# lab1

April 16, 2022

## 1 Statistisc, lab 1

[1]:
```
import numpy as np
import seaborn as sns
import scipy as sp
from  matplotlib import pyplot as plt
from math import sqrt, gamma
from statsmodels.distributions.empirical_distribution import ECDF
```

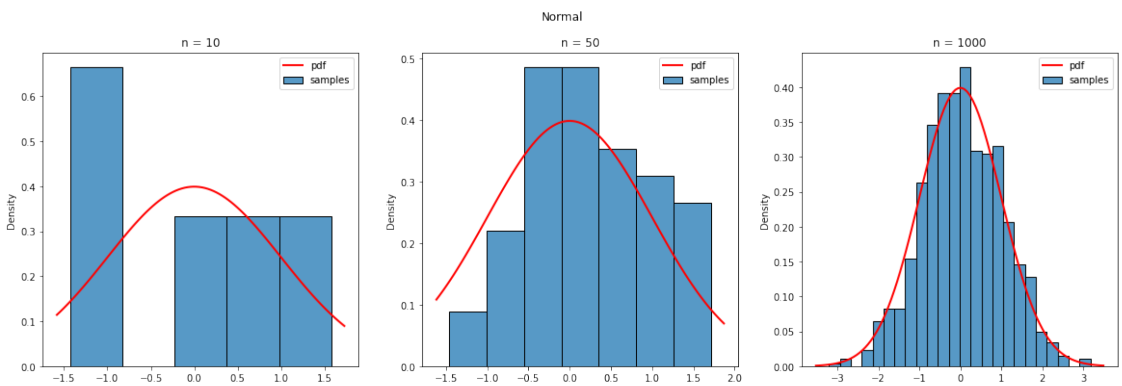
[2]:
```
dists = {'Normal': {
            'gen_data': np.random.standard_normal,
            'cdf': sp.stats.norm.cdf,
            'pdf': sp.stats.norm.pdf
        },
         'Cauchy': {
             'gen_data': np.random.standard_cauchy,
             'cdf': sp.stats.cauchy.cdf,
             'pdf':sp.stats.cauchy.pdf
         },
         'Laplace': {
             'gen_data': lambda n: np.random.laplace(0, 1 / sqrt(2), n),
             'cdf': lambda x: sp.stats.laplace.cdf(x, 0, 1 / sqrt(2)),
             'pdf': lambda x: sp.stats.laplace.pdf(x, 0, 1 / sqrt(2))
         },
         'Poisson': {
             'gen_data': lambda n: np.random.poisson(10, n),
             'cdf': lambda x: sp.stats.poisson.cdf(x, 10),
             'pdf': lambda x: 10 ** x * np.exp(-10) / gamma(x + 1)
         },
         'Uniform': {
             'gen_data': lambda n: np.random.uniform(-sqrt(3), sqrt(3), n),
             'cdf': lambda x: sp.stats.uniform.cdf(x, -sqrt(3), 2 * sqrt(3)),
             'pdf': lambda x: sp.stats.uniform.pdf(x, -sqrt(3), 2 * sqrt(3))
        }}
```

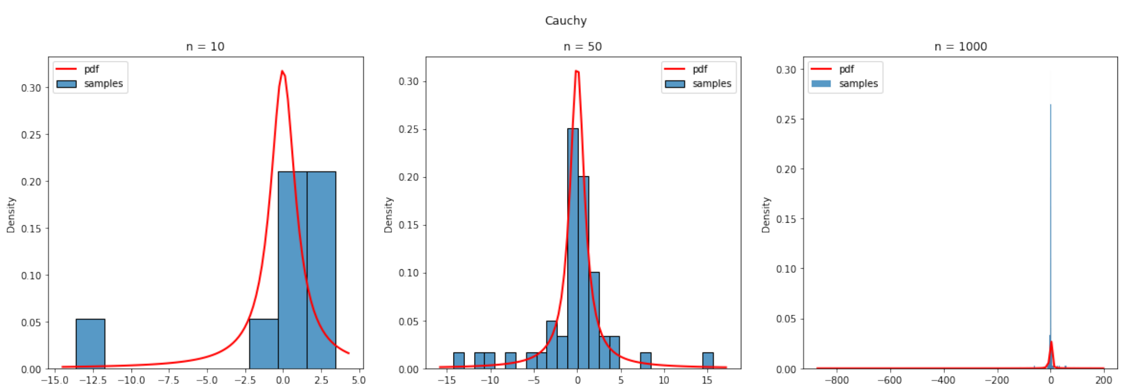
## 1.1 Task 1

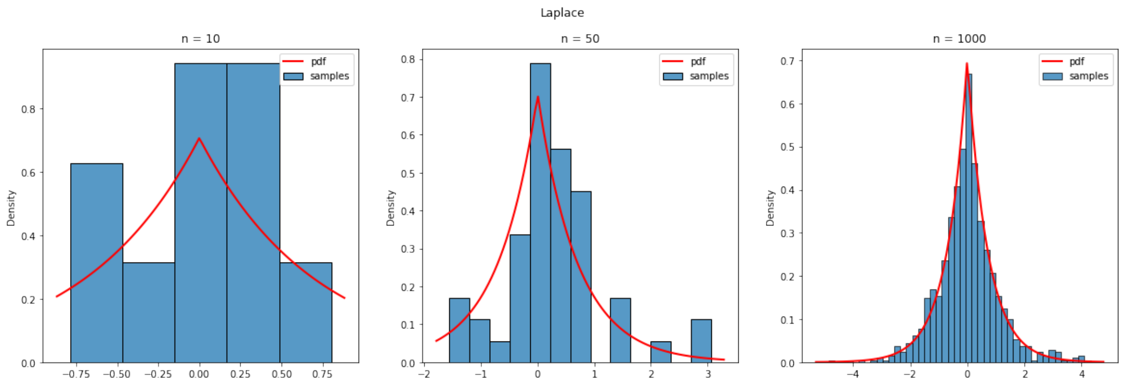
### 1.1.1 Histograms

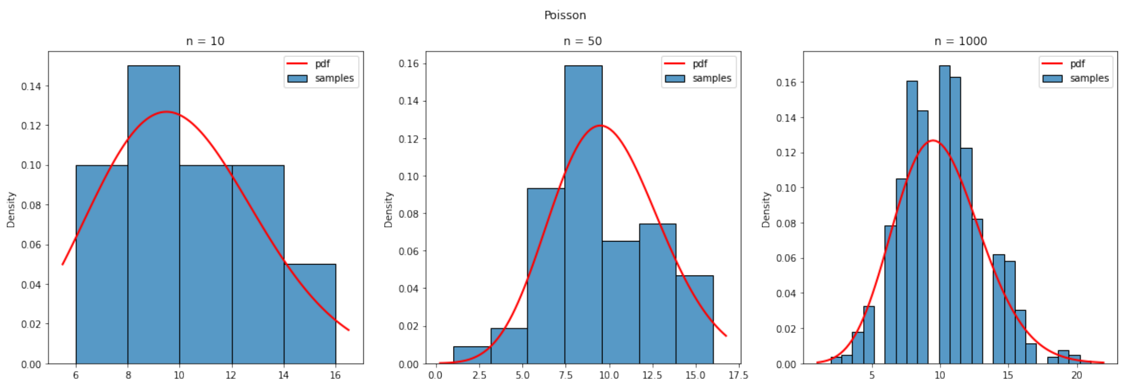
[124]:
```python
def build_hist(datasets, pdf, name):
    fig, axes = plt.subplots(1, len(datasets), figsize=(20, 6))
    fig.suptitle(name)
    for i, data in enumerate(datasets):
        sns.histplot(data, kde=False, stat='density', label='samples', ax=axes[i])
        x0, x1 = axes[i].get_xlim()
        x_pdf = np.linspace(x0, x1, 100)
        y_pdf = [pdf(x) for x in x_pdf]

        axes[i].plot(x_pdf, y_pdf, 'r', lw=2, label='pdf')
        axes[i].legend()
        axes[i].set_title(f'n = {len(data)}')
```

