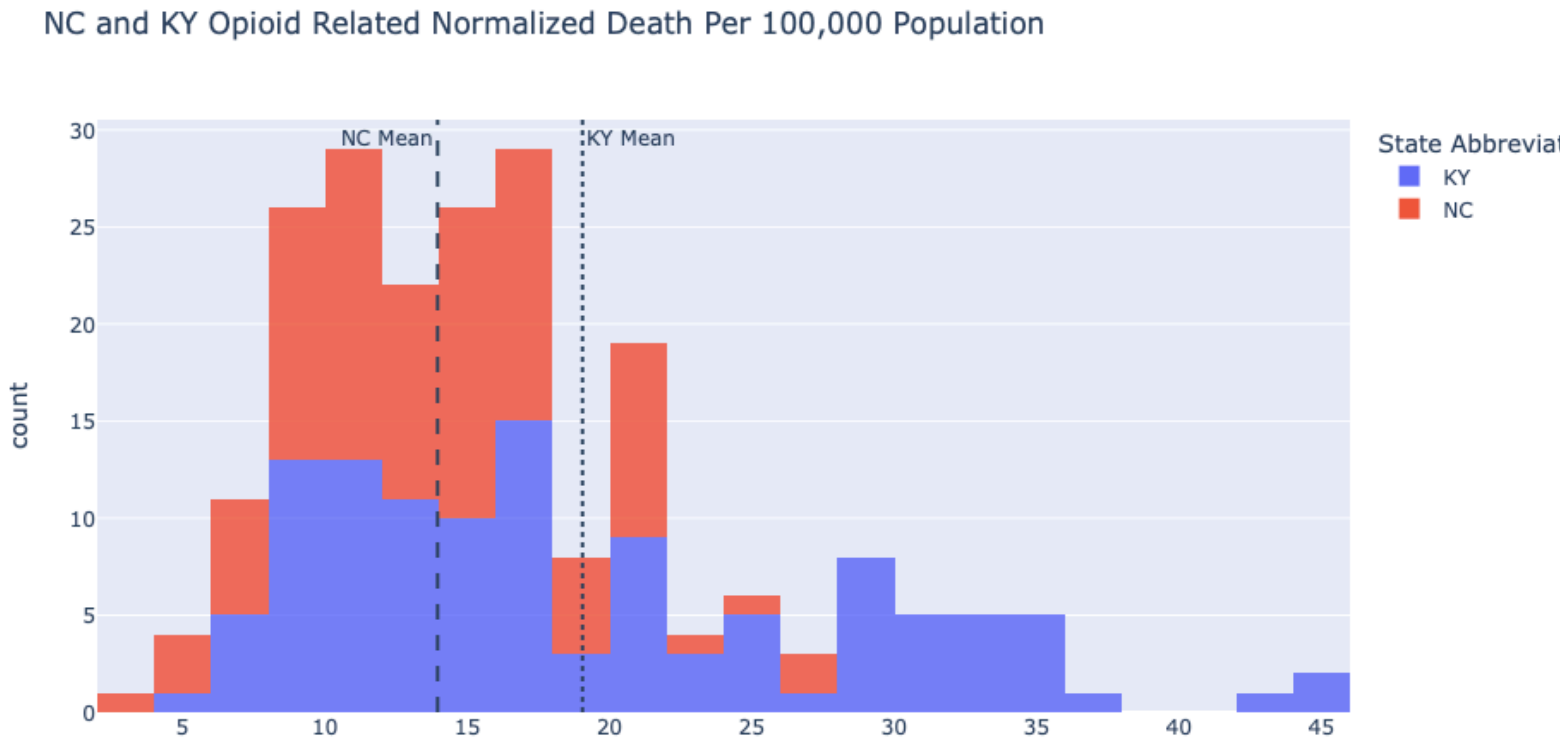
- In the first task we want to compare the distribution of NC and KY mortality rate (by using plotly to draw a histogram).
- From the histogram we can see that KY has a higher mean for normalized death compare to NC.

In [27]:

Slide Type -

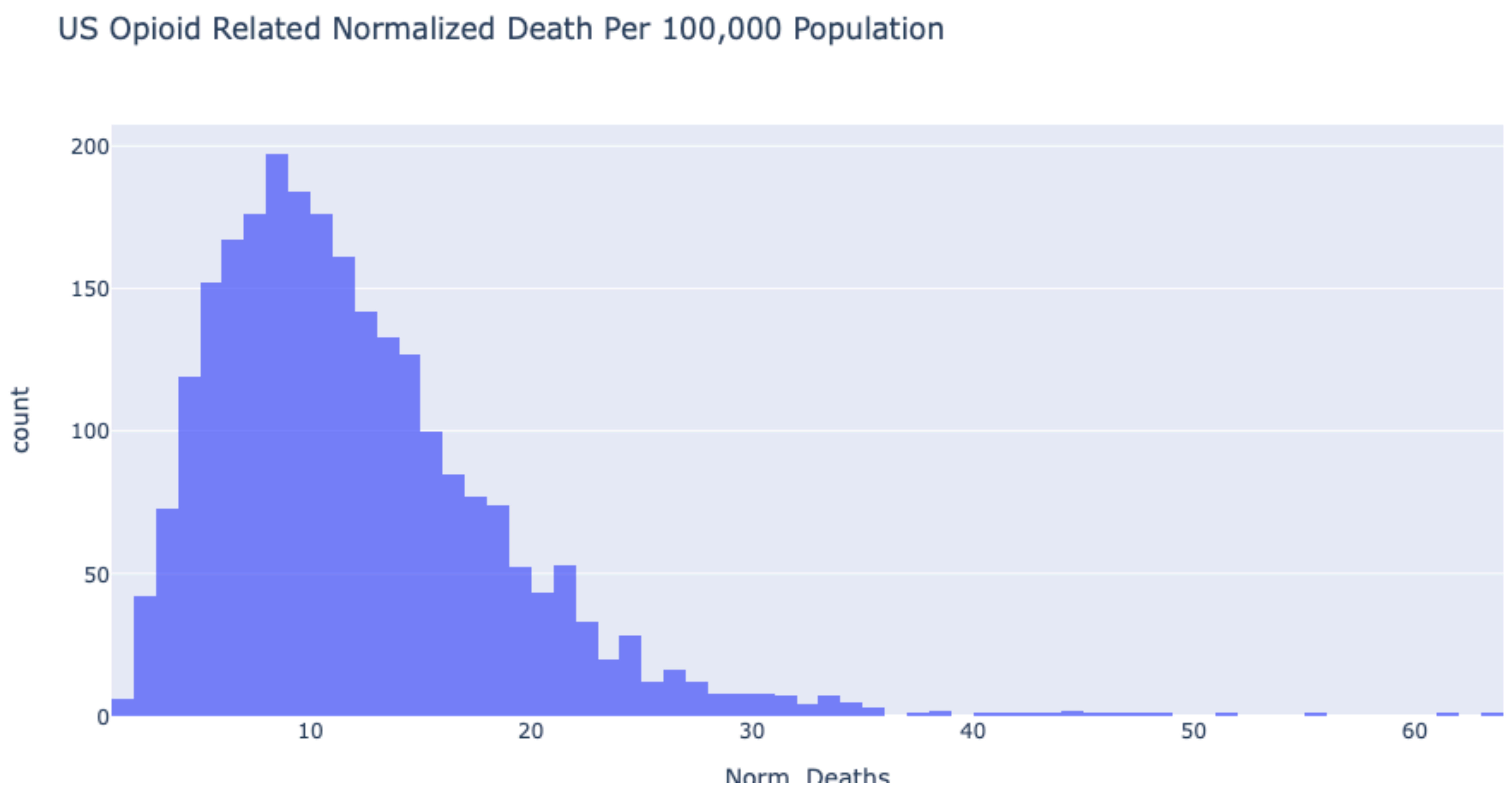
```
import plotly.express as px
# Create a histogram of Normalized Deaths variable for NC and KY as a distribution.
fig = px.histogram(super_df_NC_KY, x="Norm_Deaths", opacity = 0.85, color = 'State Abbreviation', title = "NC a

# Add a vertical line to the histogram representing the ***mean*** Normalized Opioid Death for NC.
fig.add_vline(x=super_df[super_df['State Abbreviation'] =='NC']['Norm_Deaths'].mean(),line_dash="dash",annotati

# Add a vertical line to the histogram representing the ***mean*** Normalized Opioid Death for KY.
fig.add_vline(x=super_df[super_df['State Abbreviation'] =='KY']['Norm_Deaths'].mean(),line_dash="dot",annotatic
```



- For the next task we are asked to select a distribution for opioid related mortality rate. First I created a histogram for US Opioid Mortality Related Normalized Death.

In [28]:

```
# visuallizing the normalized death data
px.histogram(super_df, x="Norm_Deaths", opacity = 0.85,  title = "US Opioid Related Normalized Death Per 100,000 Pop
```



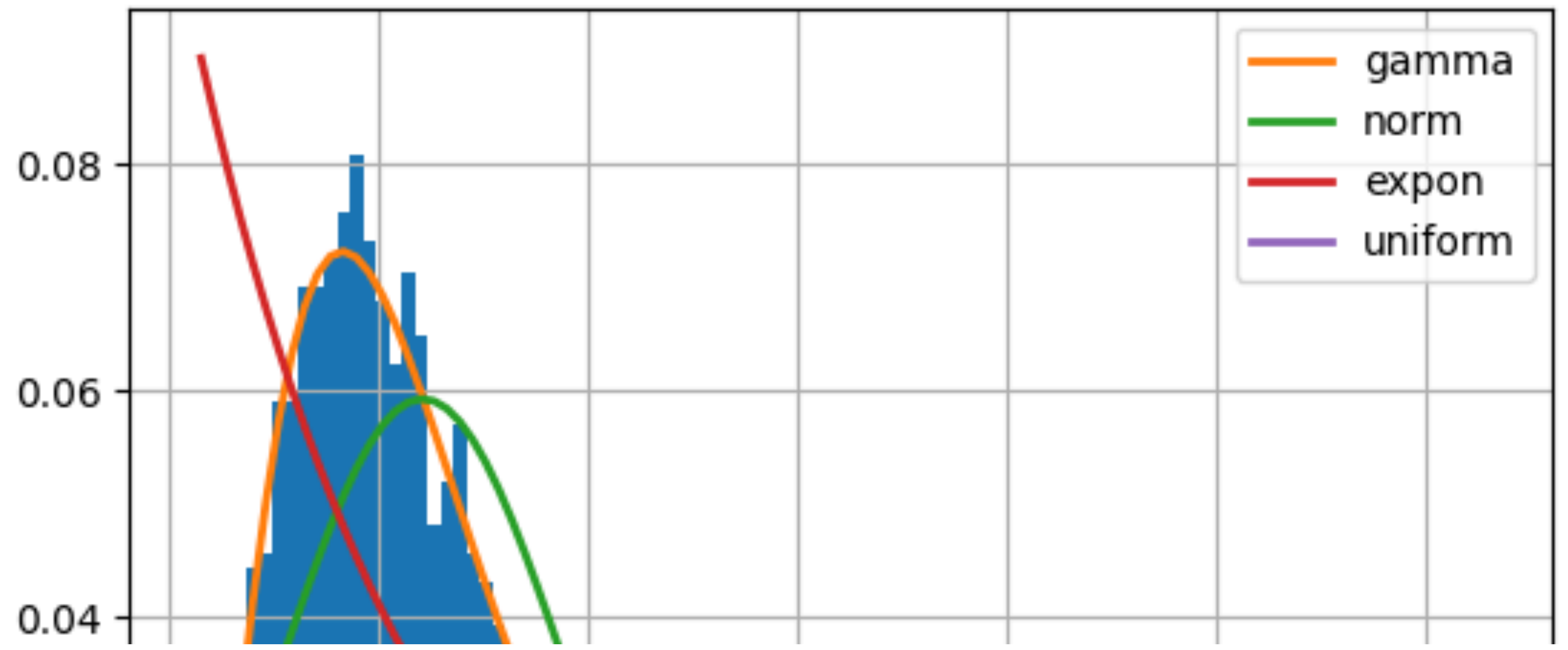
- For choosing the best distribution, I used fitter library.
- The fitter method compares the selected distribution based on the “sum square-error”, so we can choose the one that has the lowest “sum square-error”, which is gamma in this case.

```python
import fitter
from fitter import Fitter
f = Fitter(super_df['Norm_Deaths'].values,
distributions=['gamma','expon',"norm","uniform"])
f.fit()
print(f.get_best(method = 'sumsquare_error'))
f.summary()
```

```
Fitting 4 distributions: 100%|██████████| 4/4 [00:00<00:00, 71.44it/s]
{'gamma': {'a': 2.9061086491605264, 'loc': 1.021490483700656, 'scale': 3.828966301269168}}
```

Out[29]:

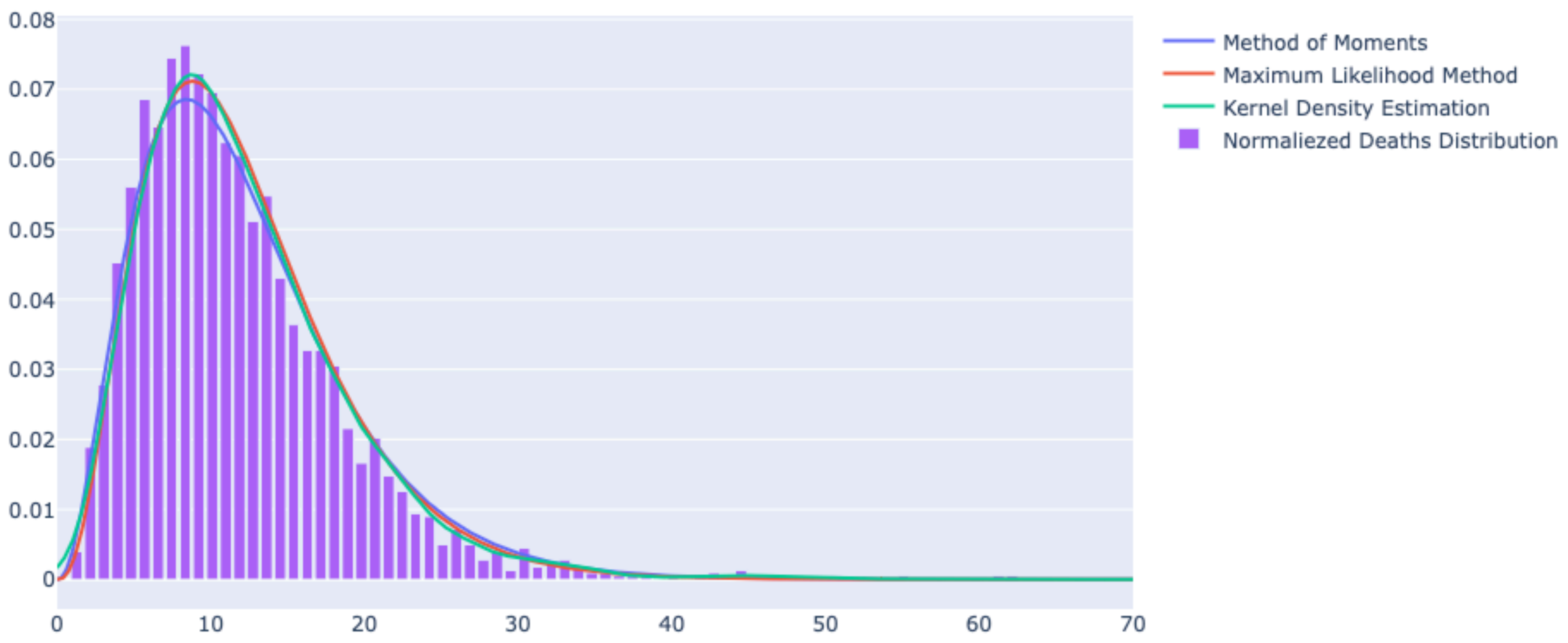
	sumsquare_error	aic	bic	kl_div	ks_statistic	ks_pvalue
gamma	0.000779	1332.689989	-37861.313867	inf	0.011361	8.962659e-01
norm	0.007670	2148.998957	-32090.378404	inf	0.088590	1.048767e-17
expon	0.030673	1048.966330	-28587.844949	inf	0.199862	6.018522e-89
uniform	0.057077	828.602613	-27018.529072	inf	0.590991	0.000000e+00



- It is also asked to develop distribution estimator with MoM, MLE and KDE for calculating the attributes related each of them I used the exact codes which is covered in the class.
- By using the go function from the plotly, we can have the distribution and estimators in the same chart.

In [34]:

```python
import plotly.graph_objects as go
x = np.linspace(0, 70,1000)
fig = go.Figure()
fig.add_trace(go.Scatter(x=x, y=gamma_dist_MoM,mode='lines',name='Method of Moments'))
fig.add_trace(go.Scatter(x=x, y=gamma_dist_MLM,mode='lines',name='Maximum Likelihood Method'))
fig.add_trace(go.Scatter(x=x, y=gamma_dist_KDE,mode='lines', name='Kernel Density Estimation'))
fig.add_trace(go.Bar(x=bin_edges, y = counts,name = 'Normaliezed Deaths Distribution'))
fig.show()
```



- For comparing the three estimators, we need to calculate mean square.
- As we can see KDE has the lowest mean square, so it is the best estimator.

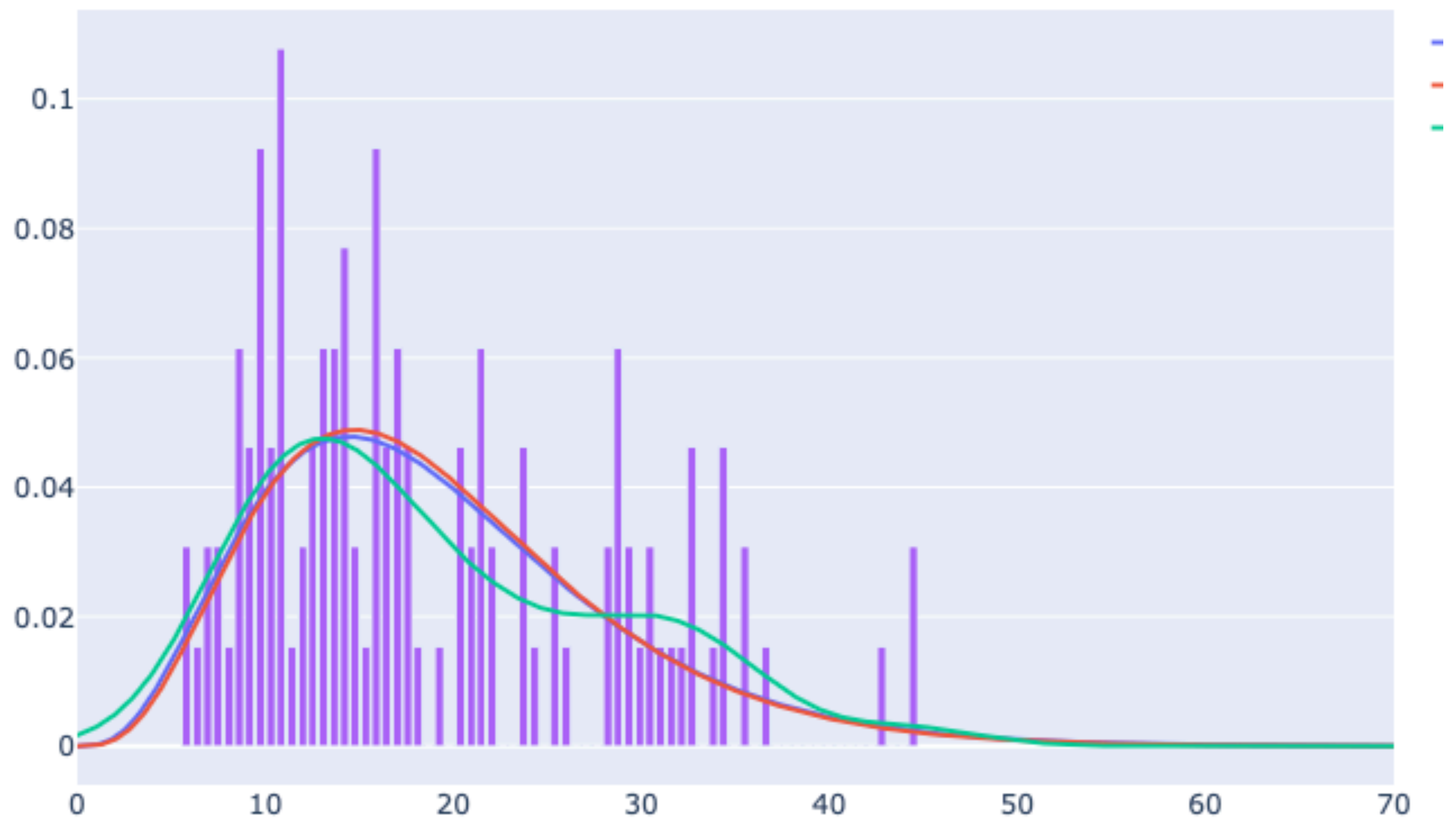
In [35]:

Slide Type -

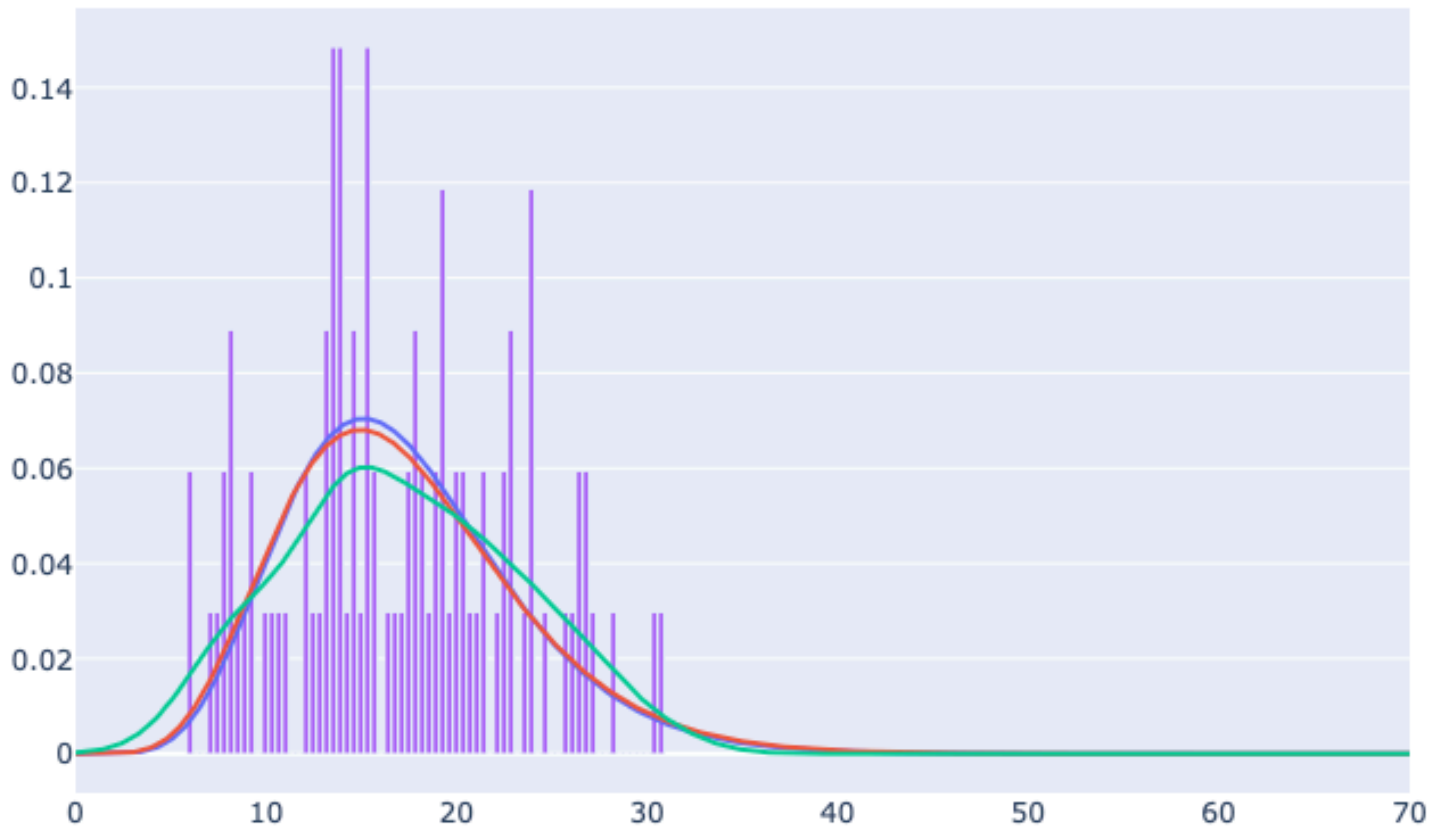
```
OBS, bins = np.histogram(super_df['Norm_Deaths'], density = True, bins = np.linspace(0, 70,1001))
mse = np.mean((OBS - gamma_dist_KDE)**2)
print('mse for KDE is: ' + str(mse))
mse = np.mean((OBS - gamma_dist_MoM)**2)
print('mse for MOM is: ' + str(mse))
mse = np.mean((OBS - gamma_dist_MLM)**2)
print('mse for MLM is: ' + str(mse))
```

```
mse for KDE is: 7.703600732145187e-05
mse for MOM is: 7.935161626223034e-05
mse for MLM is: 7.707464251209312e-05
```

- I did the same thing for they and TN.
- These two states has the higher average value and smaller range compare to the US.



KY



TN

- The next step is to perform hypothesis testing for the 5 variables we selected from the previous stages. For each of the variables we need to separate data into high and low categories.
- I separated them based on the median normalized death.

In [38]:

```
import scipy.stats as stats
# create high and low data frame
selected_df_high = super_df[super_df['Norm_Deaths']>super_df['Norm_Deaths'].median()]
selected_df_low = super_df[super_df['Norm_Deaths']<=super_df['Norm_Deaths'].median()]
# filter thosed data frame for five selected variables
selected_df_low = selected_df_low [['Primary care physicians raw value','Life expectancy raw value','Premature death
,'Unemployment raw value','Some college raw value']]
selected_df_high = selected_df_high [['Primary care physicians raw value','Life expectancy raw value','Premature dea
,'Unemployment raw value','Some college raw value']]
```

- We need to chose the method for hypothesis testing. Because the variables are the same and continuous, so I think T-Test would be a good choice.
- After performing the T-Test, we can see that the last one failed to reject the null hypothesis.
- There is some difference that exists in the top four variables listed in the table for high and low normalized death values.

In [39]:

Slide Type -

```
# perform t-test
import scipy.stats as stats
p_value = []
variable = []
for column in selected_df_high.columns:
    a = selected_df_high[column].dropna()
    b = selected_df_low[column].dropna()
    p_value.append(stats.ttest_ind(a,
                b)[1])
    variable.append(column)


tmp_df = pd.DataFrame()
tmp_df['variable'] = variable
tmp_df['p_value'] = p_value
tmp_df['outcome'] = ['Reject the null hypothesis' if item<0.05 else 'Fail to Reject the null hypothesis' for item in
tmp_df
```

Out[39]:

	variable	p_value	outcome
0	Primary care physicians raw value	1.346543e-05	Reject the null hypothesis
1	Life expectancy raw value	4.559017e-23	Reject the null hypothesis
2	Premature death raw value	1.617207e-24	Reject the null hypothesis
3	Unemployment raw value	9.408775e-08	Reject the null hypothesis
4	Some college raw value	8.262440e-01	Fail to Reject the null hypothesis

- For the next task we want to perform the linear and non linear regression for the selected variables.
- First it is asked to normalize the Opioid_Dispensing_Rate, the Opioid_Dispensing_Rate is expressed per 100 person, so we need to divide it by 1,000 to be expressed per 100,000 population.

Slide Type -

```
# Opiod_Dispensing_Rate is expressed per 100 person, so we need to devide it by 1,000 to be expressed per 100,000 po
super_df['Norm_Opiod_Dispensing_Rate'] = (super_df['Opiod_Dispensing_Rate']/super_df['Population'])*1000
```

- In the next step I used "statsmodels" for creating a fitted linear regression model. Also, I replace the "college raw value" with "norm opioid dispensing rate".
- This linear regression provides us some parameters which are intercept and the coefficient for the other variables. So if we want to predict the Normalized Death, it is gonna be a combination of these parameters.

In [41]: Slide Type -

```
# I replace "Some college raw value" by "Norm_Opiod_Dispensing_Rate" due to two reasons:
# 1) the new variable should be considered because it expresses opioid related drug consumption
# 2) It does not differ for two groups with low and high normalized deaths.

import statsmodels.formula.api as smf
# create a fitted model in one line
lm = smf.ols(formula='Norm_Deaths ~ Q("Norm_Opiod_Dispensing_Rate") + Q("Primary care physicians raw value") + Q("Li

# print the coefficients
lm.params
```

Out[41]:

```
Intercept                                    -4.050354
Q("Norm_Opiod_Dispensing_Rate")              -4.395861
Q("Primary care physicians raw value")     3961.287115
Q("Life expectancy raw value")                0.043944
Q("Premature death raw value")                0.001133
Q("Unemployment raw value")                  25.986149
dtype: float64
```

- In the summary of results we can see the performance of the linear regression model.
- R-Square is low
- P-value for two parameters is greater than 5 percent, which means these two variables are not significant and it is better to be replaced with new variable.

Dep. Variable:	Norm_Deaths	**R-squared:**	0.158
Model:	OLS	**Adj. R-squared:**	0.156
Method:	Least Squares	**F-statistic:**	92.84
Date:	Mon, 21 Nov 2022	**Prob (F-statistic):**	8.59e-90
Time:	14:19:42	**Log-Likelihood:**	-8043.3
No. Observations:	2479	**AIC:**	1.610e+04
Df Residuals:	2473	**BIC:**	1.613e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.0504	10.921	-0.371	0.711	-25.465	17.364
Q("Norm_Opiod_Dispensing_Rate")	-4.3959	0.776	-5.668	0.000	-5.917	-2.875
Q("Primary care physicians raw value")	3961.2871	401.393	9.869	0.000	3174.186	4748.388
Q("Life expectancy raw value")	0.0439	0.127	0.346	0.730	-0.205	0.293
Q("Premature death raw value")	0.0011	0.000	7.615	0.000	0.001	0.001
Q("Unemployment raw value")	25.9861	9.168	2.834	0.005	8.008	43.964

Omnibus:	504.817	**Durbin-Watson:**	1.976
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1661.591
Skew:	1.008	**Prob(JB):**	0.00
Kurtosis:	6.467	**Cond. No.**	2.84e+07

- For the non linear regression model, I used the stats models too, but I squared one of the variables to create a non linear function.
- I added the "Some college raw value" and omitted the intercept for this model.

Slide Type -

```
#omitting the intercept
nlm = smf.ols(formula='Norm_Deaths ~ 0 + Q("Norm_Opiod_Dispensing_Rate")**2 + Q("Primary care physicians raw value")

# print the coefficients
nlm.params
```

```
Q("Norm_Opiod_Dispensing_Rate")          -4.087989
Q("Primary care physicians raw value")   2915.435745
Q("Life expectancy raw value")           -0.074986
Q("Premature death raw value")            0.001166
Q("Unemployment raw value")              35.265332
Q("Some college raw value")               8.705625
dtype: float64
```

- In the summary of the non linear regression, we can see that the R-square has increased significantly and there is no p-value greater than 5 percent.
- My non-linear regression performs better than the linear one.

```
nlm.summary()
```

OLS Regression Results

Dep. Variable:	Norm_Deaths	**R-squared (uncentered):**	0.804
Model:	OLS	**Adj. R-squared (uncentered):**	0.804
Method:	Least Squares	**F-statistic:**	1691.
Date:	Mon, 21 Nov 2022	**Prob (F-statistic):**	0.00
Time:	16:58:24	**Log-Likelihood:**	-8027.7
No. Observations:	2479	**AIC:**	1.607e+04
Df Residuals:	2473	**BIC:**	1.610e+04
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Q("Norm_Opiod_Dispensing_Rate")	-4.0880	0.772	-5.292	0.000	-5.603	-2.573
Q("Primary care physicians raw value")	2915.4357	439.280	6.637	0.000	2054.040	3776.831
Q("Life expectancy raw value")	-0.0750	0.015	-5.092	0.000	-0.104	-0.046
Q("Premature death raw value")	0.0012	5.77e-05	20.211	0.000	0.001	0.001
Q("Unemployment raw value")	35.2653	9.230	3.821	0.000	17.165	53.365
Q("Some college raw value")	8.7056	1.551	5.612	0.000	5.663	11.748

Omnibus:	496.287	**Durbin-Watson:**	1.982
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1705.631
Skew:	0.977	**Prob(JB):**	0.00
Kurtosis:	6.562	**Cond. No.**	3.13e+07