.grid(b=True)
plt.show()
```



1.6 Bernoulli distribution

```
In [34]: # Parameters
         p = 0.5
         n = 1       # defines Bernoulli as special case of binomial dist.

         # Statistical properties
         mu = p
         sd = np.sqrt(p * (1 - p))
         print("mean=", np.round(mu, 2), ";", "std=", np.round(sd, 2))
```

```
mean= 0.5 ; std= 0.5
```

```
In [35]: # Theoretical CDF
         xlim0 = stats.binom.ppf(0.01, n, p, loc=0)
         xlim1 = stats.binom.ppf(0.99, n, p, loc=0)
         x0 = np.linspace(xlim0, xlim1, 100)
         y0 = stats.binom.cdf(x0, n, p, loc=0)
```

```
In [36]: # Sample size
         N = 100
```

```
In [37]: # Random sample (scipy)
         y = stats.binom.rvs(n, p, loc=0, size=N)
```

```
print("mean=", np.mean(y).round(2), ";", "std=", np.std(y).round(2))

# Empirical CDF
ecdf = ECDF(y)
x2 = ecdf.x
y2 = ecdf.y
```

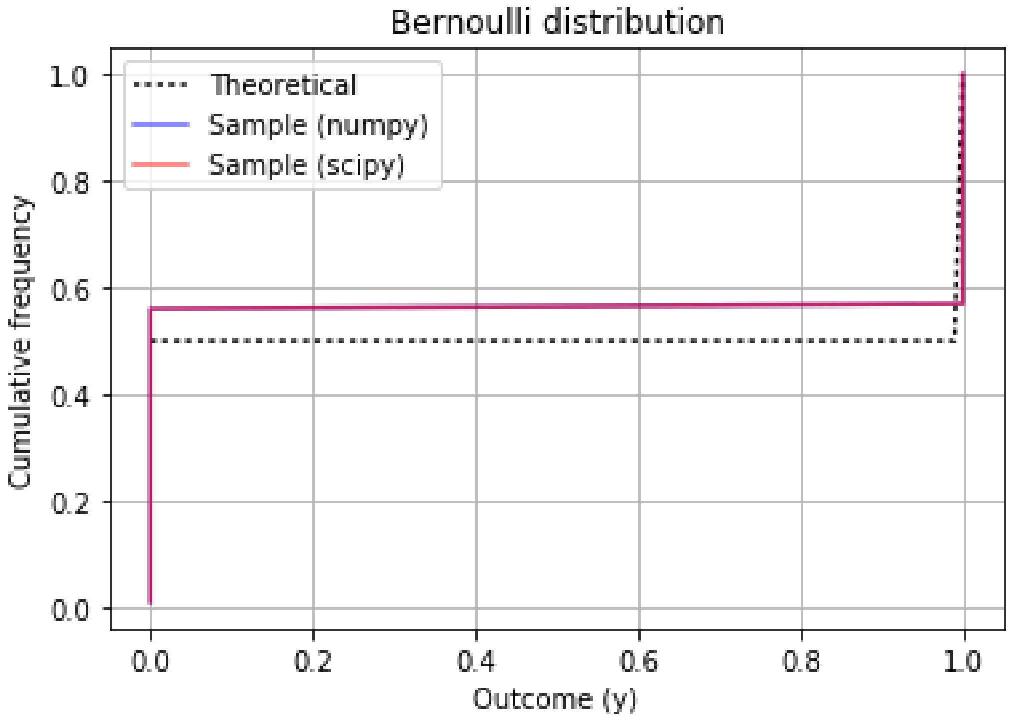
```
mean= 0.44 ; std= 0.5
```

In [38]:
```
# Random sample (numpy)
y1 = np.random.binomial(n, p, size=N)
print("mean=", np.mean(y).round(2), ";", "std=", np.std(y).round(2))

# Empirical CDF
ecdf = ECDF(y)
x1 = ecdf.x
y1 = ecdf.y
```

```
mean= 0.44 ; std= 0.5
```

In [39]:
```
# Plot
fig, ax = plt.subplots(1, 1)
_ = ax.plot(x0, y0, marker="", ls=":", c="k", alpha=1.0, label="Theoretica
_ = ax.plot(x1, y1, marker="", ls="-", c="b", alpha=0.6, label="Sample (nu
_ = ax.plot(x2, y2, marker="", ls="-", c="r", alpha=0.6, label="Sample (sc
ax.set_xlabel("Outcome (y)")
ax.set_ylabel("Cumulative frequency")
ax.set_title("Bernoulli distribution")
ax.legend()
ax.grid(b=True)
plt.show()
```



1.7 Binomial distribution

In [40]:
```
# Parameters
p = 0.5
n = 10

# Statistical properties
mu = n * p
sd = np.sqrt(n * p * (1 - p))
print("mean=", np.round(mu, 2), ";", "std=", np.round(sd, 2))
```

```
mean= 5.0 ; std= 1.58
```

In [41]:
```
# Theoretical CDF
xlim0 = stats.binom.ppf(0.01, n, p, loc=0)
xlim1 = stats.binom.ppf(0.99, n, p, loc=0)
x0 = np.linspace(xlim0, xlim1, 100)
y0 = stats.binom.cdf(x0, n, p, loc=0)
```

In [42]:
```
# Sample size
N = 100
```

In [43]:
```
# Random sample (scipy)
y = stats.binom.rvs(n, p, loc=0, size=N)
```

```
print("mean=", np.mean(y).round(2), ";", "std=", np.std(y).round(2))

# Empirical CDF
ecdf = ECDF(y)
x2 = ecdf.x
y2 = ecdf.y
```

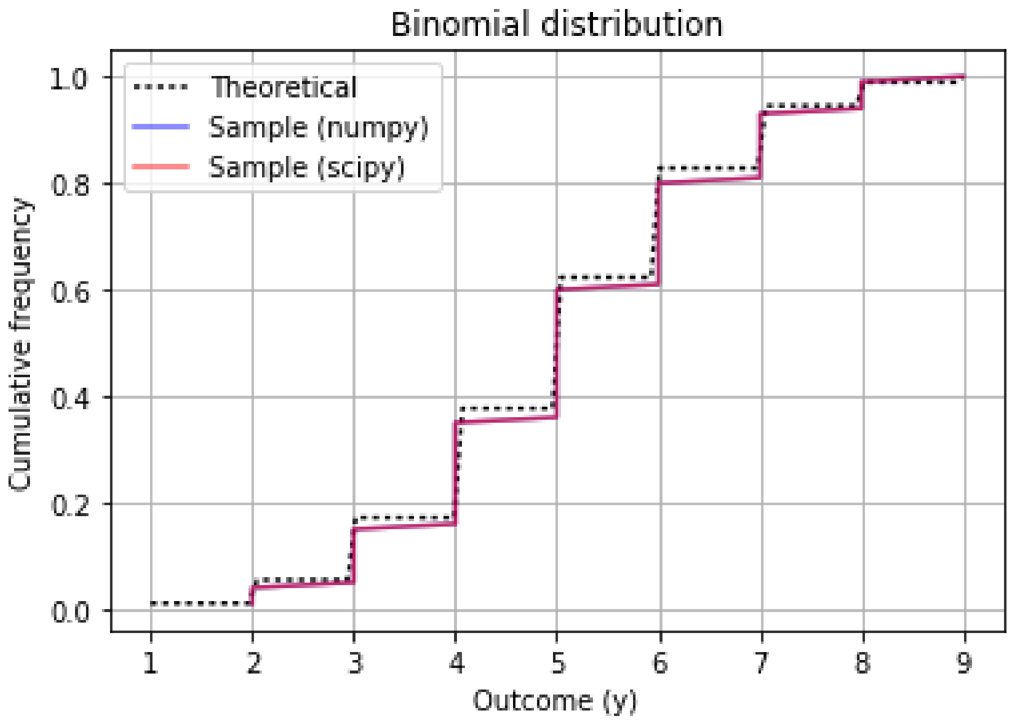
```
mean= 5.14 ; std= 1.55
```

```
In [44]: # Random sample (numpy)
y1 = np.random.binomial(n, p, size=N)
print("mean=", np.mean(y).round(2), ";", "std=", np.std(y).round(2))

# Empirical CDF
ecdf = ECDF(y)
x1 = ecdf.x
y1 = ecdf.y
```

```
mean= 5.14 ; std= 1.55
```

```
In [45]: # Plot
fig, ax = plt.subplots(1, 1)
_ = ax.plot(x0, y0, marker="", ls=":", c="k", alpha=1.0, label="Theoretica
_ = ax.plot(x1, y1, marker="", ls="-", c="b", alpha=0.6, label="Sample (nu
_ = ax.plot(x2, y2, marker="", ls="-", c="r", alpha=0.6, label="Sample (sc
ax.set_xlabel("Outcome (y)")
ax.set_ylabel("Cumulative frequency")
ax.set_title("Binomial distribution")
ax.legend()
ax.grid(b=True)
plt.show()
```



1.8 Categorical distribution

In [46]:
```
# Parameters
k = 3
x = np.random.uniform(0, 1, k)
p = x / sum(x)
n = 1

# Statistical properties
mu = p
sd = np.sqrt(p * (1 - p))
print("mean=", np.round(mu, 2), ";", "std=", np.round(sd, 2))
```

```
mean= [0.3 0.5 0.2] ; std= [0.46 0.5  0.4 ]
```

In [47]:
```
# Sample size
N = 100
```

In [48]:
```
# Random sample (numpy)
y = np.random.multinomial(n, p, size=N)
print("mean=", np.mean(y, 0).round(2), ";", "std=", np.std(y, 0).round(2))
```

```
mean= [0.36 0.48 0.16] ; std= [0.48 0.5  0.37]
```

```
In [49]: # stats.multinomial.xxx() not implemented yet
```

1.9 Multinomial distribution

```
In [50]: # Parameters
         k = 3
         x = np.random.uniform(0, 1, k)
         p = x / sum(x)
         n = 5

         # Statistical properties
         mu = n * p
         sd = np.sqrt(n * p * (1 - p))
         print("mean=", np.round(mu, 2), ";", "std=", np.round(sd, 2))

mean= [0.83 1.27 2.91] ; std= [0.83 0.97 1.1 ]


In [51]: # Theoretical CDF
         # stats.multinomial.xxx() not implemented yet

In [52]: # Sample size
         N = 100

In [53]: # Random sample (numpy)
         y = np.random.multinomial(n, p, size=N)
         print("mean=", np.mean(y, 0).round(2), ";", "std=", np.std(y, 0).round(2))

mean= [0.79 1.23 2.98] ; std= [0.79 0.99 1.03]


In [49]: # stats.multinomial.xxx() not implemented yet
```