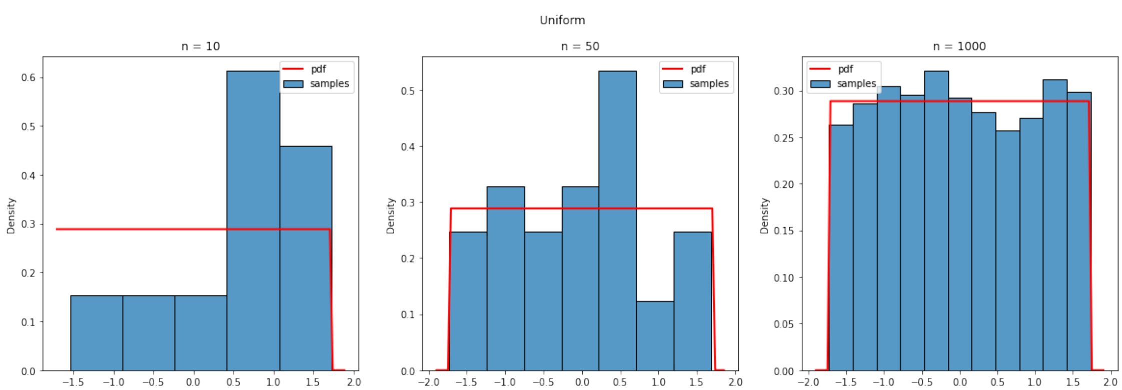
[125]:
```python
for name, dist in dists.items():
    build_hist([dist['gen_data'](n) for n in [10, 50, 1000]], dist['pdf'], name)
```




Laplace
n = 10
n = 50
n = 1000
pdf
samples
Density


Poisson
n = 10
n = 50
n = 1000
pdf
samples
Density


Uniform
n = 10
n = 50
n = 1000
pdf
samples
Density

## 1.2 Task 2

### 1.2.1 Position and dispersion characteristics

[204]:
```
def calc_chars(data_generator, n):
    iters = 1000
    mean = 0
    med = 0
    z_r = 0
    z_q = 0
    z_tr = 0
    mean_2 = 0
    med_2 = 0
    z_r_2 = 0
    z_q_2 = 0
    z_tr_2 = 0
    for i in range(iters):
        data = data_generator(n)
        data.sort()

        tmp = data.mean()
        mean += tmp
        mean_2 += tmp ** 2

        tmp = np.median(data)
        med += tmp
        med_2 += tmp ** 2

        tmp = (data[0] + data[-1]) / 2
        z_r += tmp
        z_r_2 += tmp ** 2

        tmp = (np.quantile(data, 0.25) + np.quantile(data, 0.75)) / 2
        z_q += tmp
        z_q_2 += tmp ** 2

        r = n // 4
        tmp = sum(data[r:-r]) / (n - 2 * r)
        z_tr += tmp
        z_tr_2 += tmp ** 2

    mean /= iters
    med /= iters
    z_r /= iters
    z_q /= iters
    z_tr /= iters
    mean_2 /= iters
    med_2 /= iters
```

```
    z_r_2 /= iters
    z_q_2 /= iters
    z_tr_2 /= iters

    return tuple(map(lambda x: round(x, 4) ,(mean, med, z_r, z_q, z_tr,\
           mean_2 - mean ** 2, \
           med_2 - med ** 2,\
           z_r_2 - z_r ** 2,\
           z_q_2 - z_q ** 2,\
           z_tr_2 - z_tr ** 2)))
```

[210]:

```
for name, dist in dists.items():
    for n in [10, 100, 1000]:
        print(f'{name}, n = {n}:', ' & '.join(map(str,␣
↪calc_chars(dist['gen_data'], n))))
    print()
```

```
Normal, n = 10: 0.012 & 0.0108 & 0.0107 & 0.0122 & 0.0106 & 0.0992 & 0.1356 &
0.1737 & 0.1158 & 0.1138
Normal, n = 100: -0.0008 & 0.0017 & -0.0013 & 0.0001 & -0.0001 & 0.0098 & 0.0158
& 0.095 & 0.0119 & 0.0119
Normal, n = 1000: 0.001 & 0.0011 & -0.015 & 0.0011 & 0.0011 & 0.001 & 0.0016 &
0.0618 & 0.0012 & 0.0012

Cauchy, n = 10: -7.1878 & 0.0024 & -35.9147 & -0.0344 & -0.0157 & 49162.7683 &
0.3346 & 1228851.7662 & 1.0214 & 0.5423
Cauchy, n = 100: -0.5918 & -0.0053 & -29.7943 & -0.0093 & -0.0062 & 115.0363 &
0.0235 & 274417.1969 & 0.0495 & 0.025
Cauchy, n = 1000: 8.4377 & 0.0016 & 4233.7402 & 0.0011 & 0.0019 & 55991.2183 &
0.0025 & 13994883205.8307 & 0.0048 & 0.0025

Laplace, n = 10: -0.0021 & -0.007 & 0.0095 & -0.0038 & -0.0046 & 0.0983 & 0.0719
& 0.4186 & 0.0842 & 0.071
Laplace, n = 100: 0.0021 & 0.0051 & -0.0152 & 0.0025 & 0.0043 & 0.0108 & 0.006 &
0.4215 & 0.0105 & 0.0065
Laplace, n = 1000: -0.0016 & -0.0015 & -0.0163 & -0.0006 & -0.0014 & 0.001 &
0.0005 & 0.4326 & 0.001 & 0.0006

Poisson, n = 10: 10.0003 & 9.829 & 10.3005 & 9.9149 & 9.8798 & 1.0265 & 1.4918 &
1.8919 & 1.1816 & 1.1816
Poisson, n = 100: 10.0022 & 9.8555 & 10.983 & 9.9157 & 9.8598 & 0.1034 & 0.2034
& 1.0062 & 0.1523 & 0.1204
Poisson, n = 1000: 9.9993 & 9.9955 & 11.624 & 9.9935 & 9.8575 & 0.0105 & 0.0042
& 0.6256 & 0.0037 & 0.0119

Uniform, n = 10: -0.0235 & -0.0398 & -0.0014 & -0.0267 & -0.0339 & 0.1065 &
```

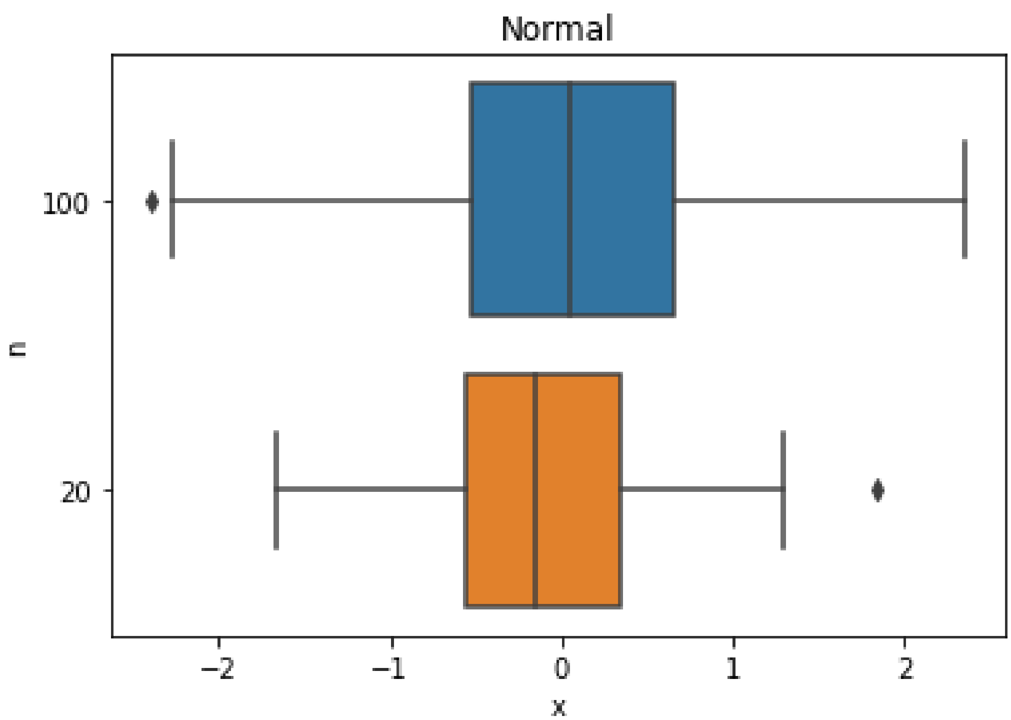
```
0.2307 & 0.0523 & 0.1451 & 0.1673
Uniform, n = 100: -0.0012 & -0.0006 & 0.0008 & -0.0022 & -0.0004 & 0.0097 &
0.0284 & 0.0006 & 0.0145 & 0.0192
Uniform, n = 1000: 0.0006 & 0.0018 & 0.0001 & 0.0 & 0.0012 & 0.0009 & 0.0029 &
0.0 & 0.0014 & 0.0018
```

