topdiscovery

November 28, 2022

0.1 Frequentist and Bayesian analyses of Dzero $p\bar{p} \to t\bar{t}$ discovery data

Created: 12 Mar. 2018, Bhubaneswar, India, Harrison B. Prosper **Updated**: 27 Nov. 2022, adapted for ASP 2022, South Africa, HBP

In 1995, the Dzero and CDF Collaborations announced the discovery of the top quark, ending a 4-decade long search. The Dzero discovery was based on the following experimental results:

$$N = 17 \text{ observed events}$$
$$B \pm \delta B = 3.8 \pm 0.6 \text{ predicted background events}$$

We take the likelihood for N to be

$$p(N|s, b) = (s + b)^N \exp(-(s + b))/N!.$$

In order to write down a likelihood for the background, we shall use the ansatz

$$B = M/k,$$
$$\delta B = \sqrt{M}/k,$$

that is, we model the background estimate as a number obtained by scaling down a much larger count M by a scale factor k. Therefore, the likelihood for the background is taken to be

$$p(M|kb) = (kb)^M \exp(-kb)/M!.$$

Solving for M and k, we find

$$M = 40.11 \text{ events}$$
$$k = 10.56.$$

The non-integral value for M is taken into accout by writing $M! = \Gamma(M+1)$. The overall likelihood for the data $D = N, M, k$ is therefore

$$\boxed{p(D|s,b) = \frac{(s+b)^N \exp(-(s+b))}{N!} \frac{(kb)^M \exp(-kb)}{\Gamma(M+1)}}.$$

The **parameter of interest** is the mean signal count s, while b, the mean background, is an example of a **nuisance parameter**. A nuisance parameter is a parameter of the probability model that is needed to describe the model, but is not of intrinsic interest. In order to make inferences about the parameter (or parameters) of interest, all nuisance parameters must be removed from the model. This can be done in two ways: the nuisance parameters can be profiled away or they can be integrated out with respect to some prior density $\pi(*)$. Note that since integration (i.e., **marginalization**) follows directly from the rules of probability theory, this method for removing nuisance parameters is natural in the Bayesian approach. Strictly speaking however, it is conceptually inconsistent within the frequentist approach because the notion of a probability density of a parameter does not exist within that approach. Nevertheless, for pragmatic reasons, both integration and profiling are used in the frequentist approach.

0.2 Method 1 (Profiling)

In the method, nuisance parameters are removed by replacing them with their maximum likelihood estimates for a given values of the parameter of interest. Since replacing true values by estimates thereof is an approximation, the frequentist principle is not guaranteed to be satisfied exactly.

In this example, the nuisance parameter b in the likelihood is replaced by its maximum likelihood estimate, $\widehat{b}(s)$, of b as a function of the mean signal s. We find,

$$\widehat{b}(s) = \frac{g + \sqrt{g^2 + 4(1+k)Ms}}{2(1+k)}, \text{ where}$$
$$g = N + M - (1+k)s.$$

This procedure yields the profile likelihood $L_p(s) \equiv p(D|s, \widehat{b}(s))$, which is then to be used as if it were a 1-parameter likelihood.

0.3 Method 2 (Integration)

For simplicity, let's integrate over b with a ***flat*** prior density given by \$ (b) db = k db\$, so that $k \int_0^\infty p(M|kb)\, db = 1$, where the background likelihood is $p(M \mid kb) = (kb)^M \exp(-kb)/\Gamma(M+1)$.

$$
\begin{aligned}
p(D|s) &= \int_0^\infty p(N|s,b)\, p(M|kb)\, d(kb), \\
&= \frac{k}{N!\Gamma(M+1)} e^{-s} \int_0^\infty e^{-(1+\kappa)b} (s+b)^N\, (\kappa b)^M\, db \\
&= \frac{k}{N!\Gamma(M+1)} e^{-s} \int_0^\infty e^{-(1+\kappa)b} \left[\sum_{r=0}^{N} \binom{N}{r} s^{N-r}\, b^r \right] (\kappa b)^M\, db \\
&= \frac{k}{\Gamma(M+1)} \sum_{r=0}^{N} e^{-s} \frac{s^{N-r}}{(N-r)!} \frac{1}{r!} \int_0^\infty e^{-(1+\kappa)b}\, b^{M+r}\, \kappa^M\, db \\
&= k \sum_{r=0}^{N} \frac{\kappa^M}{(1+\kappa)^{M+r+1}\, \Gamma(M+1)} \,\mathrm{Poisson}(N-r,s) \frac{1}{r!} \int_0^\infty e^{-(1+\kappa)b} [(1+\kappa)b]^{M+r}\, d(1+\kappa)b \\
&= k \sum_{r=0}^{N} \frac{\kappa^M}{(1+\kappa)^{M+r+1}\, \Gamma(M+1)} \frac{\Gamma(M+r+1)}{r!} \,\mathrm{Poisson}(N-r,s), \\
&= \frac{k}{M} \sum_{r=0}^{N} \frac{\kappa^M}{(1+\kappa)^{M+r+1}} \frac{\Gamma(M+r+1)}{\Gamma(M)\Gamma(r+1)} \,\mathrm{Poisson}(N-r,s), \\
\text{let } x &= \frac{1}{1+\kappa} \text{ and } 1-x = \frac{\kappa}{1+\kappa}, \text{ then} \\
&= k \frac{x(1-x)}{M} \sum_{r=0}^{N} x^{r+1-1}\, (1-x)^{M-1} \frac{\Gamma(M+r+1)}{\Gamma(M)\Gamma(r+1)} \,\mathrm{Poisson}(N-r,s), \\
&= \frac{(1-x)^2}{M} \sum_{r=0}^{N} \mathrm{beta}(x, r+1, M) \,\mathrm{Poisson}(N-r,s).
\end{aligned}
$$

We shall compute an interval $[l(N), u(N)]$ using the Dzero data and assert that the signal $s \in [l(N), u(N)]$ with a confidence level of $p = 0.683$. The interpretation of the probability p depends on which approach has been adopted, frequentist or Bayesian. In the former, p is interpreted as the minimum fraction of statements of this form that are true in an infinite ensemble of statements, not necessarily about the same quantity. The lower and upper bounds of the interval are found by solving

$$
\begin{aligned}
D_L(l) &= (1-p)/2, \\
D_R(u) &= (1-p)/2 = 1 - D_L(u), \\
D_L(u) &= (1+p)/2,
\end{aligned}
$$

where

$$
\begin{aligned}
D_L(z) &= \int_0^z p(s|D)\, ds, \\
&= \sum_{r=0}^{N} \mathrm{beta}(x, r+1, M)\, P(N-r+1, z) / \sum_{r=0}^{N} \mathrm{beta}(x, r+1, M),
\end{aligned}
$$

0.3.1 Import required modules

modules	description
pandas	a versatile module for manipulating data, typically, loaded from csv files
numpy	the standard array manipulation and numerical module in Python
scipy	the standard scientific tools module in Python
matplotlib	a widely used plotting module for producing high quality plots

[4]:
```
import os, sys
import pandas as pd
import numpy as np

import scipy as sp
import scipy.stats as st
import scipy.integrate as integrate

import matplotlib as mp
import matplotlib.pyplot as plt

# make plots appear inline
%matplotlib inline

# update fonts
LABEL_FONT_SIZE = 20
font = {'family' : 'serif',
        'weight' : 'normal',
        'size'   : LABEL_FONT_SIZE
        }
mp.rc('font', **font)

#mp.rc('xtick', labelsize='x-small')
#mp.rc('ytick', labelsize='x-small')

# set usetex = False if Latex is not
# available on your system
mp.rc('text', usetex=True)
```

0.3.2 Dzero Results

[5]:
```
N = 17
B = 3.8
dB= 0.6

S = N - B   # MLE of signal
```

```
M =(B / dB)**2
k = M / B

print("Effective background count (M):        %10.2f" % M)
print("Effective background scale factor (k): %10.2f" % k)
```

```
Effective background count (M):             40.11
Effective background scale factor (k):      10.56
```

0.3.3 Profile Likelihood $L_p(s) \equiv p(D|s, \hat{b}(s))$

Write $L_p(s)$ so that there can be an implied loop over n.

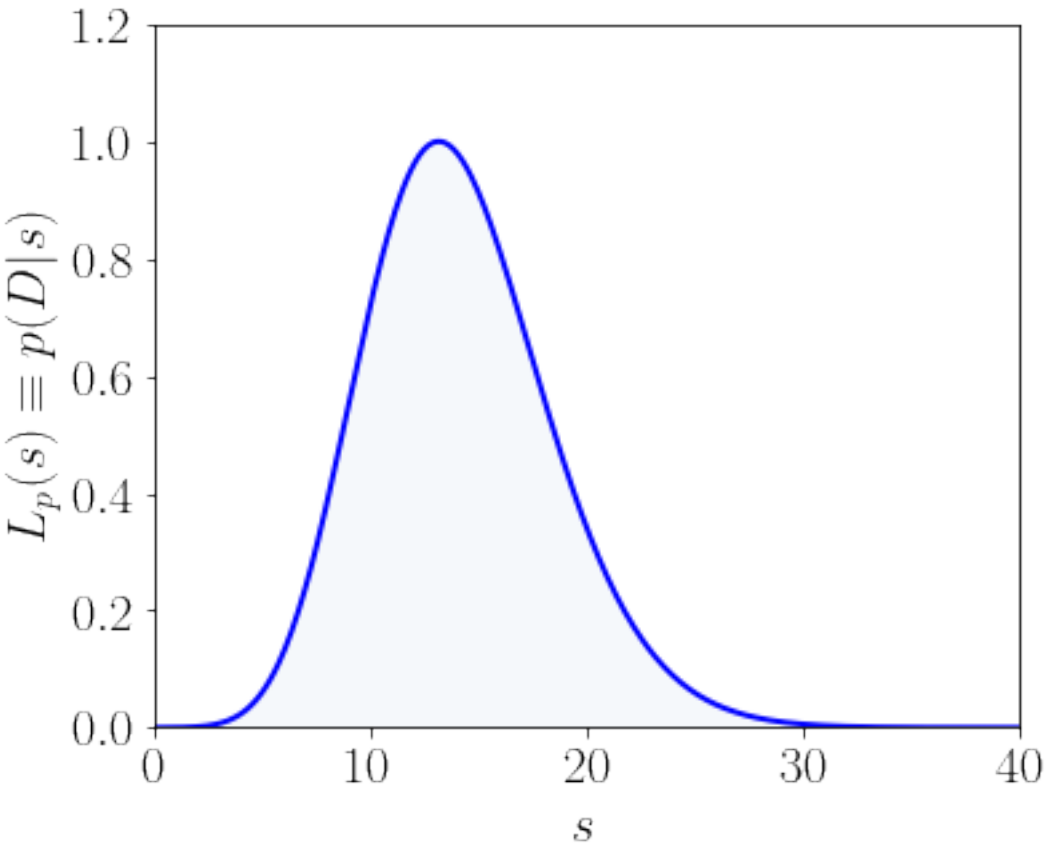
$$\hat{b}(s) = \frac{g + \sqrt{g^2 + 4(1+k)Ms}}{2(1+k)}, \text{ where}$$
$$g = N + M - (1+k)s.$$

[6]:
```
def Lp(N, M, k, s):
    g = N + M - (1+k)*s
    b_hat  = g + np.sqrt(g*g+4*(1+k)*M*s)
    b_hat /= 2*(1+k)

    # use gamma distribution to compute Poisson since
    # it can cope better with large counts.
    p1 = sp.stats.gamma.pdf(s + b_hat, N + 1)
    p2 = sp.stats.gamma.pdf(k * b_hat, M + 1)
    return p1 * p2
```

[7]:
```
def plot_profile(Lp, N, M, k, limits=None):

    # set size of figure
    plt.figure(figsize=(6, 5))

    xmin, xmax = 0, 40
    ymin, ymax = 0, 1.2

    # set up x, y limits
    plt.xlim(xmin, xmax)
    plt.ylim(ymin, ymax)

    plt.xlabel(r'$s$', fontsize=LABEL_FONT_SIZE)
    plt.ylabel(r'$L_p(s) \equiv p(D|s)$', fontsize=LABEL_FONT_SIZE)

    x = np.arange(xmin, xmax, 0.1) # x = [0, 0.1, ...]

    # compute profile likelihood; set max(L) = 1
```

```
    y = [Lp(N, M, k, s) for s in x]; y = y / max(y)

    # plot profile likelihood
    plt.plot(x, y,
             color='blue',
             linewidth=2)

    # fill beneath the curve
    plt.fill_between(x, y, alpha=0.05, color='steelblue')

    if limits:
        # fill between [l(N), u(N)]
        lN, uN = limits
        x = np.arange(lN, uN, 0.1)
        y = [Lp(N, M, k, z) for z in x]; y = y / max(y)
        plt.fill_between(x, y, alpha=0.4, color='green')

        plt.text(20, 1.00, f'$s \in [{lN:5.1f}, {uN:5.1f}]$')
        plt.text(20, 0.85, '@ 68.3\% CL')

    plt.tight_layout()
    plt.savefig("fig/top_quark_profile.jpg")
    plt.show()
```

[8]:
```
plot_profile(Lp, N, M, k)
```



0.3.4 Compute statistic $\lambda(s) = L_P(s)/L_p(\hat{s})$ and $t_{\mathbf{obs}}(s) = -2\ln\lambda(s)$.

Note that $0 < \lambda(s) < 1$ and reaches its upper bound for $s = \hat{s}$. In effect, $\lambda(s)$ compares the likelihood of hypothesis s to the hypothesis $\hat{s}$ which agrees best with data.

[9]:
```
def t(s, N, M, k):
    L    = Lp(N, M, k, s)
    Lmax = Lp(N, M, k, S)
    return -2*np.log(L/Lmax)
```

0.3.5 Find (approximate) 68% confidence interval for s.

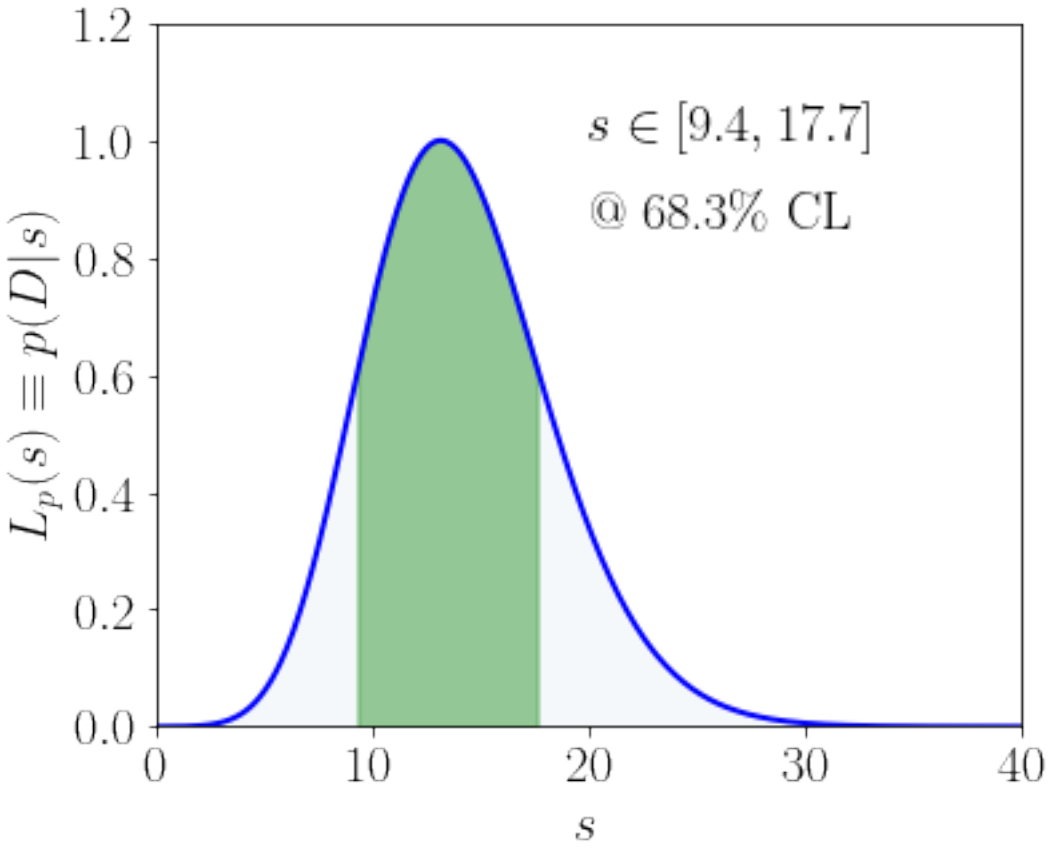
Solve $t(s) = 1$ for s_L and s_U.

[10]:
```
def findLimit(f, N, M, k, smin, smax, alpha):
    def func(x):
        return f(x, N, M, k) - alpha
    return sp.optimize.brenth(func, smin, smax)
```

[11]:
```
alpha= 1

smin = 1.e-2  # avoid 0
smax = S      # mode of Lp(s) = p(D|s)
lN   = findLimit(t, N, M, k, smin, smax, alpha)
print("lower bound l(%d): %10.2f" % (N, lN))

smin = S
smax = 50
uN = findLimit(t, N, M, k, smin, smax, alpha)
print("upper bound u(%d): %10.2f" % (N, uN))
```

```
lower bound l(17):       9.35
upper bound u(17):      17.70
```

[12]:
```
plot_profile(Lp, N, M, k, limits=[lN, uN])
```



0.3.6 Compute observed value of statistic $t(s)$ with $H_0 : s = 0$.

[13]:
```
tobs = t(0, N, M, k)
Z    = np.sqrt(tobs)
print("t_obs(0): %6.1f\tZ: %6.1f" % (tobs, Z))
```

```
t_obs(0):   20.9        Z:    4.6
```

0.4 Bayesian Approach

0.4.1 Marginal (or Integrated) Likelihood

$$p(D \mid s) = \frac{(1-x)^2}{M} \sum_{r=0}^{N} \text{beta}(x, r+1, M)\, \text{Poisson}(N-r, s).$$

Write $p(D|s)$ so that there can be an implied loop over n.

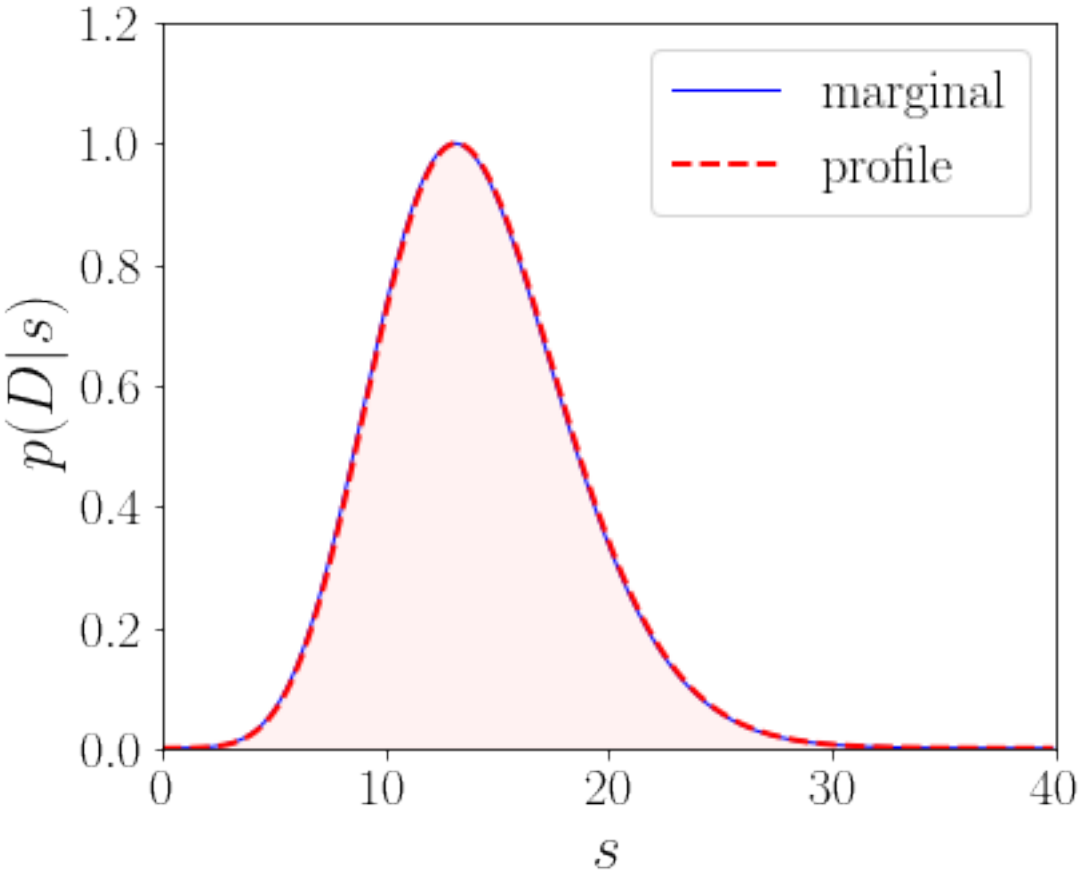
[14]:
```
def pDs(N, M, k, s):
    x = 1.0/(1 + k)
    A = (1-x)**2
    if type(N) == type(0):

        # create list r = [0, 1, ..., N]
        r = np.arange(0, N+1)

        # probability density function (pdf)
        beta = sp.stats.beta.pdf(x, r + 1, M)

        # probability mass function (pmf)
        poisson = sp.stats.poisson.pmf(N - r, s)

        p = A * sum(beta * poisson) / M
        return p

    else:
        p = []
        for n in N:
            r = np.arange(0, n+1)
            beta = sp.stats.beta.pdf(x, r + 1, M)
            poisson = sp.stats.poisson.pmf(n - r, s)
            p.append(A * sum(beta * poisson) / M)
        return np.array(p)
```

0.4.2 Plot likelihoods

Superimpose profile and marginal likelihoods.

[15]:
```
def plotLikelihoods(Lp, pDs, N, M, k):

    # set size of figure
    plt.figure(figsize=(6, 5))

    xmin, xmax = 0, 40
    ymin, ymax = 0, 1.2

    # set up x, y limits
    plt.xlim(xmin, xmax)
    plt.ylim(ymin, ymax)

    plt.xlabel(r'$s$', fontsize=24)
    plt.ylabel(r'$p(D|s)$', fontsize=24)

    x  = np.arange(xmin, xmax, 0.1) # x = [0, 0.1, ...]

    # compute likelihoods set max(L) = 1
    y1 = [pDs(N, M, k, s) for s in x]; y1 = y1 / max(y1)
    y2 = [Lp(N, M, k, s)  for s in x]; y2 = y2 / max(y2)



    # plot marginal likelihood
    plt.plot(x, y1,
             color='blue',
             linewidth=1,
             label=r'marginal')

    # plot profile likelihood
    plt.plot(x, y2,
             color='red',
             linewidth=2,
             linestyle='dashed',
             label='profile')

    # fill beneath the curve
    plt.fill_between(x, y2, alpha=0.05, color='red')

    # add a legend with labels given in plt.plot
    plt.legend()

    plt.savefig("fig/top_quark_profile_vs_marginal.jpg")
    plt.show()
```

[16]:
```
plotLikelihoods(Lp, pDs, N, M, k)
```



0.4.3 Compute $D_L(z)$

$$
\begin{aligned}
D_L(z) &= \int_0^z p(s|D)\, ds, \\
&= \sum_{r=0}^{N} \text{beta}(x, r+1, M)\, P(N-r+1, z) / \sum_{r=0}^{N} \text{beta}(x, r+1, M),
\end{aligned}
$$

[17]:
```
def DL(z, N, M, k):
    x = 1.0/(1 + k)
    r = np.arange(0, N + 1)
    beta = sp.stats.beta.pdf(x, r + 1, M)
    igam = sp.special.gammainc(N - r + 1, z)
    p = sum(beta * igam) / sum(beta)
    return p

def DR(z, N, M, k):
    return 1 - DL(z, N, M, k)
```

0.4.4 Find roots of

$$D_L(l) = \alpha_L,$$
$$D_R(u) = \alpha_R.$$

For central limits we set $\alpha_L = \alpha_R = (1 - \mathrm{CL})/2$, where the confidence level is 0.683.

Find lower and upper bounds $l(N)$, $u(N)$

[18]:
```
CL    = 0.683
alpha= (1-CL)/2

smin = 1.e-2  # avoid 0
smax = S      # approximate location of peak of p(s|D)
lN   = findLimit(DL, N, M, k, smin, smax, alpha)
print("lower bound l(%d): %10.2f" % (N, lN))



smin = S         # approximate location of peak of p(s|D)
smax = 50
uN = findLimit(DR, N, M, k, smin, smax, alpha)
print("upper bound u(%d): %10.2f" % (N, uN))
```

```
lower bound l(17):       9.86
upper bound u(17):      18.35
```

...then check lower and upper bounds. Probability content should be 0.683.

[19]:
```
p, perr = integrate.quad(lambda x: pDs(N, M, k, x), lN, uN)
print(f"Probability content in interval "\
      f"[{lN:6.2f}, {uN:6.2f}] = {p:6.3f}")
```

```
Probability content in interval [  9.86,  18.35] =  0.683
```

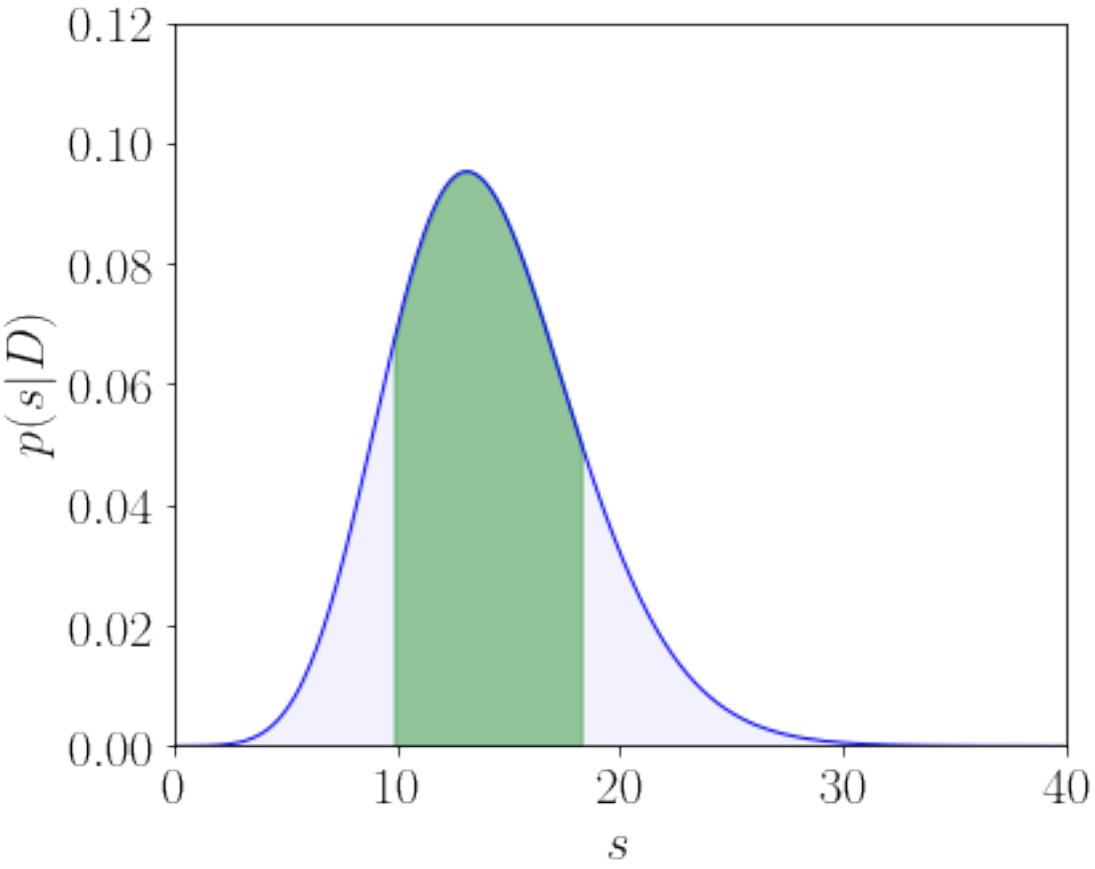
0.5 Plot Posterior Density

$$p(s \mid D) \quad = \quad \frac{\sum_{r=0}^{N} \mathrm{beta}(x, r+1, M)\, \mathrm{Poisson}(N-r, s)}{\sum_{r=0}^{N} \mathrm{beta}(x, r+1, M)}.$$

[20]:
```
def psD(s, N, M, k):
    x = 1.0/(1 + k)
    r = np.arange(0, N + 1)
    beta = sp.stats.beta.pdf(x, r + 1, M)
    pois = sp.stats.gamma.pdf(s, N - r +1)
    p = sum(beta*pois)/sum(beta)
    return p
```

[21]:
```
def plotPost(psD, N, M, k, lN, uN):

    # set size of figure
    plt.figure(figsize=(6, 5))

    xmin, xmax = 0, 40
    ymin, ymax = 0, 0.12

    # set up x, y limits
    plt.xlim(xmin, xmax)
    plt.ylim(ymin, ymax)

    plt.xlabel(r'$s$', fontsize=LABEL_FONT_SIZE)
    plt.ylabel(r'$p(s|D)$', fontsize=LABEL_FONT_SIZE)

    x = np.arange(xmin, xmax, 0.1)
    y = [psD(z, N, M, k) for z in x]

    plt.plot(x, y,
             color='blue',
             linewidth=1,
             label=r'marginal')
    plt.fill_between(x, y, alpha=0.05, color='blue')

    # fill between [l(N), u(N)]
    x = np.arange(lN, uN, 0.1)
    y = [psD(z, N, M, k) for z in x]
    plt.fill_between(x, y, alpha=0.4, color='green')

    plt.savefig("fig/top_quark_posterior_density.jpg")
    plt.show()
```

[22]:
```
plotPost(psD, N, M, k, lN, uN)
```



0.5.1 Compute Bayes factor

[23]:
```
pH1 = psD(S, N, M, k)
pH0 = psD(0, N, M, k)
B10 = pH1 / pH0
Z   = np.sqrt(2*np.log(B10))
print("Bayes factor, Z: %10.2e, %6.1f" % (B10, Z))
```

```
Bayes factor, Z:   3.17e+04,    4.6
```

0.5.2 Compute p-value with and without inclusion of the background uncertainty

[24]:
```
def pvalue_noerror(N, B):
    pv = sp.special.gammainc(N, B)
    Z  = sp.special.erfinv(1-2*pv)*np.sqrt(2)
    return (pv, Z)

def pvalue(N, M, k):
    n  = np.arange(0, N)
```

```
    pv = 1 - sum(pDs(n, M, k, 0))
    Z  = sp.special.erfinv(1-2*pv)*np.sqrt(2)
    return (pv, Z)
```

[25]:
```
pv0, Z0 = pvalue_noerror(N, B)
pv1, Z1 = pvalue(N, M, k)
print(f"p-value(no unc.):   {pv0:10.2e}\tZ: {Z0:6.1f}")
print(f"p-value(with unc.): {pv1:10.2e}\tZ: {Z1:6.1f}")
```

```
p-value(no unc.):     5.71e-07  Z:    4.9
p-value(with unc.):   4.15e-06  Z:    4.5
```

[ ]:
```

```