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.

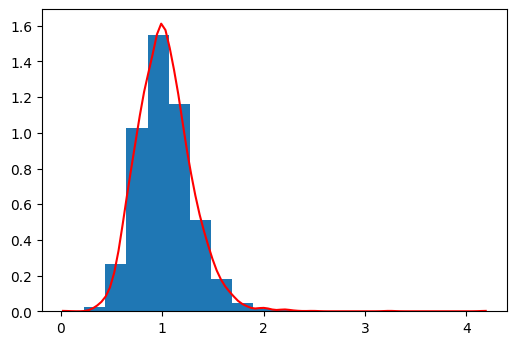


Figure 1 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “Reproduction rate”

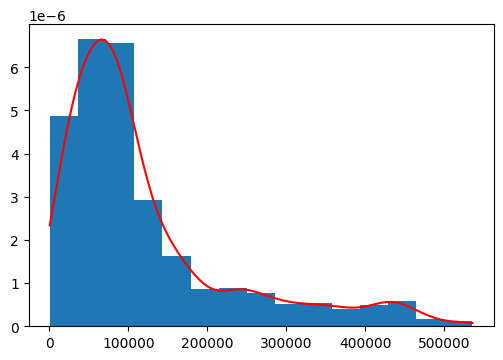


Figure 2 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “Total cases per million”

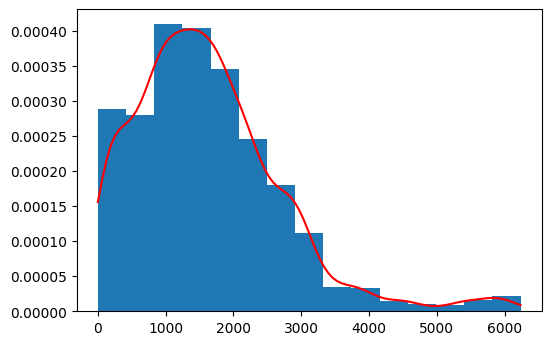


Figure 3 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “Total deaths per million”

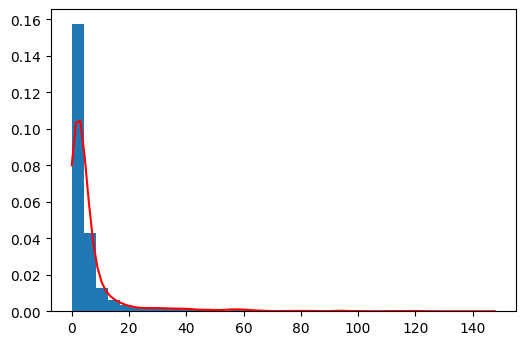


Figure 4 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “New tests smoothed per thousand”

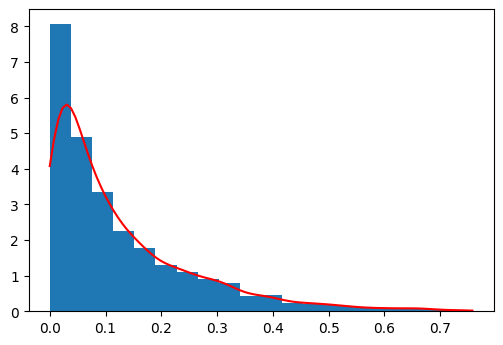


Figure 5 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “Positive rate”

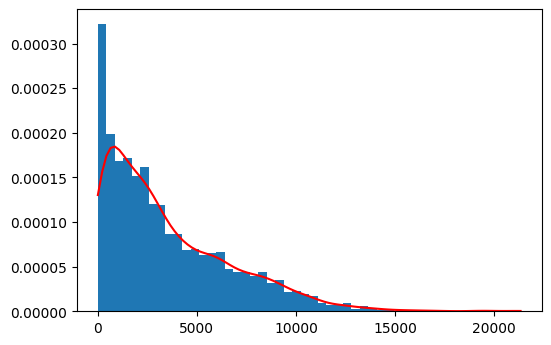


Figure 6 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “New vaccinations smoothed per million”

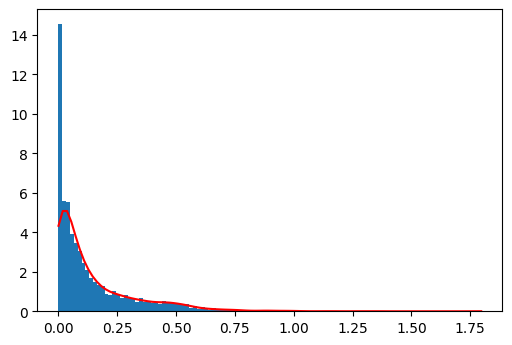


Figure 7 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “New people vaccinated smoothed per hundred”

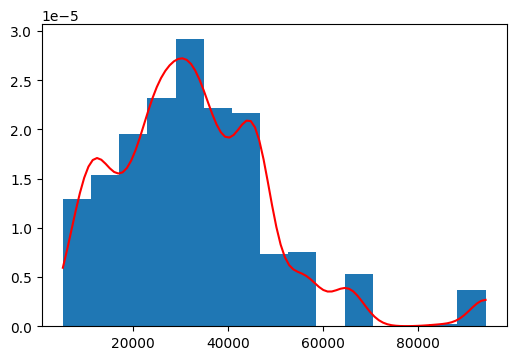


Figure 8 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “GDP per capita”

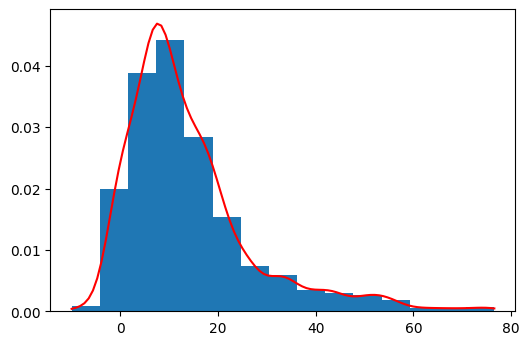


Figure 9 – Non-parametric estimation of PDF in form of histogram and using kernel density function for “Excess mortality cumulative”

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.

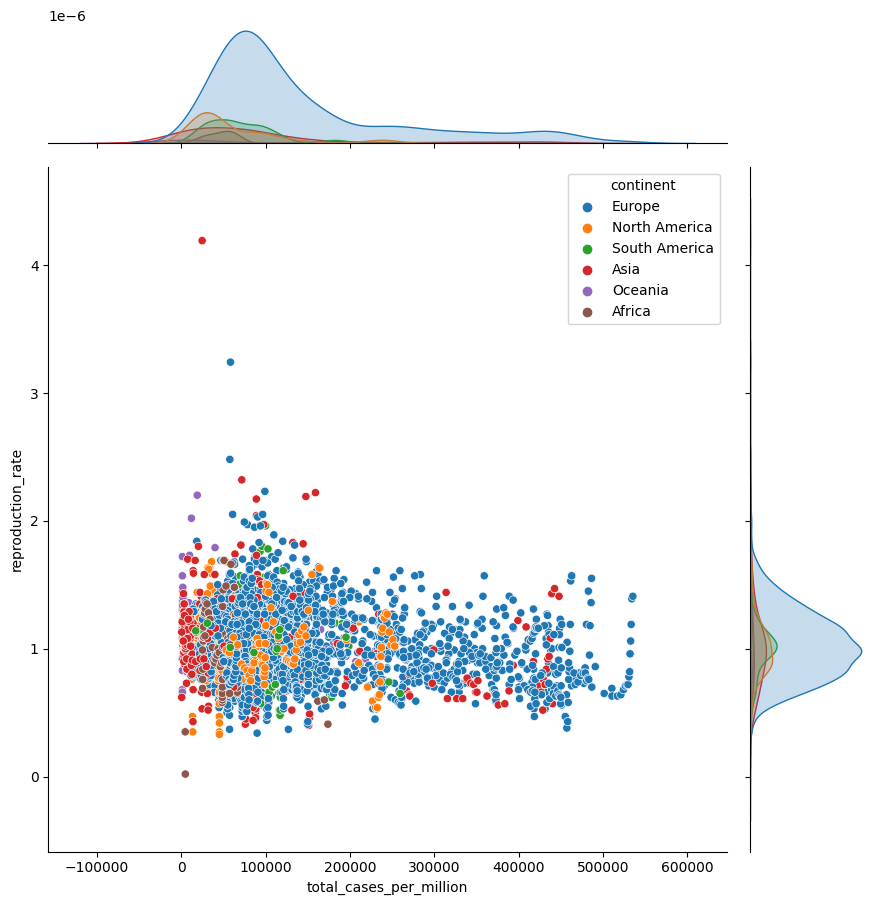


Figure 10 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “Total cases per million”

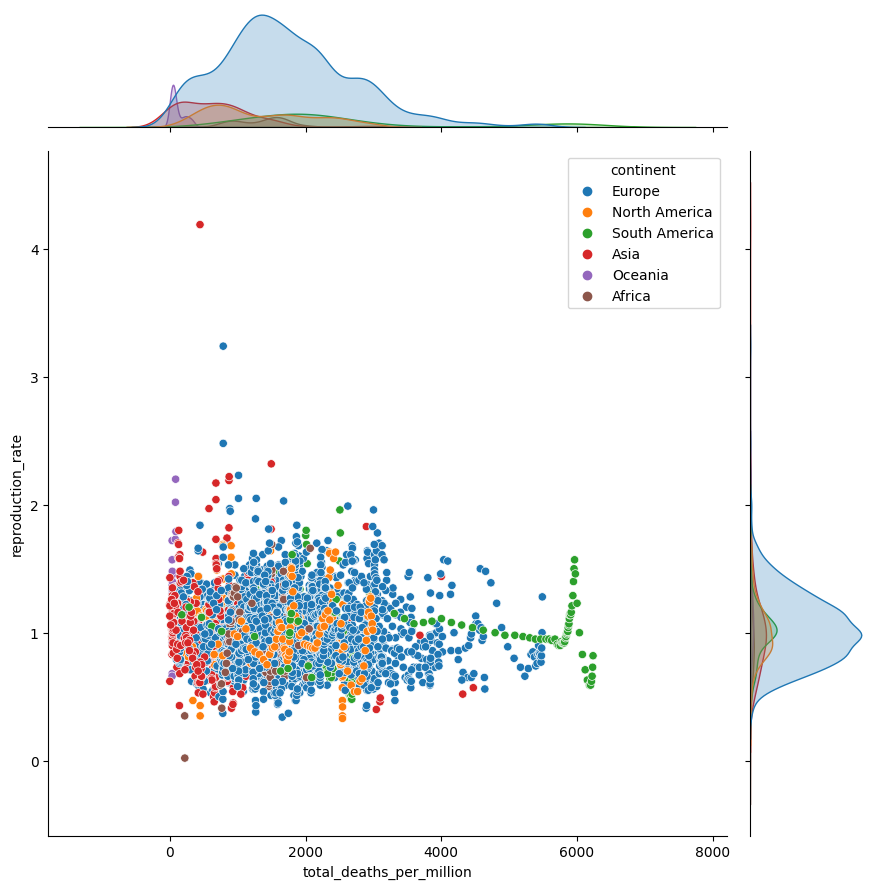


Figure 11 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “Total deaths per million”

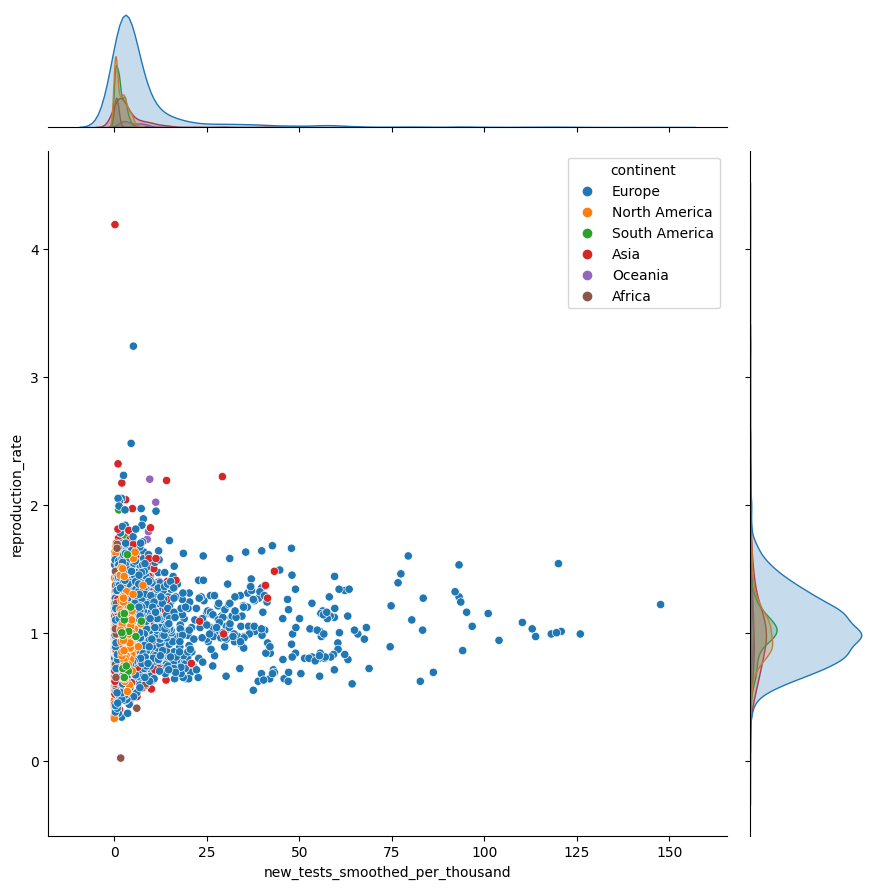


Figure 12 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “New tests smoothed per thousand”

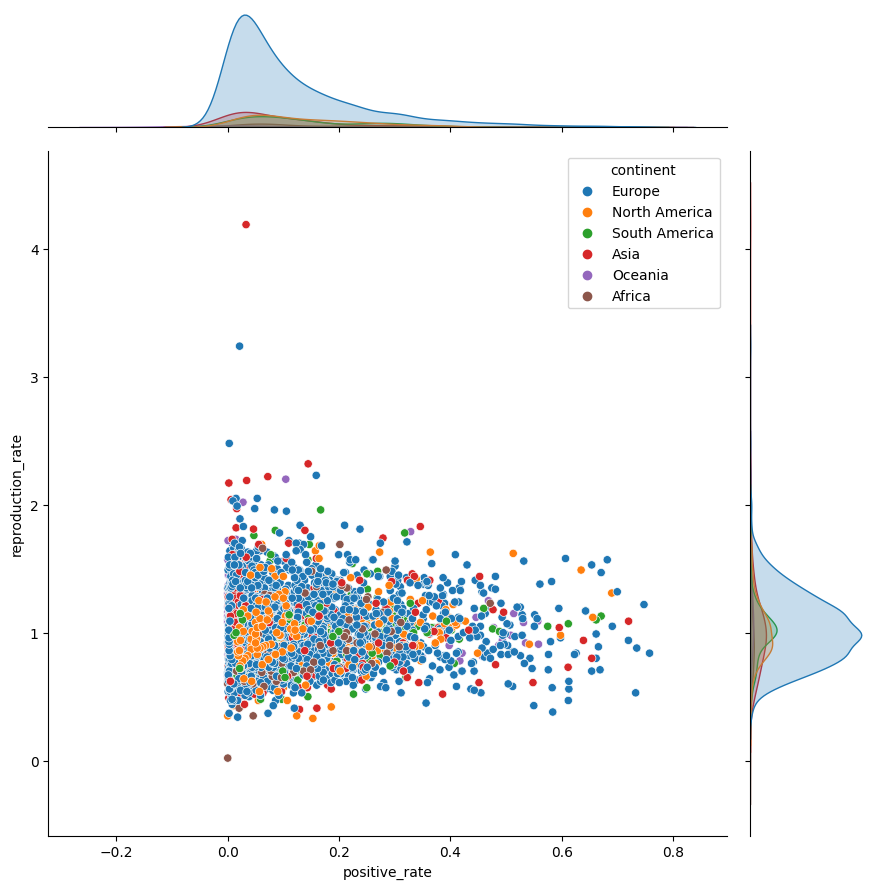


Figure 13 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “Positive rate”

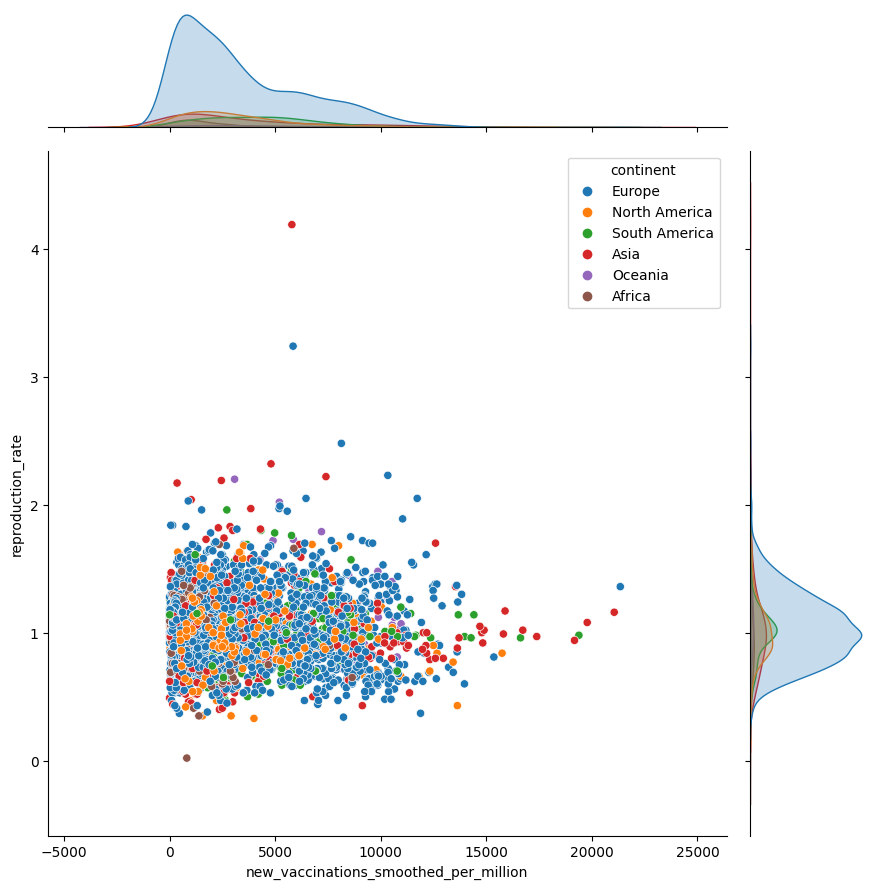


Figure 14 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “New vaccinations smoothed per million”

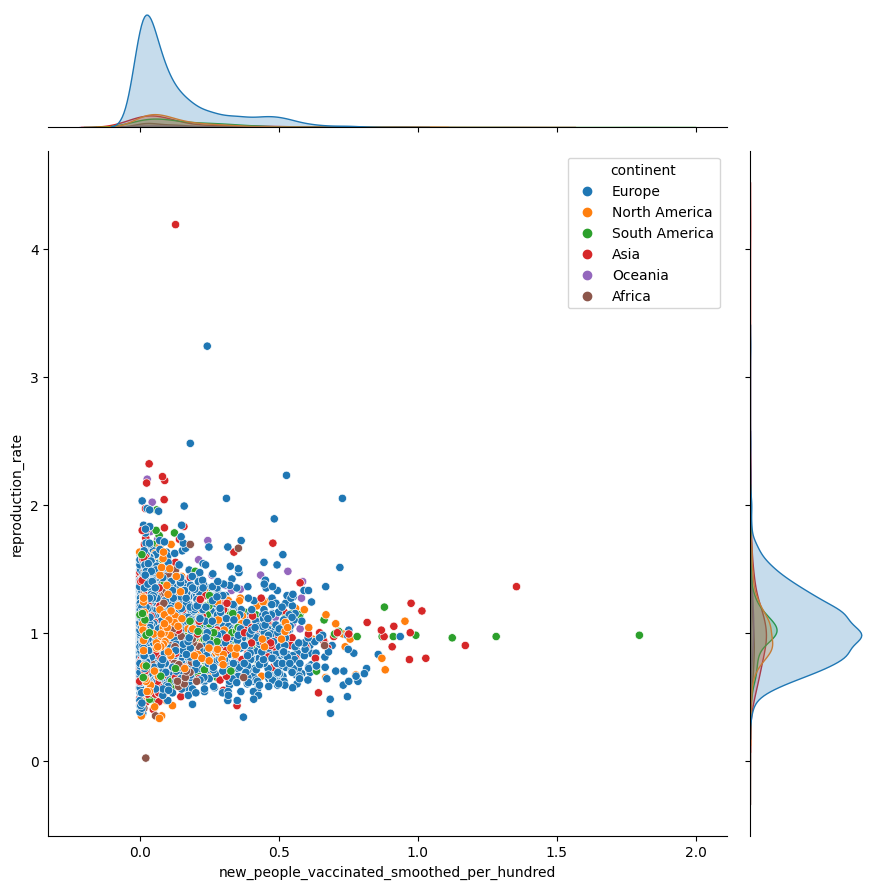


Figure 15 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “New people vaccinated smoothed per hundred”

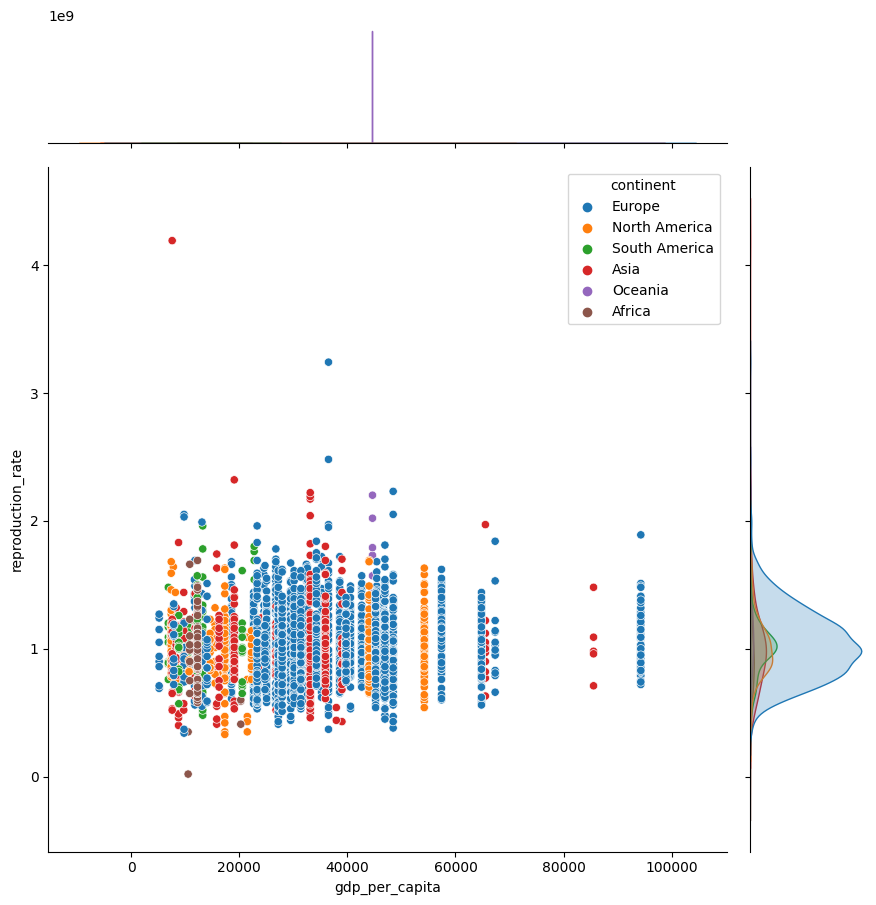


Figure 16 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “GDP per capita”

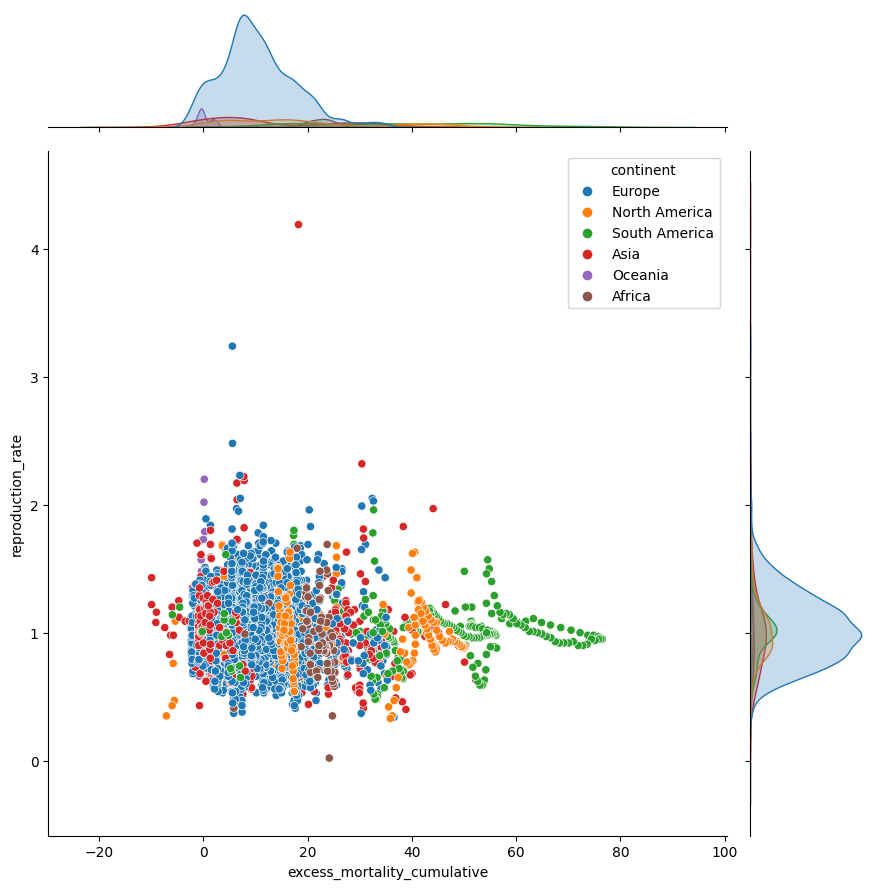


Figure 17 – Joint probability density plot in form of histogram between target “Reproduction rate” and predictor “Excess mortality cumulative”

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.

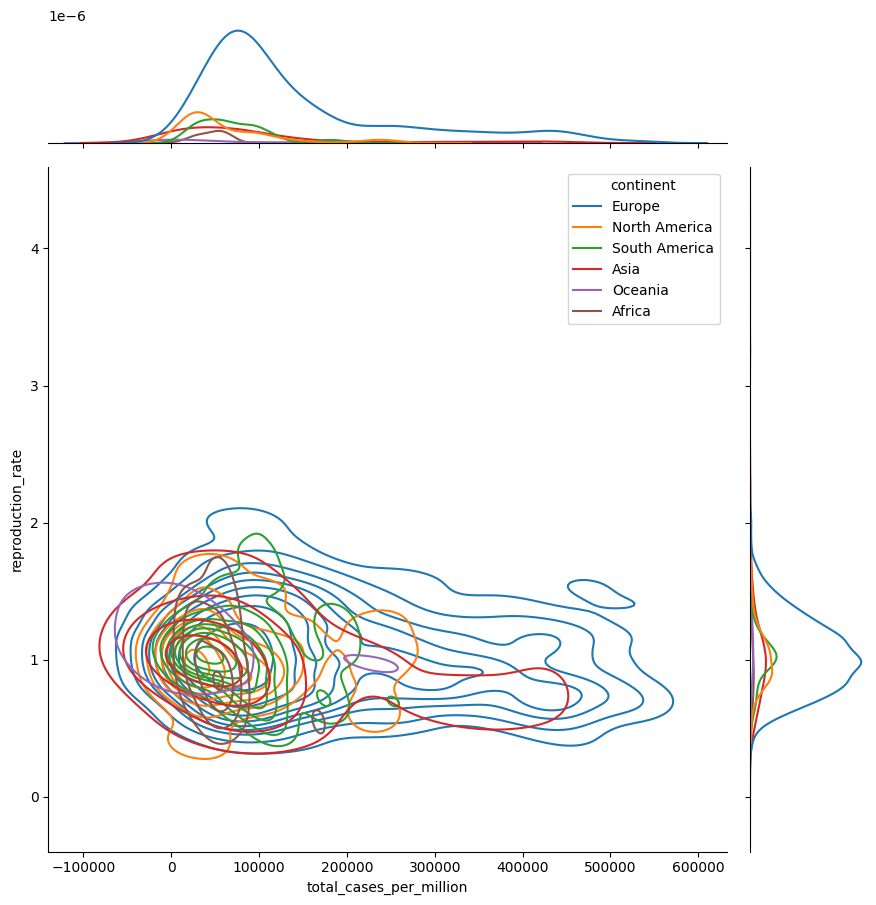


Figure 18 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “Total cases per million”

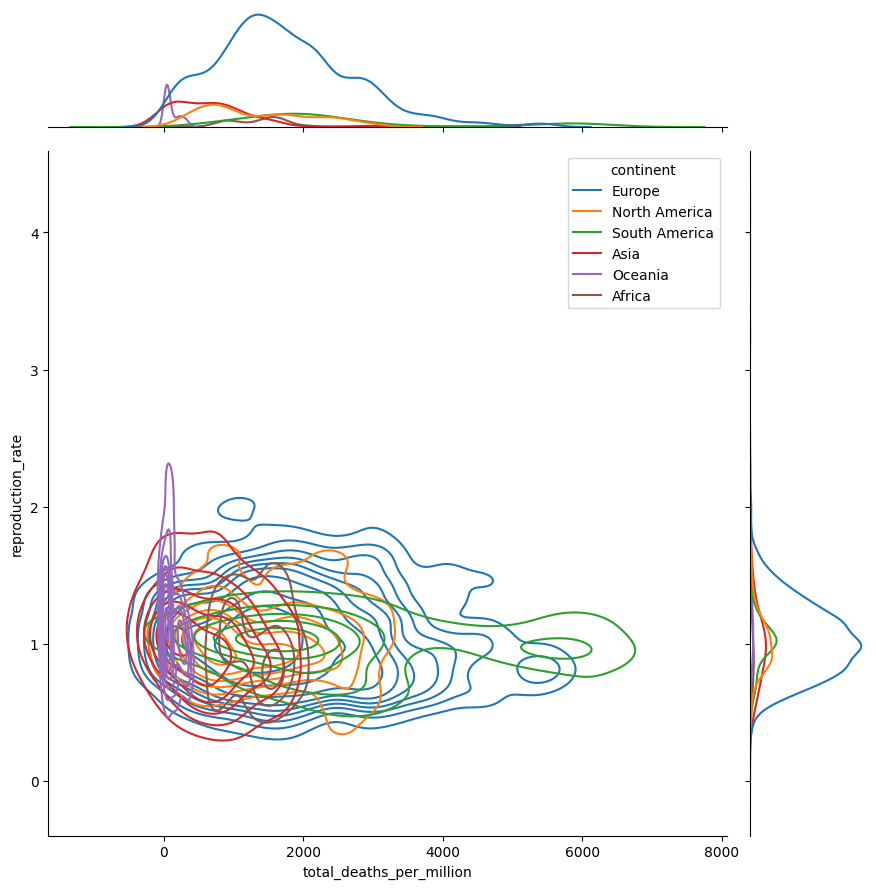


Figure 19 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “Total deaths per million”

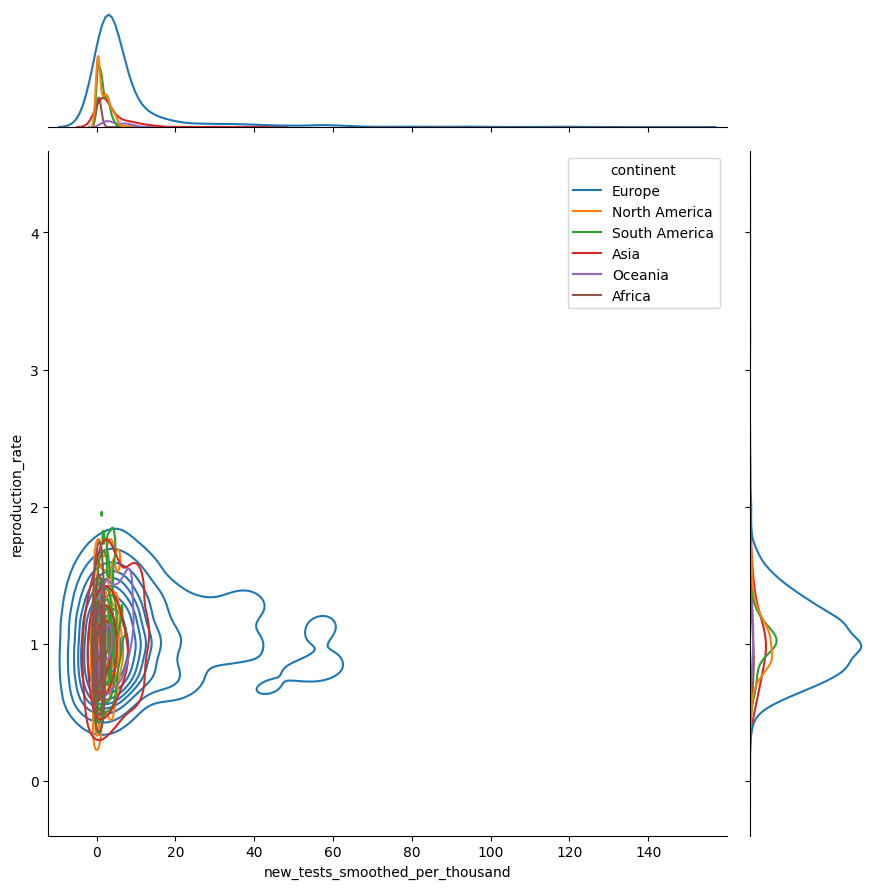


Figure 20 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “New tests smoothed per thousand”

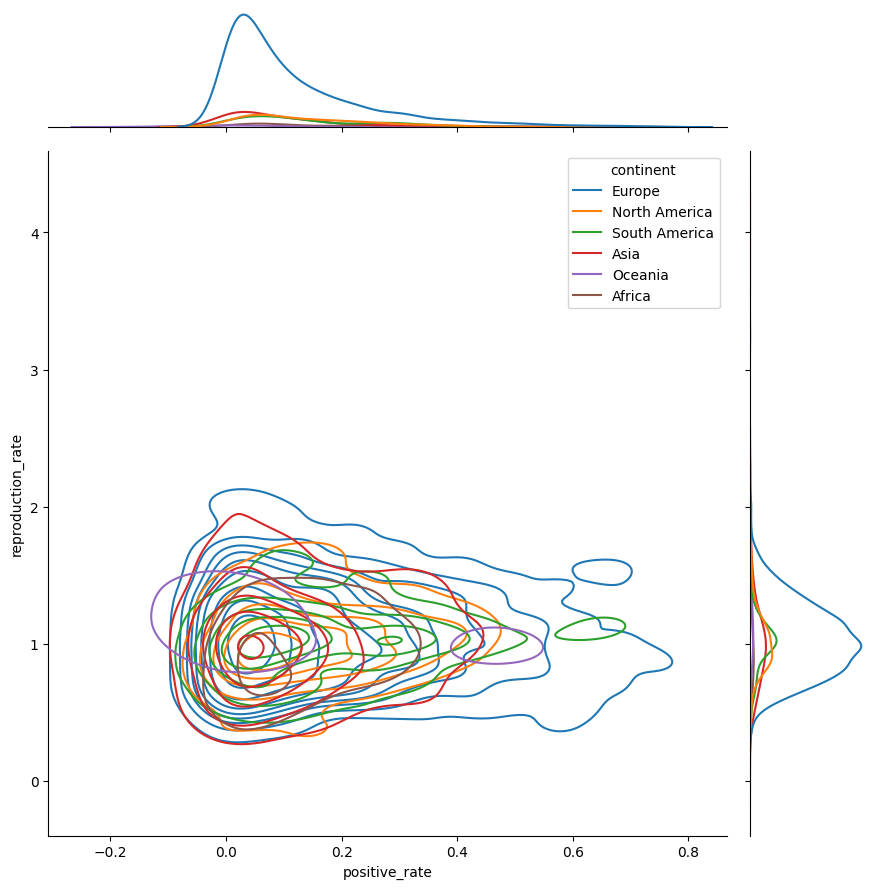


Figure 21 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “Positive rate”

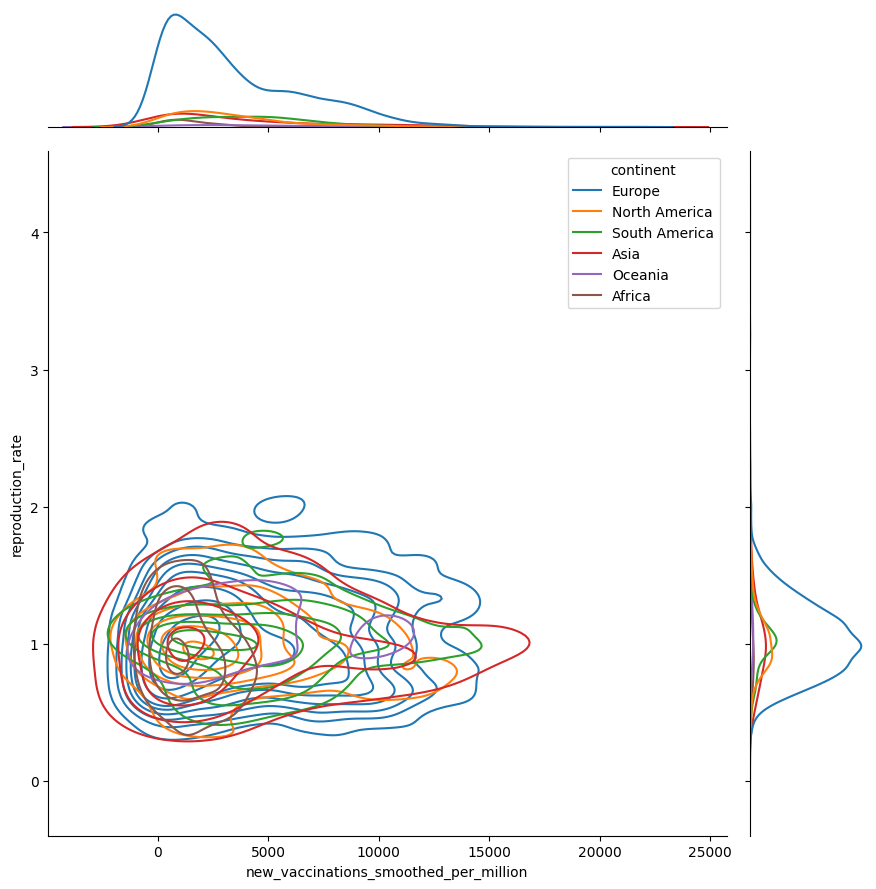


Figure 22 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “New vaccinations smoothed per million”

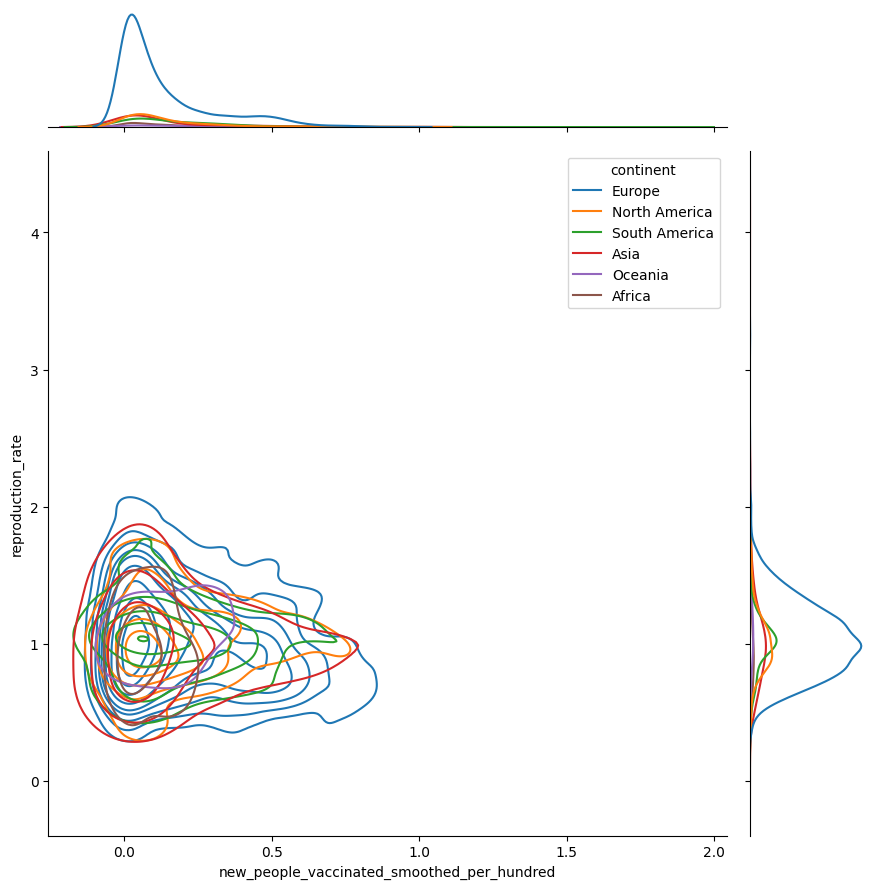


Figure 23 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “New people vaccinated smoothed per hundred”

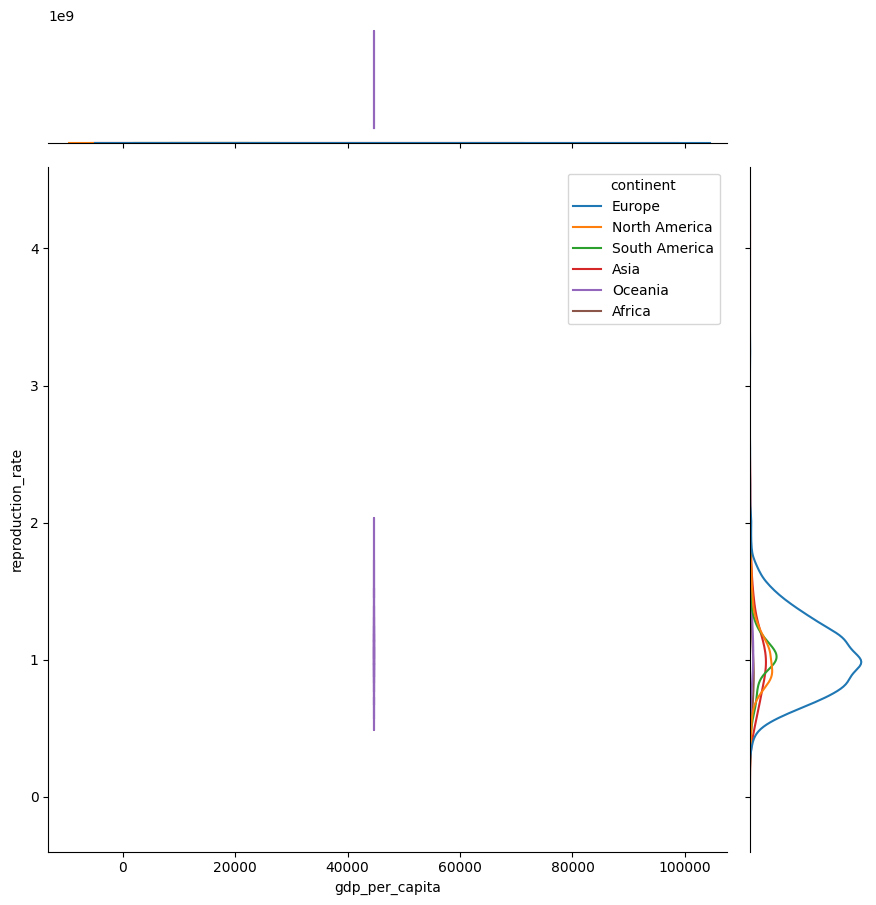


Figure 24 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “GDR per capita”

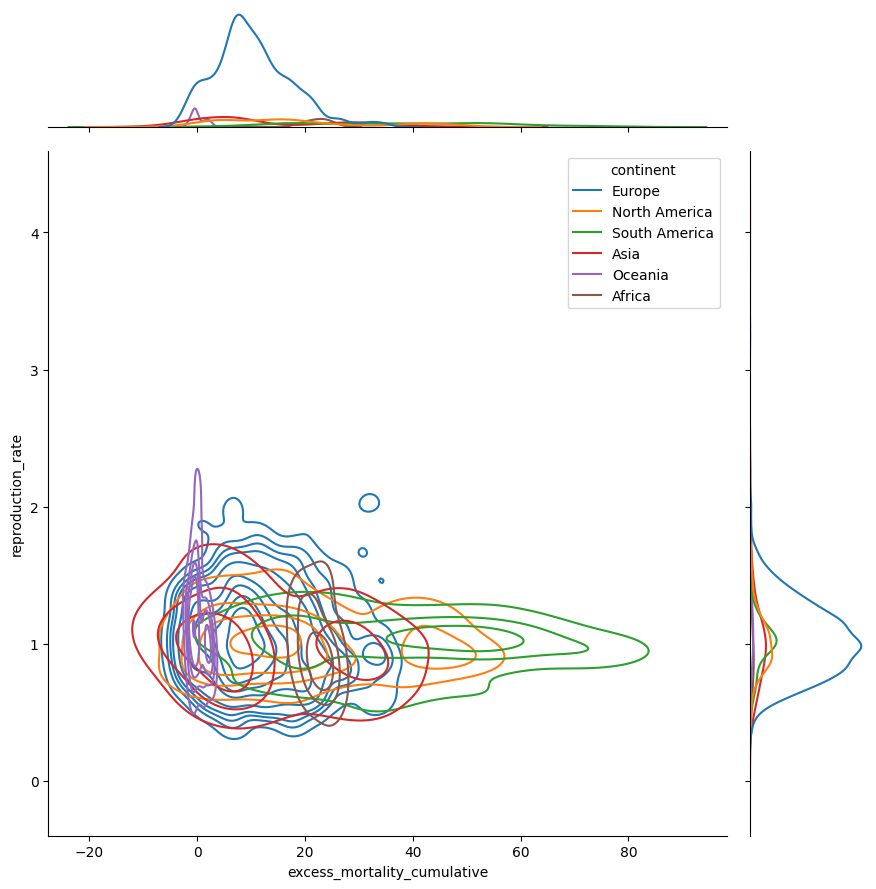


Figure 25 – Joint probability density plot in form of kernel density function between target “Reproduction rate” and predictor “Excess mortality cumulative”

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

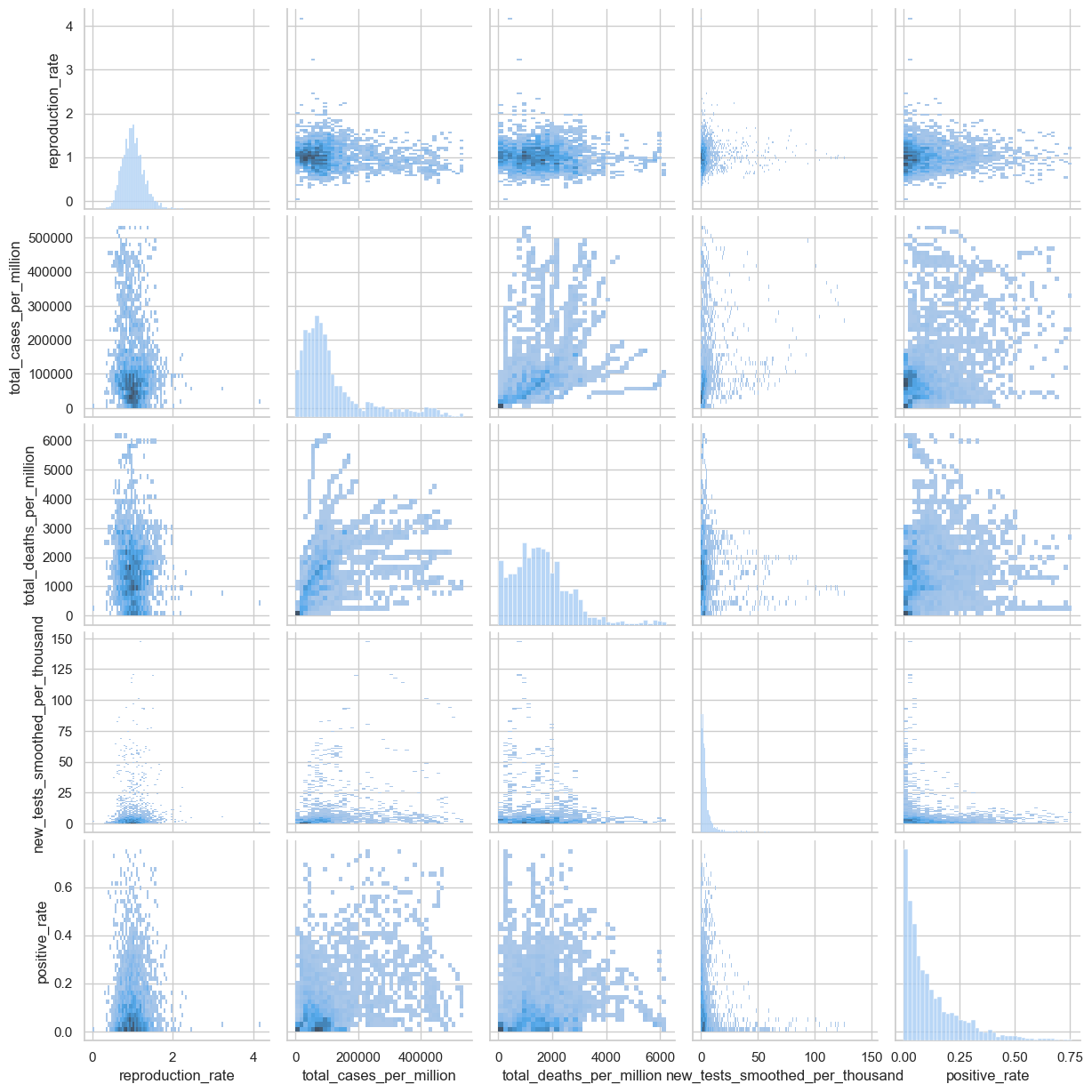


Figure 26 – Pairwise joint probability density between five random variables

## Estimation of multivariate mean and variance

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 |  |  |

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 be . So, Figure 27 depicts distribution of variables under this condition. And Table 2 contains conditional mathematical expectation and conditional variance.

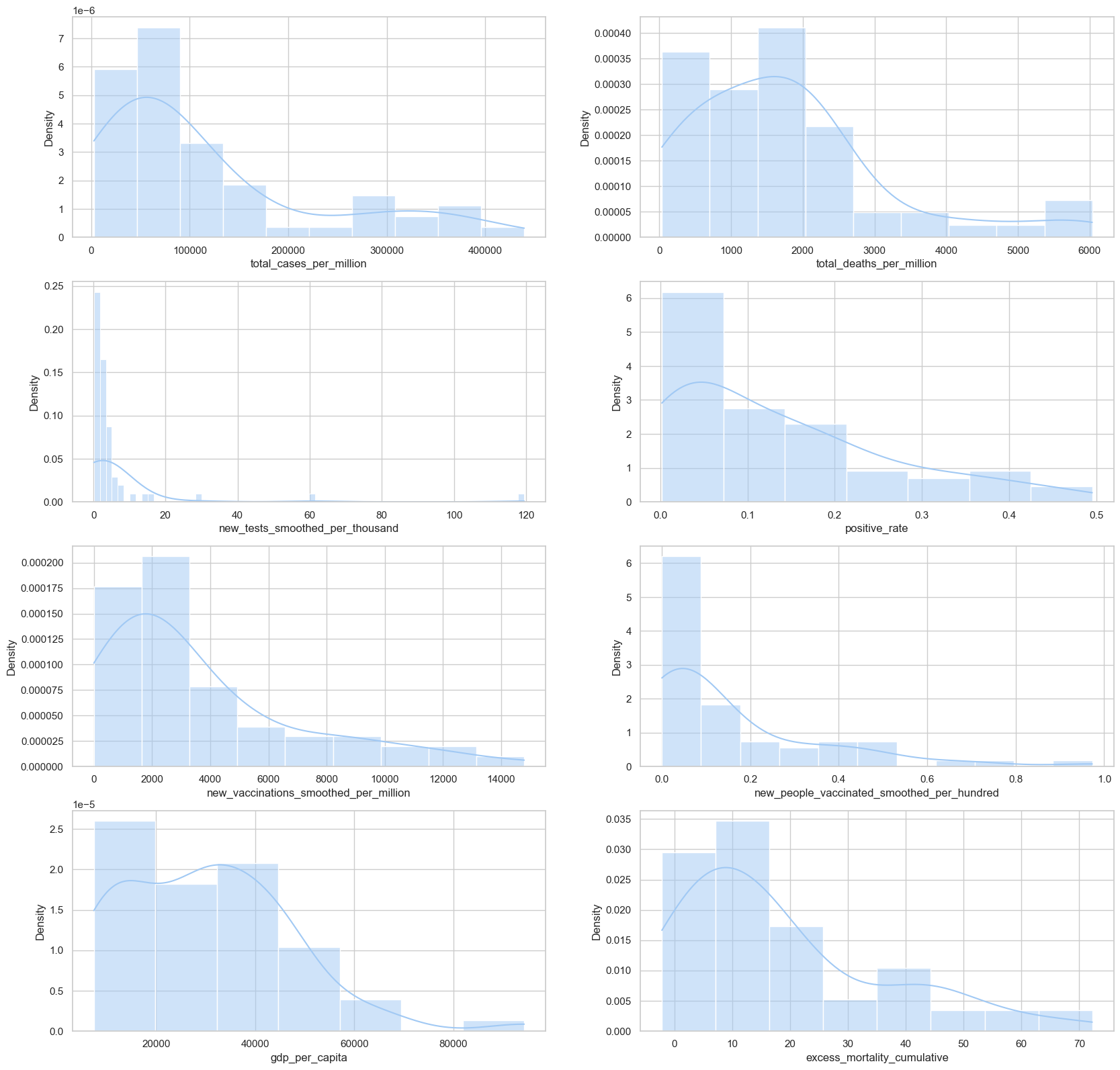


Figure 27 – Conditional distributions of variables

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 |  |  |

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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Coefficient | Conf. interval | Signif. level |
| Total cases per million | Total deaths per million |  |  |  |
| New vaccinations smoothed per million | Excess mortality cumulative |  |  |  |
| New people vaccinated smoothed per hundred | Excess mortality cumulative |  |  |  |
| GDP per capita | Excess mortality cumulative |  |  |  |
| Total cases per million | New vaccinations smoothed per million |  |  |  |

Table 3 – Pair correlation coefficients confidence intervals and significance levels

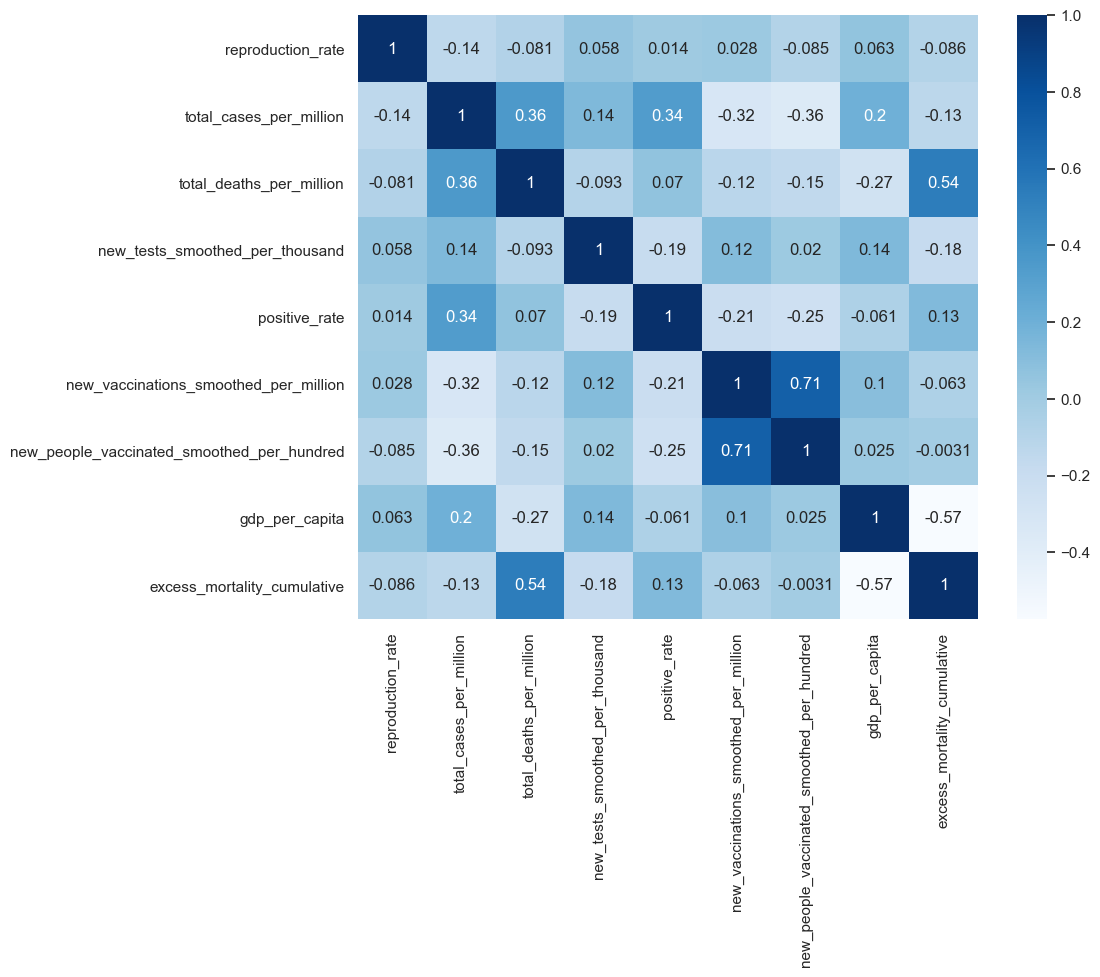


Figure 28 – Pair correlation coefficients

## 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 |  |
| Mean Squared Error |  |
| R2 Score |  |
| Mean Absolute Percentage Error |  |

Table 4 – Ordinal regression model metrics

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

|  |  |
| --- | --- |
| Metric | Value |
| Mean Absolute Error |  |
| Mean Squared Error |  |
| R2 Score |  |

Table 5 – Lasso-regression model metrics

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

|  |  |
| --- | --- |
| Metric | Value |
| Mean Absolute Error |  |
| Mean Squared Error |  |
| R2 Score |  |

Table 6 – LARS model metrics

Also 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.

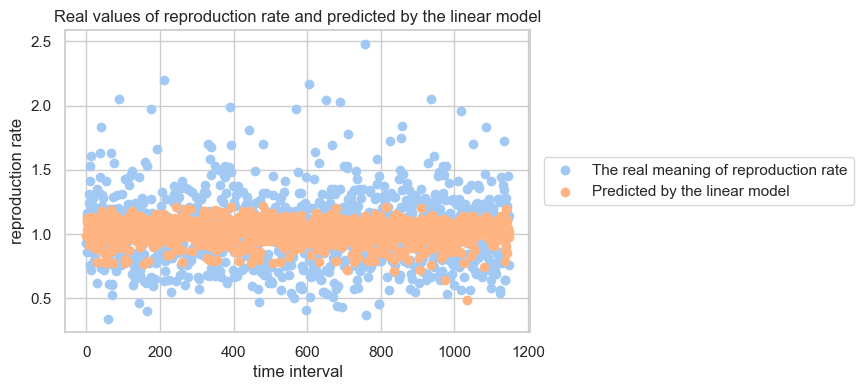


Figure 29 – Real and predicted points by linear model

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

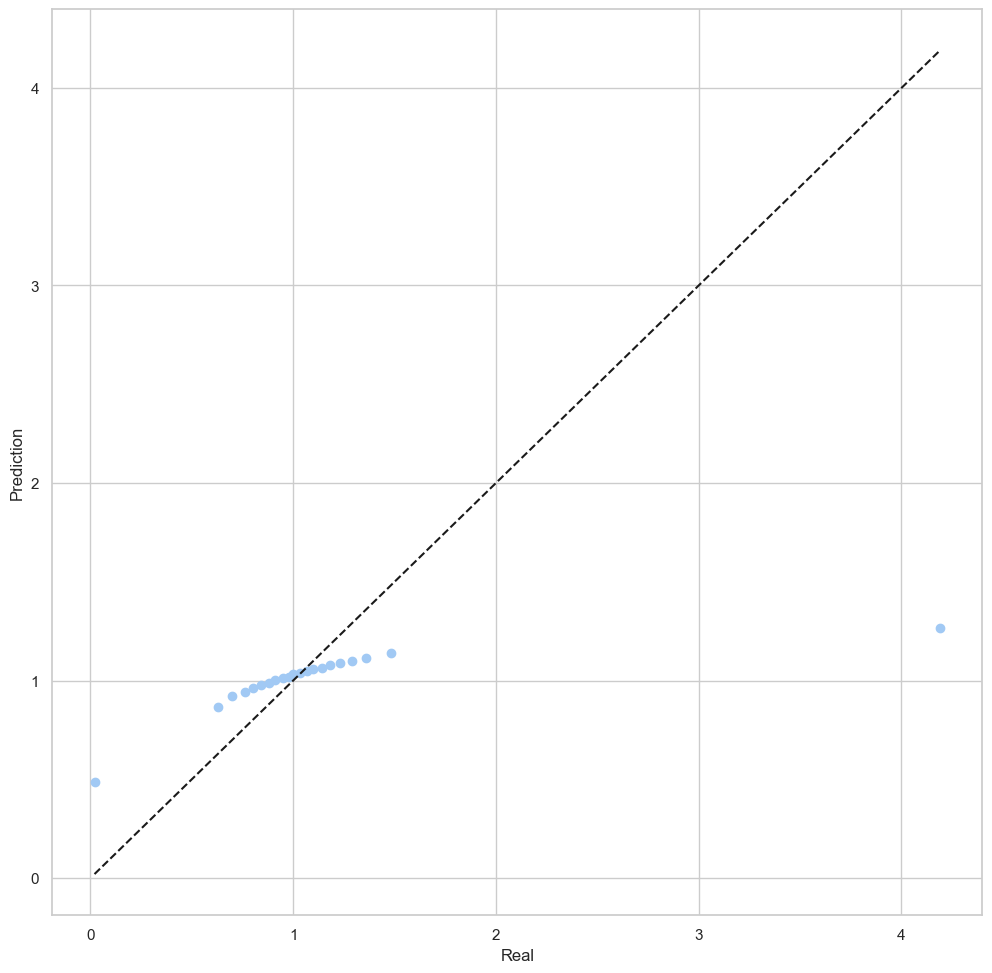


Figure 30 – QQ-plot for linear model

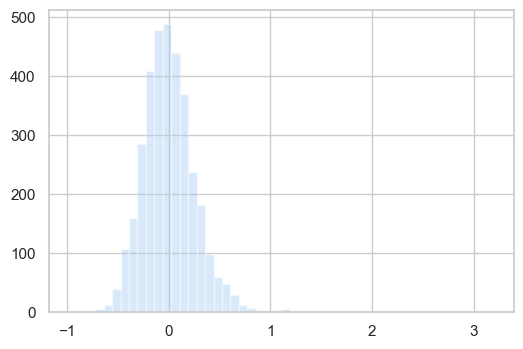


Figure 31 – Distribution of residuals for linear model

|  |  |  |
| --- | --- | --- |
|  | Interval left | Interval right |
| 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 |  |  |

Table 7 – Confidence interval of regression coefficients

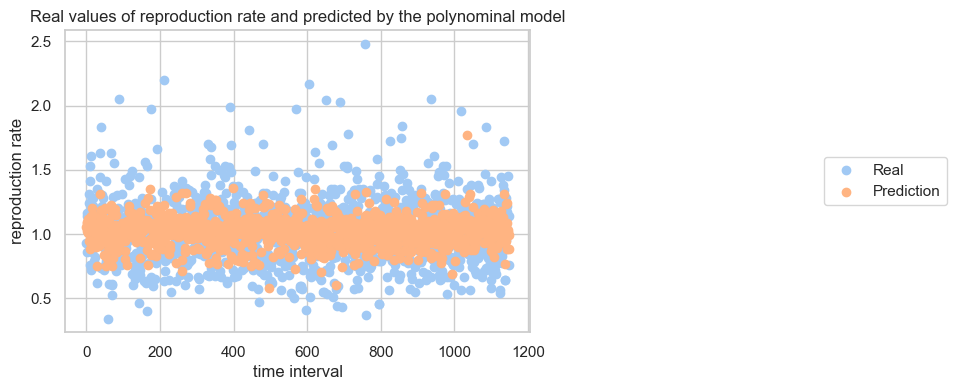


Figure 32 – Real and predicted points by polynomial model

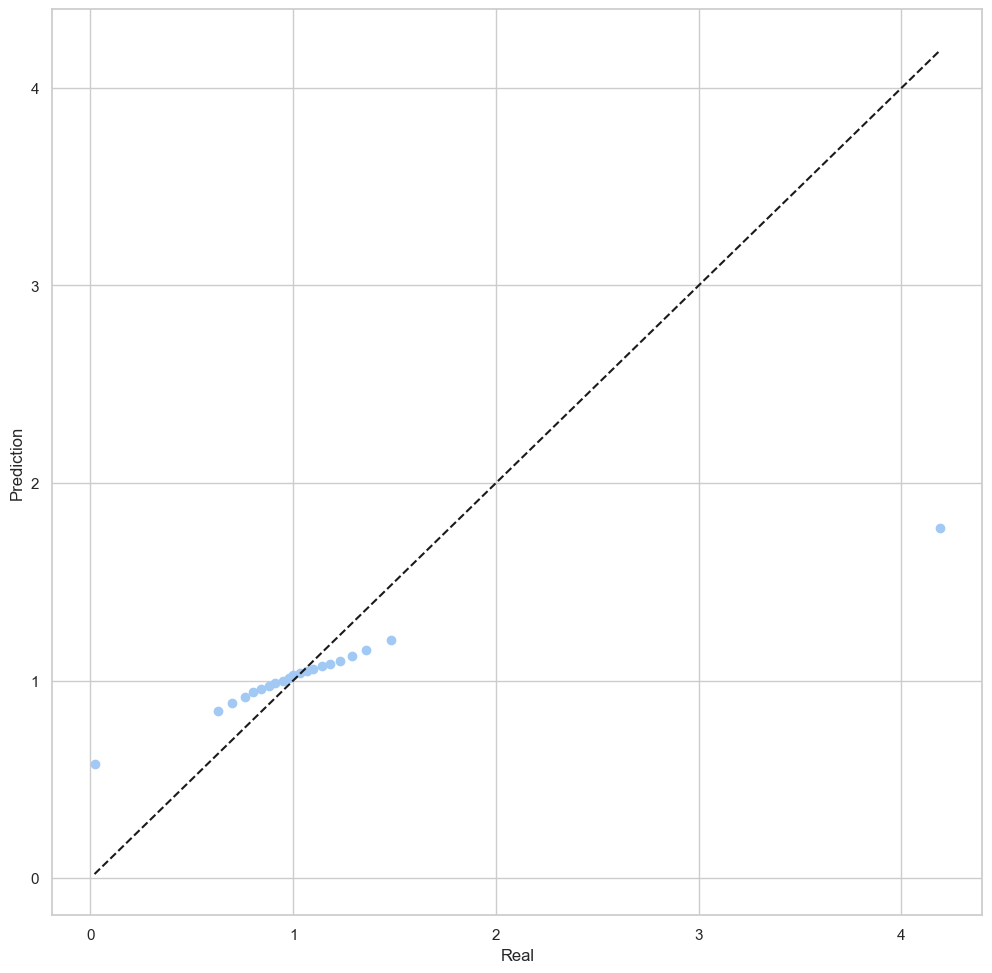


Figure 33 – QQ-plot for polynomial model

**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\_thousand')

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\_smoothed\_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\_vaccinated\_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\_thousand')

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\_smoothed\_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\_vaccinated\_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))