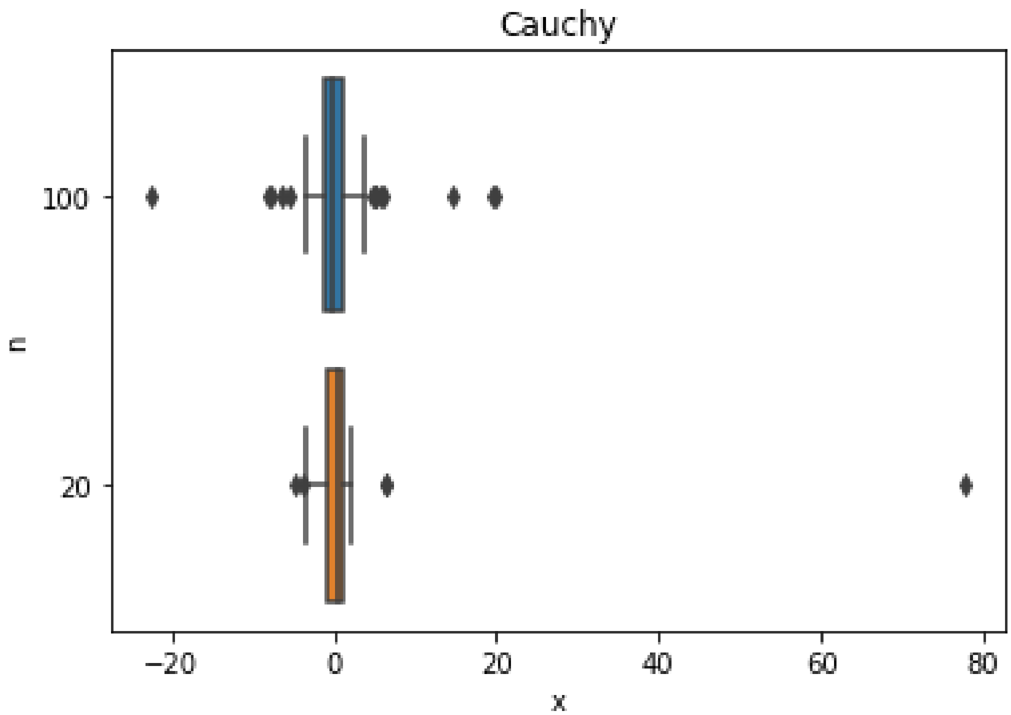
## 1.3 Task 3

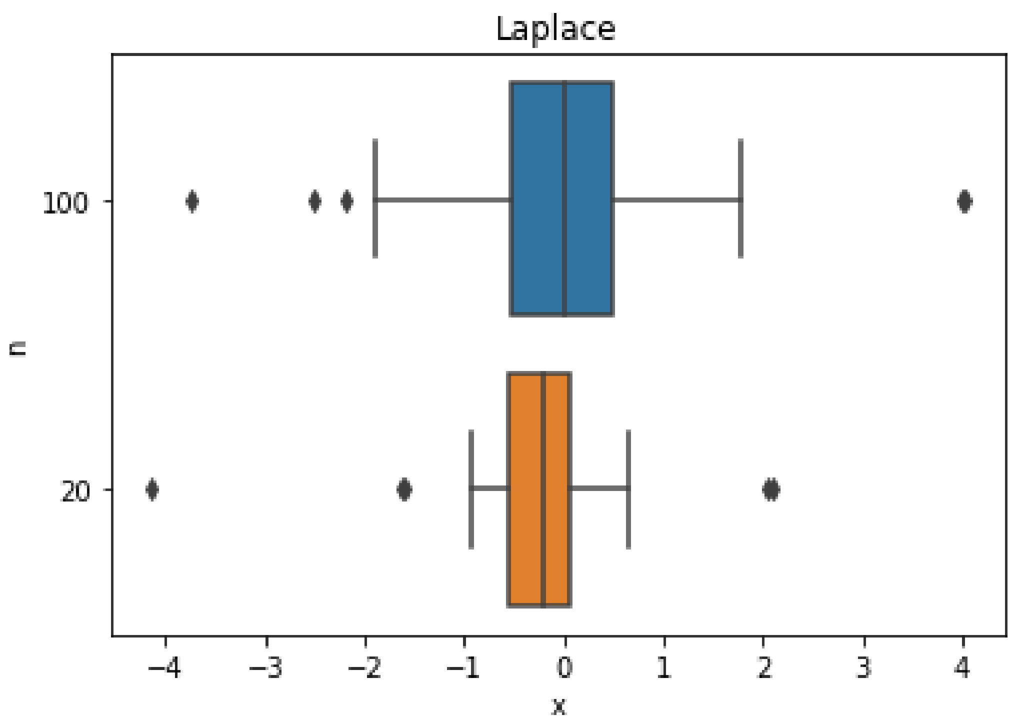
### 1.3.1 Boxplots

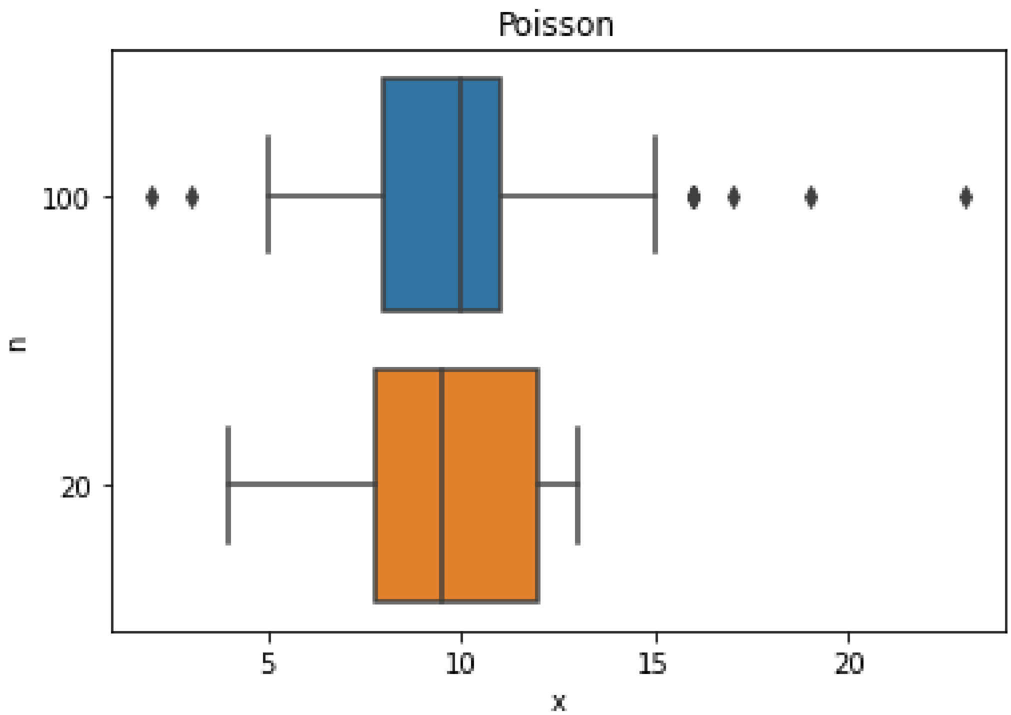
[65]:
```python
def build_boxplot(dataset, title):
    names=[str(len(data)) for data in dataset]
    fig, ax = plt.subplots(1, 1)
    sns.boxplot(data=dataset, orient='h', ax=ax)
    ax.set(xlabel='x', ylabel='n')
    ax.set(yticklabels=names)
    ax.set_title(title)
```

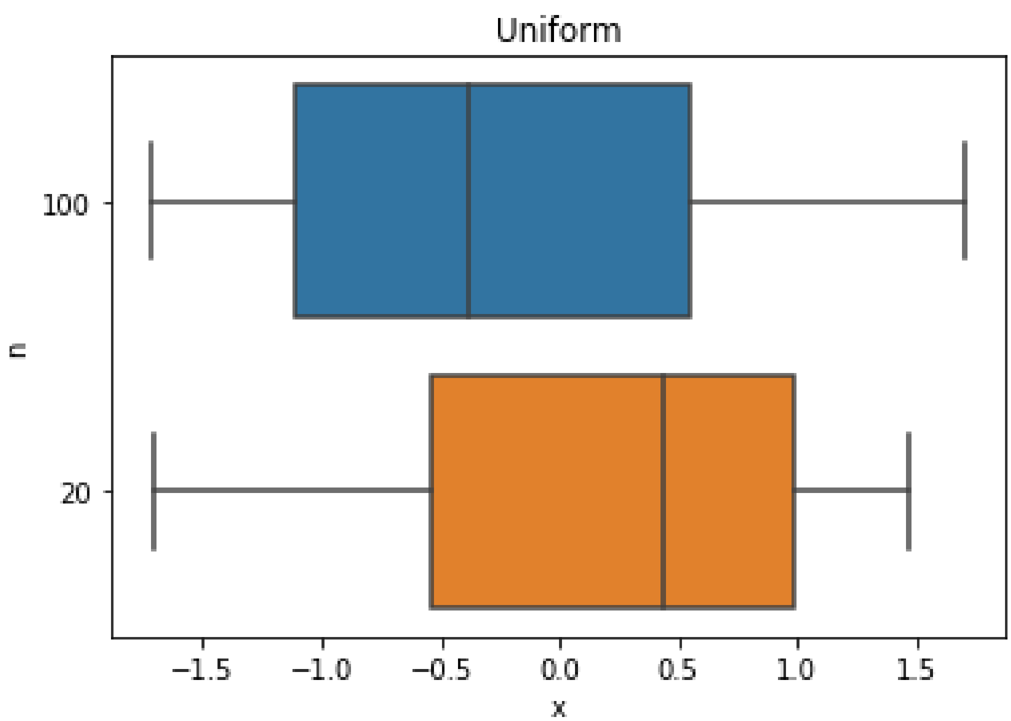
[66]:
```python
for name, dist in dists.items():
    build_boxplot([dist['gen_data'](100), dist['gen_data'](20)], name)
```




Cauchy
100
20
n
−20
0
20
40
60
80
x


Laplace
100
20
n
−4
−3
−2
−1
0
1
2
3
4
x


Poisson
100
20
n
5
10
15
20
x


Uniform
100
20
n
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x

### 1.3.2 Share of outliers

[69]:
```
def calc_outlier(data_generator, n, iters=1000):
    num = 0
    for i in range(iters):
        data = data_generator(n)
        q1 = np.quantile(data, 0.25)
        q3 = np.quantile(data, 0.75)
        iqr = q3 - q1
        x1 = q1 - 1.5 * iqr
        x2 = q3 + 1.5 * iqr
        num += np.count_nonzero((data < x1) | (x2 < data)) / n
    return round(num / iters, 2)
```

[72]:
```
for name, dist in dists.items():
    for n in [20, 100]:
        print(f'{name} {n}:', calc_outlier(dist['gen_data'], n))
```

```
Normal 20: 0.02
Normal 100: 0.01
Cauchy 20: 0.15
Cauchy 100: 0.16
Laplace 20: 0.07
Laplace 100: 0.06
Poisson 20: 0.02
Poisson 100: 0.01
Uniform 20: 0.0
Uniform 100: 0.0
```

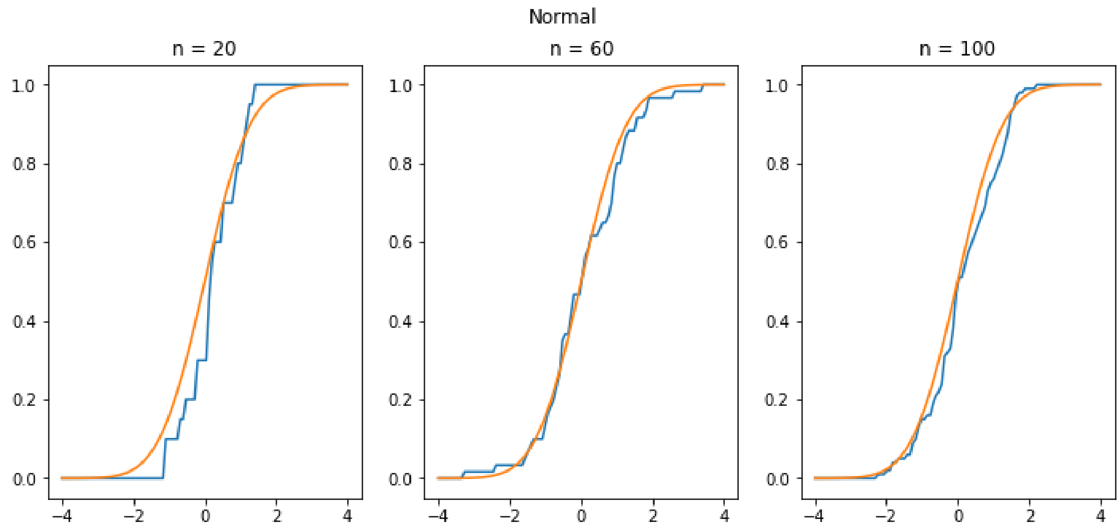
## 1.4 Task 4

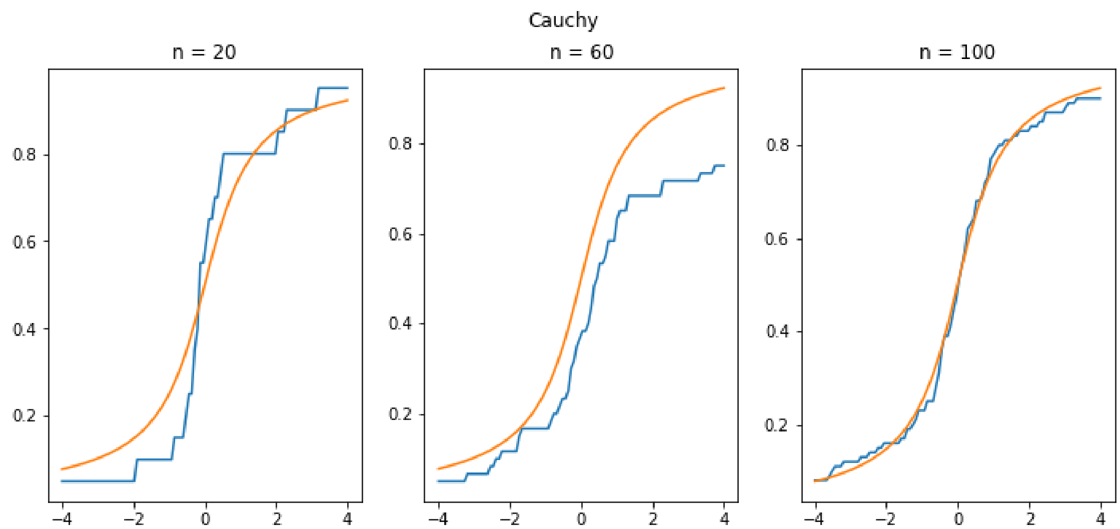
### 1.4.1 Empirical distribution function

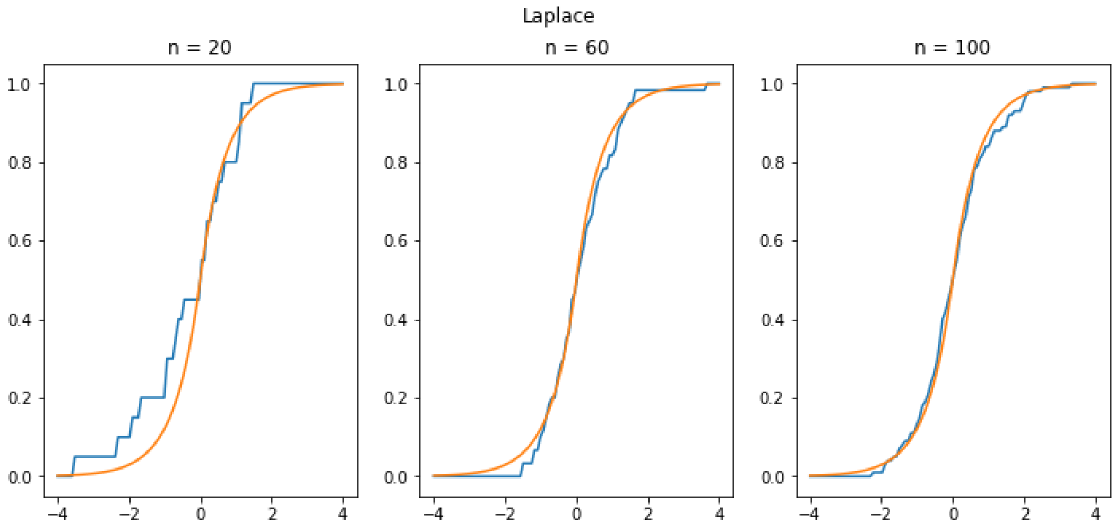
[78]:
```
def build_edf(name, datasets, cdf, x):
    fig, axes = plt.subplots(1, len(datasets), figsize=(12, 5))
    fig.suptitle(name)
    for i, data in enumerate(datasets):
        y1 = ECDF(data)(x)
        y2 = cdf(x)
        axes[i].plot(x, y1)
        axes[i].plot(x, y2)
        axes[i].set_title(f'n = {len(data)}')
```

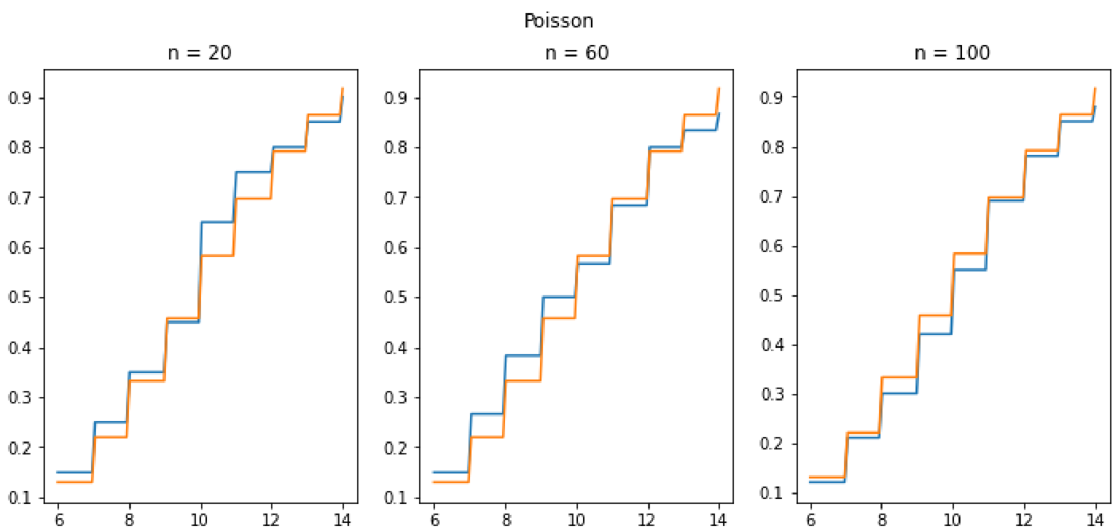
[79]:
```
for name, dist in dists.items():
    if name != 'Poisson':
        build_edf(name, [dist['gen_data'](n) for n in [20, 60, 100]],␣
 ↪dist['cdf'], np.linspace(-4, 4, 100))
    else:
```

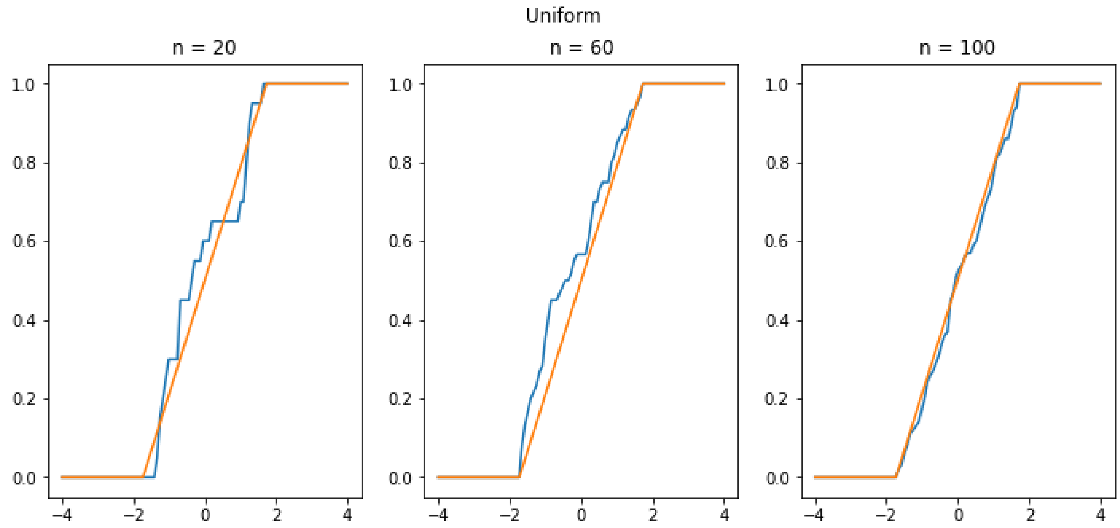
```
        build_edf(name, [dist['gen_data'](n) for n in [20, 60, 100]],␣
↪dist['cdf'], np.linspace(6, 14, 100))
```




Laplace
n = 20
n = 60
n = 100

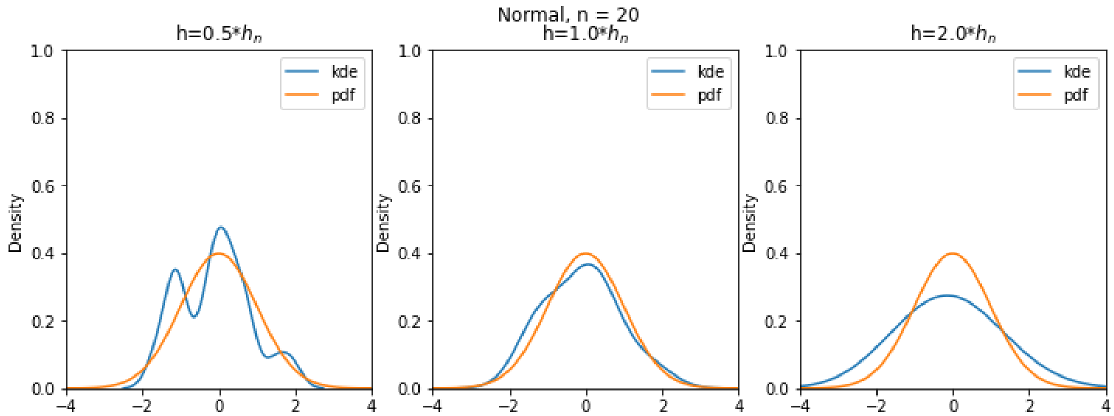

Poisson
n = 20
n = 60
n = 100

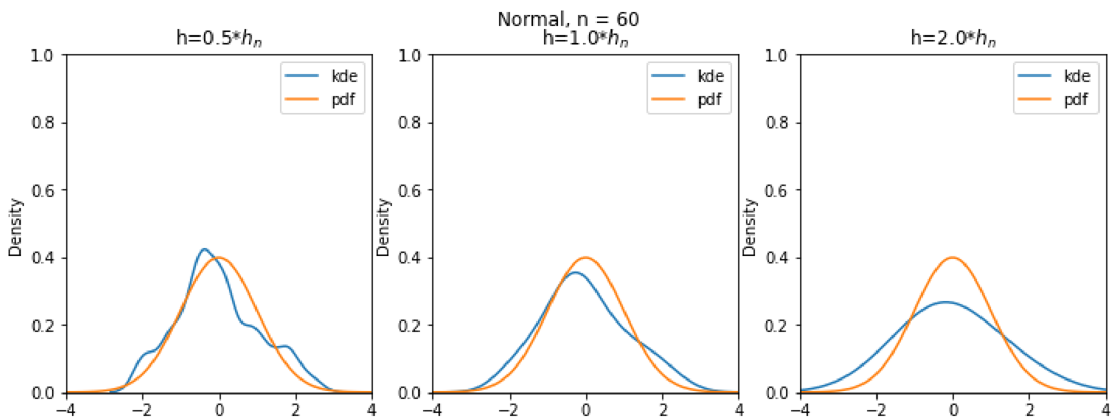


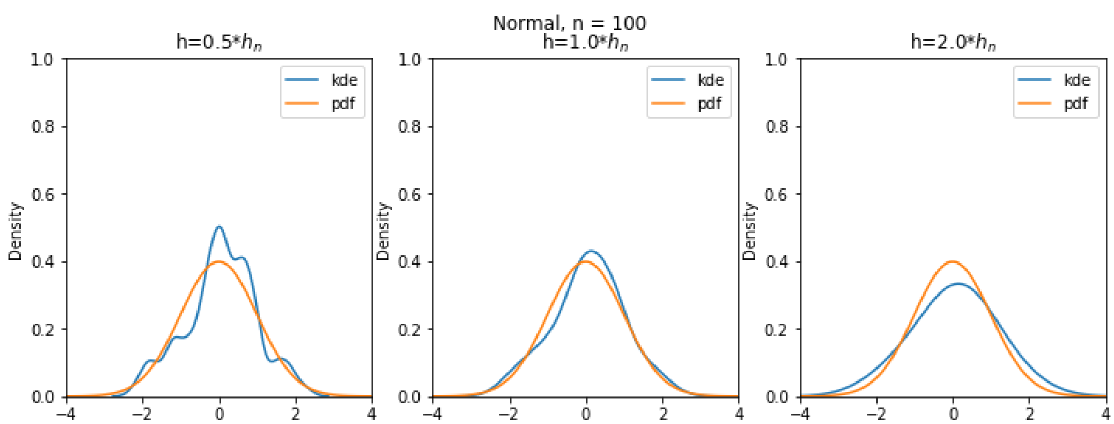

### 1.4.2 KDE

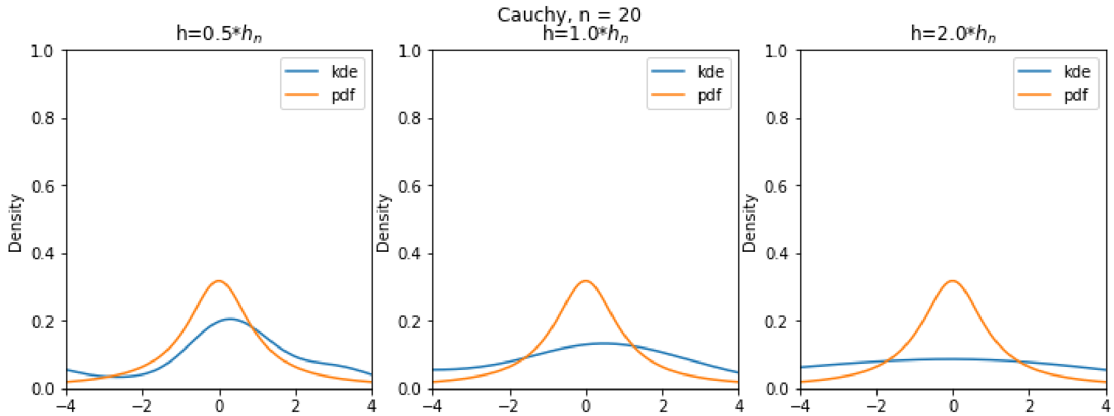
[181]:
```
def build_kde(name, data, pdf, x):
    scales = [0.5, 1.0, 2.0]
    fig, ax = plt.subplots(1, len(scales), figsize=(12, 4))
    fig.suptitle(f'{name}, n = {len(data)}')
    for i, scale in enumerate(scales):
        sns.kdeplot(data, ax=ax[i], bw_method='silverman', bw_adjust=scale,␣
↪label='kde')
        ax[i].set_xlim([x[0], x[-1]])
        ax[i].set_ylim([0, 1])
        ax[i].plot(x, [pdf(xk) for xk in x], label='pdf')
        ax[i].legend()
        ax[i].set_title(f'h={str(scale)}*$h_n$')
```

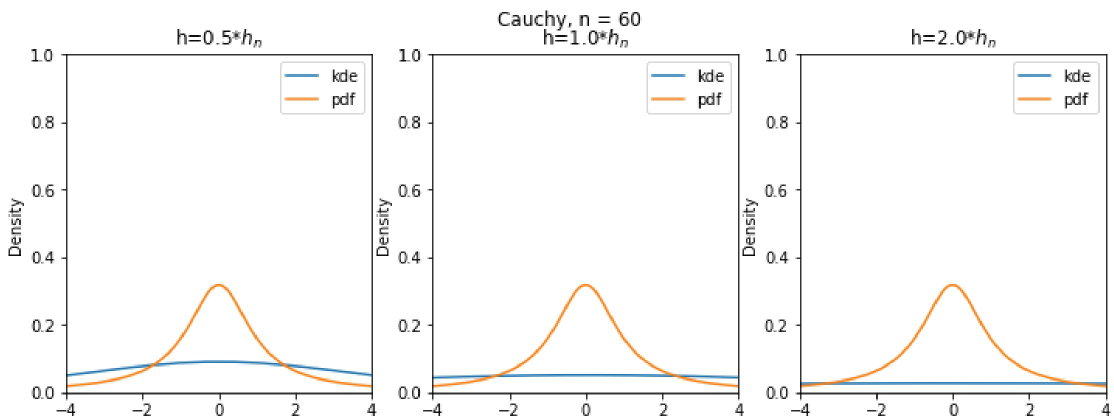
[182]:
```
for name, dist in dists.items():
    for n in [20, 60, 100]:
        if name != 'Poisson':
            build_kde(name, dist['gen_data'](n), dist['pdf'], np.linspace(-4,␣
↪4, 100))
        else:
            build_kde(name, dist['gen_data'](n), dist['pdf'], np.linspace(6,␣
↪14, 100))
```

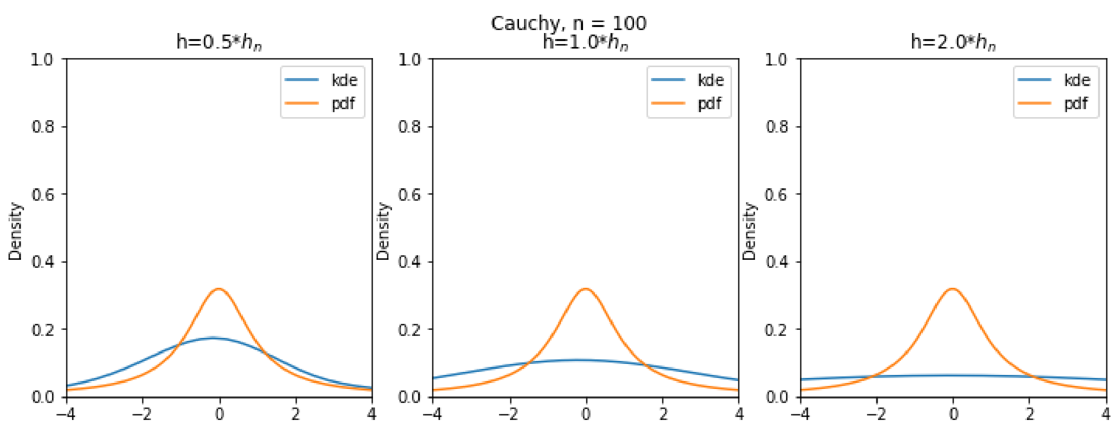

Normal, n = 20
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
Density
kde
pdf

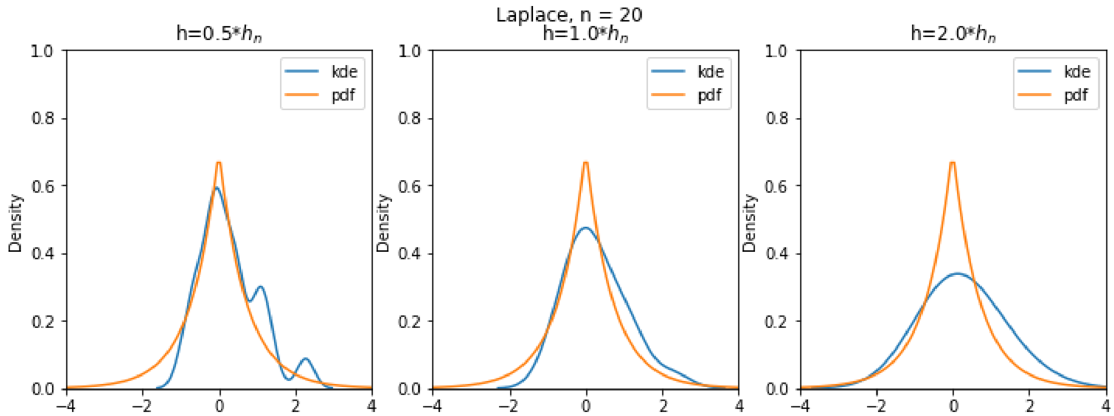

Normal, n = 60
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
Density
kde
pdf

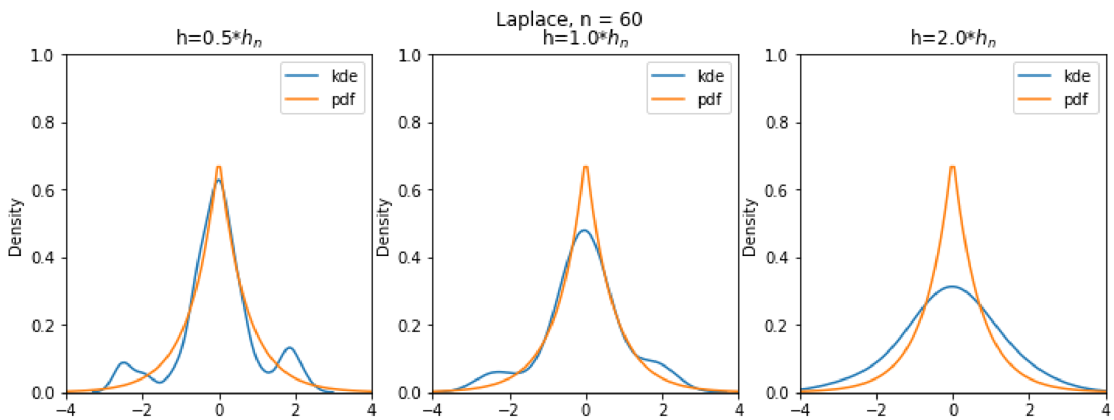

Normal, n = 100
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
Density
kde
pdf

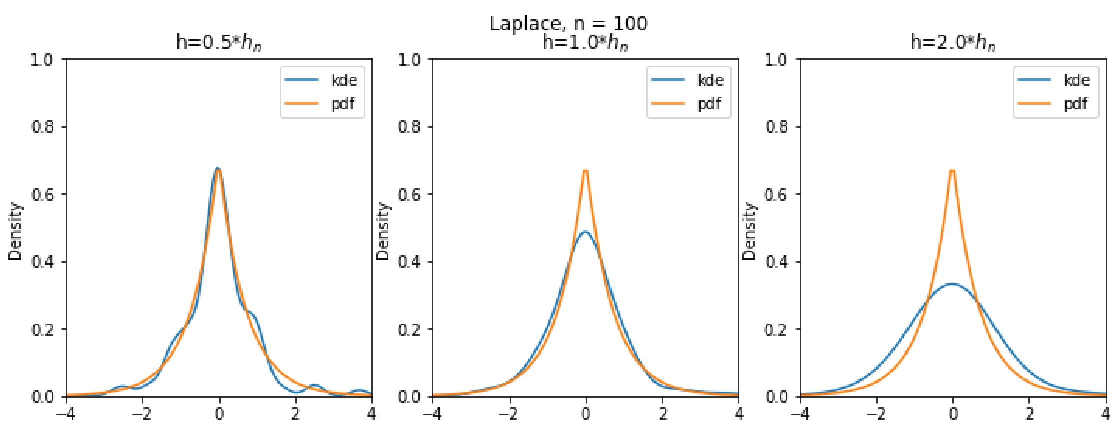

Cauchy, n = 20
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
Density
kde
pdf

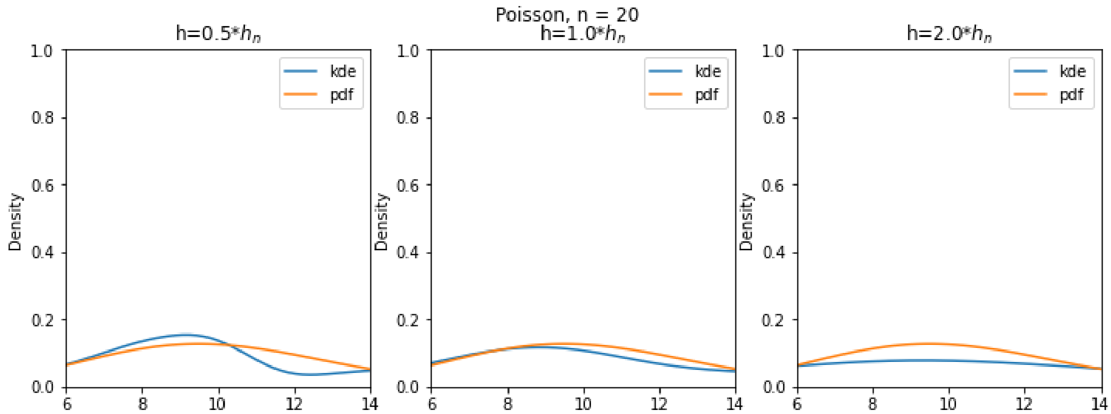

Cauchy, n = 60
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
Density
kde
pdf

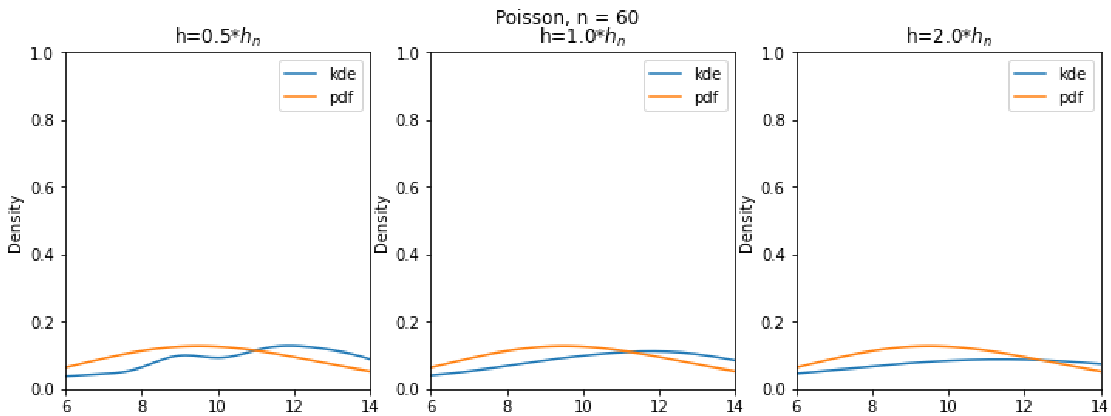

Cauchy, n = 100
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
Density
kde
pdf

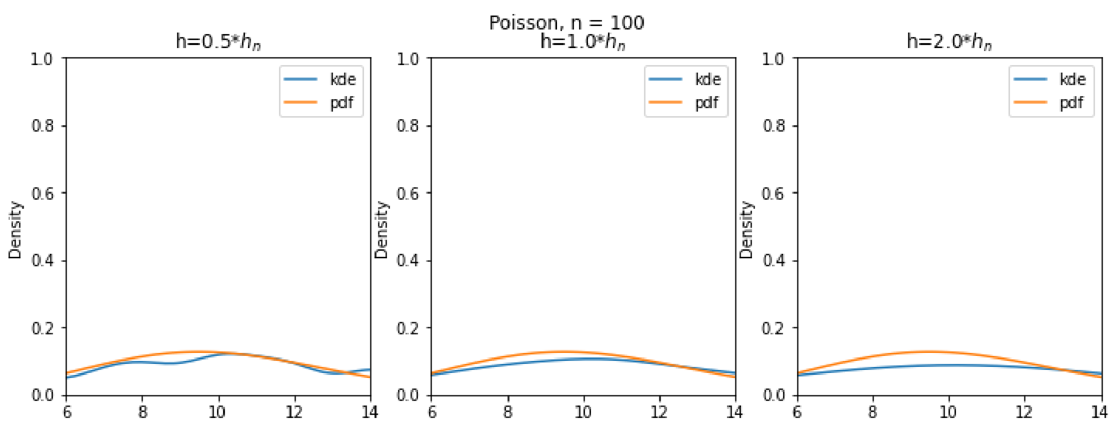

Laplace, n = 20
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
kde
pdf
Density

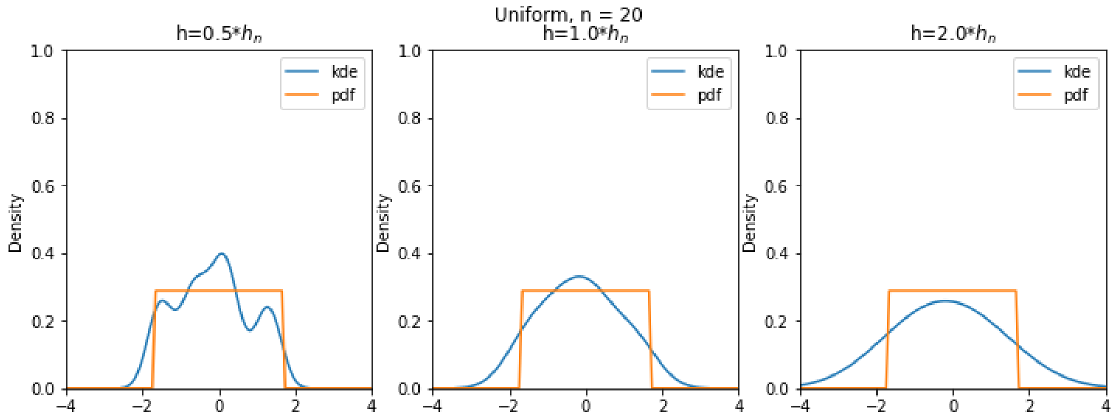

Laplace, n = 60
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
kde
pdf
Density

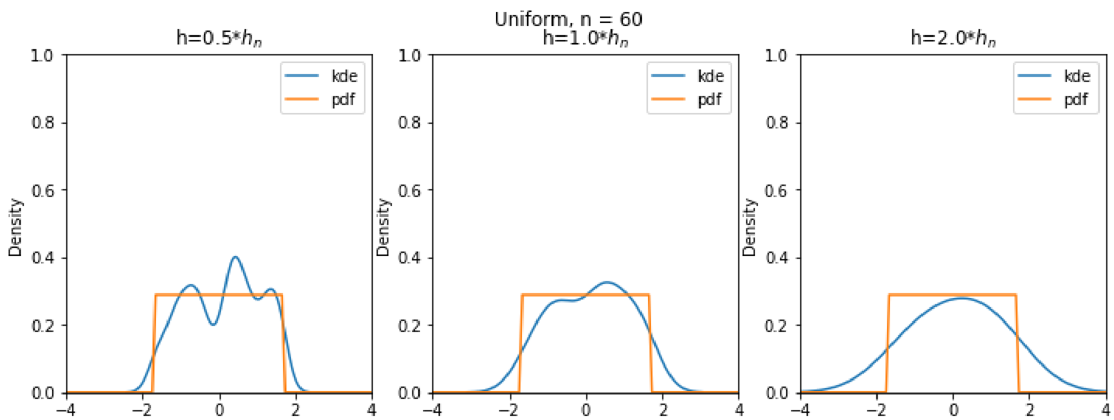

Laplace, n = 100
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
kde
pdf
Density

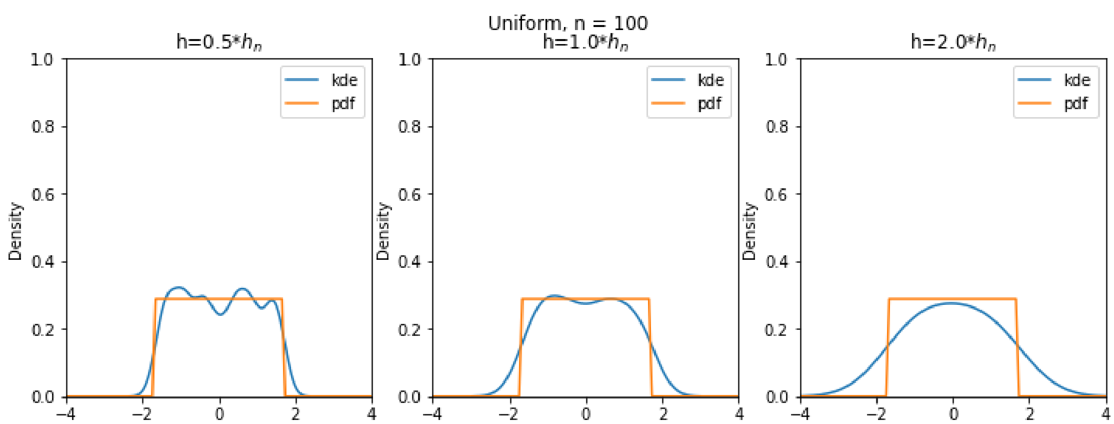

Poisson, n = 20
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
kde
pdf
Density


Poisson, n = 60
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
kde
pdf
Density


Poisson, n = 100
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
kde
pdf
Density


Uniform, n = 20
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
kde
pdf
Density


Uniform, n = 60
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
kde
pdf
Density


Uniform, n = 100
h=0.5*$h_n$
h=1.0*$h_n$
h=2.0*$h_n$
kde
pdf
Density

[ ]: