FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTE
OF HIGHER EDUCATION
ITMO UNIVERSITY

# Report on learning practice # 2

# Analysis of multivariate random variables

Performed by:

Igor Vernyy, j4134c

Kirill Mukhin, j4134c

Alexander Petrov, j4134c

Bogdan Chertkov, j4132c

Saint-Petersburg

2022

**Non-parametric estimation of PDF for MRV**

Figures 1 – 9 depict non-parametric estimations in form of histogram and using kernel density function for the chosen random variables.

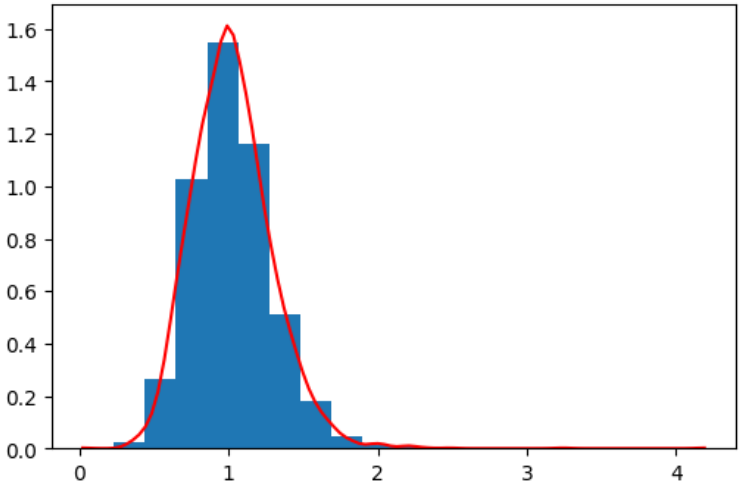


Figure 1 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “Reproduction rate”

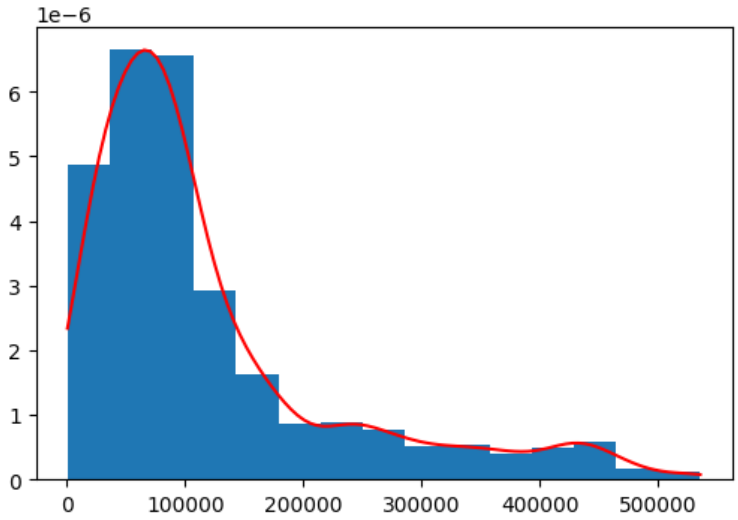


Figure 2 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “Total cases per million”

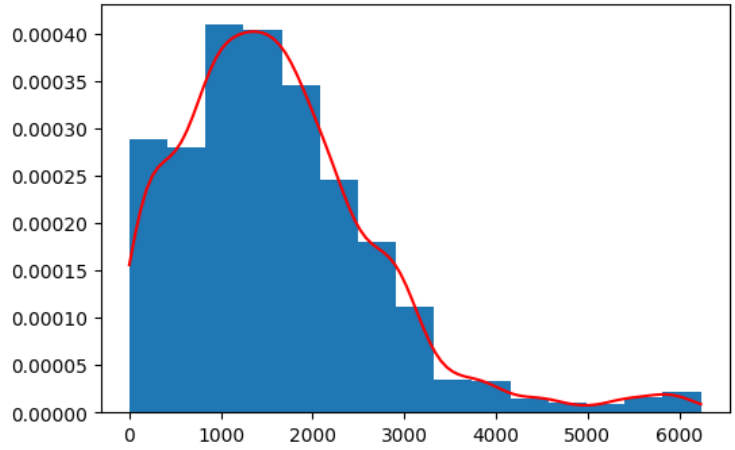


Figure 3 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “Total deaths per million”

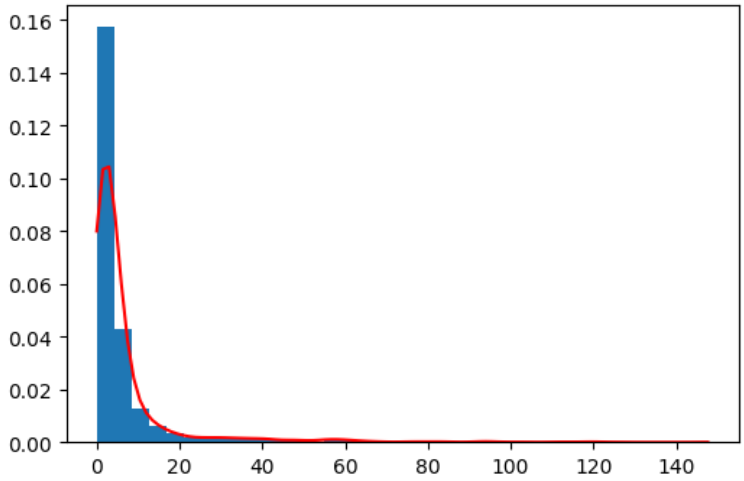


Figure 4 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “New tests smoothed per thousand”

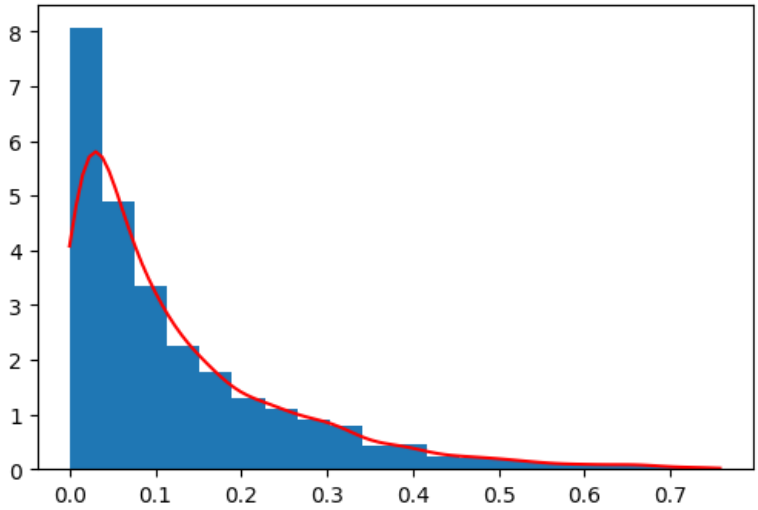


Figure 5 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “Positive rate”

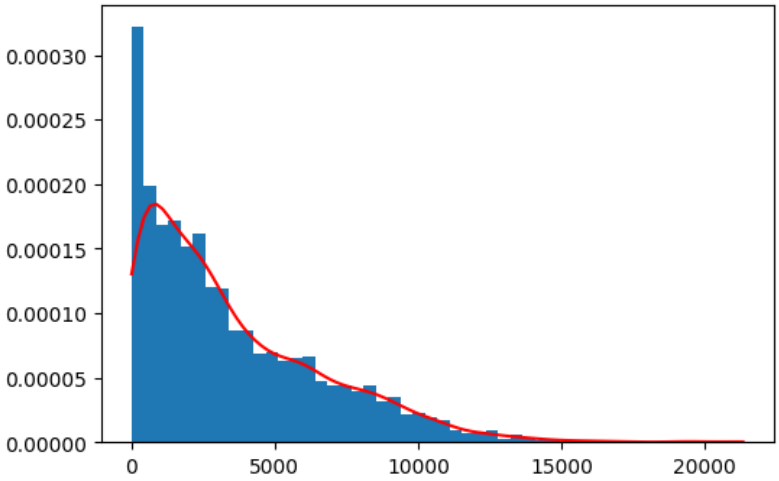


Figure 6 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “New vaccinations smoothed per million”

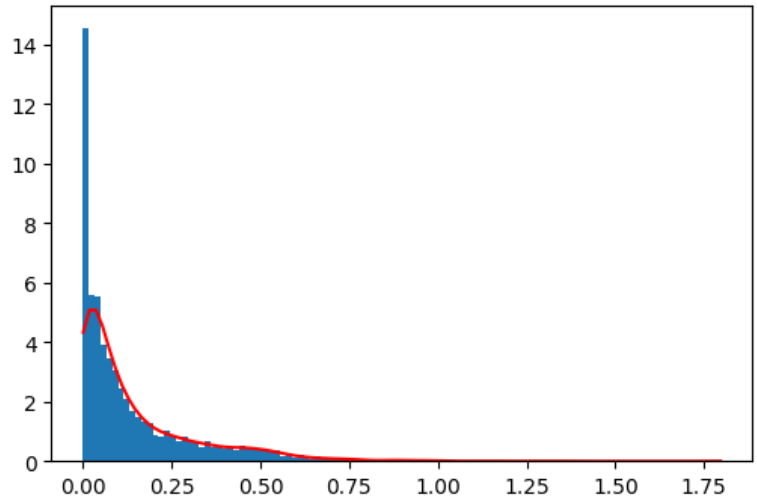


Figure 7 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “New people vaccinated smoothed per hundred”

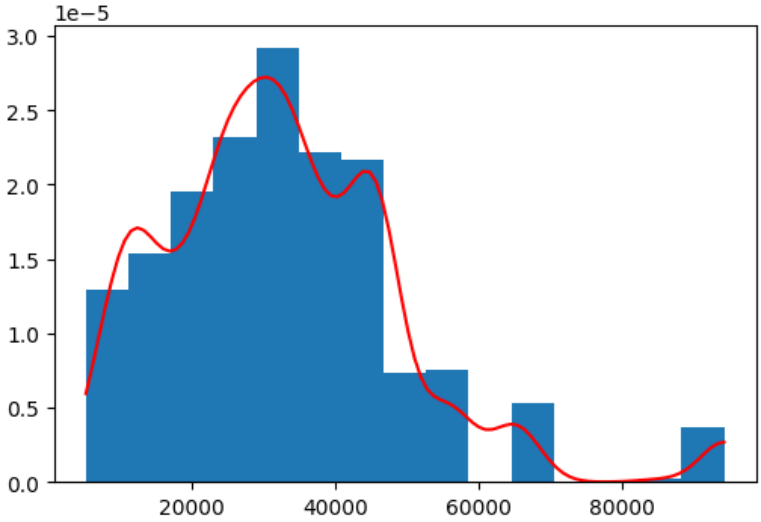


Figure 8 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “GDP per capita”

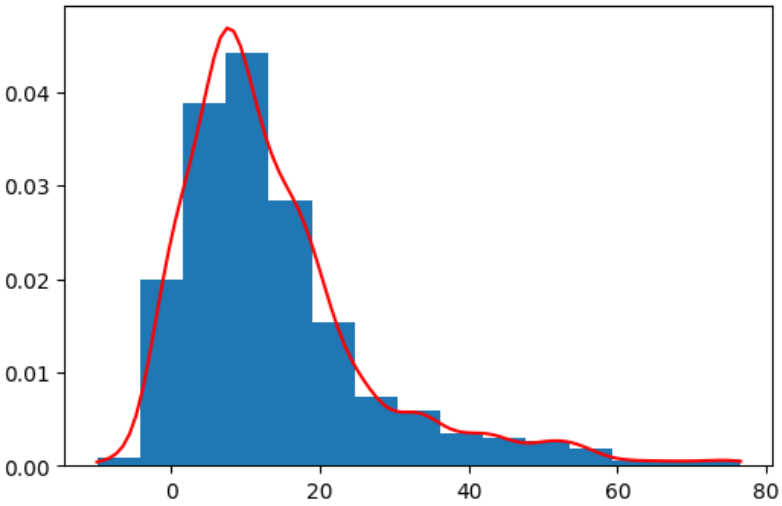


Figure 9 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “Excess mortality cumulative”

Among given above-mentioned random variables, the first one is a target and the rest are predictors. Figures 10 – 17 show us joint probability density plots in form of histogram for variations between target and each predictor. Each image also contains points of different color. Colors represent data for different continents.

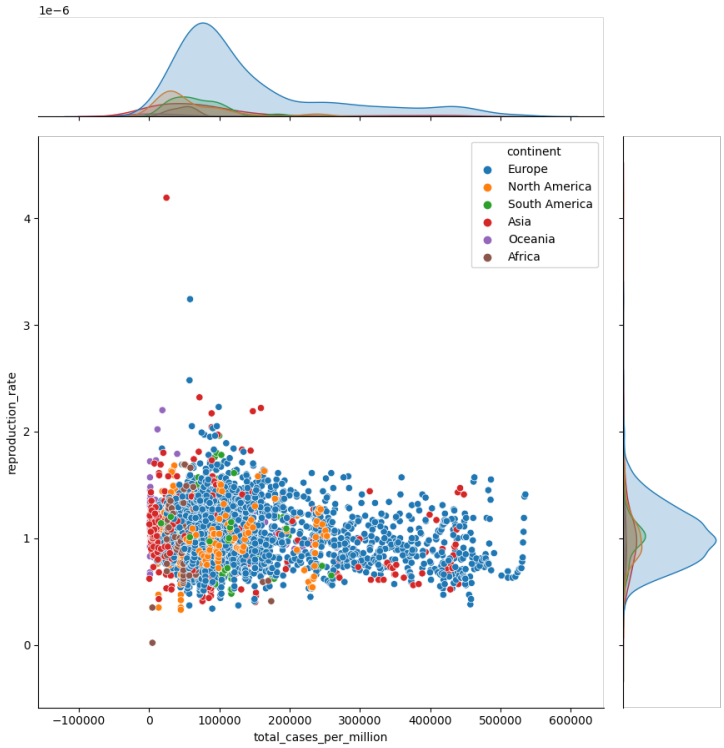


Figure 10 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “Total cases per million”

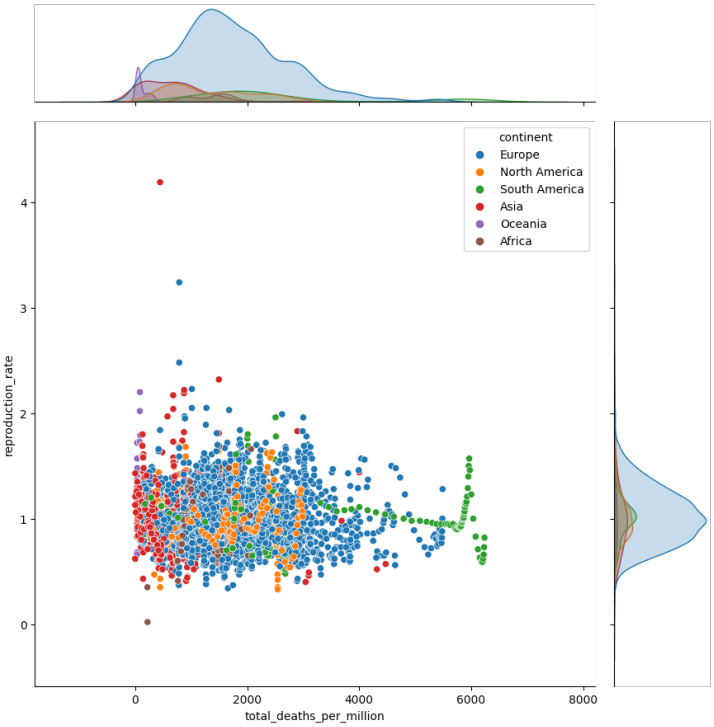


Figure 11 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “Total deaths per million”

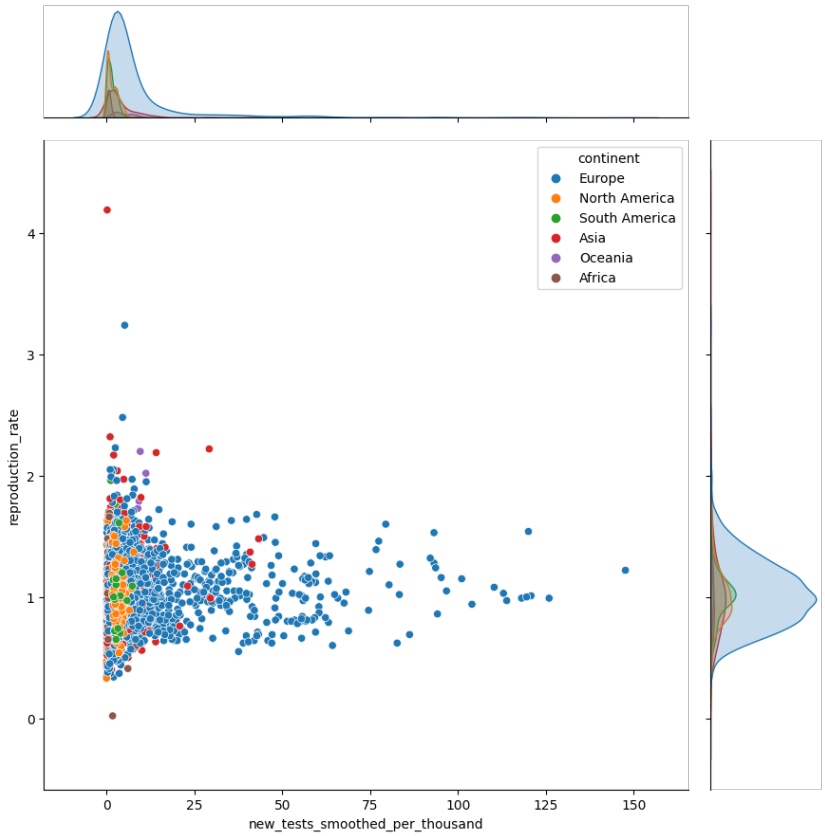


Figure 12 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “New tests smoothed per thousand”

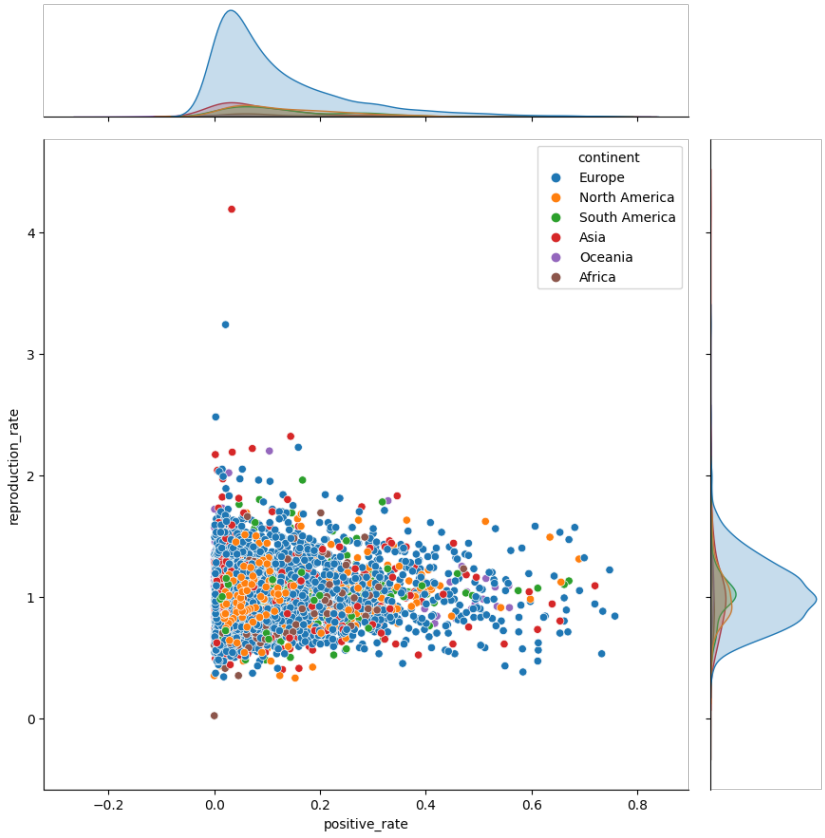


Figure 13 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “Positive rate”

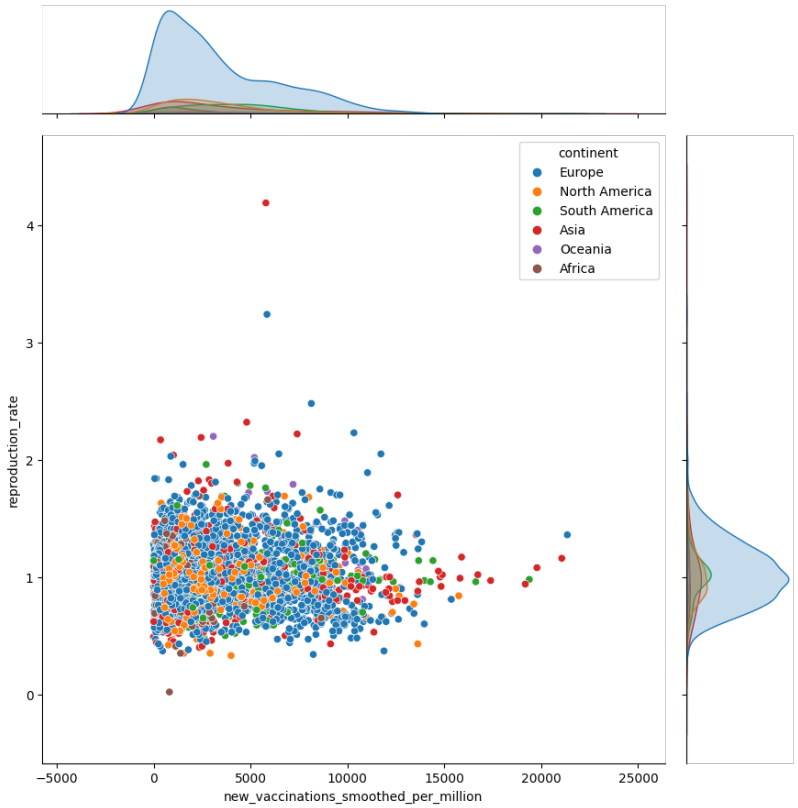


Figure 14 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “New vaccinations smoothed per million”

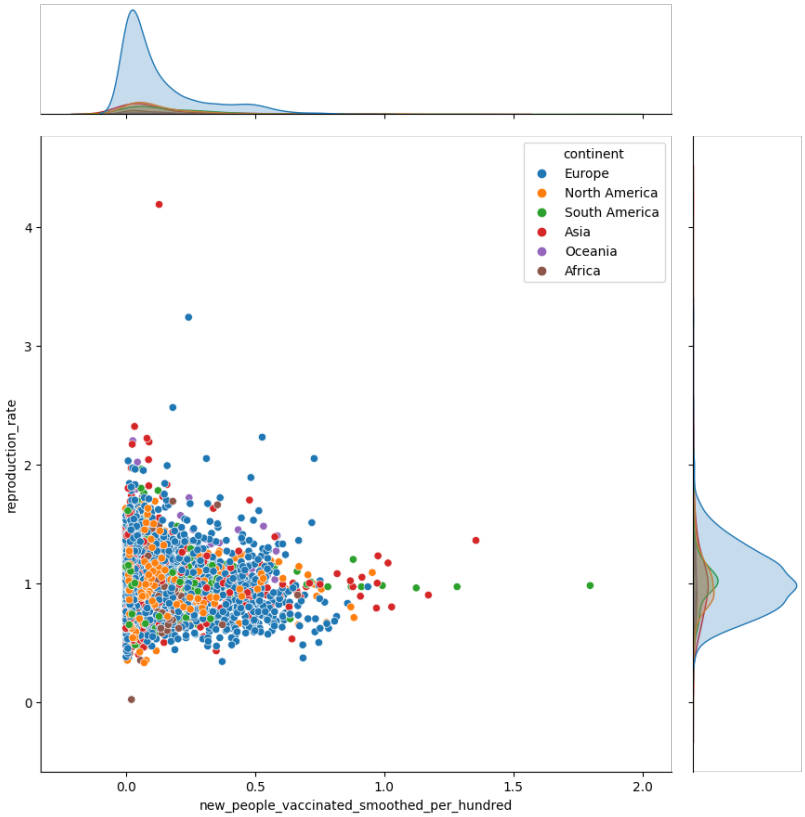


Figure 15 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “New people vaccinated smoothed per hundred”

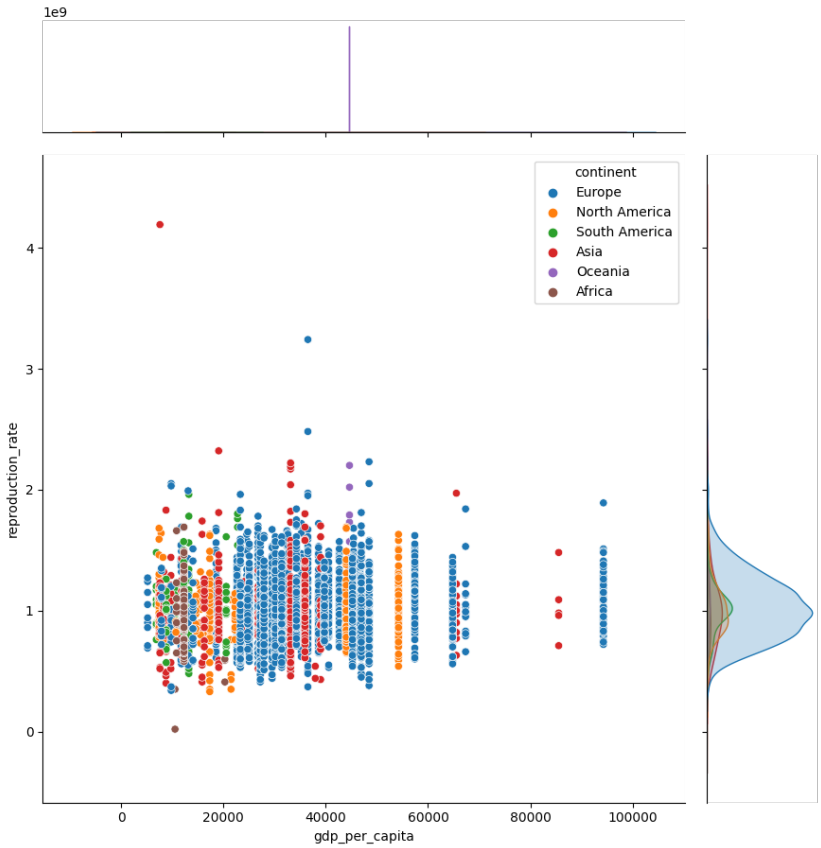


Figure 16 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “GDP per capita”

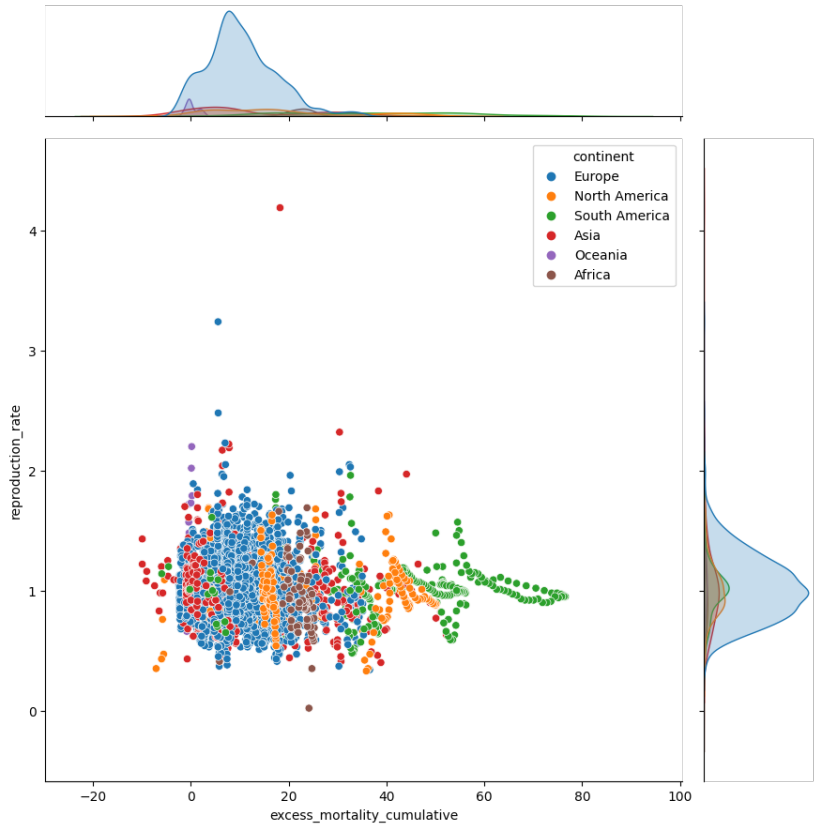


Figure 17 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “Excess mortality cumulative”

Figures 18 – 25 show us joint probability density plots in form of kernel density function for variations between target and each predictor. Each image also contains lines of different color. Colors represent data for different continents.

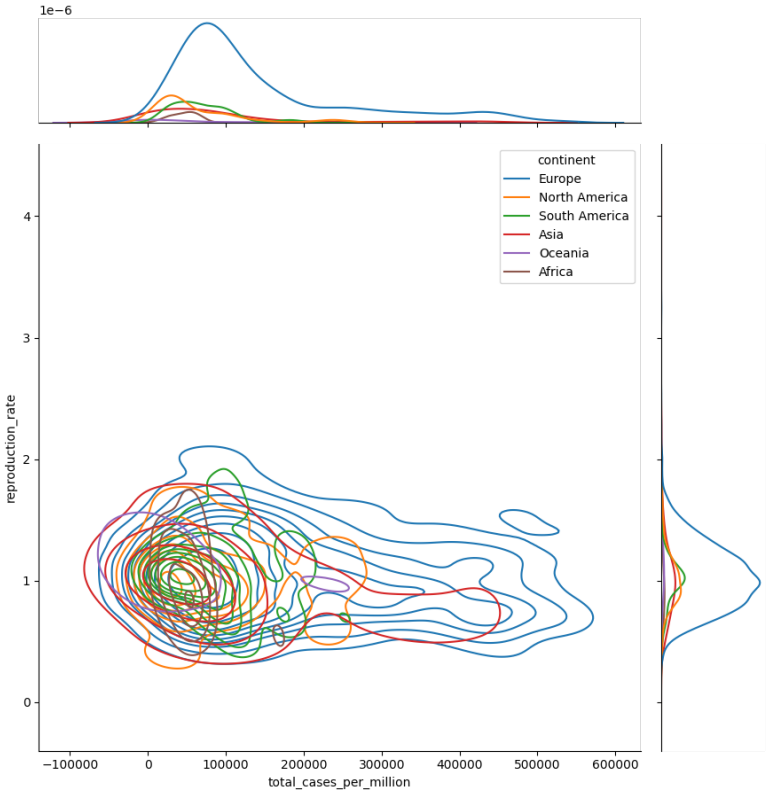


Figure 18 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “Total cases per million”

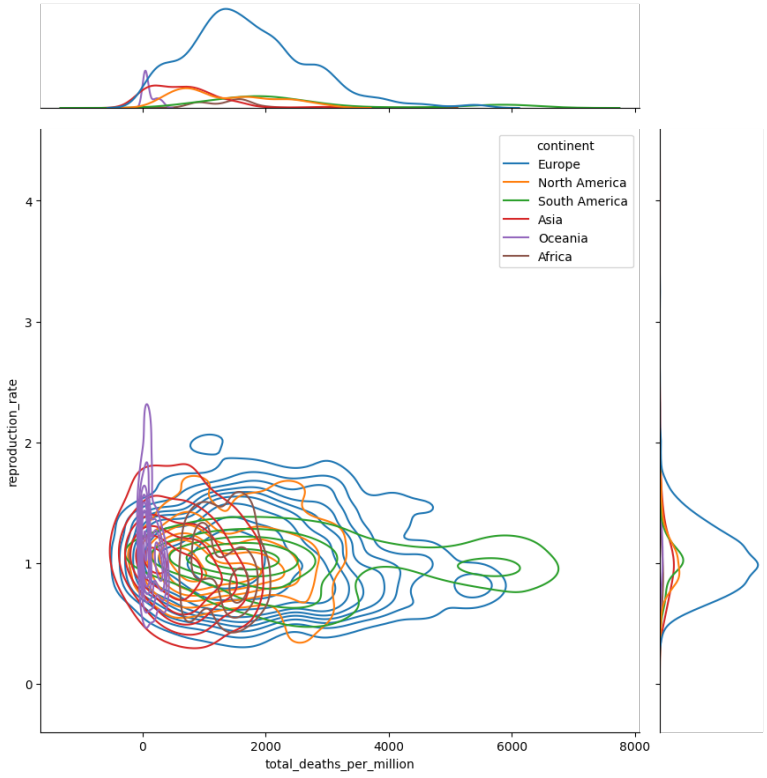


Figure 19 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “Total deaths per million”

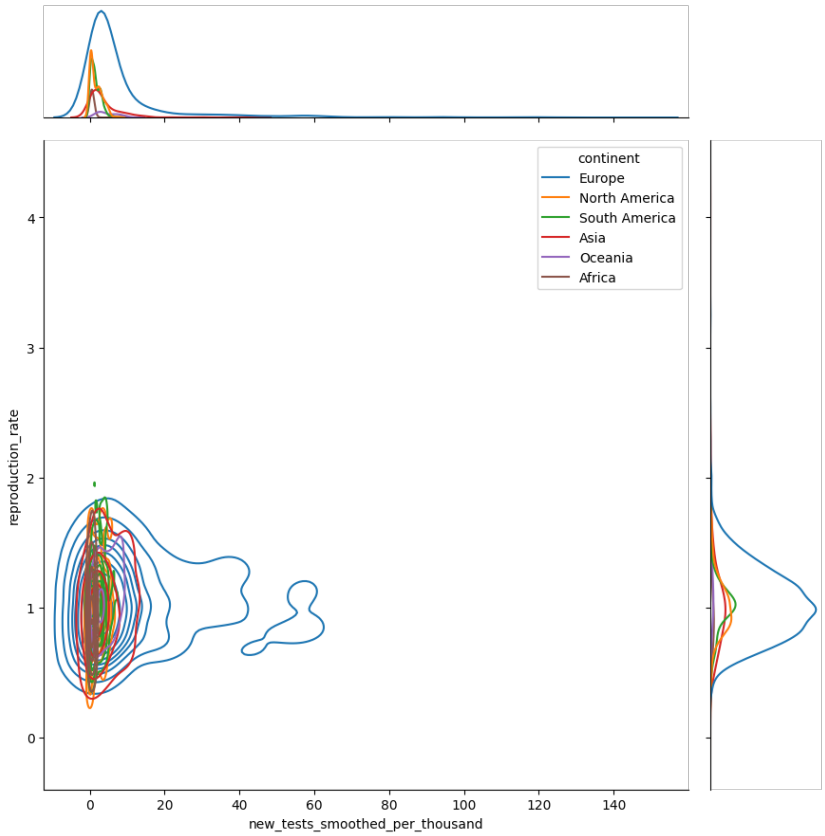


Figure 20 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “New tests smoothed per thousand”

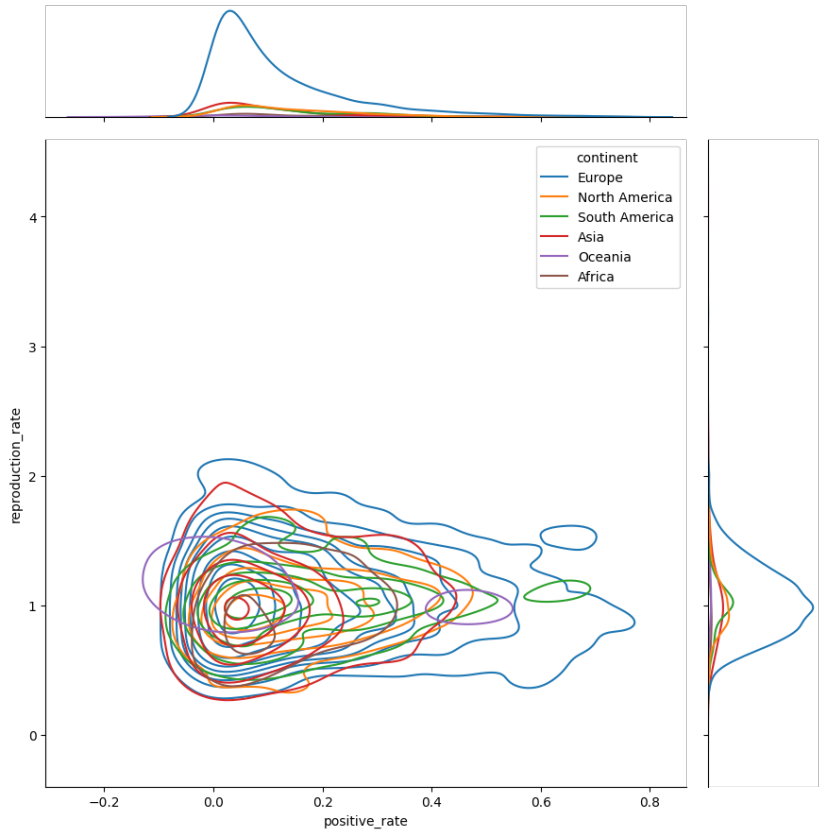


Figure 21 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “Positive rate”

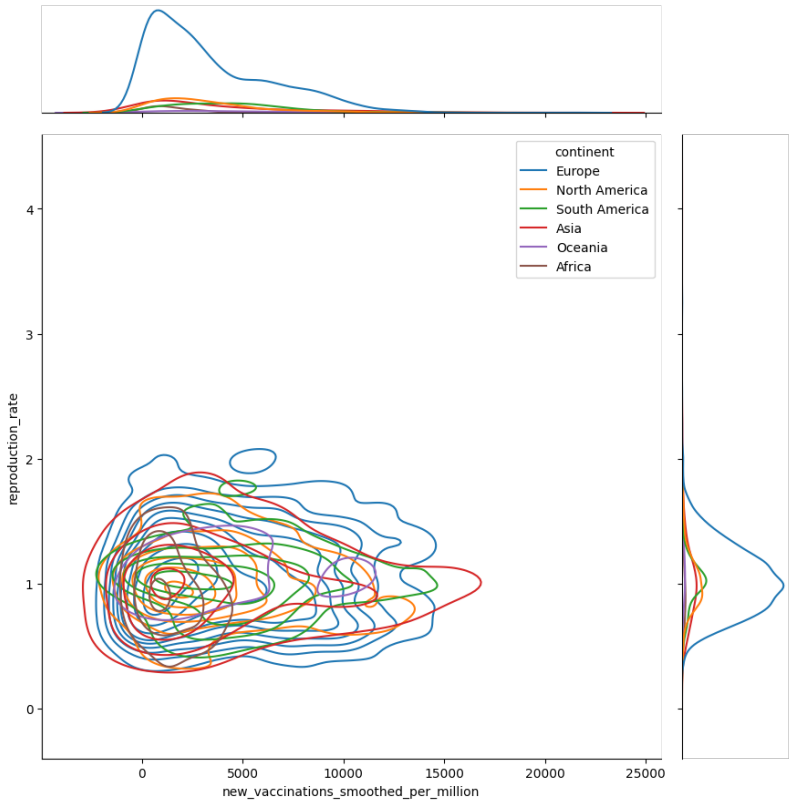


Figure 22 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “New vaccinations smoothed per million”

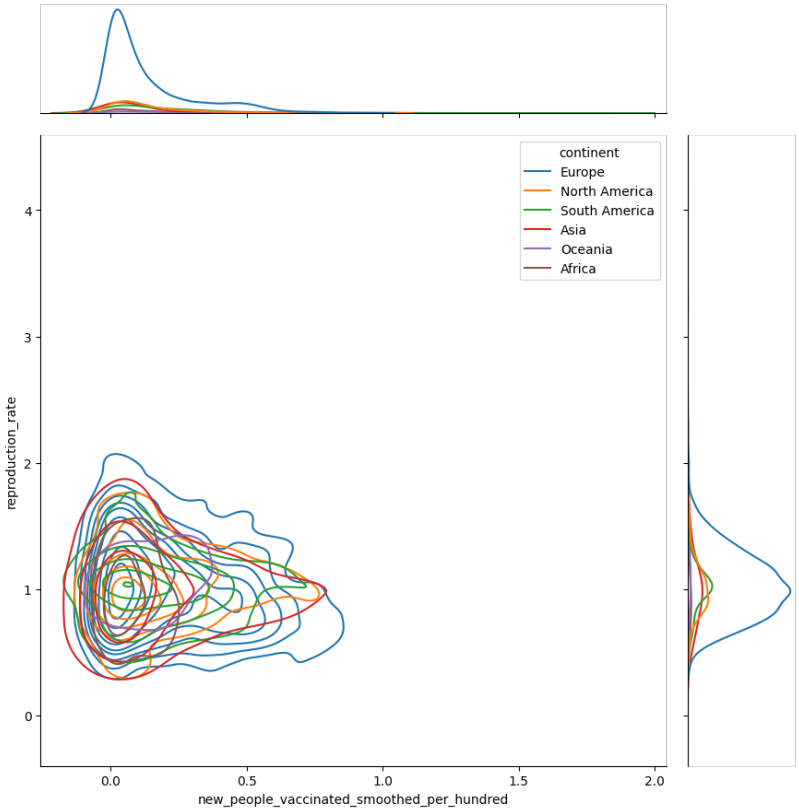


Figure 23 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “New people vaccinated smoothed per hundred”

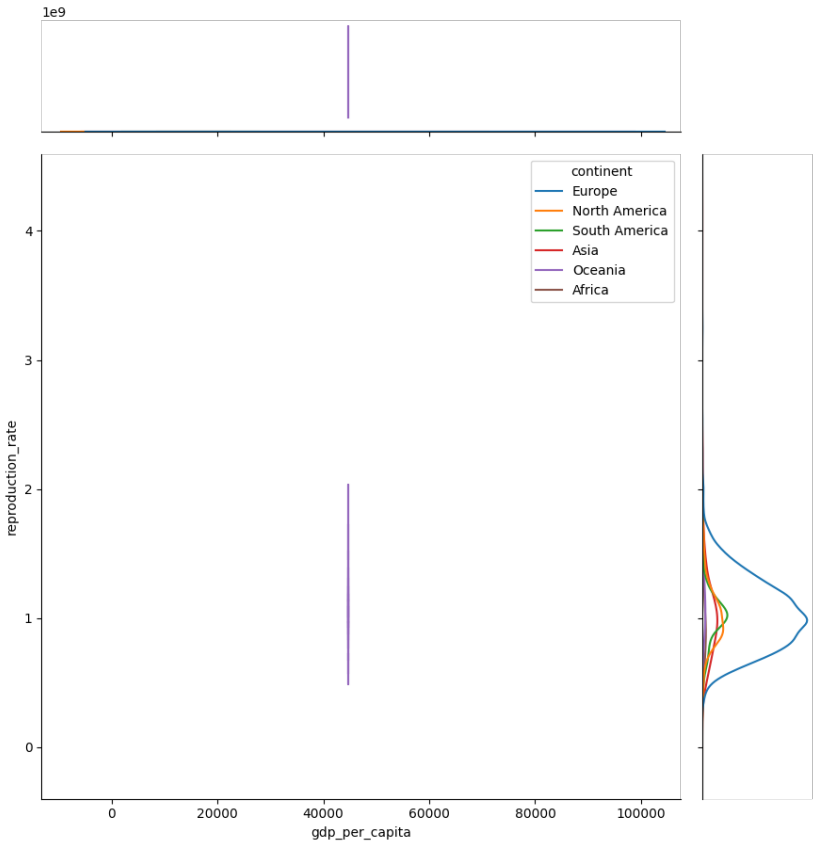


Figure 24 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “GDR per capita”

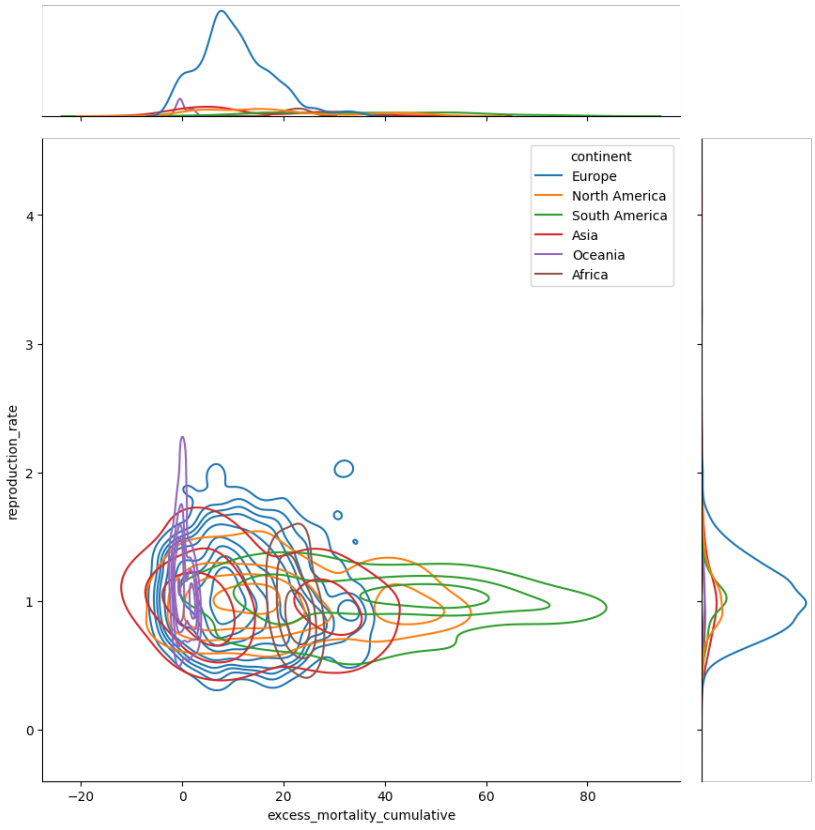


Figure 25 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “Excess mortality cumulative”

The next Figure 26 depicts pairwise joint probability density histogram between five variables.

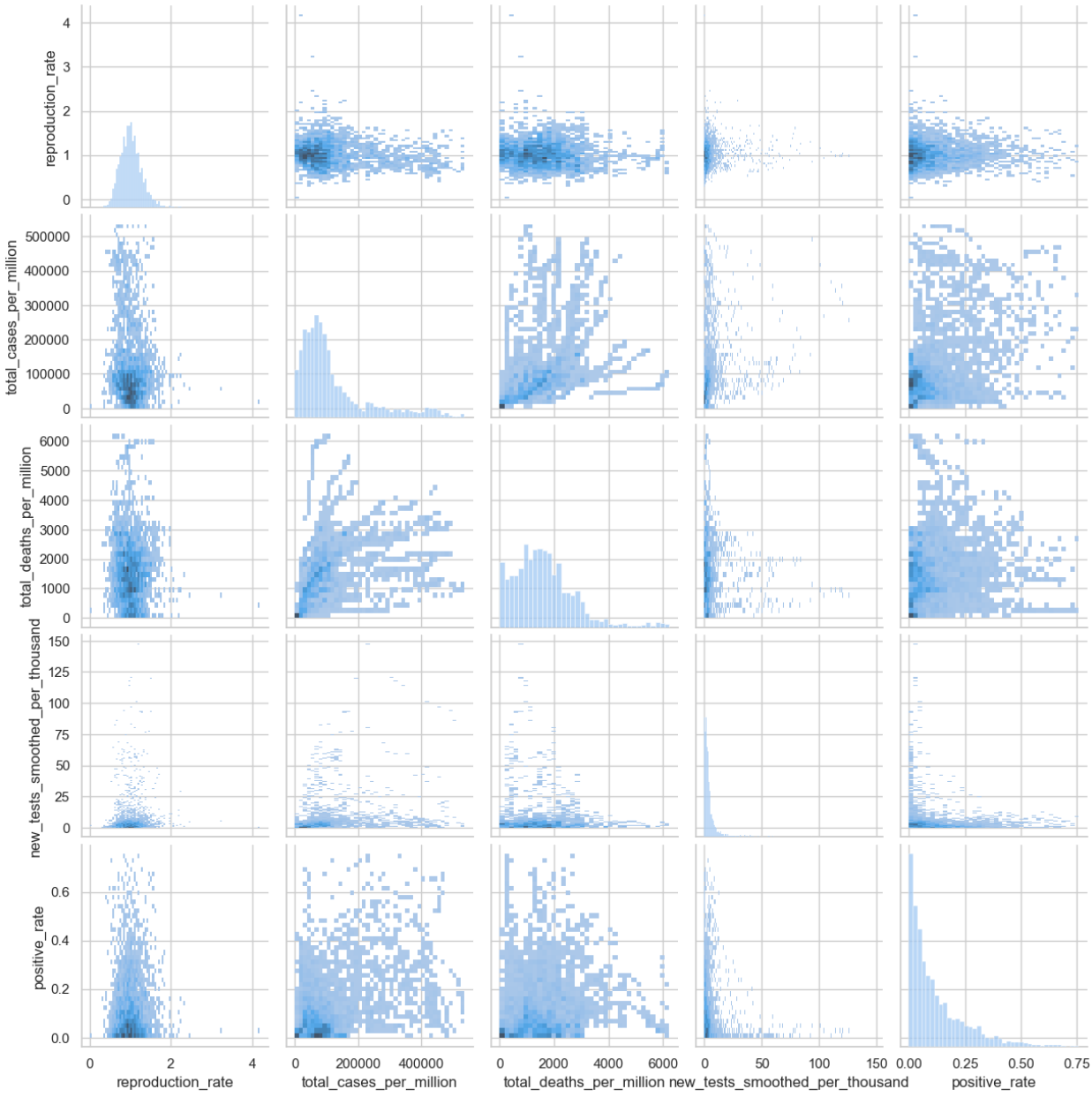


Figure 26 – Pairwise joint probability density between five random variables

**Estimation of multivariate mean and variance**

| $X$ | $E(X)$ | $Var(X)$ |
|---|---|---|
| Reproduction rate | 1.023385 | $7.484235 \cdot 10^{-2}$ |
| Total cases per million | 119878.364158 | $1.217849 \cdot 10^{10}$ |
| Total deaths per million | 1617.485002 | $1.206946 \cdot 10^{6}$ |
| New tests smoothed per thousand | 6.443801 | $1.595720 \cdot 10^{2}$ |
| Positive rate | 0.123713 | $1.702659 \cdot 10^{-2}$ |
| New vaccinations smoothed per million | 3508.352299 | $1.004960 \cdot 10^{7}$ |

| | | |
|---|---|---|
| New people vaccinated smoothed per hundred | 0.139452 | $3.113892 \cdot 10^{-2}$ |
| GDP per capita | 32880.455968 | $2.839982 \cdot 10^{8}$ |
| Excess mortality cumulative | 13.950828 | $1.721878 \cdot 10^{2}$ |

Table 1 – Estimation of multivariate mathematical expectation and variance for the chosen variables

**Non-parametric estimation of conditional distributions, mathematical expectations and variances**

Let condition $C$ be $Reproduction\,rate = 1.0$. So, Figure 27 depicts distribution of variables under this condition. And Table 2 contains conditional mathematical expectation and conditional variance.

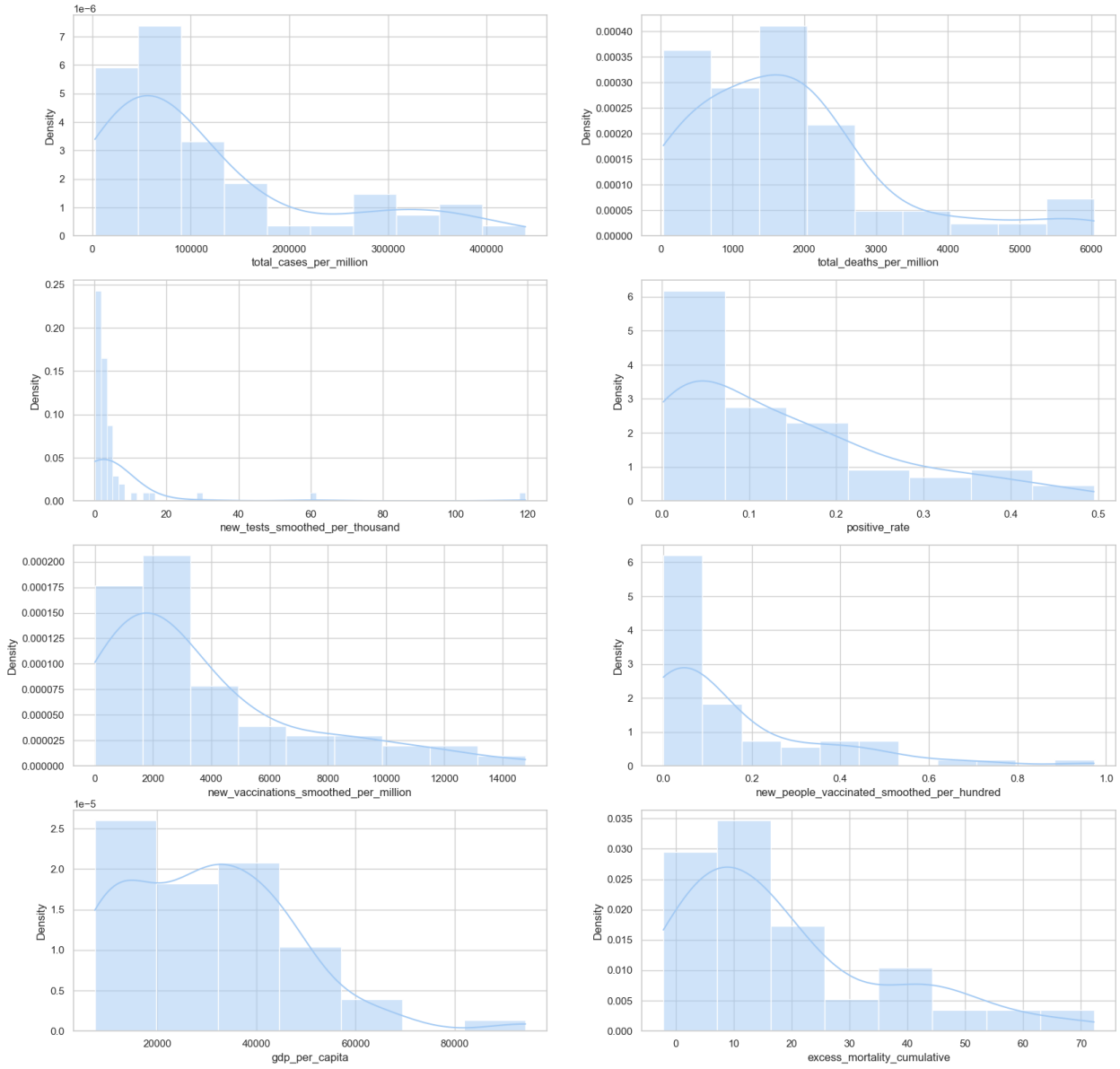


Figure 27 – Conditional distributions of variables

| $X$ | $E(X\|C)$ | $Var(X\|C)$ |
|---|---|---|
| Total cases per million | 113775.970403 | $1.250462 \cdot 10^{10}$ |
| Total deaths per million | 1743.478855 | $1.921855 \cdot 10^{6}$ |
| New tests smoothed per thousand | 6.184790 | $1.875900 \cdot 10^{2}$ |
| Positive rate | 0.135581 | $1.613816 \cdot 10^{-2}$ |
| New vaccinations smoothed per million | 3569.225806 | $1.184156 \cdot 10^{7}$ |
| New people vaccinated smoothed per hundred | 0.153016 | $4.130244 \cdot 10^{-2}$ |
| GDP per capita | 30048.161435 | $3.001686 \cdot 10^{8}$ |
| Excess mortality cumulative | 18.273065 | $3.198207 \cdot 10^{2}$ |

Table 2 – Conditional mathematical expectation and conditional variance for predictors

**Estimation of pair correlation coefficients, confidence intervals for them and significance levels**

Figure 28 shows pair correlation coefficients represented as a heatmap diagram. Table 3 contains information about confidence intervals for them and significance level.

| $X$ | $Y$ | Coefficient | Conf. interval | Signif. level |
|---|---|---|---|---|
| Total cases per million | Total deaths per million | 0.3592 | $(0.3427, 0.4092)$ | 0.000000 |
| New vaccinations smoothed per million | Excess mortality cumulative | $-0.0630$ | $(-0.0963, -0.0298)$ | 0.000202 |
| New people vaccinated smoothed per hundred | Excess mortality cumulative | $-0.0031$ | $(-0.0363,0.0302)$ | 0.856135 |
| GDP per capita | Excess mortality cumulative | $-0.5742$ | $(-0.6871, -0.6206)$ | 0.000000 |
| Total cases per million | New vaccinations smoothed per million | $-0.3183$ | $(-0.3629, -0.2965)$ | 0.000000 |

Table 3 – Pair correlation coefficients confidence intervals and significance levels

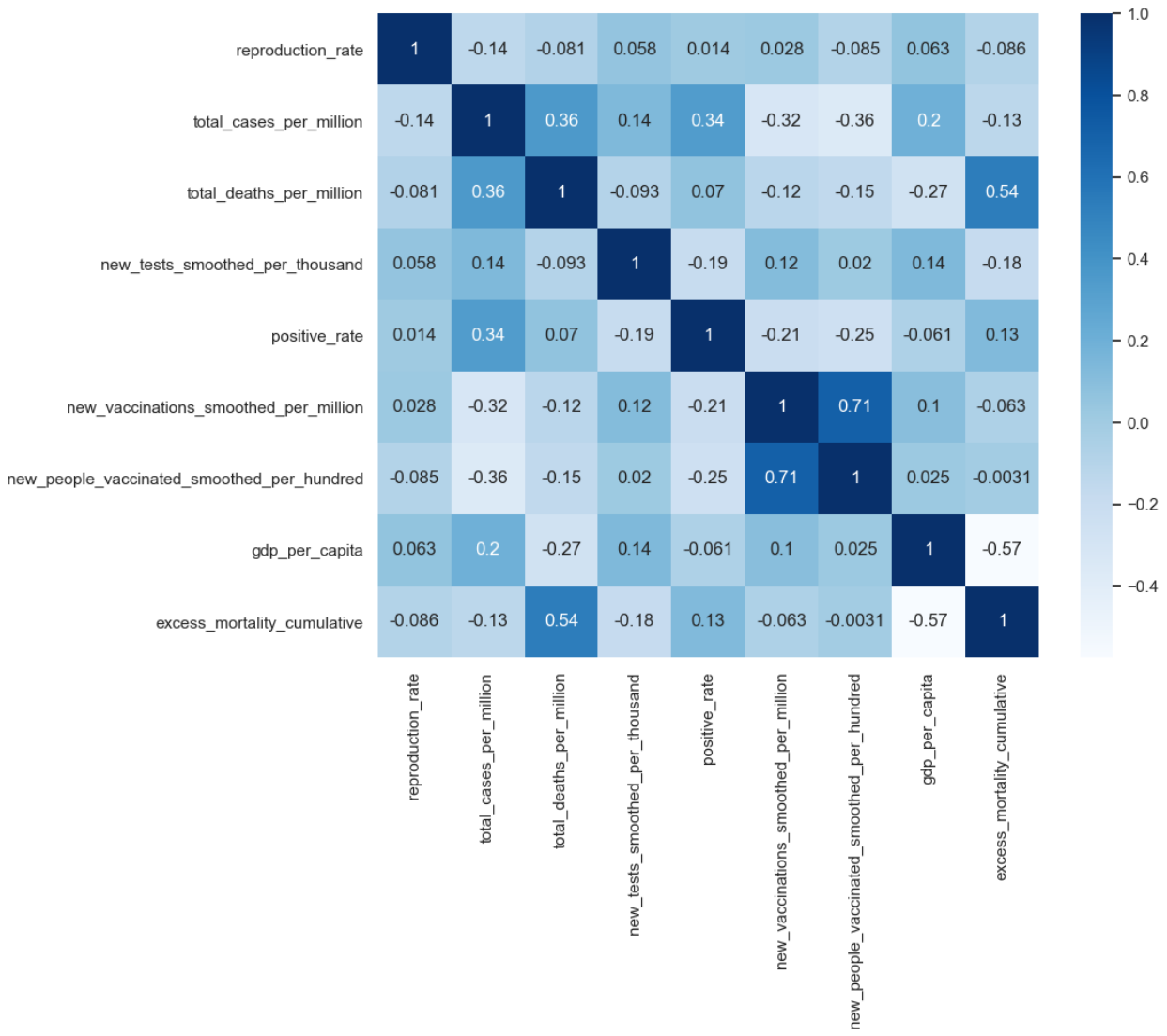


Figure 28 – Pair correlation coefficients

**Task formulation for regression, multivariate correlation**

Training a model for predicting “Reproduction rate” based on “Total cases per million”, “Total deaths per million”, “New tests smoothed per thousand”, “Positive rate”, “New vaccinations smoothed per million”, “New people vaccinated smoothed per hundred”, “GDP per capita”, “Excess mortality cumulative” variables.

**Regression model, multicollinearity and regularization**

We have built regression model. Table 4 consist of metrics obtained after fitting ordinal regression model.

| Metric | Value |
|---|---|
| Mean Absolute Error | 0.192338 |
| Mean Squared Error | 0.653560 |
| R2 Score | 0.063770 |
| Mean Absolute Percentage Error | 20.379919 |

Table 4 – Ordinal regression model metrics

After that we have applied Lasso regularization to model (with $\alpha = 0.1$). Table 5 consist of metrics obtained after fitting Lasso-regression model.

| Metric | Value |
|---|---|
| Mean Absolute Error | 0.193504 |
| Mean Squared Error | 0.067059 |
| R2 Score | 0.039364 |

Table 5 – Lasso-regression model metrics

Finally, a LARS regression was used. Table 6 shows its metrics.

| Metric | Value |
|---|---|
| Mean Absolute Error | 0.192338 |
| Mean Squared Error | 0.065356 |
| R2 Score | 0.063770 |

Table 6 – LARS model metrics

Also $2^{nd}$ degree polynomial regression models was fitted.

**Quality analysis for regression model**

Figure 29 illustrates real values and values predicted by linear model for "Reproduction rate" variable. Figure 30 contains Quantile-Quantile plot for this model. Distribution of residuals for linear model is depicted at Figure 31. Table 7 contains confident intervals for regression coefficients for linear regression.

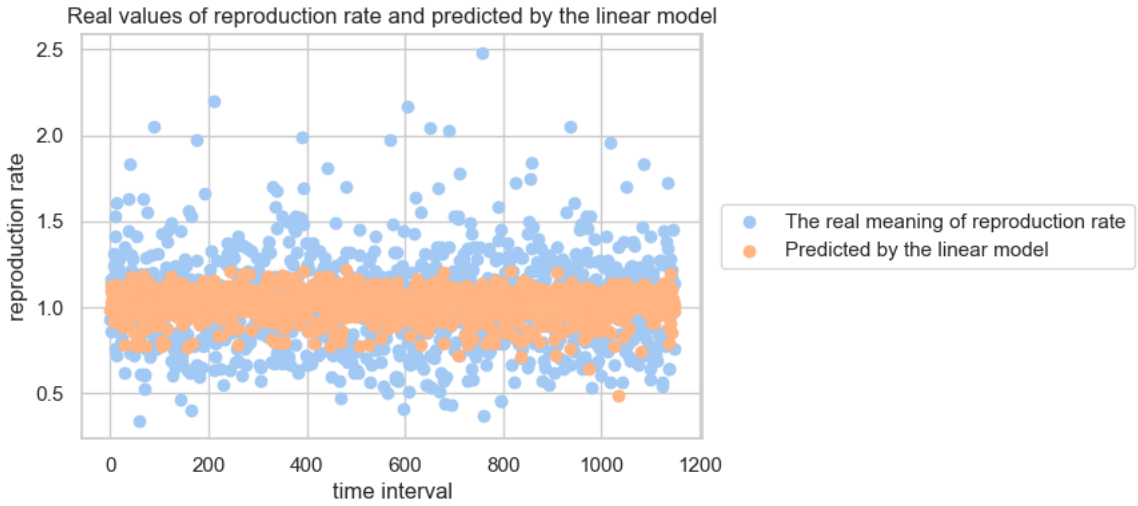


Figure 29 – Real and predicted points by linear model

Figure 32 illustrates real values and values predicted by polynomial model for "Reproduction rate" variable. Figure 33 contains Quantile-Quantile plot for this model.

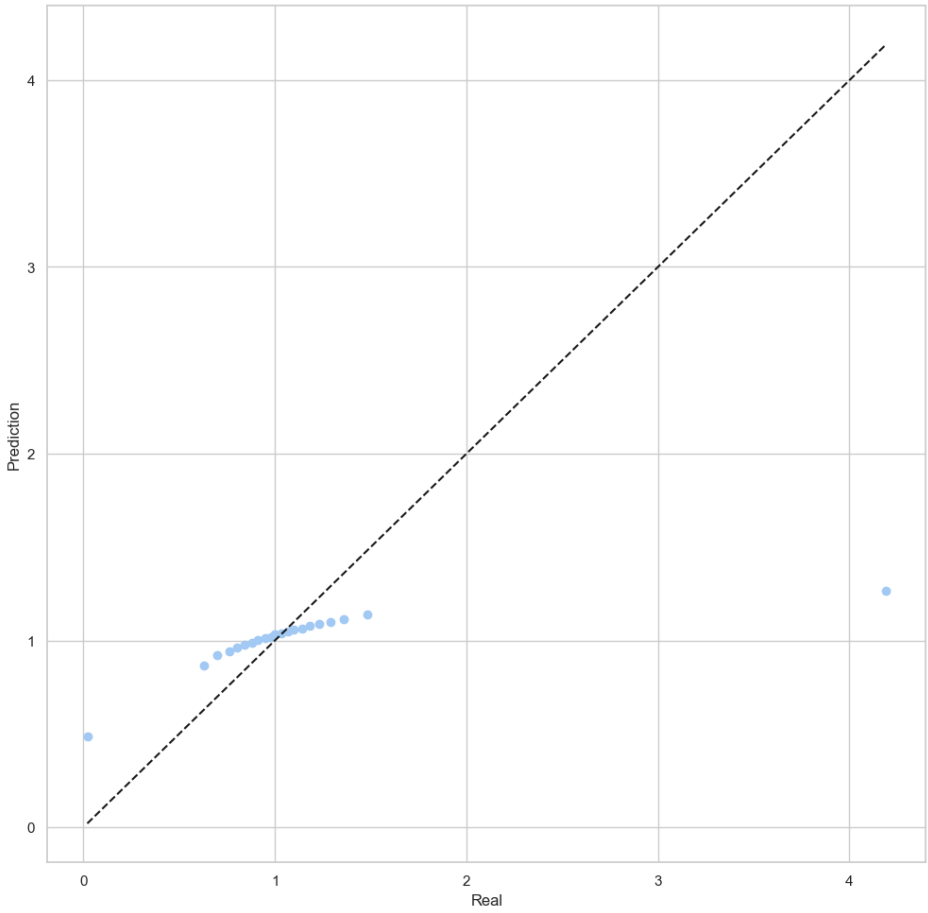


Figure 30 – QQ-plot for linear model

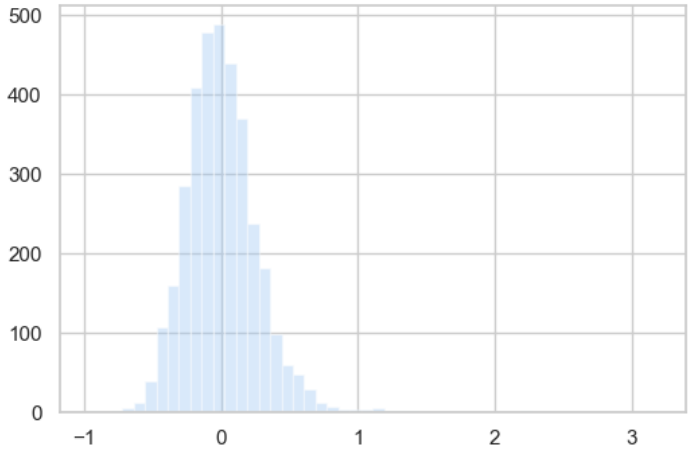


Figure 31 – Distribution of residuals for linear model

| $X$ | Interval left | Interval right |
|---|---|---|
| Total cases per million | $-8.315253 \cdot 10^{-7}$ | $-3.065669 \cdot 10^{-7}$ |
| Total deaths per million | $7.420195 \cdot 10^{-5}$ | $1.260864 \cdot 10^{-4}$ |
| New tests smoothed per thousand | $4.017558 \cdot 10^{-3}$ | $7.307548 \cdot 10^{-3}$ |
| Positive rate | $5.893598 \cdot 10^{-1}$ | $9.321734 \cdot 10^{-1}$ |
| New vaccinations smoothed per million | $1.731372 \cdot 10^{-5}$ | $3.541832 \cdot 10^{-5}$ |
| New people vaccinated smoothed per hundred | $-2.714395 \cdot 10^{-1}$ | $6.117958 \cdot 10^{-2}$ |
| GDP per capita | $1.425285 \cdot 10^{-5}$ | $1.633377 \cdot 10^{-5}$ |

| | | |
|---|---|---|
| Excess mortality cumulative | $8.211649 \cdot 10^{-3}$ | $1.238924 \cdot 10^{-2}$ |

Table 7 – Confidence interval of regression coefficients

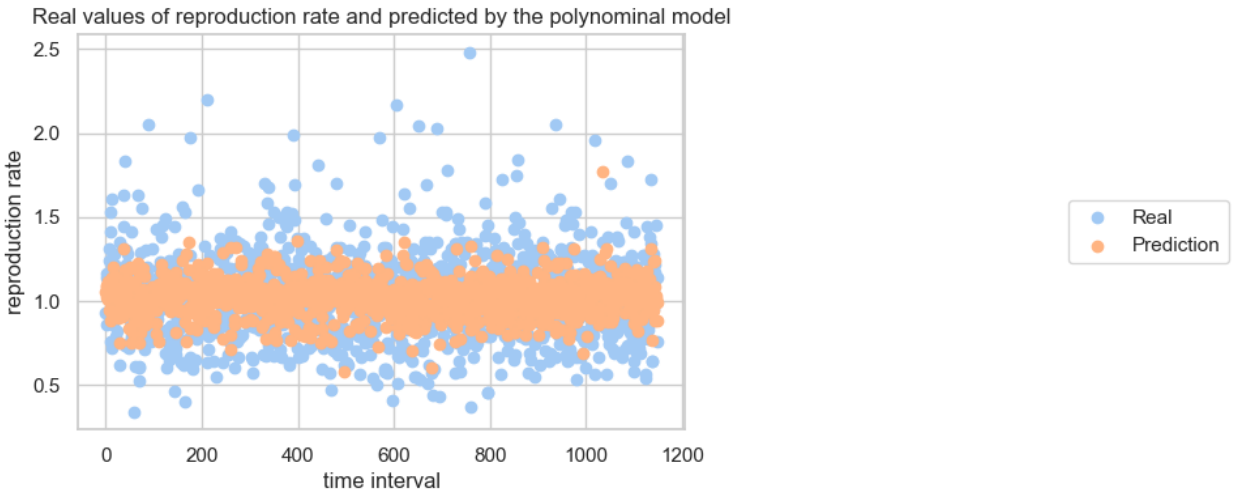


Figure 32 – Real and predicted points by polynomial model

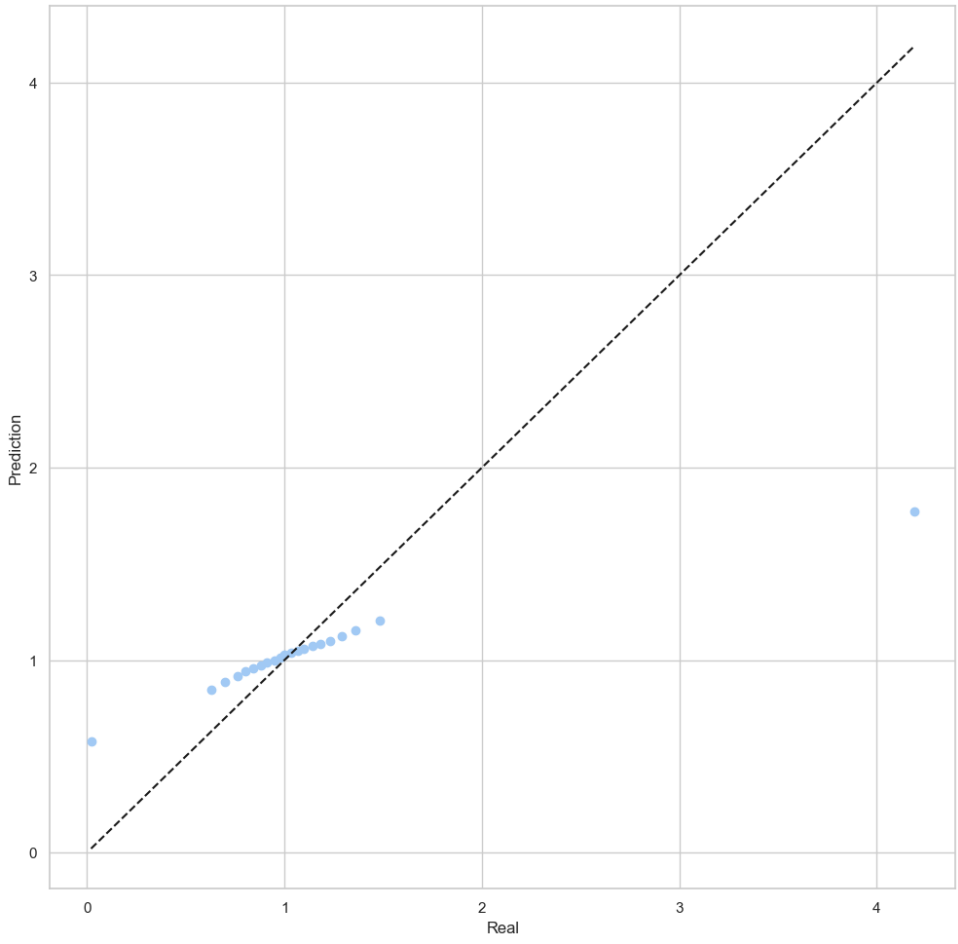


Figure 33 – QQ-plot for polynomial model

## Appendix

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

# In[1]:


import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy.stats
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lars
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import ElasticNet
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.linear_model import  LassoLarsIC


# In[2]:


pd.set_option('display.max_columns', 70)


# In[3]:


source_df = pd.read_csv('owid-covid-data.csv')
source_df


# ## 1. Make a non-parametric estimation of PDF in form of histogram and
using kernel density function for MRV (or probability law in case of
discrete MRV).

# Dataset content:
# - total_cases_per_million - Total confirmed cases of COVID-19 per
1,000,000 people
# - total_deaths_per_million - Total deaths attributed to COVID-19 per
1,000,000 people
# - reproduction_rate - Real-time estimate of the effective reproduction
rate (R) of COVID-19
# - new_tests_smoothed_per_thousand - New tests for COVID-19 (7-day
smoothed) per 1,000 people
# - positive_rate - The share of COVID-19 tests that are positive, given
as a rolling 7-day average (this is the inverse of tests_per_case)
# - new_vaccinations_smoothed_per_million - New COVID-19 vaccination doses
administered (7-day smoothed) per 1,000,000 people in the total population
# - new_people_vaccinated_smoothed_per_hundred - Daily number of people
receiving their first vaccine dose (7-day smoothed) per 100 people in the
total population
# - gdp_per_capita - Share of the population that is 70 years and older in
2015
```

```
# - excess_mortality_cumulative - Percentage difference between the
cumulative number of deaths since 1 January 2020 and the cumulative
projected deaths for the same period based on previous years
#

# In[4]:


source_df.continent.unique()


# In[5]:


columns = ['continent',
           'reproduction_rate',
           'total_cases_per_million',
            'total_deaths_per_million',
            'new_tests_smoothed_per_thousand',
            'positive_rate',
            'new_vaccinations_smoothed_per_million',
            'new_people_vaccinated_smoothed_per_hundred',
            'gdp_per_capita',
            'excess_mortality_cumulative']
df = source_df[columns].copy().dropna()
df.index = np.arange(0, len(df))
df


# In[6]:


target = columns[1]
predictors = columns[2:]
categorial = columns[0]


# In[7]:


df.describe()


# In[8]:


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-')


# In[9]:


cases = df['total_cases_per_million']
kernel_density_estimation(cases, 15)
```

```
# In[10]:


deaths = df['total_deaths_per_million']
kernel_density_estimation(deaths, 15)


# In[11]:


reproduction = df['reproduction_rate']
kernel_density_estimation(reproduction, 20)


# In[12]:


tests = df['new_tests_smoothed_per_thousand']
kernel_density_estimation(tests, 35)


# In[13]:


positive = df['positive_rate']
kernel_density_estimation(positive, 20)


# In[14]:


vaccinations = df['new_vaccinations_smoothed_per_million']
kernel_density_estimation(vaccinations, 50)


# In[15]:


vac_people = df['new_people_vaccinated_smoothed_per_hundred']
kernel_density_estimation(vac_people, 110)


# In[16]:


gdp = df['gdp_per_capita']
kernel_density_estimation(gdp, 15)


# In[17]:


emc = df['excess_mortality_cumulative']
kernel_density_estimation(emc, 15)


# In[18]:
```

```
for predictor in predictors:
    sns.jointplot(data=df, x=predictor, y=target, hue=categorial,
height=9)
    plt.plot()


# In[19]:


for predictor in predictors:
    sns.jointplot(data=df, x=predictor, y=target, kind='kde',
hue=categorial, height=9)
    plt.plot()


# In[20]:


# sns.pairplot(data=df[predictors + [target]], kind='kde',
plot_kws={'levels':[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]})


# In[21]:


sns.set_theme(style='whitegrid', palette='pastel')

ax = sns.pairplot(df[['reproduction_rate',
           'total_cases_per_million',
            'total_deaths_per_million',
            'new_tests_smoothed_per_thousand',
            'positive_rate']], kind='hist', diag_kind='hist')
plt.show()


# In[22]:


# ax = sns.pairplot(df[['reproduction_rate',
#            'total_cases_per_million',
#             'total_deaths_per_million',
#             'new_tests_smoothed_per_thousand',
#             'positive_rate']], diag_kind='kde')
# ax.map_lower(sns.kdeplot, levels=5, color='.1')

# plt.show()


# ## 2. Estimation of Multivariate Mathematical Expectation and Variance

# In[53]:


df[columns].mean()


# In[54]:


df[columns].var()
```

```
# ## 3. Non-parametric estimation of conditional distributions,
mathematical expectations and variances

# In[25]:


df_conditional = df[df.reproduction_rate == 1]
df_conditional


# In[26]:


df_conditional[['total_cases_per_million',
            'total_deaths_per_million',
            'new_tests_smoothed_per_thousand',
            'positive_rate',
            'new_vaccinations_smoothed_per_million',
            'new_people_vaccinated_smoothed_per_hundred',
            'gdp_per_capita',
            'excess_mortality_cumulative']].mean()


# In[27]:


df_conditional[['total_cases_per_million',
            'total_deaths_per_million',
            'new_tests_smoothed_per_thousand',
            'positive_rate',
            'new_vaccinations_smoothed_per_million',
            'new_people_vaccinated_smoothed_per_hundred',
            'gdp_per_capita',
            'excess_mortality_cumulative']].var()


# In[28]:


figure, ax = plt.subplots(4, 2, figsize=(20, 20))
sns.set_theme(style='whitegrid', palette='pastel')

total_cases_per_million = sns.histplot(df.total_cases_per_million,
ax=ax[0, 0], kde=True, stat='density')
total_cases_per_million.set(xlabel='total_cases_per_million')

total_deaths_per_million = sns.histplot(df.total_deaths_per_million,
ax=ax[0, 1], kde=True, stat='density')
total_deaths_per_million.set(xlabel='total_deaths_per_million')

new_tests_smoothed_per_thousand =
sns.histplot(df.new_tests_smoothed_per_thousand, ax=ax[1, 0], kde=True,
stat='density')
new_tests_smoothed_per_thousand.set(xlabel='new_tests_smoothed_per_thousan
d')

positive_rate = sns.histplot(df.positive_rate, ax=ax[1, 1], kde=True,
stat='density')
positive_rate.set(xlabel='positive_rate')
```

```
new_vaccinations_smoothed_per_million =
sns.histplot(df.new_vaccinations_smoothed_per_million, ax=ax[2, 0],
kde=True, stat='density')
new_vaccinations_smoothed_per_million.set(xlabel='new_vaccinations_smoothe
d_per_million')

new_people_vaccinated_smoothed_per_hundred =
sns.histplot(df.new_people_vaccinated_smoothed_per_hundred, ax=ax[2, 1],
kde=True, stat='density')
new_people_vaccinated_smoothed_per_hundred.set(xlabel='new_people_vaccinat
ed_smoothed_per_hundred')

gdp_per_capita = sns.histplot(df.gdp_per_capita, ax=ax[3, 0], kde=True,
stat='density')
gdp_per_capita.set(xlabel='gdp_per_capita')

excess_mortality_cumulative = sns.histplot(df.excess_mortality_cumulative,
ax=ax[3, 1], kde=True, stat='density')
excess_mortality_cumulative.set(xlabel='excess_mortality_cumulative')

plt.show()


# In[29]:


figure, ax = plt.subplots(4, 2, figsize=(20, 20))
sns.set_theme(style='whitegrid', palette='pastel')

total_cases_per_million =
sns.histplot(df_conditional.total_cases_per_million, ax=ax[0, 0],
kde=True, stat='density')
total_cases_per_million.set(xlabel='total_cases_per_million')

total_deaths_per_million =
sns.histplot(df_conditional.total_deaths_per_million, ax=ax[0, 1],
kde=True, stat='density')
total_deaths_per_million.set(xlabel='total_deaths_per_million')

new_tests_smoothed_per_thousand =
sns.histplot(df_conditional.new_tests_smoothed_per_thousand, ax=ax[1, 0],
kde=True, stat='density')
new_tests_smoothed_per_thousand.set(xlabel='new_tests_smoothed_per_thousan
d')

positive_rate = sns.histplot(df_conditional.positive_rate, ax=ax[1, 1],
kde=True, stat='density')
positive_rate.set(xlabel='positive_rate')

new_vaccinations_smoothed_per_million =
sns.histplot(df_conditional.new_vaccinations_smoothed_per_million,
ax=ax[2, 0], kde=True, stat='density')
new_vaccinations_smoothed_per_million.set(xlabel='new_vaccinations_smoothe
d_per_million')

new_people_vaccinated_smoothed_per_hundred =
sns.histplot(df_conditional.new_people_vaccinated_smoothed_per_hundred,
ax=ax[2, 1], kde=True, stat='density')
new_people_vaccinated_smoothed_per_hundred.set(xlabel='new_people_vaccinat
ed_smoothed_per_hundred')
```

```
gdp_per_capita = sns.histplot(df_conditional.gdp_per_capita, ax=ax[3, 0],
kde=True, stat='density')
gdp_per_capita.set(xlabel='gdp_per_capita')


excess_mortality_cumulative =
sns.histplot(df_conditional.excess_mortality_cumulative, ax=ax[3, 1],
kde=True, stat='density')
excess_mortality_cumulative.set(xlabel='excess_mortality_cumulative')

plt.show()


# ## 4. Estimation of pair correlation coefficients, confidence intervals
for them and significance levels

# In[55]:


figure, ax = plt.subplots(1, 1, figsize=(10, 8))
sns.set_theme(style='whitegrid', palette='pastel')

sns.heatmap(df[columns].corr(), cmap='Blues', annot=True)

plt.show()


# In[31]:


def return_estimation(array, target, alpha=0.05):
    r, p_value = scipy.stats.pearsonr(array, target)

    r_to_z = np.arctanh(r) # Matches Fisher transform
    se = 1 / np.sqrt(array.size - 3) # Corresponding standard deviation
    z = scipy.stats.norm.ppf(1 - alpha / 2)
    lo_z, hi_z = r_to_z-z*se, r_to_z+z*se
#   lo_z, hi_z = np.tanh(lo_z), np.tanh(hi_z)

    print('Correlation Coefficient: {:.4f}'.format(r))
    print('Confidence Interval for the correlation coefficient: ({:.4f},
{:.4f})'.format(lo_z, hi_z))
    print('Significance Level: {:.6f}'.format(p_value))


# In[32]:


return_estimation(df.total_cases_per_million, df.total_deaths_per_million)


# In[33]:


return_estimation(df.new_vaccinations_smoothed_per_million,
df.excess_mortality_cumulative)


# In[34]:
```

```
return_estimation(df.new_people_vaccinated_smoothed_per_hundred,
df.excess_mortality_cumulative)


# In[35]:


return_estimation(df.gdp_per_capita, df.excess_mortality_cumulative)


# In[36]:


return_estimation(df.total_cases_per_million,
df.new_vaccinations_smoothed_per_million)


# ## 5. Task formulation for regression. Estimate multivariate correlation
(target - predictors)

# Training a model for predicting reproduction_rate based on
total_cases_per_million, total_deaths_per_million,
new_tests_smoothed_per_thousand, positive_rate,
new_vaccinations_smoothed_per_million,
new_people_vaccinated_smoothed_per_hundred, gdp_per_capita,
excess_mortality_cumulative variables.

# In[56]:


figure, ax = plt.subplots(1, 1, figsize=(10, 8))
sns.set_theme(style='whitegrid', palette='pastel')

sns.heatmap(df[columns].corr(), cmap='Blues', annot=True)

plt.show()


# ## 6. Build regression model and make an analysis of multicollinearity
and regularization (if needed)

# In[38]:


# Highlight predictors
X = df[['total_cases_per_million',
            'total_deaths_per_million',
            'new_tests_smoothed_per_thousand',
            'positive_rate',
            'new_vaccinations_smoothed_per_million',
            'new_people_vaccinated_smoothed_per_hundred',
            'gdp_per_capita',
            'excess_mortality_cumulative']]
# Allocate the target variable
y = df[['reproduction_rate']]
# Division into training and test samples
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=42)

x = []
for i in range(len(y_test)):
```

```
    x.append(i)
# Create a linear regression model
reg = LinearRegression(normalize=True)
# Train a linear regression model
reg.fit(X_train, y_train)
# Forecast on a test sample
y_pred = reg.predict(X_test)
params = np.append(reg.intercept_,reg.coef_)


# In[39]:


# Calculate regression metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('Mean absolute error = ', mae)
print('Mean squared error = ', mse)
print('R2 Score = ', r2)


# In[40]:


def mean_absolute_percentage_error(y_true, y_pred):
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mape = mean_absolute_percentage_error(y_test, y_pred)
print('Mean absolute percentage error = ', mape)


# In[41]:


from sklearn import linear_model
clf = linear_model.Lasso(alpha=0.1)
clf.fit(X_train, y_train)
print(clf.coef_)


# In[42]:


y_pred_lasso = clf.predict(X_test)
mae_lasso = mean_absolute_error(y_test, y_pred_lasso)
mse_lasso = mean_squared_error(y_test, y_pred_lasso)
r2_lasso = r2_score(y_test, y_pred_lasso)
print('Mean Absolute Error with lasso =', mae_lasso)
print('Mean Squared Error with lasso =', mse_lasso)
print('R2 Score with lasso =', r2_lasso)


# In[43]:


lars = Lars()
lars.fit(X_train, y_train)

mae = mean_absolute_error(y_test, lars.predict(X_test))
mse = mean_squared_error(y_test, lars.predict(X_test))
```

```
r2 = r2_score(y_test, lars.predict(X_test))
print('Mean Absolute Error with lars =', mae)
print('Mean Squared Error with lars =', mse)
print('R2 Score with lars =', r2)


# ## 7. Analyze the quality of regression model (distribution of
residuals, determination coefficient)

# In[44]:


# Graph of real and predicted values
plt.scatter(x, y_test, label = u'The real meaning of reproduction rate')
plt.scatter(x, y_pred, label = u'Predicted by the linear model')
plt.title(u'Real values of reproduction rate and predicted by the linear
model')
plt.legend(loc="center right",borderaxespad=0.1, bbox_to_anchor=(1.7,
0.5))
plt.xlabel(u'time interval')
plt.ylabel(u'reproduction rate')


# In[45]:


y_pred_all = np.array(reg.predict(X))


# In[46]:


# Plotting a quantile biplot based on real and predicted values
percs = np.linspace(0, 100, 21)
qn_first = np.percentile(y, percs)
qn_second = np.percentile(y_pred_all, percs)
plt.figure(figsize=(12, 12))

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(u'Real')
plt.ylabel(u'Prediction')


# In[47]:


# Building and training of the 2nd degree polynomial regression
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(2)
X_train_new = poly.fit_transform(X_train)
poly = PolynomialFeatures(2)
X_test_new = poly.fit_transform(X_test)
reg = LinearRegression(normalize=True)
reg.fit(X_train_new, y_train)
y_pred_poly = reg.predict(X_test_new)
```

```
X_new = poly.fit_transform(X)
y_pred_poly_all = reg.predict(X_new)


# In[48]:


# Plotting a quantile biplot based on real and predicted values
percs = np.linspace(0, 100, 21)
qn_first = np.percentile(y, percs)
qn_second = np.percentile(y_pred_poly_all , percs)
plt.figure(figsize=(12, 12))

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(u'Real')
plt.ylabel(u'Prediction')


# In[58]:


mae_poly = mean_absolute_error(y_test, y_pred_poly)
mse_poly = mean_squared_error(y_test, y_pred_poly)
r2_poly = r2_score(y_test, y_pred_poly)
print('Mean absolute error with Polynomial model = ', mae_poly)
print('Mean squared error with Polynomial model = ', mse_poly)
print('R2 Score with Polynomial model = ', r2)


# In[49]:


x = []
# Visualization of real and predicted values with polynominal model
for i in range(len(y_test)):
    x.append(i)
plt.scatter(x, y_test, label = u'Real')
plt.scatter(x, y_pred_poly, label = u'Prediction')
plt.title(u'Real values of reproduction rate and predicted by the
polynominal model')
plt.legend(loc="center right",borderaxespad=0.1, bbox_to_anchor=(1.9,
0.5))
plt.xlabel(u'time interval')
plt.ylabel('reproduction rate')


# In[50]:


# Plotting the distribution of residuals
y1 = np.array(y)
y2 = np.array(y_pred_all)
y_diff = y1[:,0] - y2[:,0]
sns.distplot(y_diff, kde=False)
```

```
# In[51]:


# Confidence interval of regression coef
import numpy as np, statsmodels.api as sm
mod = sm.OLS(y_train, X_train)
res = mod.fit()
print (res.conf_int(0.01))
```