dielectron_rng

August 21, 2021

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

[2]:
```
import sys

import matplotlib.pyplot as plt

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'
```

[3]:
```
from NIST_tests import RNG_test
from random_helper_functions import get_bitstring, binary_tree_walk
from benford_helper_functions import get_first_digit, benfords_test, normalize
from stat_tests import chi2_test, ks_test
```

https://www.kaggle.com/fedesoriano/cern-electron-collision-data

1 Load data

[4]:
```
df = pd.read_csv('dielectron_data/dielectron.csv')
```

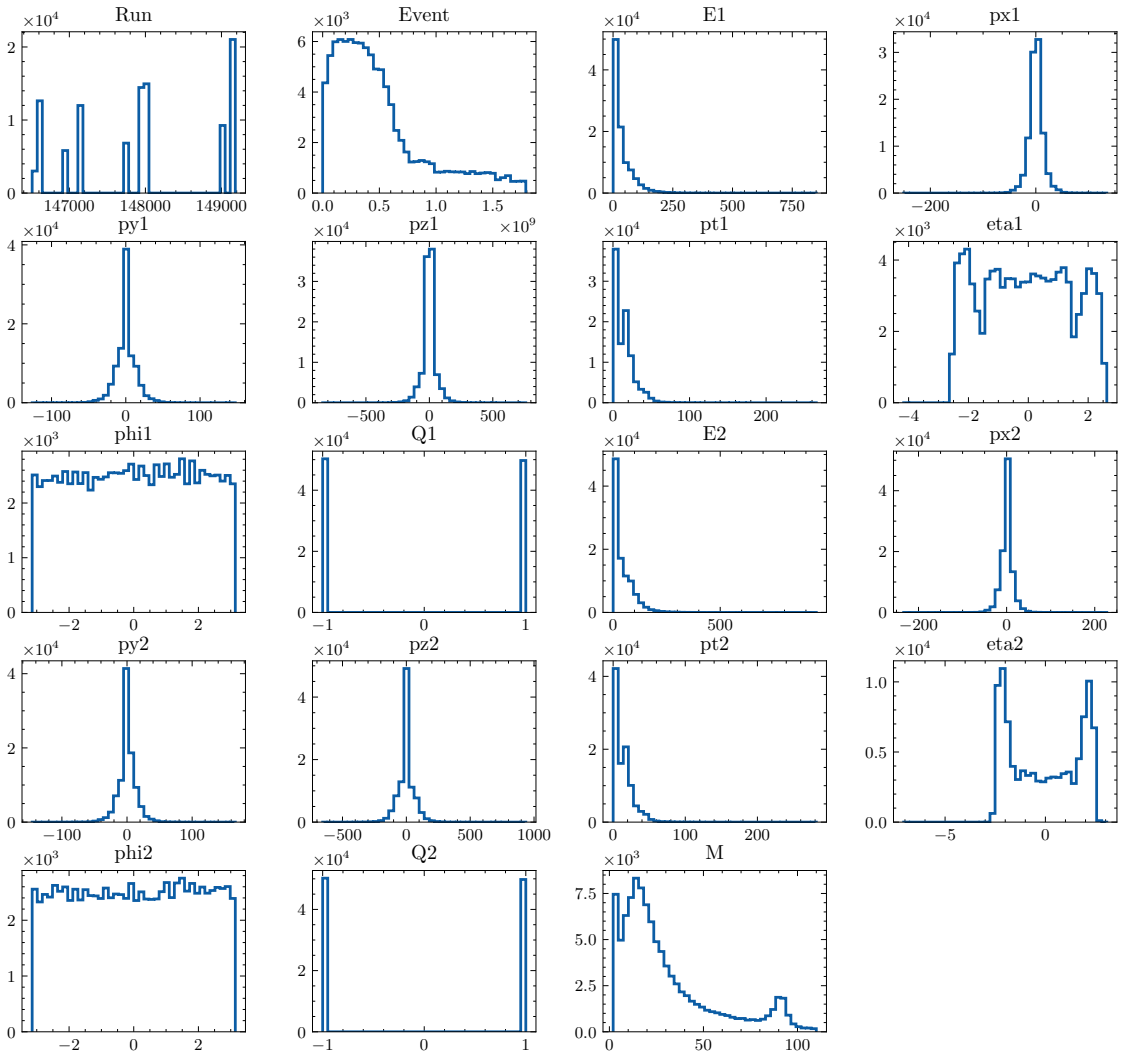
[5]:
```
df.head()
```

[5]:

	Run	Event	E1	px1	py1	pz1	pt1	\
0	147115	366639895	58.71410	-7.31132	10.531000	-57.29740	12.82020	
1	147115	366704169	6.61188	-4.15213	-0.579855	-5.11278	4.19242	
2	147115	367112316	25.54190	-11.48090	2.041680	22.72460	11.66100	
3	147115	366952149	65.39590	7.51214	11.887100	63.86620	14.06190	
4	147115	366523212	61.45040	2.95284	-14.622700	-59.61210	14.91790	

```
      eta1     phi1  Q1       E2        px2      py2      pz2       pt2  \
0 -2.20267  2.17766   1  11.2836  -1.032340 -1.88066 -11.0778   2.14537
1 -1.02842 -3.00284  -1  17.1492 -11.713500  5.04474  11.4647  12.75360
2  1.42048  2.96560   1  15.8203  -1.472800  2.25895 -15.5888   2.69667
3  2.21838  1.00721   1  25.1273   4.087860  2.59641  24.6563   4.84272
4 -2.09375 -1.37154  -1  13.8871  -0.277757 -2.42560 -13.6708   2.44145

       eta2      phi2  Q2         M
0 -2.344030 -2.072810  -1   8.94841
1  0.808077  2.734920   1  15.89300
2 -2.455080  2.148570   1  38.38770
3  2.330210  0.565865  -1   3.72862
4 -2.423700 -1.684810  -1   2.74718
```

[6]:
```
fig, axs = set_size_decorator(plt.subplots, fraction=1.8, ratio='4:3')(5, 4)

axs[-1, -1].set_visible(False)

axs = df.hist(bins=40, histtype='step', ax=axs.flatten()[:-1], lw=1.5)
axs = [thiner_border(ax) for ax in axs]

for ax in axs:
    ax.grid(False)
    ax.ticklabel_format(style='sci', axis='y', scilimits=(0, 0))

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



2 Naboji

[7]:
```
df['Q1'][df['Q1'] == -1] = 0
df['Q2'][df['Q2'] == -1] = 0
```

[8]:
```
Q1 = df['Q1'].values
Q2 = df['Q2'].values

q1 = Q1.astype(str)
q2 = Q2.astype(str)
q1 = ''.join(q1)
q2 = ''.join(q2)
```

[9]:
```
q = np.concatenate((df['Q1'].values, df['Q2'].values)).astype(str)
q = ''.join(q)
```

[10]:
```
Q = np.vstack((Q1, Q2)).T.flatten().astype(str)
Q = ''.join(Q)
```

[11]:
```
t1 = RNG_test(q1, short_df=True)
t2 = RNG_test(q2, short_df=True)
t3 = RNG_test(q, short_df=True)
t4 = RNG_test(Q, short_df=True)

test_q_df = pd.concat((t1, t2, t3, t4))
```

```
100%|
                    | 16/16 [00:01<00:00, 14.24it/s]
100%|
                    | 16/16 [00:00<00:00, 21.19it/s]
100%|
                    | 16/16 [00:01<00:00, 11.18it/s]
100%|
                    | 16/16 [00:01<00:00, 11.19it/s]
```

[12]:
```
test_q_df.columns = [i for i in range(1, 15+1)]
test_q_df.index = [r'$p_{Q_1}$', r'$p_{Q_2}$', r'$p_{Q_1 Q_2}$', r'$p_{Q_1,␣
↪Q_2}$']
```

[13]:
```
test_q_df
```

[13]:

	1	2	3	4	5	6	7	8	9	10	\
p_{Q_1}	0.08	0.38	0.08	0.29	0.58	0.18	0.84	0.22	nan	0.10	
p_{Q_2}	0.18	0.17	0.64	0.26	0.33	0.06	0.05	0.41	nan	0.74	
$p_{Q_1 Q_2}$	0.03	0.09	0.12	0.29	0.12	0.00	0.62	0.61	nan	0.22	
p_{Q_1, Q_2}	0.03	1.00	0.00	0.00	0.70	0.00	0.00	0.00	nan	0.75	

	11	12	13	14	15
p_{Q_1}	0.38	0.16	0.07	0.39	0.75
p_{Q_2}	0.65	0.10	0.11	0.72	0.83
$p_{Q_1 Q_2}$	0.60	0.54	0.02	0.39	0.75
p_{Q_1, Q_2}	0.27	0.00	0.02	0.80	0.48

[14]:
```
np.unique(Q1 == Q2, return_counts=True)
```

[14]:
```
(array([False,  True]), array([57053, 42947]))
```

[15]:
```
np.unique(Q1, return_counts=True)
```

[15]:
```
(array([0, 1]), array([50274, 49726]))
```

[16]:
```
np.unique(Q2, return_counts=True)
```

[16]:
```
(array([0, 1]), array([50211, 49789]))
```

[17]:
```
np.unique(df['Q1'].values, return_counts=True)
```

[17]:
```
(array([0, 1]), array([50274, 49726]))
```

[18]:
```
np.unique(df['Q2'].values, return_counts=True)
```

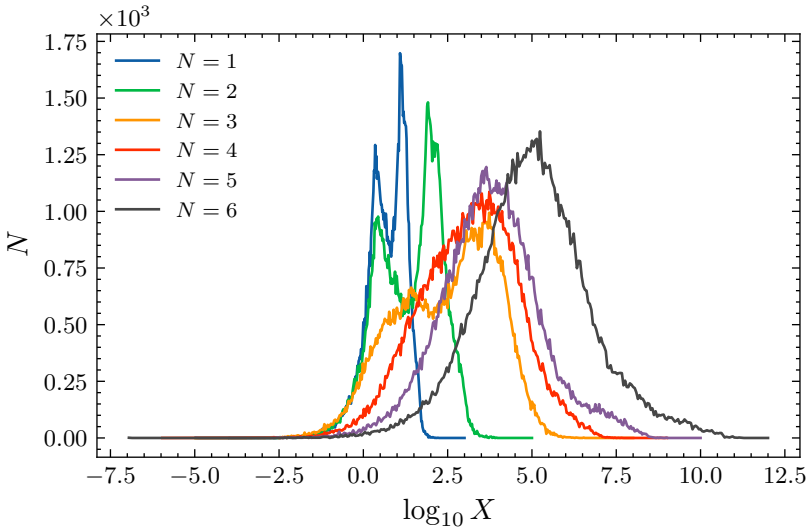
[18]:
```
(array([0, 1]), array([50211, 49789]))
```

3 Mnozenje

[19]:
```
ps = [df['px1 '].values, df['py1'].values, df['pz1'].values, df['px2'].values,␣
 ↪df['py2'].values, df['pz2'].values]
ps = np.abs(np.array(ps))

r = np.arange(0, len(ps))

dists = []

for i in r:
    m = np.prod(ps[:i+1], axis=0)
    dists.append(m)
```

[20]:
```
fig, ax = set_size_decorator(plt.subplots, fraction=0.8, ratio='golden')(1, 1)

for i, lognorm in enumerate(dists):
    bins = np.logspace(np.floor(np.log10(lognorm.min())),
                       np.floor(np.log10(lognorm.max())) + 1,
                       400)

    n, bins = np.histogram(lognorm, bins=bins)
    bins = bins[1:]

    ax.plot(np.log10(bins), n, lw=1, label=f'$N={i+1}$')

ax.legend(fontsize=8, loc='upper left')
ax.ticklabel_format(style='sci', axis='y', scilimits=(0, 0))

ax.set_xlabel(r'$\log_{10} X$')
ax.set_ylabel('$N$')

# savefig('dielectron_lognorm')
```

[20]: Text(0, 0.5, 'N')



[21]:
```
from benford_helper_functions import benford_ft
```

[22]:
```
f1s = []
first_digits = []
fracs = []
rng_tests = []
chi2_tests, ks_tests = [], []

alpha = 0.01

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

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

    # f1 = benfords_test(n, bins)
    freq, SF, sf, PDF, OST, ost = benford_ft(pdf, bins, shift=True)
    ind = np.argsort(np.abs(SF))
    f1 = np.abs(PDF)[ind[1]]

    f1s.append(f1)
```

```
    first_digit = get_first_digit(lognorm)
    _, c = np.unique(first_digit, return_counts=True)
    c = c / np.sum(c)
    first_digits.append(c)

    frac = np.log10(lognorm) % 1
    fracs.append(frac)

    chi2_tests.append(chi2_test(frac, n_bins=int(np.sqrt(len(frac))),␣
↪alpha=alpha))
    ks_tests.append(ks_test(frac, alpha=alpha))

    # bits = binary_tree_walk(frac).astype(str)
    bits = get_bitstring(frac, length=32)
    bits = ''.join(bits)
    test = RNG_test(bits, short_df=True)
    rng_tests.append(test)

df = pd.concat([i for i in rng_tests])
df.index = [f'$p_{i}$' for i in range(1, len(df)+1)]
df.columns = [i + 1 for i in range(len(df.columns))]
```

```
100%|
                    | 16/16 [00:16<00:00,  1.01s/it]
100%|
                    | 16/16 [00:16<00:00,  1.00s/it]
100%|
                    | 16/16 [00:16<00:00,  1.03s/it]
100%|
                    | 16/16 [00:16<00:00,  1.06s/it]
100%|
                    | 16/16 [00:16<00:00,  1.05s/it]
100%|
                    | 16/16 [00:16<00:00,  1.03s/it]
```

[23]:
```
df
```

[23]:

	1	2	3	4	5	6	7	8	9	10	11	12	\
p_1	0.00	0.06	0.47	0.81	0.60	0.52	0.34	0.37	0.90	0.24	0.00	0.08	
p_2	0.00	0.51	0.00	0.49	0.60	0.24	0.21	0.06	0.06	0.57	0.91	0.79	
p_3	0.29	0.93	0.53	0.77	0.86	0.44	0.56	0.13	0.67	0.02	0.42	0.18	
p_4	0.79	0.35	0.69	0.92	0.91	0.10	0.61	0.50	0.16	0.51	0.25	0.05	
p_5	0.86	0.68	0.13	0.90	0.18	0.03	0.20	0.89	0.99	0.25	0.57	0.78	
p_6	0.19	0.72	0.87	0.91	0.15	0.01	0.91	0.66	0.64	0.83	0.30	0.07	

	13	14	15
p_1	0.00	0.21	0.07

```
$p_2$  0.00  0.64  0.12
$p_3$  0.18  0.26  0.08
$p_4$  0.82  0.99  0.57
$p_5$  0.96  0.28  0.90
$p_6$  0.24  0.06  0.48
```

[24]:
```
dct = {r'$n_1$': [f'{i[0]:.4f}' for i in first_digits],
       r'$\Delta n_1$': [f'{abs(i[0] - np.log10(2)):.4f}' for i in␣
↪first_digits],
       r'$f_1$': [f'{i:.5f}' for i in f1s],
       r'$\chi^2$': [f'{i[0][0][0]:.2f}' for i in chi2_tests],
       r'$d$': [f'{i[0][0][0]:.4f}' for i in ks_tests],
       r'$p_{\chi^2}$': [f'{i[0][0][1]:.4f}' for i in chi2_tests],
       r'$p_d$': [f'{i[0][0][1]:.4f}' for i in ks_tests]}
```

[25]:
```
test_df = pd.DataFrame(dct)
test_df.index = [f'$N={i}$' for i in range(1, len(df)+1)]
test_df.sort_values(by=['$\chi^2$'], inplace=True)
```

[ ]:
```
```