lognorm_stat_testi

August 21, 2021

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import lognorm

from benford_helper_functions import normalize, get_first_digit, benfords_test
```

[2]:
```
p = 1
low, high = p * np.log(10), (p+1) * np.log(10)

x_ = np.random.uniform(low=low, high=high, size=100000)
x = np.exp(x_)

np.unique(get_first_digit(x), return_counts=True)
```

[2]:
```
(array([1, 2, 3, 4, 5, 6, 7, 8, 9]),
 array([30284, 17461, 12395,  9715,  8027,  6693,  5708,  5108,  4609]))
```

[3]:
```
from stat_tests import chi2_test, ks_test
```

[4]:
```
import sys

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

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

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

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

[5]:
```
def get_lognorm_pdf(x, mu, sigma):
    return (1 / (x * sigma * np.sqrt(2 * np.pi))) * np.exp(-((np.log10(x) -␣
 ↪mu)**2) / (2*sigma**2))

def lognormMu(x, mu, s):
    tempX = x / np.exp(mu)
```

```
    return lognorm.pdf(tempX, s)
```

[6]:
```
from scipy.stats import lognorm


def lognorm_params(mode, stddev):
    """
    Given the mode and std. dev. of the log-normal distribution, this function
    returns the shape and scale parameters for scipy's parameterization of the
    distribution.

    https://stackoverflow.com/questions/41164753/
 ↪generate-random-numbers-from-lognormal-distribution-in-python
    """
    p = np.poly1d([1, -1, 0, 0, -(stddev/mode)**2])
    r = p.roots
    sol = r[(r.imag == 0) & (r.real > 0)].real
    shape = np.sqrt(np.log(sol))
    scale = mode * sol
    return shape, scale


def lognorm_params_exact(mode, stddev):
    a = stddev**2 / mode**2
    x = 1/4*np.sqrt(-(16*(2/3)**(1/3)*a)/(np.sqrt(3)*np.
 ↪sqrt(256*a**3+27*a**2)-9*a)**(1/3) +
                    2*(2/3)**(2/3)*(np.sqrt(3)*np.
 ↪sqrt(256*a**3+27*a**2)-9*a)**(1/3)+1) + \
          1/2*np.sqrt((4*(2/3)**(1/3)*a)/(np.sqrt(3)*np.
 ↪sqrt(256*a**3+27*a**2)-9*a)**(1/3) -
                    (np.sqrt(3)*np.sqrt(256*a**3+27*a**2)-9*a)**(1/3)/(2**(1/
 ↪3)*3**(2/3)) +
                    1/(2*np.sqrt(-(16*(2/3)**(1/3)*a)/(np.sqrt(3)*np.
 ↪sqrt(256*a**3+27*a**2)-9*a)**(1/3) +
                                    2*(2/3)**(2/3)*(np.sqrt(3)*np.
 ↪sqrt(256*a**3+27*a**2)-9*a)**(1/3)+1))+1/2) + \
          1/4
    shape = np.sqrt(np.log(x))
    scale = mode * x
    return shape, scale


N = 10**6
mu = 10**3

# sigmas = np.array([100, 1000, 10000, 100000]) # -> [0.5, 0.8, 1.2]
```

```
sigmas = np.logspace(2, 12, 100)

lognorm_dists, lognorm_sigmas = [], []
for s in sigmas:
    sigma, scale = lognorm_params(mu, s)
    lognorm_sigmas.append(sigma[0])

    np.random.seed(1)
    lognorm_rng = lognorm.rvs(sigma, 0, scale, size=N)

    lognorm_dists.append(lognorm_rng)
```

[7]:
```
f1s = []

for i, lognorm in enumerate(lognorm_dists):
    bins = np.logspace(np.floor(np.log10(lognorm.min())),
                       np.floor(np.log10(lognorm.max())) + 1,
                       len(lognorm))

    pdf, _ = np.histogram(lognorm, bins=bins)

    bins = bins[:-1]
    bins = np.log10(bins)
    pdf = normalize(pdf, bins)

    f1 = benfords_test(pdf, bins)
    f1s.append(f1)

    first_digits = get_first_digit(lognorm)
    _, n = np.unique(first_digits, return_counts=True)
    n = n / np.sum(n)

    print(f'delez 1: {n[0]:.4f}, f1: {f1:.3e}, normal sigma: {sigmas[i]:.2f},␣
↪lognorm sigma: {lognorm_sigmas[i]:.2f}')
```

```
delez 1: 0.5393, f1: 9.647e-01, normal sigma: 100.00, lognorm sigma: 0.10
delez 1: 0.5489, f1: 9.454e-01, normal sigma: 126.19, lognorm sigma: 0.12
delez 1: 0.5607, f1: 9.168e-01, normal sigma: 159.23, lognorm sigma: 0.15
delez 1: 0.5747, f1: 8.759e-01, normal sigma: 200.92, lognorm sigma: 0.19
delez 1: 0.5888, f1: 8.201e-01, normal sigma: 253.54, lognorm sigma: 0.23
delez 1: 0.5964, f1: 7.484e-01, normal sigma: 319.93, lognorm sigma: 0.28
delez 1: 0.5904, f1: 6.627e-01, normal sigma: 403.70, lognorm sigma: 0.33
delez 1: 0.5688, f1: 5.676e-01, normal sigma: 509.41, lognorm sigma: 0.39
delez 1: 0.5355, f1: 4.701e-01, normal sigma: 642.81, lognorm sigma: 0.45
delez 1: 0.4959, f1: 3.769e-01, normal sigma: 811.13, lognorm sigma: 0.51
delez 1: 0.4541, f1: 2.934e-01, normal sigma: 1023.53, lognorm sigma: 0.57
delez 1: 0.4150, f1: 2.224e-01, normal sigma: 1291.55, lognorm sigma: 0.64
```

```
delez 1: 0.3801, f1: 1.648e-01, normal sigma: 1629.75, lognorm sigma: 0.70
delez 1: 0.3517, f1: 1.197e-01, normal sigma: 2056.51, lognorm sigma: 0.75
delez 1: 0.3307, f1: 8.552e-02, normal sigma: 2595.02, lognorm sigma: 0.81
delez 1: 0.3154, f1: 6.018e-02, normal sigma: 3274.55, lognorm sigma: 0.87
delez 1: 0.3057, f1: 4.179e-02, normal sigma: 4132.01, lognorm sigma: 0.92
delez 1: 0.2998, f1: 2.867e-02, normal sigma: 5214.01, lognorm sigma: 0.97
delez 1: 0.2977, f1: 1.943e-02, normal sigma: 6579.33, lognorm sigma: 1.02
delez 1: 0.2973, f1: 1.299e-02, normal sigma: 8302.18, lognorm sigma: 1.07
delez 1: 0.2978, f1: 8.545e-03, normal sigma: 10476.16, lognorm sigma: 1.12
delez 1: 0.2984, f1: 5.491e-03, normal sigma: 13219.41, lognorm sigma: 1.17
delez 1: 0.2992, f1: 3.401e-03, normal sigma: 16681.01, lognorm sigma: 1.21
delez 1: 0.2995, f1: 1.975e-03, normal sigma: 21049.04, lognorm sigma: 1.26
delez 1: 0.3001, f1: 1.009e-03, normal sigma: 26560.88, lognorm sigma: 1.30
delez 1: 0.3008, f1: 3.902e-04, normal sigma: 33516.03, lognorm sigma: 1.34
delez 1: 0.3017, f1: 2.684e-04, normal sigma: 42292.43, lognorm sigma: 1.38
delez 1: 0.3015, f1: 4.985e-04, normal sigma: 53366.99, lognorm sigma: 1.42
delez 1: 0.3014, f1: 6.808e-04, normal sigma: 67341.51, lognorm sigma: 1.46
delez 1: 0.3010, f1: 7.976e-04, normal sigma: 84975.34, lognorm sigma: 1.50
delez 1: 0.3011, f1: 8.630e-04, normal sigma: 107226.72, lognorm sigma: 1.54
delez 1: 0.3004, f1: 8.899e-04, normal sigma: 135304.78, lognorm sigma: 1.57
delez 1: 0.3004, f1: 8.887e-04, normal sigma: 170735.26, lognorm sigma: 1.61
delez 1: 0.3010, f1: 8.670e-04, normal sigma: 215443.47, lognorm sigma: 1.64
delez 1: 0.3009, f1: 8.311e-04, normal sigma: 271858.82, lognorm sigma: 1.68
delez 1: 0.3010, f1: 7.858e-04, normal sigma: 343046.93, lognorm sigma: 1.71
delez 1: 0.3009, f1: 7.351e-04, normal sigma: 432876.13, lognorm sigma: 1.75
delez 1: 0.3007, f1: 6.821e-04, normal sigma: 546227.72, lognorm sigma: 1.78
delez 1: 0.3008, f1: 6.293e-04, normal sigma: 689261.21, lognorm sigma: 1.81
delez 1: 0.3002, f1: 5.790e-04, normal sigma: 869749.00, lognorm sigma: 1.84
delez 1: 0.3008, f1: 5.328e-04, normal sigma: 1097498.77, lognorm sigma: 1.87
delez 1: 0.3012, f1: 4.922e-04, normal sigma: 1384886.37, lognorm sigma: 1.90
delez 1: 0.3020, f1: 4.586e-04, normal sigma: 1747528.40, lognorm sigma: 1.93
delez 1: 0.3017, f1: 4.329e-04, normal sigma: 2205130.74, lognorm sigma: 1.96
delez 1: 0.3015, f1: 4.158e-04, normal sigma: 2782559.40, lognorm sigma: 1.99
delez 1: 0.3014, f1: 4.073e-04, normal sigma: 3511191.73, lognorm sigma: 2.02
delez 1: 0.3010, f1: 4.074e-04, normal sigma: 4430621.46, lognorm sigma: 2.05
delez 1: 0.3007, f1: 4.155e-04, normal sigma: 5590810.18, lognorm sigma: 2.08
delez 1: 0.3007, f1: 4.308e-04, normal sigma: 7054802.31, lognorm sigma: 2.11
delez 1: 0.3005, f1: 4.529e-04, normal sigma: 8902150.85, lognorm sigma: 2.13
delez 1: 0.3010, f1: 4.812e-04, normal sigma: 11233240.33, lognorm sigma: 2.16
delez 1: 0.3008, f1: 5.156e-04, normal sigma: 14174741.63, lognorm sigma: 2.19
delez 1: 0.3008, f1: 5.557e-04, normal sigma: 17886495.29, lognorm sigma: 2.21
delez 1: 0.3006, f1: 6.015e-04, normal sigma: 22570197.20, lognorm sigma: 2.24
delez 1: 0.3008, f1: 6.527e-04, normal sigma: 28480358.68, lognorm sigma: 2.26
delez 1: 0.3011, f1: 7.087e-04, normal sigma: 35938136.64, lognorm sigma: 2.29
delez 1: 0.3010, f1: 7.688e-04, normal sigma: 45348785.08, lognorm sigma: 2.32
delez 1: 0.3009, f1: 8.320e-04, normal sigma: 57223676.59, lognorm sigma: 2.34
delez 1: 0.3013, f1: 8.969e-04, normal sigma: 72208090.18, lognorm sigma: 2.37
delez 1: 0.3015, f1: 9.623e-04, normal sigma: 91116275.61, lognorm sigma: 2.39
```

```
delez 1: 0.3007, f1: 1.026e-03, normal sigma: 114975699.54, lognorm sigma: 2.41
delez 1: 0.3009, f1: 1.088e-03, normal sigma: 145082877.85, lognorm sigma: 2.44
delez 1: 0.3012, f1: 1.145e-03, normal sigma: 183073828.03, lognorm sigma: 2.46
delez 1: 0.3018, f1: 1.196e-03, normal sigma: 231012970.01, lognorm sigma: 2.49
delez 1: 0.3021, f1: 1.240e-03, normal sigma: 291505306.28, lognorm sigma: 2.51
delez 1: 0.3021, f1: 1.275e-03, normal sigma: 367837977.18, lognorm sigma: 2.53
delez 1: 0.3021, f1: 1.300e-03, normal sigma: 464158883.36, lognorm sigma: 2.55
delez 1: 0.3013, f1: 1.315e-03, normal sigma: 585702081.81, lognorm sigma: 2.58
delez 1: 0.3013, f1: 1.319e-03, normal sigma: 739072203.35, lognorm sigma: 2.60
delez 1: 0.3001, f1: 1.312e-03, normal sigma: 932603346.88, lognorm sigma: 2.62
delez 1: 0.3003, f1: 1.293e-03, normal sigma: 1176811952.43, lognorm sigma: 2.64
delez 1: 0.3001, f1: 1.263e-03, normal sigma: 1484968262.25, lognorm sigma: 2.67
delez 1: 0.3001, f1: 1.222e-03, normal sigma: 1873817422.86, lognorm sigma: 2.69
delez 1: 0.3003, f1: 1.171e-03, normal sigma: 2364489412.65, lognorm sigma: 2.71
delez 1: 0.3011, f1: 1.112e-03, normal sigma: 2983647240.28, lognorm sigma: 2.73
delez 1: 0.3010, f1: 1.044e-03, normal sigma: 3764935806.79, lognorm sigma: 2.75
delez 1: 0.3011, f1: 9.685e-04, normal sigma: 4750810162.10, lognorm sigma: 2.77
delez 1: 0.3007, f1: 8.877e-04, normal sigma: 5994842503.19, lognorm sigma: 2.79
delez 1: 0.3010, f1: 8.025e-04, normal sigma: 7564633275.55, lognorm sigma: 2.81
delez 1: 0.3010, f1: 7.142e-04, normal sigma: 9545484566.62, lognorm sigma: 2.83
delez 1: 0.3012, f1: 6.246e-04, normal sigma: 12045035402.59, lognorm sigma:
2.86
delez 1: 0.3013, f1: 5.351e-04, normal sigma: 15199110829.53, lognorm sigma:
2.88
delez 1: 0.3012, f1: 4.477e-04, normal sigma: 19179102616.72, lognorm sigma:
2.90
delez 1: 0.3013, f1: 3.650e-04, normal sigma: 24201282647.94, lognorm sigma:
2.92
delez 1: 0.3013, f1: 2.909e-04, normal sigma: 30538555088.33, lognorm sigma:
2.94
delez 1: 0.3012, f1: 2.323e-04, normal sigma: 38535285937.11, lognorm sigma:
2.96
delez 1: 0.3009, f1: 1.997e-04, normal sigma: 48626015800.65, lognorm sigma:
2.97
delez 1: 0.3007, f1: 2.018e-04, normal sigma: 61359072734.13, lognorm sigma:
2.99
delez 1: 0.3004, f1: 2.336e-04, normal sigma: 77426368268.11, lognorm sigma:
3.01
delez 1: 0.3009, f1: 2.817e-04, normal sigma: 97700995729.92, lognorm sigma:
3.03
delez 1: 0.3015, f1: 3.361e-04, normal sigma: 123284673944.21, lognorm sigma:
3.05
delez 1: 0.3016, f1: 3.919e-04, normal sigma: 155567614393.05, lognorm sigma:
3.07
delez 1: 0.3013, f1: 4.464e-04, normal sigma: 196304065004.03, lognorm sigma:
3.09
delez 1: 0.3010, f1: 4.985e-04, normal sigma: 247707635599.17, lognorm sigma:
3.11
```

```
delez 1: 0.3015, f1: 5.476e-04, normal sigma: 312571584968.82, lognorm sigma:
3.13
delez 1: 0.3013, f1: 5.935e-04, normal sigma: 394420605943.76, lognorm sigma:
3.15
delez 1: 0.3010, f1: 6.363e-04, normal sigma: 497702356433.21, lognorm sigma:
3.16
delez 1: 0.3014, f1: 6.760e-04, normal sigma: 628029144183.42, lognorm sigma:
3.18
delez 1: 0.3009, f1: 7.130e-04, normal sigma: 792482898353.92, lognorm sigma:
3.20
delez 1: 0.3011, f1: 7.472e-04, normal sigma: 1000000000000.00, lognorm sigma:
3.22
```

(Uniform distribution characterization). A sequence of real numbers (respectively, a Borel measurable function, a random variable, a Borel probability measure) is Benford if and only if the decimal logarithm of its absolute value is uniformly distributed modulo 1.

[8]:
```
sigmas = np.sqrt(np.log10(1 + sigmas / mu**2))
```

[9]:
```
lognorm_fracs = []

for dist in lognorm_dists:
    # lognorm_fracs.append(get_number_fracs_math(np.log10(dist[dist > 0])))
    lognorm_fracs.append(np.log10(dist) % 1)
```

[10]:
```
lognorm_fracs_hists = []

for i, fracs in enumerate(lognorm_fracs):
    n, bins, _ = plt.hist(fracs, bins=30, density=False, histtype='step')
    lognorm_fracs_hists.append([n, bins[1:]])

    plt.close()
```

1 Test χ^2

[11]:
```
# chi2_test(np.array(lognorm_fracs), n_bins=30)
```

[12]:
```
from scipy.stats import chisquare
from scipy.stats import chi2
```

[13]:
```
chi2_test = []

N, n_bins = np.sum(lognorm_fracs_hists[0][0]), len(lognorm_fracs_hists[0][1])

for hist in lognorm_fracs_hists:
    chi2_ = chisquare(hist[0], f_exp=N/n_bins)
    chi2_test.append([chi2_.statistic, chi2_.pvalue])
```

[14]:
```
for c in chi2_test:
    print(f'{c[0]:.3f}, {c[1]:.3f}')
```

```
5445502.694, 0.000
4201962.289, 0.000
3207930.557, 0.000
2417048.101, 0.000
1800594.890, 0.000
1320276.298, 0.000
948841.878, 0.000
663092.915, 0.000
445008.712, 0.000
283916.566, 0.000
171555.056, 0.000
98570.562, 0.000
54121.416, 0.000
28610.910, 0.000
14648.218, 0.000
7242.391, 0.000
3508.612, 0.000
1675.201, 0.000
794.073, 0.000
381.696, 0.000
174.293, 0.000
95.251, 0.000
56.577, 0.002
49.820, 0.009
32.397, 0.303
52.016, 0.005
34.217, 0.231
28.004, 0.518
42.637, 0.049
25.386, 0.658
28.536, 0.489
32.745, 0.288
31.983, 0.321
37.829, 0.126
47.098, 0.018
41.112, 0.067
32.430, 0.301
34.468, 0.223
21.481, 0.841
46.234, 0.022
40.160, 0.081
49.189, 0.011
37.330, 0.138
27.647, 0.537
```

33.843, 0.245
29.176, 0.456
31.339, 0.350
27.141, 0.564
40.260, 0.080
26.219, 0.614
25.656, 0.644
38.929, 0.103
25.860, 0.633
27.570, 0.541
32.804, 0.286
34.197, 0.232
51.092, 0.007
35.972, 0.174
25.474, 0.653
36.920, 0.148
42.216, 0.054
38.056, 0.121
35.618, 0.185
28.619, 0.485
45.330, 0.027
39.287, 0.096
52.800, 0.004
44.134, 0.036
45.431, 0.027
46.371, 0.022
47.700, 0.016
57.100, 0.001
39.289, 0.096
39.671, 0.089
30.650, 0.382
30.227, 0.403
27.182, 0.562
36.655, 0.155
34.635, 0.217
32.177, 0.312
25.302, 0.662
34.494, 0.222
34.967, 0.206
32.174, 0.312
30.365, 0.396
33.923, 0.242
36.046, 0.172
19.678, 0.903
19.631, 0.904
29.615, 0.433
26.657, 0.590
33.142, 0.272

```
37.524, 0.133
50.174, 0.009
49.563, 0.010
27.185, 0.562
29.716, 0.428
24.375, 0.710
46.138, 0.023
30.857, 0.372
```

[15]:
```
alpha = 0.01 # stopnja pomembnosti

#                          stopnja zaupanja
critical_value_chi2 = chi2.ppf(1 - alpha, n_bins)
critical_value_chi2
```

[15]:
```
50.89218131151707
```

2 Test Kolmogorov-Smirnova

[16]:
```
ks_test(np.array(lognorm_fracs))
```

[16]:
```
(array([[4.27647627e-001, 0.00000000e+000],
        [4.12863214e-001, 0.00000000e+000],
        [3.95750415e-001, 0.00000000e+000],
        [3.76083786e-001, 0.00000000e+000],
        [3.53442606e-001, 0.00000000e+000],
        [3.27730384e-001, 0.00000000e+000],
        [2.99392307e-001, 0.00000000e+000],
        [2.68281036e-001, 0.00000000e+000],
        [2.36003538e-001, 0.00000000e+000],
        [2.01724347e-001, 0.00000000e+000],
        [1.67375939e-001, 0.00000000e+000],
        [1.34319390e-001, 0.00000000e+000],
        [1.03688112e-001, 0.00000000e+000],
        [7.66349102e-002, 0.00000000e+000],
        [5.43322610e-002, 0.00000000e+000],
        [3.66844702e-002, 0.00000000e+000],
        [2.34325619e-002, 0.00000000e+000],
        [1.38814678e-002, 8.26037701e-168],
        [7.85264784e-003, 5.46130671e-054],
        [4.36224377e-003, 5.90436355e-017],
        [3.41470867e-003, 1.48616029e-010],
        [3.01190645e-003, 2.63432380e-008],
        [2.13730047e-003, 2.15103434e-004],
        [1.70640438e-003, 5.90695544e-003],
        [1.28047034e-003, 7.52477854e-002],
```

```
[7.53866941e-004, 6.20380797e-001],
[9.64710748e-004, 3.09565203e-001],
[1.06724679e-003, 2.04605714e-001],
[7.69549439e-004, 5.94102102e-001],
[5.62185952e-004, 9.09877341e-001],
[7.83946809e-004, 5.70193011e-001],
[1.02269533e-003, 2.46294833e-001],
[9.13742392e-004, 3.73814857e-001],
[1.03190945e-003, 2.37190224e-001],
[7.79679821e-004, 5.77252644e-001],
[1.06901042e-003, 2.03074412e-001],
[9.29362921e-004, 3.53269993e-001],
[1.28699522e-003, 7.27684466e-002],
[8.51206331e-004, 4.63235828e-001],
[8.85191585e-004, 4.13261174e-001],
[8.21507453e-004, 5.09314894e-001],
[7.45528631e-004, 6.34417805e-001],
[9.67592596e-004, 3.06173914e-001],
[1.14535074e-003, 1.44908311e-001],
[9.15331272e-004, 3.71691085e-001],
[7.98210018e-004, 5.46785939e-001],
[5.94710852e-004, 8.70992100e-001],
[7.80684613e-004, 5.75588090e-001],
[7.30195677e-004, 6.60275523e-001],
[7.13839986e-004, 6.87809744e-001],
[5.94551501e-004, 8.71198420e-001],
[5.23498924e-004, 9.46763613e-001],
[7.10020619e-004, 6.94216426e-001],
[8.77858425e-004, 4.23775796e-001],
[8.89463835e-004, 4.07206267e-001],
[8.33730821e-004, 4.90100696e-001],
[8.73992182e-004, 4.29380390e-001],
[1.03989612e-003, 2.29504396e-001],
[8.32180114e-004, 4.92519745e-001],
[6.84492962e-004, 7.36597923e-001],
[8.28253183e-004, 4.98670094e-001],
[8.18047960e-004, 5.14811723e-001],
[7.75976417e-004, 5.83398629e-001],
[1.00383416e-003, 2.65735687e-001],
[1.10320387e-003, 1.75103396e-001],
[1.33330723e-003, 5.70868257e-002],
[1.65095778e-003, 8.57178979e-003],
[1.20934989e-003, 1.07220499e-001],
[1.05602407e-003, 2.14558992e-001],
[9.75213375e-004, 2.97331053e-001],
[1.04774804e-003, 2.22132784e-001],
[1.02813996e-003, 2.40884011e-001],
```

```
        [1.31455472e-003, 6.30473454e-002],
        [1.29017495e-003, 7.15855481e-002],
        [9.06877552e-004, 3.83078381e-001],
        [9.36275132e-004, 3.44418019e-001],
        [7.36486807e-004, 6.49664138e-001],
        [9.78444090e-004, 2.93637004e-001],
        [9.16782687e-004, 3.69757750e-001],
        [9.06993820e-004, 3.82920307e-001],
        [7.30094605e-004, 6.60445969e-001],
        [6.57112244e-004, 7.80653084e-001],
        [8.05120192e-004, 5.35565181e-001],
        [7.27976401e-004, 6.64017641e-001],
        [6.24211322e-004, 8.30454190e-001],
        [5.60533071e-004, 9.11669898e-001],
        [5.10566364e-004, 9.56662094e-001],
        [6.08871872e-004, 8.52093225e-001],
        [7.88792884e-004, 5.62205645e-001],
        [5.67977022e-004, 9.03451099e-001],
        [6.78458704e-004, 7.46459894e-001],
        [8.13875386e-004, 5.21474327e-001],
        [6.77807202e-004, 7.47520131e-001],
        [8.41567782e-004, 4.77961389e-001],
        [7.65628400e-004, 6.00653179e-001],
        [8.65908505e-004, 4.41232112e-001],
        [7.16143034e-004, 6.83941144e-001],
        [8.87857572e-004, 4.09476595e-001],
        [7.72785164e-004, 5.88707672e-001],
        [7.32287734e-004, 6.56747141e-001]]),
[0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
 0.0016274566551782183,
```

0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,

```
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183,
0.0016274566551782183])
```

[17]:
```
from scipy.stats import kstest
from scipy.stats import kstwo
```

[18]:
```
ks_test = []

for dist in lognorm_fracs:
    ks = kstest(dist, cdf='uniform', alternative='two-sided', args=(0, 1))
    stat, p = ks.statistic, ks.pvalue
    ks_test.append([stat, p])
    print(f'{stat:.2e}, {p:.3f}')
```

```
4.28e-01, 0.000
```

```
4.13e-01, 0.000
3.96e-01, 0.000
3.76e-01, 0.000
3.53e-01, 0.000
3.28e-01, 0.000
2.99e-01, 0.000
2.68e-01, 0.000
2.36e-01, 0.000
2.02e-01, 0.000
1.67e-01, 0.000
1.34e-01, 0.000
1.04e-01, 0.000
7.66e-02, 0.000
5.43e-02, 0.000
3.67e-02, 0.000
2.34e-02, 0.000
1.39e-02, 0.000
7.85e-03, 0.000
4.36e-03, 0.000
3.41e-03, 0.000
3.01e-03, 0.000
2.14e-03, 0.000
1.71e-03, 0.006
1.28e-03, 0.075
7.54e-04, 0.620
9.65e-04, 0.310
1.07e-03, 0.205
7.70e-04, 0.594
5.62e-04, 0.910
7.84e-04, 0.570
1.02e-03, 0.246
9.14e-04, 0.374
1.03e-03, 0.237
7.80e-04, 0.577
1.07e-03, 0.203
9.29e-04, 0.353
1.29e-03, 0.073
8.51e-04, 0.463
8.85e-04, 0.413
8.22e-04, 0.509
7.46e-04, 0.634
9.68e-04, 0.306
1.15e-03, 0.145
9.15e-04, 0.372
7.98e-04, 0.547
5.95e-04, 0.871
7.81e-04, 0.576
7.30e-04, 0.660
```

7.14e-04, 0.688
5.95e-04, 0.871
5.23e-04, 0.947
7.10e-04, 0.694
8.78e-04, 0.424
8.89e-04, 0.407
8.34e-04, 0.490
8.74e-04, 0.429
1.04e-03, 0.230
8.32e-04, 0.493
6.84e-04, 0.737
8.28e-04, 0.499
8.18e-04, 0.515
7.76e-04, 0.583
1.00e-03, 0.266
1.10e-03, 0.175
1.33e-03, 0.057
1.65e-03, 0.009
1.21e-03, 0.107
1.06e-03, 0.215
9.75e-04, 0.297
1.05e-03, 0.222
1.03e-03, 0.241
1.31e-03, 0.063
1.29e-03, 0.072
9.07e-04, 0.383
9.36e-04, 0.344
7.36e-04, 0.650
9.78e-04, 0.294
9.17e-04, 0.370
9.07e-04, 0.383
7.30e-04, 0.660
6.57e-04, 0.781
8.05e-04, 0.536
7.28e-04, 0.664
6.24e-04, 0.830
5.61e-04, 0.912
5.11e-04, 0.957
6.09e-04, 0.852
7.89e-04, 0.562
5.68e-04, 0.903
6.78e-04, 0.746
8.14e-04, 0.521
6.78e-04, 0.748
8.42e-04, 0.478
7.66e-04, 0.601
8.66e-04, 0.441
7.16e-04, 0.684

```
8.88e-04, 0.409
7.73e-04, 0.589
7.32e-04, 0.657
```

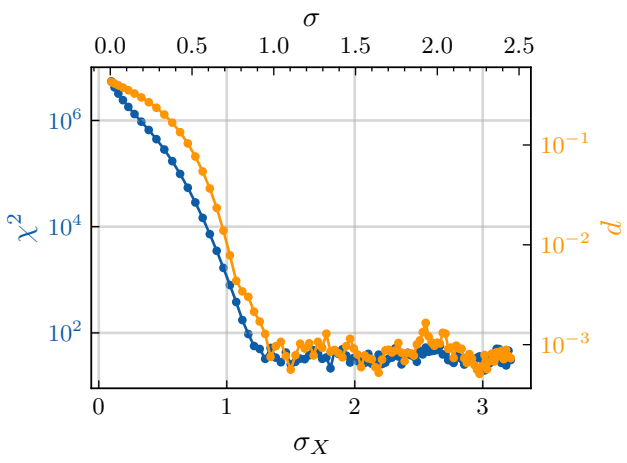
[19]:
```
critical_value_ks = kstwo.ppf(1 - alpha, len(lognorm_fracs[0]))
f'{critical_value_ks:.2e}'
```

[19]:
```
'1.63e-03'
```

[20]:
```
if False:
    fig, axs = set_size_decorator(plt.subplots, fraction=1, ratio='4:3')(2, 2)
    axs = axs.flatten()

    for i, fracs in enumerate(lognorm_fracs):
        axs[i].hist(fracs, bins=30, density=False, histtype='step')
        lognorm_fracs_hists.append([n, bins[1:]])

        # plt.close()

        axs[i].ticklabel_format(style='sci', axis='y', scilimits=(0, 0))

        an1 = f'$\sigma$ = {sigmas[i]:.2f}, $\sigma_X$ = {lognorm_sigmas[i]:.
↪2f}'
        an2 = f'\n$\chi^2$ = {chi2_test[i][0]:.2f}, $d$ = {ks_test[i][0]:.2e}'
        an3 = f'\n$\chi^2_*$ = {critical_value_chi2:.2f}, $d_*$ =␣
↪{critical_value_ks:.2e}'
        an4 = r' pri $\alpha$ = {}'.format(alpha)

        if i == 3:
            an = an1 + an2 + an3 + an4
        else:
            an = an1 + an2

        axs[i].annotate(an, xy=(0.1, 0.1), xycoords='axes fraction', fontsize=8)

    savefig('lognorm_uniform_hists')
```

[21]:
```
fig, ax = set_size_decorator(plt.subplots, fraction=0.5, ratio='4:3')(1, 1)

ax.set_yscale('log')

x = lognorm_sigmas

y = [i[0] for i in chi2_test]
ax.scatter(x, y, s=5)
ax.plot(x, y, lw=1)
```

```
ax.set_xlabel('$\sigma_X$')

ax.set_ylabel('$\chi^2$', c='C0')
ax.tick_params(axis='y', labelcolor='C0')

ax.set_ylim([0.9e1, 1e7])

ax2 = ax.twinx()

ax2.set_yscale('log')
ax2.minorticks_off()
ax2 = thiner_border(ax2)

y2 = [i[0] for i in ks_test]
ax2.scatter(x, y2, s=5, c='C2')
ax2.plot(x, y2, lw=1, c='C2')

ax2.set_ylabel('$d$', c='C2')
ax2.tick_params(axis='y', labelcolor='C2')

x3 = sigmas

ax3 = ax.twiny()
ax3 = thiner_border(ax3)
# ax3.set_xscale('log')

ax3.plot(x3, y, lw=0)

ax3.set_xlabel('$\sigma$')

ax.grid(zorder=0, alpha=0.5)

# savefig('stat_lognorm_tests', tight_layout=False)
```

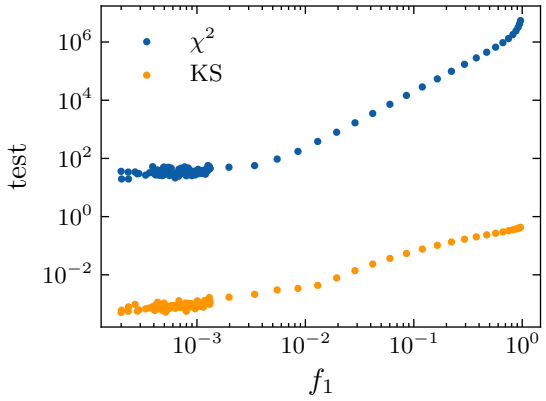


[22]:
```
chi2_lst = np.array([i[0] for i in chi2_test])
ks_lst = np.array([i[0] for i in ks_test])
```

[23]:
```
f1s = np.array(f1s)
```

[24]:
```
idx = np.argsort(f1s)

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

ax.set_xscale('log')
ax.set_yscale('log')
ax.scatter(f1s[idx], chi2_lst[idx], s=3, label='$\chi^2$')
ax.scatter(f1s[idx], ks_lst[idx], s=3, label='KS', c='C2')

ax.set_ylabel('test')
ax.set_xlabel('$f_1$')

ax.legend()

# savefig('test_lognorm_f1_chi2_KS')
```

[24]:
```
<matplotlib.legend.Legend at 0x7f8d539fbc10>
```



[ ]:
```

```