benford_ft

August 21, 2021

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

from konstante import get_constants

from benford_helper_functions import get_first_digit, normalize,␣
↪shift_multiply_integrate_pdf
from benford_helper_functions import make_sampling_function
from benford_helper_functions import benford_ft
```

https://stackoverflow.com/questions/66912677/scipy-stats-lognormal-distribution-obtain-pdf-with-given-lognormal-distribution

[2]:
```
import sys

sys.path.append('../..')
from plotting.matplotlib_setup import configure_latex, savefig,␣
↪set_size_decorator, savefig

tex_dir, images_dir = 'porocilo/main.tex', 'porocilo/images'

configure_latex(style=['science', 'notebook'], global_save_path=images_dir)

%config InlineBackend.figure_format = 'pdf'
```

[3]:
```
from scipy.stats import lognorm

def lognormMu(x, mu, s):
    tempX = x / np.exp(mu)
    return lognorm.pdf(tempX, s)

x = np.logspace(0, 6, 10000)[1:] # NON log axis

s = [0.5, 0.8, 1.2]
lognorm_pdf1 = lognormMu(x, mu=7.1, s=s[0])
lognorm_pdf2 = lognormMu(x, mu=7.1, s=s[1])
lognorm_pdf3 = lognormMu(x, mu=7.1, s=s[2])
```

[4]:
```
consts = get_constants() # constants dataset
```

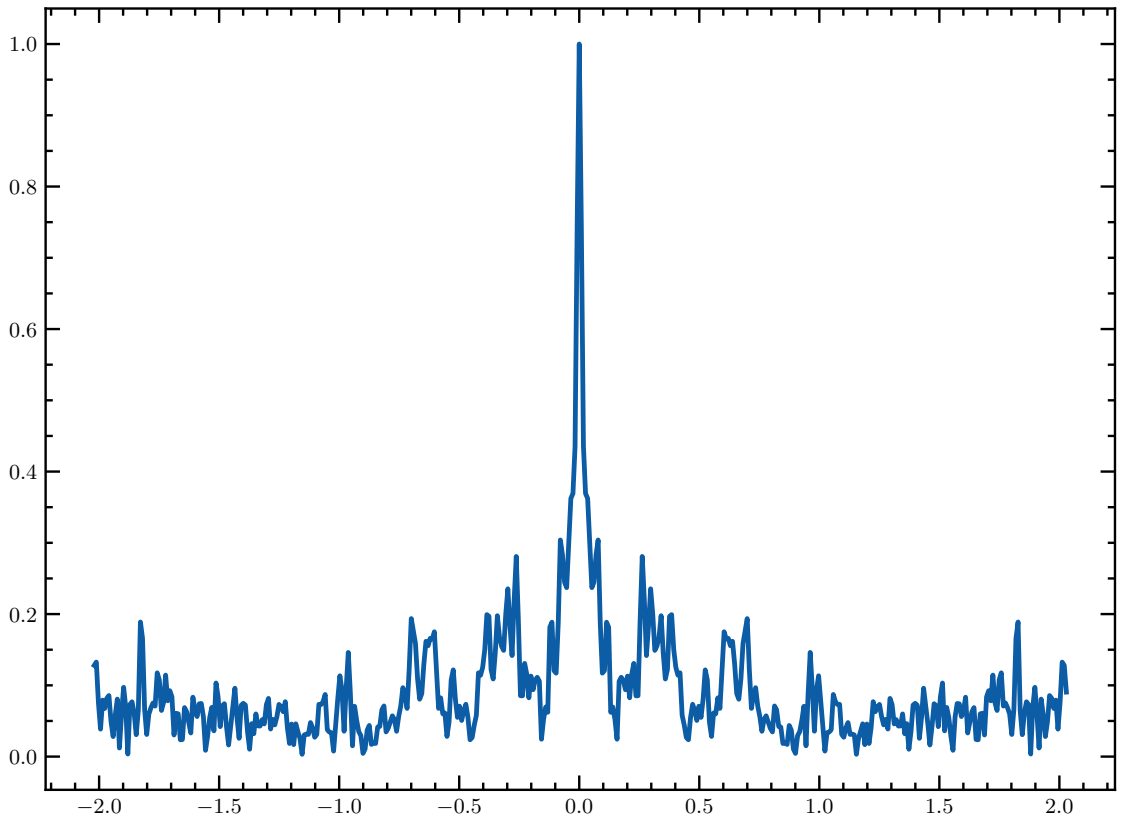
[5]:
```
lognorm_pdf1 = normalize(lognorm_pdf1, np.log10(x))
lognorm_pdf2 = normalize(lognorm_pdf2, np.log10(x))
lognorm_pdf3 = normalize(lognorm_pdf3, np.log10(x))
```

[6]:
```
log_input = np.log10(np.abs(consts))
n_, bins_ = np.histogram(log_input, bins=len(log_input), density=True)
bins_ = bins_[1:]
```

[7]:
```
f, SF, sf, PDF, OST, ost = benford_ft(n_, bins_, shift=True)
```

[8]:
```
plt.plot(f, np.abs(PDF))
```

[8]:
```
[<matplotlib.lines.Line2D at 0x7fb0dd0e7f70>]
```



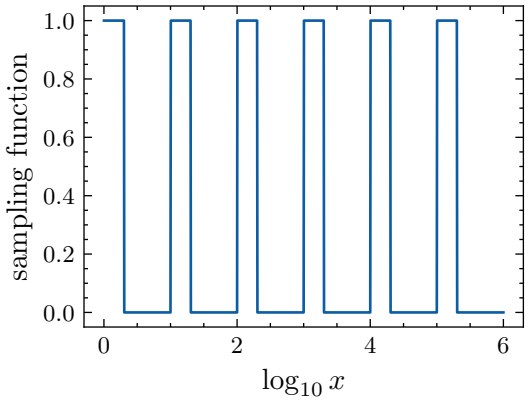
[9]:
```
bins = np.log10(x)
N = len(bins)
```

[10]:
```
ns = [lognorm_pdf1, lognorm_pdf2, lognorm_pdf3]
```

[11]:
```
res = []
for n in ns:
    f, SF, sf, PDF, OST, ost = benford_ft(n, bins, shift=True)
    res.append([f, SF, sf, PDF, OST, ost])
```

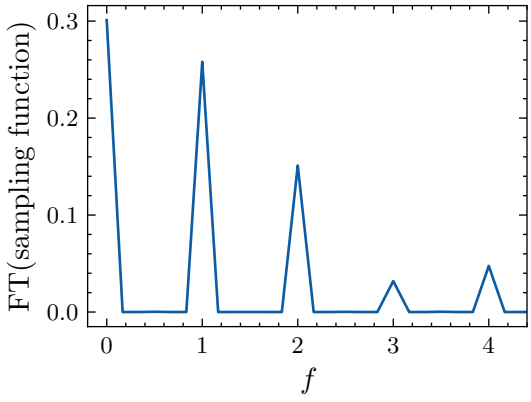
[12]:
```
fig, ax = set_size_decorator(plt.subplots, fraction=0.5, ratio='4:3')(1, 1)

ax.plot(bins, res[0][2], lw=1)
ax.set_xlabel(r'$\log_{10}x$')
ax.set_ylabel('sampling function')
#savefig('lognorm_sampling_function')
```

[12]: Text(0, 0.5, 'sampling function')



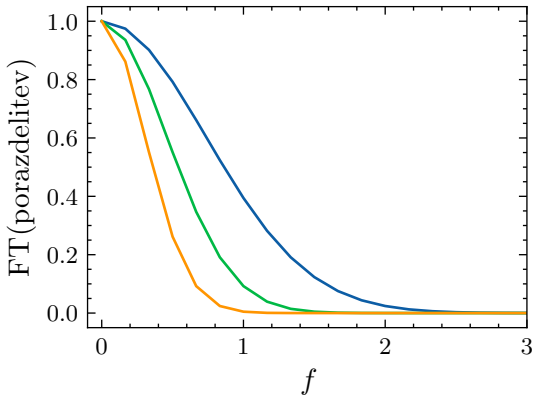
[13]:
```
fig, ax = set_size_decorator(plt.subplots, fraction=0.5, ratio='4:3')(1, 1)

ax.plot(res[0][0][N//2:], np.abs(res[0][1][N//2:]), lw=1)
ax.set_xlim([-0.2, 4.4])

ax.set_xlabel('$f$')
ax.set_ylabel('FT(sampling function)')

# savefig('lognorm_FT_sampling_function')
```

[13]: Text(0, 0.5, 'FT(sampling function)')



[14]:
```
fig, ax = set_size_decorator(plt.subplots, fraction=0.5, ratio='4:3')(1, 1)

for r in res:
    ax.plot(r[0][N//2:], np.abs(r[3][N//2:]), lw=1)

ax.set_xlim([-0.1, 3])
ax.set_xlabel('$f$')
ax.set_ylabel('FT(porazdelitev)')

#savefig('lognorm_FT_pdf')
```

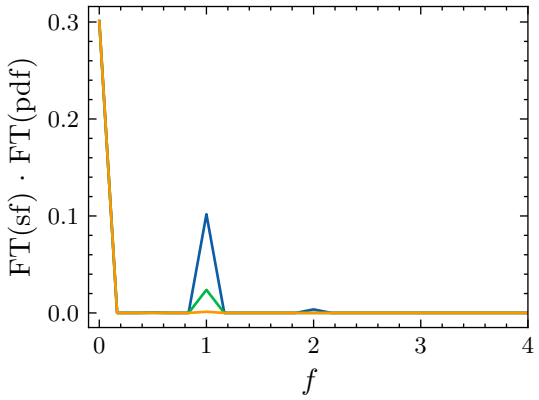
[14]:
```
Text(0, 0.5, 'FT(porazdelitev)')
```



[15]:
```
fig, ax = set_size_decorator(plt.subplots, fraction=0.5, ratio='4:3')(1, 1)

for r in res:
    ax.plot(r[0][N//2:], np.abs(r[4][N//2:]), lw=1)
```

```
ax.set_xlim([-0.1, 4])
ax.set_xlabel('$f$')
ax.set_ylabel('FT(sf) $\cdot$ FT(pdf)')

#savefig('lognorm_conv')
```

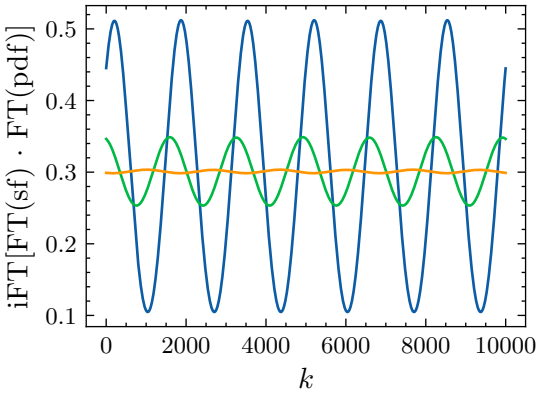
[15]:
```
Text(0, 0.5, 'FT(sf) $\\cdot$ FT(pdf)')
```



[16]:
```
fig, ax = set_size_decorator(plt.subplots, fraction=0.5, ratio='4:3')(1, 1)

for r in res:
    ax.plot(range(len(r[5])), r[5], lw=1)
    print(np.mean(r[5]))

ax.set_xlabel('$k$')
ax.set_ylabel('iFT[FT(sf) $\cdot$ FT(pdf)]')

# savefig('lognorm_ost')
```

```
0.30093009300930096
0.3009300930093009
0.30093009300930096
```

[16]:
```
Text(0, 0.5, 'iFT[FT(sf) $\\cdot$ FT(pdf)]')
```

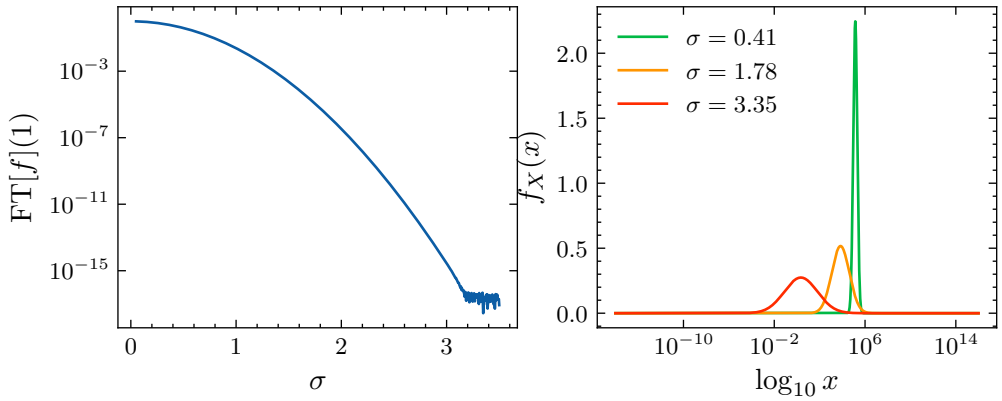


[17]:
```
from benford_helper_functions import benfords_test

x = np.logspace(-16, 16, 10000)[1:]

sigmas = np.linspace(0.05, 3.5, 1000)

res = []
dists = []
for s in sigmas:
    lognorm_pdf = lognormMu(x, mu=12, s=s)

    idx = np.argwhere(lognorm_pdf > 0)
    z = np.log10(x)

    lognorm_pdf = normalize(lognorm_pdf, z)
    t = benfords_test(lognorm_pdf, z)
    res.append(t)
    dists.append(lognorm_pdf)
```

[18]:
```
fig, axs = set_size_decorator(plt.subplots, fraction=1, ratio='4:3')(1, 2)

axs[0].set_yscale('log')
axs[0].plot(sigmas, res, lw=1, c='C0')

axs[1].set_xscale('log')
axs[1].plot(x, dists[104], label=f'$\sigma={sigmas[104]:.2f}$', lw=1, c='C1')
axs[1].plot(x, dists[len(sigmas)//2], label=f'$\sigma={sigmas[len(sigmas)//2]:.
↪2f}$', lw=1, c='C2')
axs[1].plot(x, dists[np.argmin(res)], label=f'$\sigma={sigmas[np.argmin(res)]:.
↪2f}$', lw=1, c='C3')
axs[1].legend()
```

```
axs[0].set_ylabel('FT$[f](1)$')
axs[0].set_xlabel('$\sigma$')

axs[1].set_ylabel('$f_X(x)$')
axs[1].set_xlabel('$\log_{10}x$')

# savefig('sigma_FT')
```

[18]:
```
Text(0.5, 0, '$\\log_{10}x$')
```



[ ]: