FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTE
OF HIGHER EDUCATION
ITMO UNIVERSITY

Report on learning practice # 1

Analysis of univariate random variables

Performed by:

Igor Vernyy, j4134c

Kirill Mukhin, j4134c

Alexander Petrov, j4134c

Bogdan Chertkov, j4132c

Saint-Petersburg

2022

Substantiation of chosen subsample

We have tried different combinations of dataset features to obtain well-interpreted results. A final set contains the following continuous features: “Hospital beds per thousand” (HBpT), “Excess mortality cumulative per million” (EMCpM) and “Excess mortality cumulative” (EMC).

Non-parametric estimation of PDF

Figures 1 – 3 depict non-parametric estimations in form of histogram and using kernel density function for the chosen random variables.

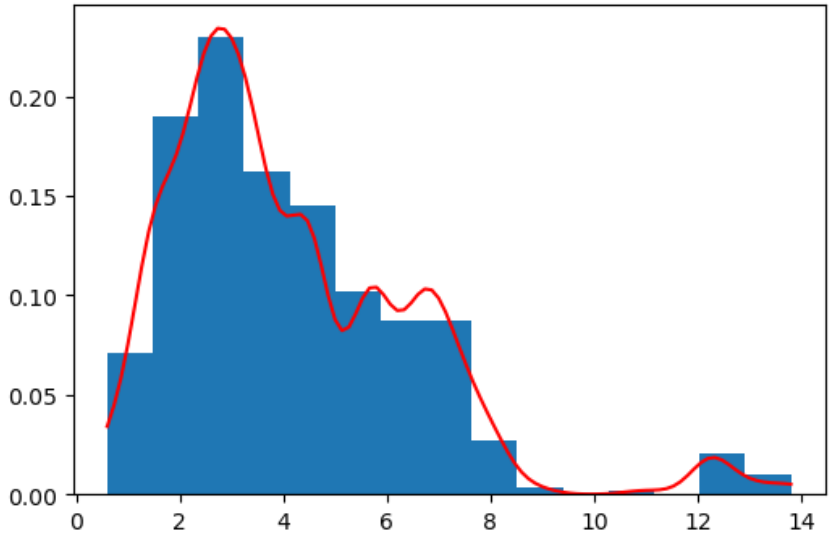

Figure 1 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “Hospital beds per thousand”

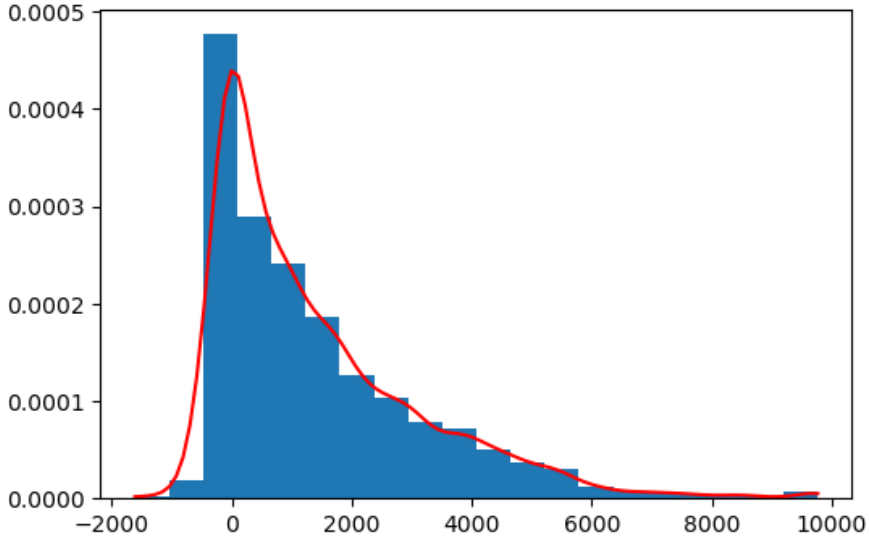

Figure 2 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “Excess mortality cumulative per million”

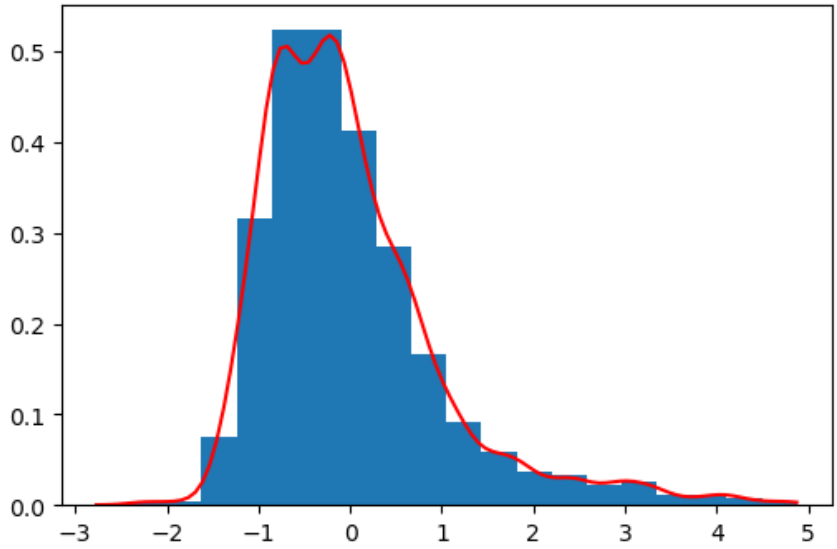

Figure 3 – Non-parametric estimation of PDF in form of histogram and using kernel density function for "Excess mortality cumulative"

Order statistics estimation

Calculated order statistics for three random variables are presented in Table 1. Due to significant differences between variable's mean and variance we have to normalize their values in order to draw them as box with whisker plot (Figure 4).

X	min (X)	max (X)	$mean(X)$	$var(X)$	$median(X)$	$Q_1(X)$	$Q_3(X)$
HBpT	0.6	13.8	4.14	5.7	3.4	2.5	5.64
EMCpM	−1618.3	9771.68	$1.41 \cdot 10^3$	$3.07 \cdot 10^6$	856.64	37.1	2228.36
EMC	−2.77	4.87	$-1.9 \cdot 10^{-15}$	1	−0.20	−0.71	0.42

Table 1 – Order statistics and some moments for the chosen features

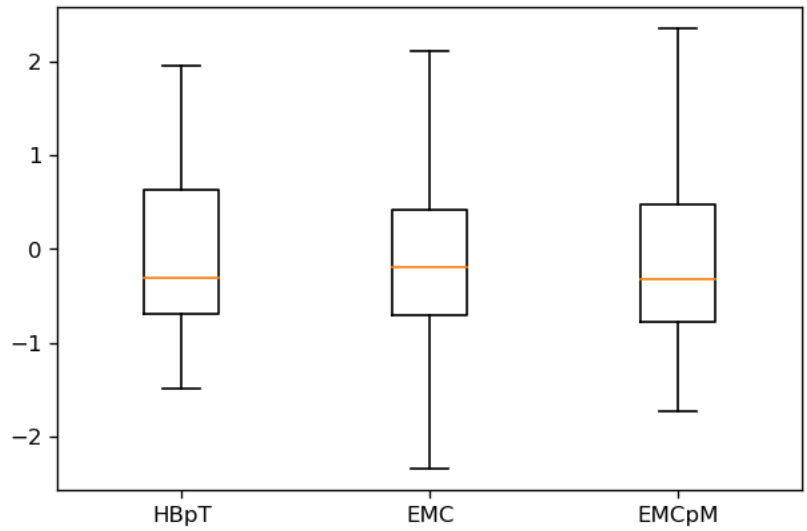

Figure 4 – Box with whisker plot for the normalized features

Selection of theoretical distributions

Observing kernel density function plots for variables we can make propose Gamma distribution for the first variable ("Hospital beds per thousand"). You can see its probability density function at Figure 5.

$$Gamma(k,\theta) = \frac{1}{\Gamma(k)\,\theta^k} x^{k-1} e^{-\frac{x}{\theta}}$$

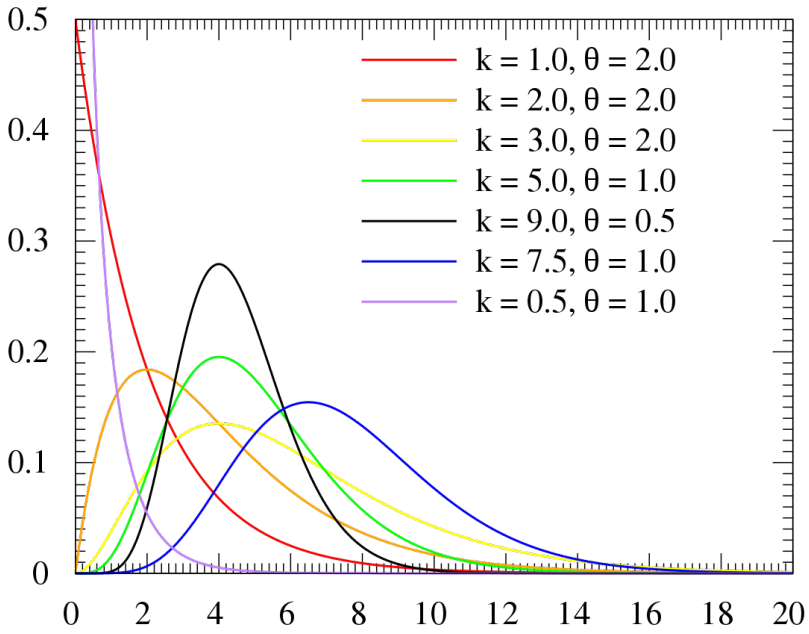

Figure 5 – Gamma distribution PDF

Figure 6 depicts chi-squared distribution which can be used to explain the second variable ("Excess mortality cumulative per million").

$$\chi^2(k) = \frac{1}{2^{\frac{k}{2}} \cdot \Gamma\left(\frac{k}{2}\right)} \cdot x^{\frac{k}{2}-1} e^{-\frac{x}{2}}$$

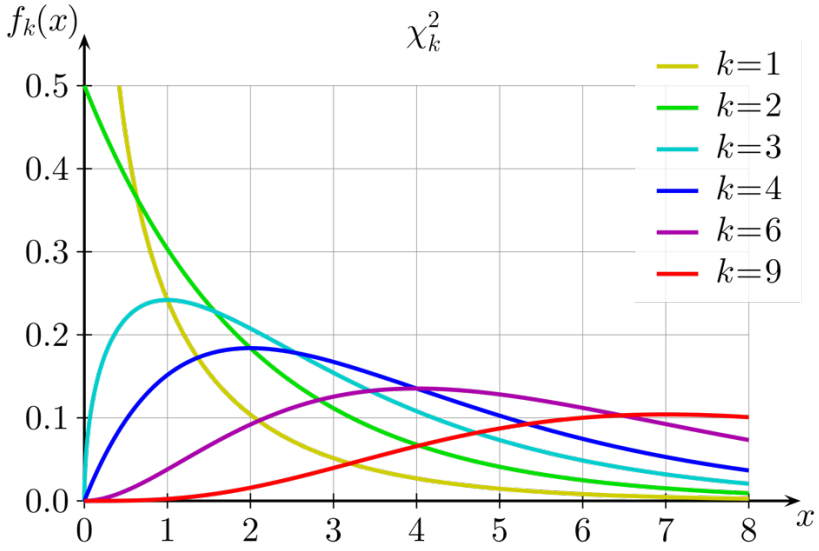

Figure 6 – Chi-squared distribution PDF

And the third variable “Excess mortality cumulative” seems like a slightly skewed normal distribution.

$$N(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

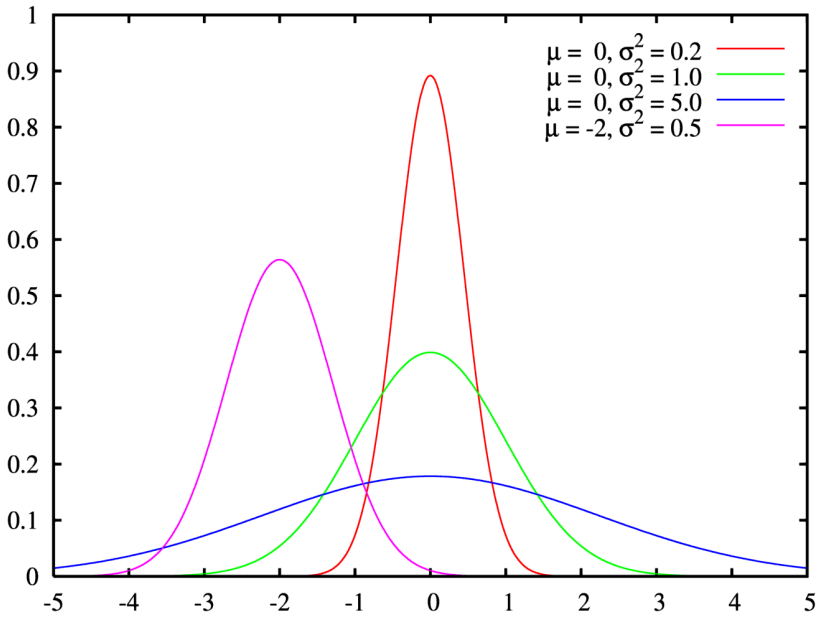

Figure 7 – Norma distribution PDF

Estimation of random variable distribution parameters

The next step is to fit parameters for above-mentioned distributions using maximum likelihood estimation and least squares method. Figures 8 – 10 contain resulting plots.

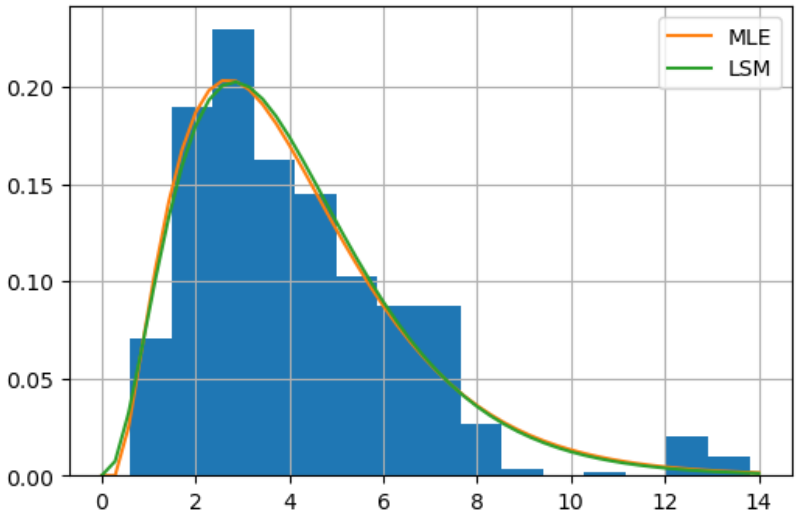

Figure 8 – Estimations of Gamma distribution using MLE and LSM

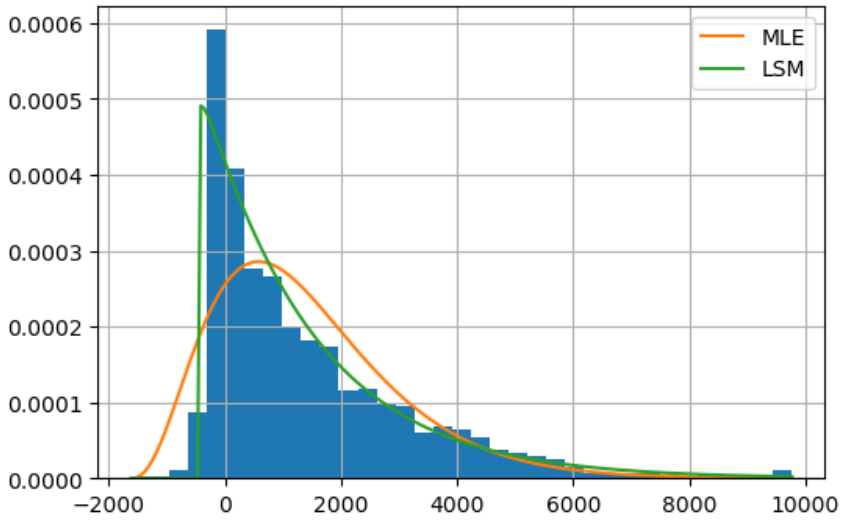

Figure 9 – Estimations of χ^2 distribution using MLE and LSM

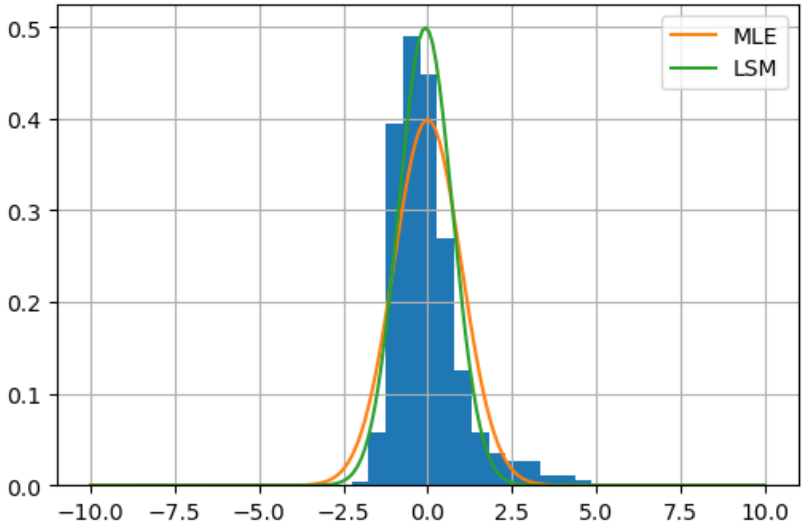

Figure 10 – Estimations of Normal distributions using MLE and LSM

QQ-plots

In order to check the correctness of the choice of theoretical distributions for the data, we have built quantile-quantile plots. Figures 11 – 13 show them. Each figure contains plot for maximum likelihood estimation (left) and least squares method (right). You can see that points are spread along 45-degree line. Which indicates the correctness of our assumptions.

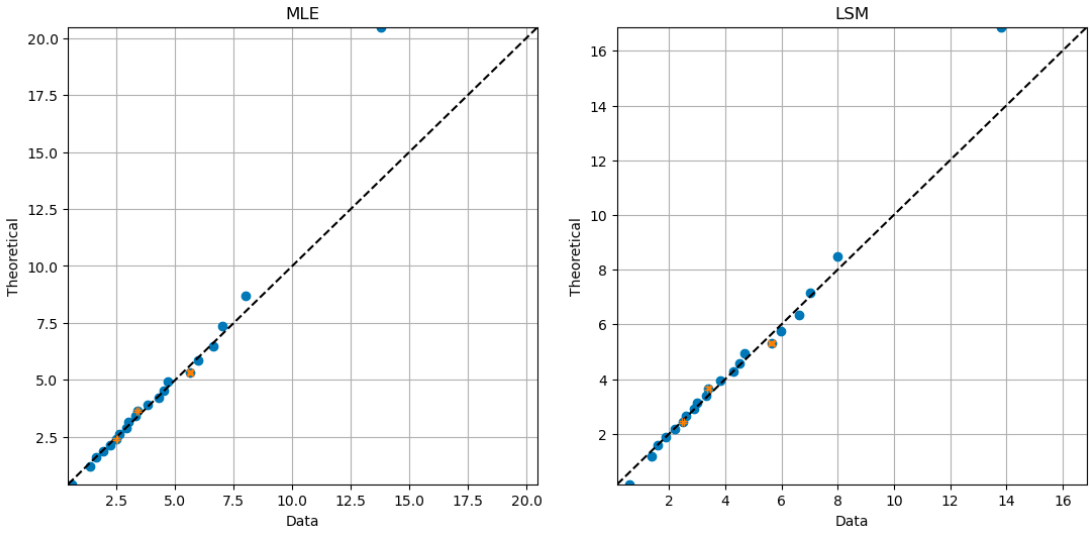

Figure 11 – QQ-plots for the first RV “Hospital beds per thousand”

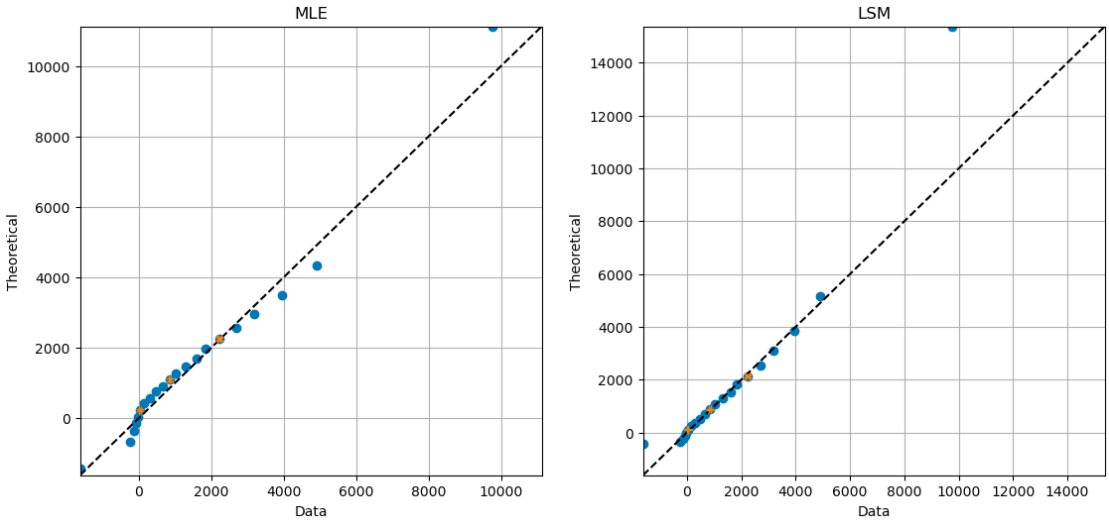

Figure 12 – QQ-plots for the second RV “Excess mortality cumulative per million”

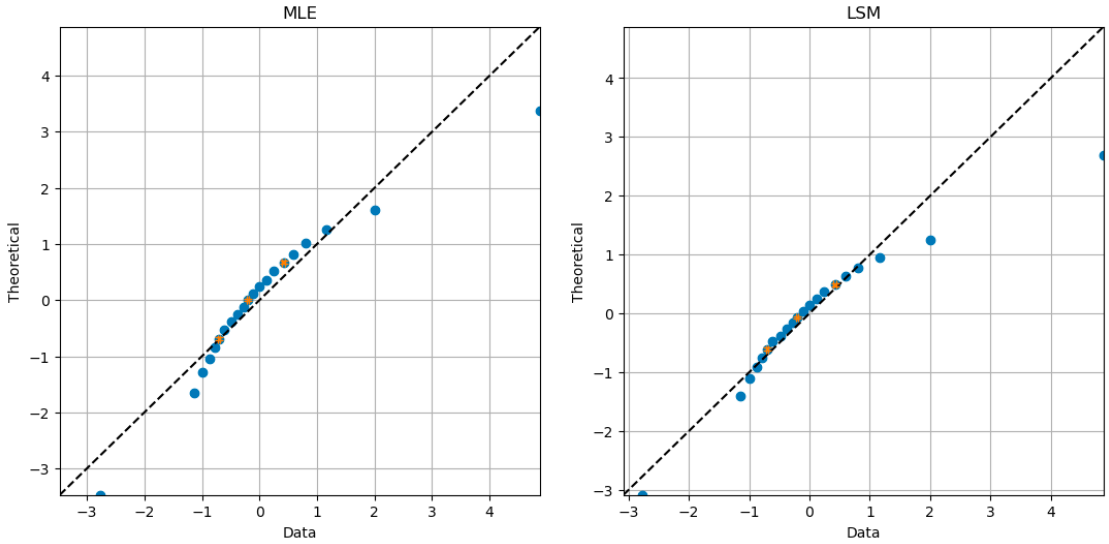

Figure 13 – QQ-plots for the third RV “Excess mortality cumulative”

Statistical tests

Statistical tests are also an important criterion for testing hypotheses about the type of distribution. To check assumptions two tests are used – Kolmogorov (Table 2 – MLE, Table 3 – LSM) and Cramer-von-Mises (Table 4 – MLE, Table 5 – LSM). The result is defined assuming $\alpha = 0.05$.

X	Distribution	Statistic	pValue	Result
HBpT	Gamma	0.131	0.328	Passed
EMCpM	χ^2	0.177	0.077	Passed
EMC	Normal	0.121	0.419	Passed

Table 2 – Kolmogorov test for MLE

X	Distribution	Statistic	pValue	Result
HBpT	Gamma	0.114	0.499	Passed
EMCpM	χ^2	0.110	0.550	Passed
EMC	Normal	0.155	0.165	Passed

Table 3 – Kolmogorov test for LSM

X	Distribution	Statistic	pValue	Result
HBpT	Gamma	0.118	0.506	Passed
EMCpM	χ^2	0.096	0.604	Passed
EMC	Normal	0.159	0.366	Passed

Table 2 – Cramer-von-Mises for MLE

X	Distribution	Statistic	pValue	Result
HBpT	Gamma	0.063	0.800	Passed
EMCpM	χ^2	0.074	0.730	Passed
EMC	Normal	0.133	0.449	Passed

Table 3 – Cramer-von-Mises for LSM

Appendix

```
#!/usr/bin/env python
# coding: utf-8

# # Lab 1

# In[1]:


import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy.stats
from scipy import optimize
from scipy import stats
from scipy.stats.distributions import gamma, expon, norm, chi2
import seaborn as sns
from IPython.display import display, Math
from functools import partial


# ## Step 1. Choose subsample with main variables for your further
analysis

# In[2]:


path_to_file = '/opt/notebooks/owid-covid-data.csv'
source_df = pd.read_csv(path_to_file)

df = source_df[['hospital_beds_per_thousand',
'excess_mortality_cumulative',
'excess_mortality_cumulative_per_million']].copy().dropna()


# In[3]:


df.head(5)


# ## Step 2. You need to make a non-parametric estimation of PDF in form
of histogram and using kernel density function (or probability law in case
of discrete RV).

# In[4]:


from scipy.stats import kde

def kernel_density_estimation(x, bins):
    density = kde.gaussian_kde(x)
    xgrid = np.linspace(x.min(), x.max(), 100)
    mpl.rcParams['figure.dpi'] = 100
    plt.hist(x, bins=bins, density = True)
    plt.plot(xgrid, density(xgrid), 'r-')
```

```
# ### Hospital beds per thousand

# In[5]:


beds = df['hospital_beds_per_thousand']


# In[6]:


kernel_density_estimation(beds, 15)


# ### Excess mortality cumulative per million

# In[7]:


emcpm = df['excess_mortality_cumulative_per_million'].dropna()


# In[8]:


kernel_density_estimation(emcpm, 20)


# ### Exces mortality cumulative

# In[9]:


emc = df['excess_mortality_cumulative'].dropna()
emc = (emc - emc.mean()) / emc.std()


# In[10]:


kernel_density_estimation(emc, 20)


# ## Step 3. You need to make an estimation of order statistics and
represent them as "box with whiskers" plot.

# In[11]:


pd.DataFrame(
    [
        ['Hospital beds per thousand (HBpT)', beds.min(), beds.max(),
beds.mean(), beds.var(), beds.median(), np.percentile(beds, 25),
np.percentile(beds, 75)],
        ['Excess mortality cumulative per million (EMCpM)', emcpm.min(),
emcpm.max(), emcpm.mean(), emcpm.var(), emcpm.median(),
np.percentile(emcpm, 25), np.percentile(emcpm, 75)],
        ['Excess mortality cumulative (EMC)', emc.min(), emc.max(),
emc.mean(), emc.var(), emc.median(), np.percentile(emc, 25),
np.percentile(emc, 75)]
```

```
    ],
    columns=['Random Variable', 'min', 'max', 'mean', 'variance',
'median', 'Q1', 'Q3']
)


# In[12]:


beds_norm = (beds - beds.mean()) / beds.std()
emc_norm = (emc - emc.mean()) / emc.std()
emcpm_norm = (emcpm - emcpm.mean()) / emcpm.std()


# In[13]:


fig1, (ax1) = plt.subplots(1, 1, dpi=120)
plt.subplots_adjust(wspace=0.4)
ax1.boxplot(
    (beds_norm, emc_norm, emcpm_norm),
    labels=('HBpT', 'EMC', 'EMCpM'),
    showfliers=False
)

# ax2.boxplot(
#     ((), emca),
#     labels=('', 'EMCA'),
#     showfliers=False
# )


# ## Step 4. Find one or several theoretical distributions that could
describe your sample on a basis of non-parametric analysis results

# ### Hospital beds per thousand
#
# It seems like a **gamma distribution**:
#
# $$x \sim \text{Gamma}(\alpha, \beta) = \frac{\beta^{\alpha}x^{\alpha-
1}e^{-\beta x}}{\Gamma(\alpha)}$$

# ### Excess mortality cumulative per million
#
# It seems like a **chi-squared** distribution

# ### Excess mortality cumulative
#
# It seems like a **Normal distribution**.

# ## Step 5. Estimate parameters of chosen distributions using methods of
maximum likelihood and least squares method

# In[14]:


def lsm_loss(func, data, params):
    space = np.linspace(0.01, 0.95)
    quantile = np.quantile(data, space)
    quantile_approximation = func.ppf(space, *params)
```

```
    return quantile - quantile_approximation

# ### Hospital beds per thousand

# #### Maximum likelihood method

# In[15]:


beds_mle = gamma.fit(beds, method='MLE')

beds_mle


# #### Least squares method

# In[16]:


beds_lsm = optimize.least_squares(partial(lsm_loss, gamma, beds), (1.0,
1.0, 1.0)).x


# In[17]:


beds_lsm


# In[39]:


from scipy.stats.distributions import gamma

mpl.rcParams['figure.dpi'] = 100
beds.hist(bins=15, density=True)
plt.plot(np.linspace(0, 14), gamma.pdf(np.linspace(0, 14), *beds_mle),
label='MLE')
plt.plot(np.linspace(0, 14), gamma.pdf(np.linspace(0, 14), *beds_lsm),
label='LSM')
# plt.title('Paramters estimation for "Hospital beds per thousand"
feature')
plt.legend()


# ### Excess mortality cumulative per million

# #### Maximum likelihood method

# In[19]:


emcpm_mle = chi2.fit(emcpm, method='MLE')

emcpm_mle


# #### Least squares method

# In[20]:
```

```
emcpm_lsm = optimize.least_squares(partial(lsm_loss, chi2, emcpm), (1.0,
1.0, 1.0)).x


# In[21]:


emcpm_lsm


# In[40]:


mpl.rcParams['figure.dpi'] = 100
emcpm.hist(bins=35, density=True)
space = np.linspace(min(emcpm), max(emcpm), 200)
plt.plot(space, chi2.pdf(space, *emcpm_mle), label='MLE')
plt.plot(space, chi2.pdf(space, *emcpm_lsm), label='LSM')
plt.legend()


# ### Excess mortality cumulative

# #### Maximum likelihood method

# In[23]:


emc_mle = norm.fit(emc, method='MLE')

emc_mle


# #### Least squares method

# In[24]:


emc_lsm = scipy.optimize.least_squares(partial(lsm_loss, norm, emc), (1.0,
1.0)).x


# In[25]:


emc_lsm


# In[41]:


mpl.rcParams['figure.dpi'] = 100
emc.hist(bins=15, density=True)
space = np.linspace(-10, 10, 200)
plt.plot(space, norm.pdf(space, *emc_mle), label='MLE')
plt.plot(space, norm.pdf(space, *emc_lsm), label='LSM')
plt.legend()
```

```
# ## Step 6. Validate your estimated parameters using QQ biplots

# In[45]:


from math import sqrt

# Calculating confidence intervals for 25%, 50% and 75% quantiles
def conf_intervals(data, qn):
    # 95% quantile of Gaussian distribution
    norm_q95 = scipy.stats.norm.ppf(0.95)
    kernel = scipy.stats.gaussian_kde(data)

    p25 = len(data[data < qn[5]]) / len(data)
    sigma25 =          (sqrt((p25 * (1 - p25)) / len(data))) /
kernel(qn[5])
    p50 = len(data[data < qn[10]]) / len(data)
    sigma50 =          (sqrt((p50 * (1 - p50)) / len(data))) /
kernel(qn[10])
    p75 = len(data[data < qn[15]]) / len(data)
    sigma75 =          (sqrt((p75 * (1 - p75)) / len(data))) /
kernel(qn[15])

    conf_q25 = norm_q95 * sigma25
    conf_q50 = norm_q95 * sigma50
    conf_q75 = norm_q95 * sigma75

    return [conf_q25, conf_q50, conf_q75]

def qq(data, theoretical, feature_name, method_name):
    # Построение квантильного биплота для двух случайных величин
    plt.figure(figsize=(6, 6))

    percs = np.linspace(0, 100, 21)
    qn_first = np.percentile(data, percs)
    qn_second = np.percentile(theoretical, percs)

    min_qn = np.min([qn_first.min(), qn_second.min()])
    max_qn = np.max([qn_first.max(), qn_second.max()])
    x = np.linspace(min_qn, max_qn)

    plt.plot(qn_first, qn_second, ls="", marker="o", markersize=6)
    plt.plot(x, x, color="k", ls="--")
    plt.xlabel('Data')
    plt.ylabel('Theoretical')
    plt.xlim([min_qn, max_qn])
    plt.ylim([min_qn, max_qn])
    plt.grid(True)

    conf_first = conf_intervals(data, qn_first)
    conf_second = conf_intervals(theoretical, qn_first)

    conf_first_list = []
    conf_second_list = []
    for element1, element2 in zip(conf_first, conf_second):
        conf_first_list.append(element1[0])
        conf_second_list.append(element2[0])

    # Добавление доверительных интервалов на график
    plt.errorbar(
        # [25%, 50%, 75%]
```

```
        [np.percentile(data, 25), np.percentile(data, 50),
np.percentile(data, 75)],
        [np.percentile(theoretical, 25), np.percentile(theoretical, 50),
np.percentile(theoretical, 75)],
        xerr=conf_first_list,
        yerr=conf_second_list,
        ls='none',
        capsize=3,
        elinewidth=2
    )

    plt.title(method_name)
    plt.show()


# ### Hospital beds per thousand

# #### Maximum likelihood method

# In[46]:


beds_theoretical_mle = gamma.rvs(*beds_mle, size=len(beds))
qq(beds, beds_theoretical_mle, 'Hospital beds per thousand', 'MLE')


# #### Least squares method

# In[47]:


beds_theoretical_lsm = gamma.rvs(*beds_lsm, size=len(beds))
qq(beds, beds_theoretical_lsm, 'Hospital beds per thousand', 'LSM')


# ### Excess mortality cumulative per million

# #### Maximum likelihood method

# In[48]:


emcpm_theoretical_mle = chi2.rvs(*emcpm_mle, size=len(emcpm))
qq(emcpm, emcpm_theoretical_mle, 'Excess mortality cumulative per
million', 'MLE')


# #### Least squares method

# In[49]:


emcpm_theoretical_lsm = chi2.rvs(*emcpm_lsm, size=len(emcpm))
qq(emcpm, emcpm_theoretical_lsm, 'Excess mortality cumulative per
million', 'LSM')


# ### Excess mortality cumulative

# #### Maximum likelihood method
```

```
# In[50]:


emc_theoretical_mle = norm.rvs(*emc_mle, size=len(emc))
qq(emc, emc_theoretical_mle, 'Excess mortality cumulative', 'MLE')


# #### Least squares method

# In[52]:


emc_theoretical_lsm = norm.rvs(*emc_lsm, size=len(emc))
qq(emc, emc_theoretical_lsm, 'Excess mortality cumulative', 'LSM')


# ## Step 7. Estimate correctness of fitted distributions using at least 2
statistical tests.

# In[34]:


def ks_test(data, distribution, params, alpha = 0.05, N = 50):
    ks = stats.kstest(data.sample(N), distribution, params, N)
    if ks[1] > alpha:
        print(f'Kolmogorov test passed. Under the null hypothesis,\n the
two {distribution} distributions are identical')
    else:
        print(f"Kolmogorov test failed. Hypothesis about {distribution} is
false")
    print(ks)

def cvm_test(data, distribution, params, alpha = 0.05):
    cvm = stats.cramervonmises(data.sample(50), distribution, params)
    if cvm.pvalue > alpha:
        print(f"Cramer-von-Mises test passed. Hypothesis about that
distribution \n have cumulative {distribution} distribution is true")
    else:
        print(f"Cramer-von-Mises failed. We reject the null hypothesis \n
that the observed sample is drawn from a {distribution} distribution")
    print(cvm)
    print('\n')


# In[35]:


def statistical_tests(data, distribution, params, alpha = 0.05, N = 50):
    ks_test(data, distribution, params, alpha, N)
    print('\n')
    cvm_test(data, distribution, params, alpha)


# ### Hospital beds per thousand

# In[62]:


statistical_tests(beds, 'gamma', beds_mle, alpha = 0.05, N = 50)
statistical_tests(beds, 'gamma', beds_lsm, alpha = 0.05, N = 50)
```

```
# ### Excess mortality cumulative per million

# In[60]:


statistical_tests(emcpm, 'chi2', emcpm_mle, alpha = 0.05, N = 50)
statistical_tests(emcpm, 'chi2', emcpm_lsm, alpha = 0.05, N = 50)


# ### Excess mortality cumulative

# In[61]:


statistical_tests(emc, 'norm', emc_mle, alpha = 0.05, N = 50)
statistical_tests(emc, 'norm', emc_lsm, alpha = 0.05, N = 50)
```