Economics 430: Project 3

By: George M, Omer A, Achmad R, Connor O

Fall 2023, UCLA

Instructor: Dr. Rojas

Due Date: Dec 6, 2023, 5:00pm

In [2]:
```
import matplotlib.pyplot as plt
import wooldridge as woo
import seaborn as sns
import statsmodels.api as sm
import statsmodels as sms
import statsmodels.formula.api as smf
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from RegscorePy import mallow
import statsmodels.stats.api as sms
from simple_colors import *
import scipy as sp
from RegscorePy import *
import scipy.stats as stats
from sklearn.ensemble import RandomForestRegressor
from statsmodels.stats.outliers_influence import variance_inflation_factor

import linearmodels as plm
```

For this project, you will work with any two datasets you like. One will be used for panel data models, and the other for qualitative dependent variable models (for this one you need to have at least 5 predictors). Your task will be to find the best fit model for each case by following the steps outlined below.

1. Briefly discuss your data and economic/finance/business question you are trying to answer with your model.

lot of airlines over short amount of time

In this analysis, we chose the airfare dataset from the wooldridge package to see what happens with the log of average fare of a ticket in relation to logs of distance, log of the average number of passengers and the market share of the flier or airline. Our panel data is Short and wide - there are many individual observations (N = 4596) over a relatively short period of time (T = 4). This dataset captures values from the year 1997 up to the year 2000.

In [3]:
```
# using the jobtraining dataset in the linearmodels package
data1 = woo.data('airfare')
year = pd.Categorical(data1.year)
data1 = data1.set_index(["id", "year"])
data1["year"] = year

data1.head()
```

Out[3]:

		dist	passen	fare	bmktshr	ldist	y98	y99	y00	lfare	ldistsq	concen	lpassen
id	**year**												
1	**1997**	528	152	106	0.8386	6.269096	0	0	0	4.663439	39.301571	0.8386	5.023880

	1998	528	265	106	0.8133	6.269096	1	0	0	4.663439	39.301571	0.8133	5.579730
	1999	528	336	113	0.8262	6.269096	0	1	0	4.727388	39.301571	0.8262	5.817111
	2000	528	298	123	0.8612	6.269096	0	0	1	4.812184	39.301571	0.8612	5.697093
2	**1997**	861	282	104	0.5798	6.758094	0	0	0	4.644391	45.671837	0.5798	5.641907

In [4]:
```
# Checking for null values in the dataset
data1.isnull().sum()
```

Out[4]:
```
dist       0
passen     0
fare       0
bmktshr    0
ldist      0
y98        0
y99        0
y00        0
lfare      0
ldistsq    0
concen     0
lpassen    0
year       0
dtype: int64
```

This dataset has zero null values

1. Provide a descriptive analysis of your variables. This should include histograms and fitted distributions, correlation plot, boxplots, scatterplots, and statistical summaries (e.g., the five-number summary). All figures must include comments.

In [5]:
```
# Statistical Summary - Five number summary
data1.describe()
```

Out[5]:

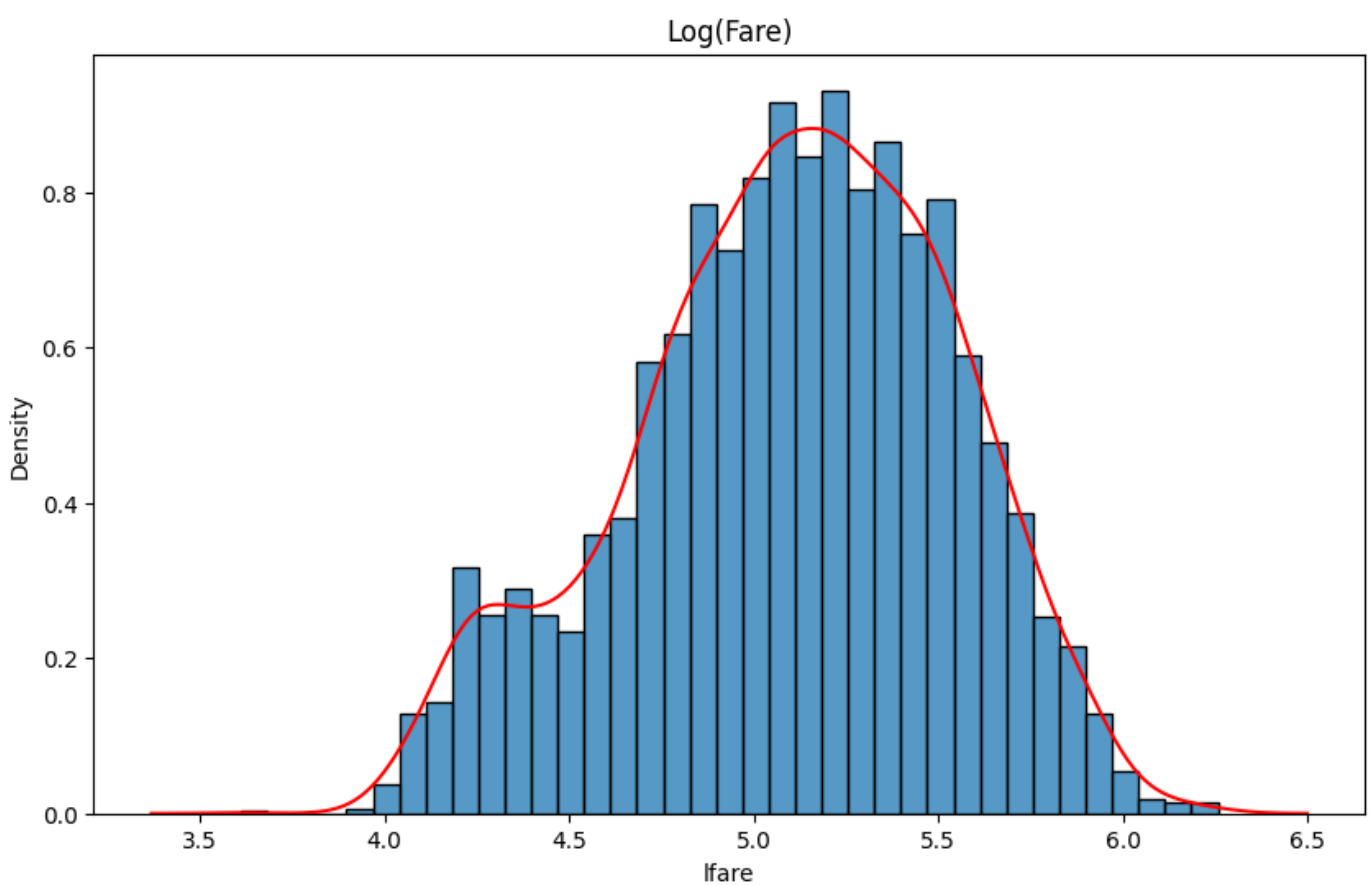
	dist	passen	fare	bmktshr	ldist	y98	y99	459
count	4596.000000	4596.000000	4596.000000	4596.000000	4596.000000	4596.00000	4596.00000	
mean	989.744996	636.824195	178.796780	0.610115	6.696482	0.25000	0.25000	
std	611.831506	811.999966	74.881513	0.196435	0.659318	0.43306	0.43306	
min	95.000000	2.000000	37.000000	0.160500	4.553877	0.00000	0.00000	
25%	505.000000	215.000000	123.000000	0.465000	6.224558	0.00000	0.00000	
50%	861.000000	357.000000	168.000000	0.603950	6.758094	0.00000	0.00000	
75%	1304.000000	717.000000	225.000000	0.753100	7.173192	0.25000	0.25000	
max	2724.000000	8497.000000	522.000000	1.000000	7.909857	1.00000	1.00000	

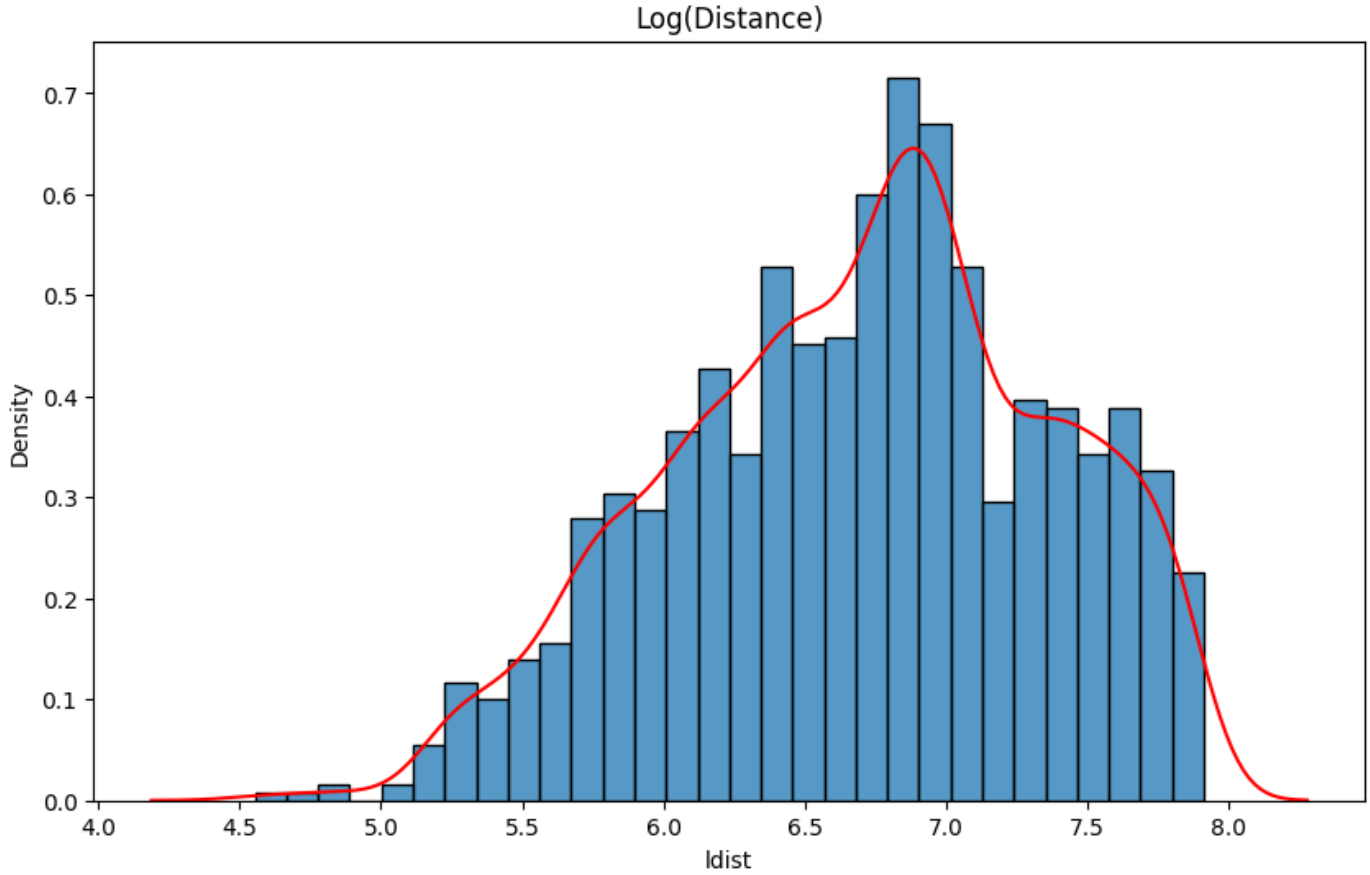
Comment on the above

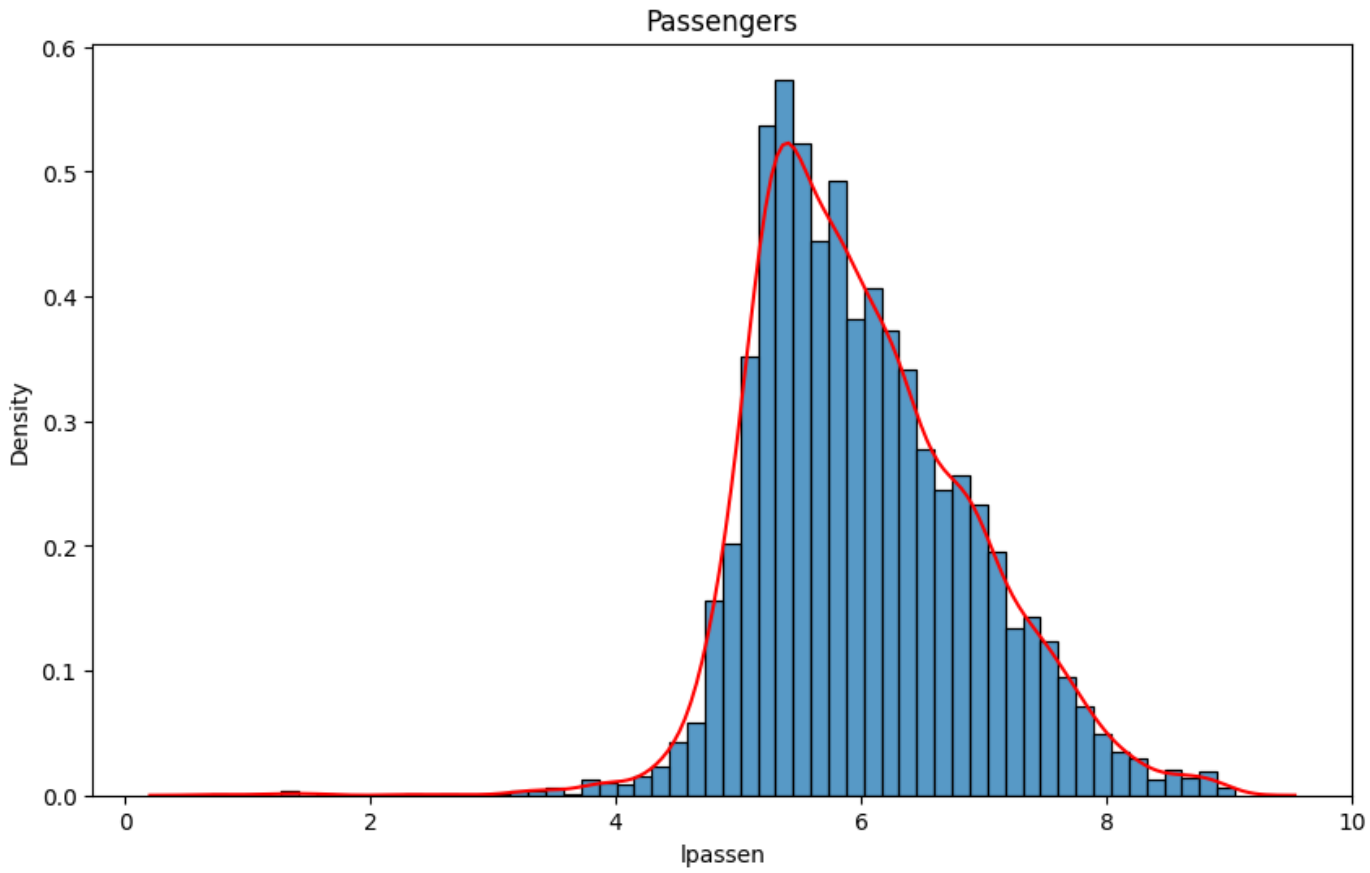
Histograms & Density plots

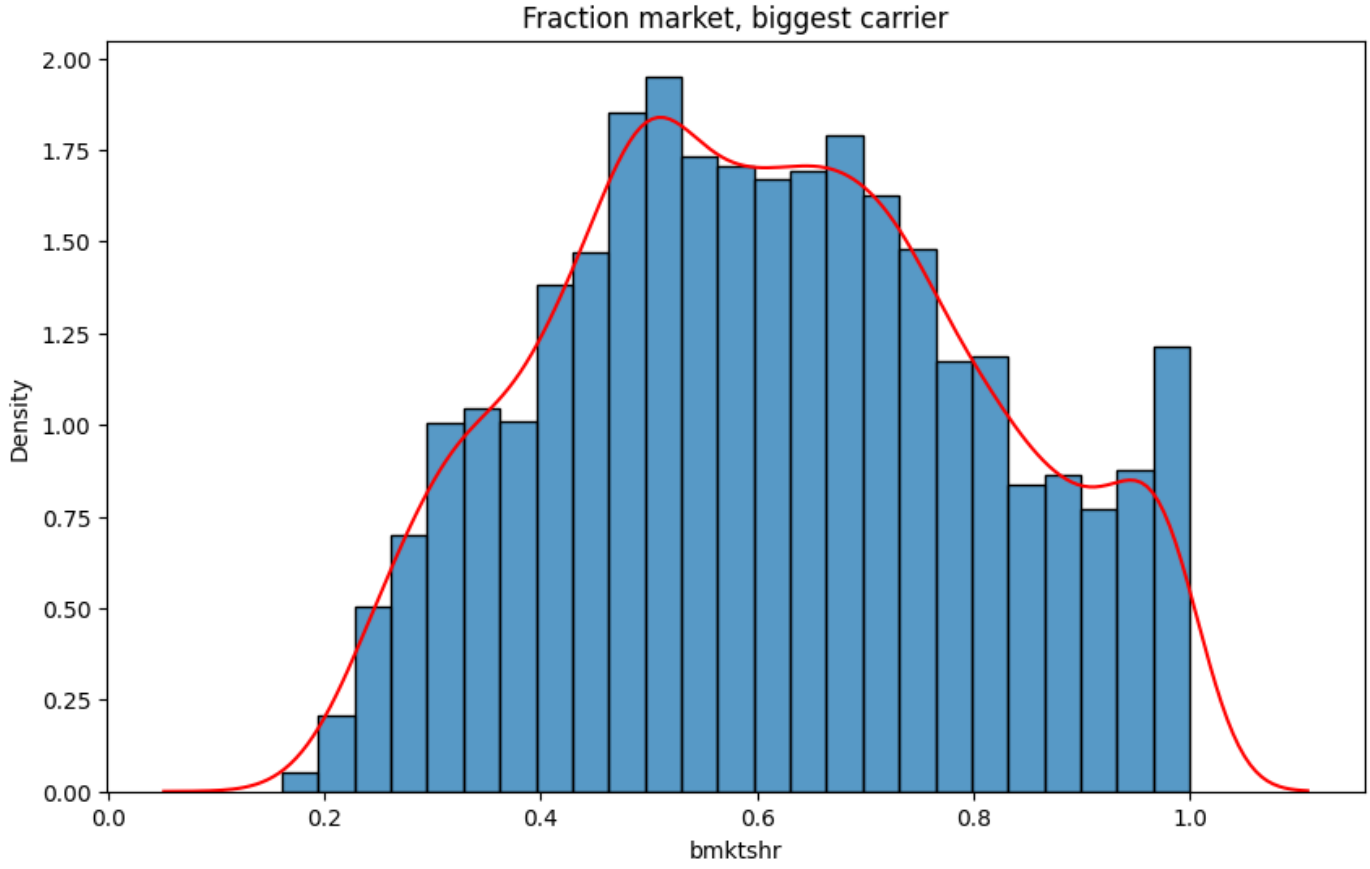
In [5]:
```
# Histograms and Density plots

plt.figure(figsize = (10,6))
sns.histplot(data1.lfare, stat = "density")
sns.kdeplot(data1.lfare, color = "red")
plt.title("Log(Fare)")
```

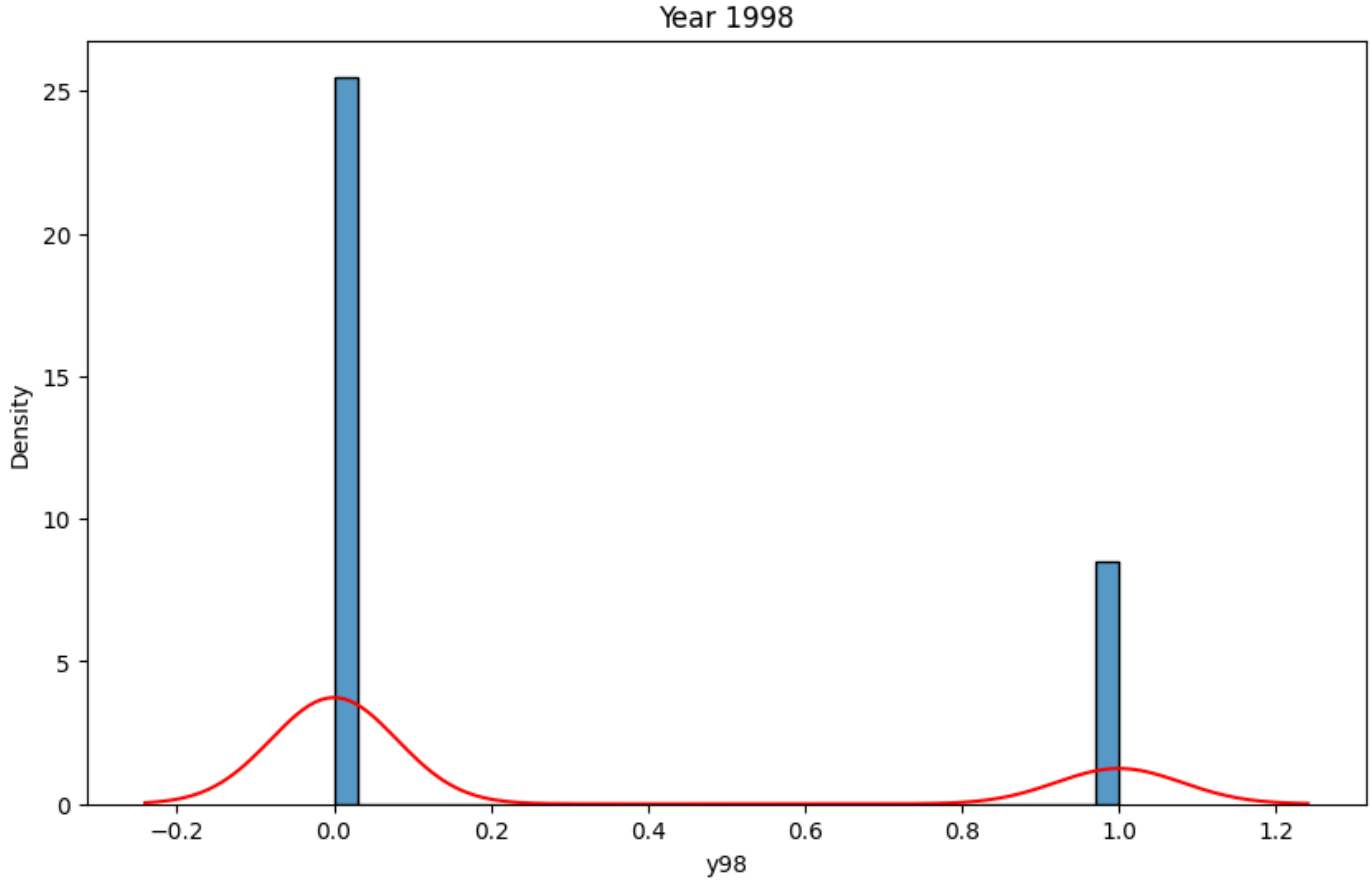
```
plt.show()

plt.figure(figsize = (10,6))
sns.histplot(data1.ldist, stat = "density")
sns.kdeplot(data1.ldist, color = "red")
plt.title("Log(Distance)")
plt.show()

plt.figure(figsize = (10,6))
sns.histplot(data1.lpassen, stat = "density")
sns.kdeplot(data1.lpassen, color = "red")
plt.title("Passengers")
plt.show()

plt.figure(figsize = (10,6))
sns.histplot(data1.bmktshr, stat = "density")
sns.kdeplot(data1.bmktshr, color = "red")
plt.title("Fraction market, biggest carrier")
plt.show()

plt.figure(figsize = (10,6))
sns.histplot(data1.y98, stat = "density")
sns.kdeplot(data1.y98, color = "red")
plt.title("Year 1998")
plt.show()

plt.figure(figsize = (10,6))
sns.histplot(data1.y99, stat = "density")
sns.kdeplot(data1.y99, color = "red")
plt.title("Year 1999")
plt.show()

plt.figure(figsize = (10,6))
sns.histplot(data1.y00, stat = "density")
sns.kdeplot(data1.y00, color = "red")
plt.title("Year 2000")
plt.show()
```

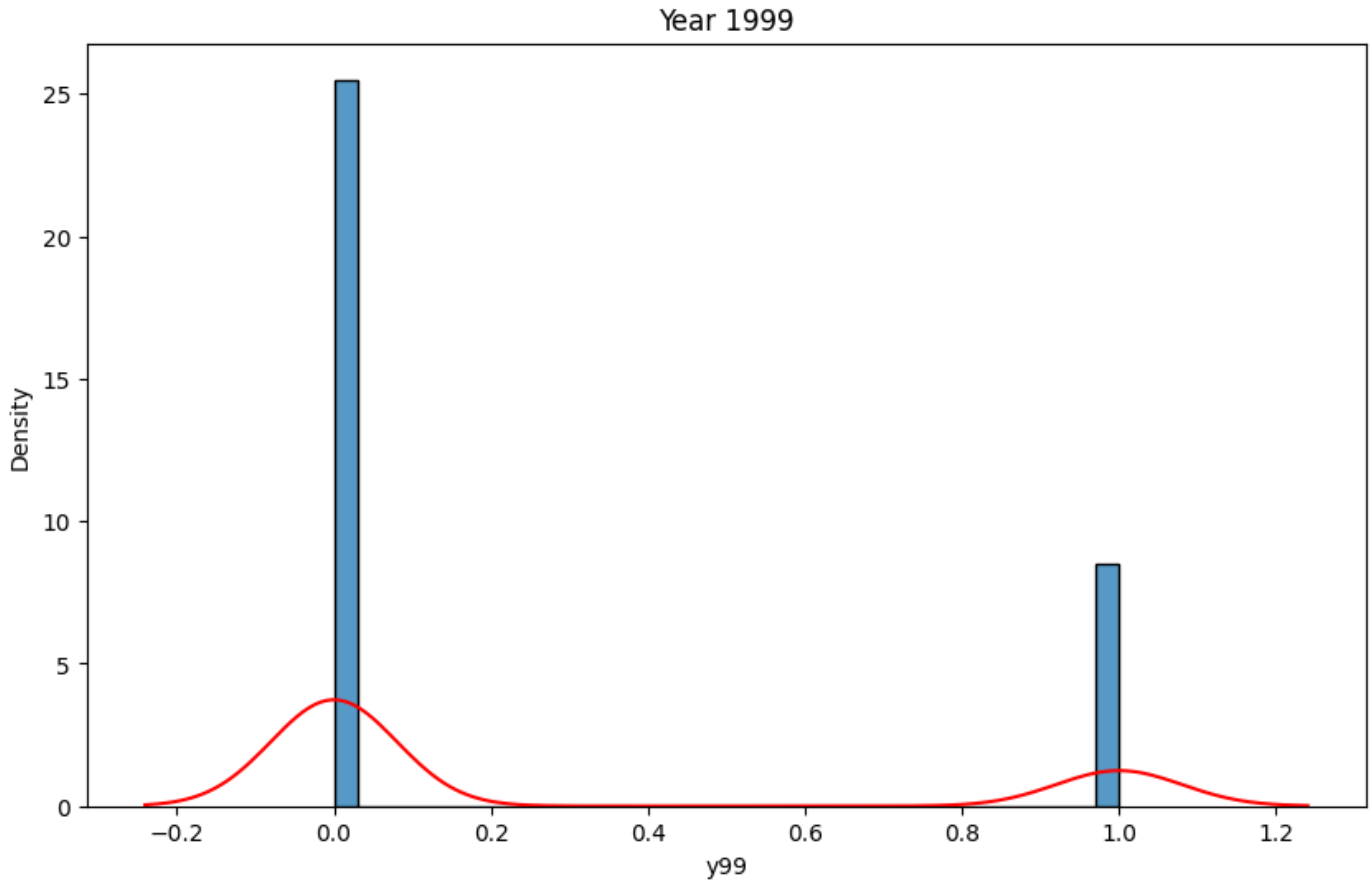


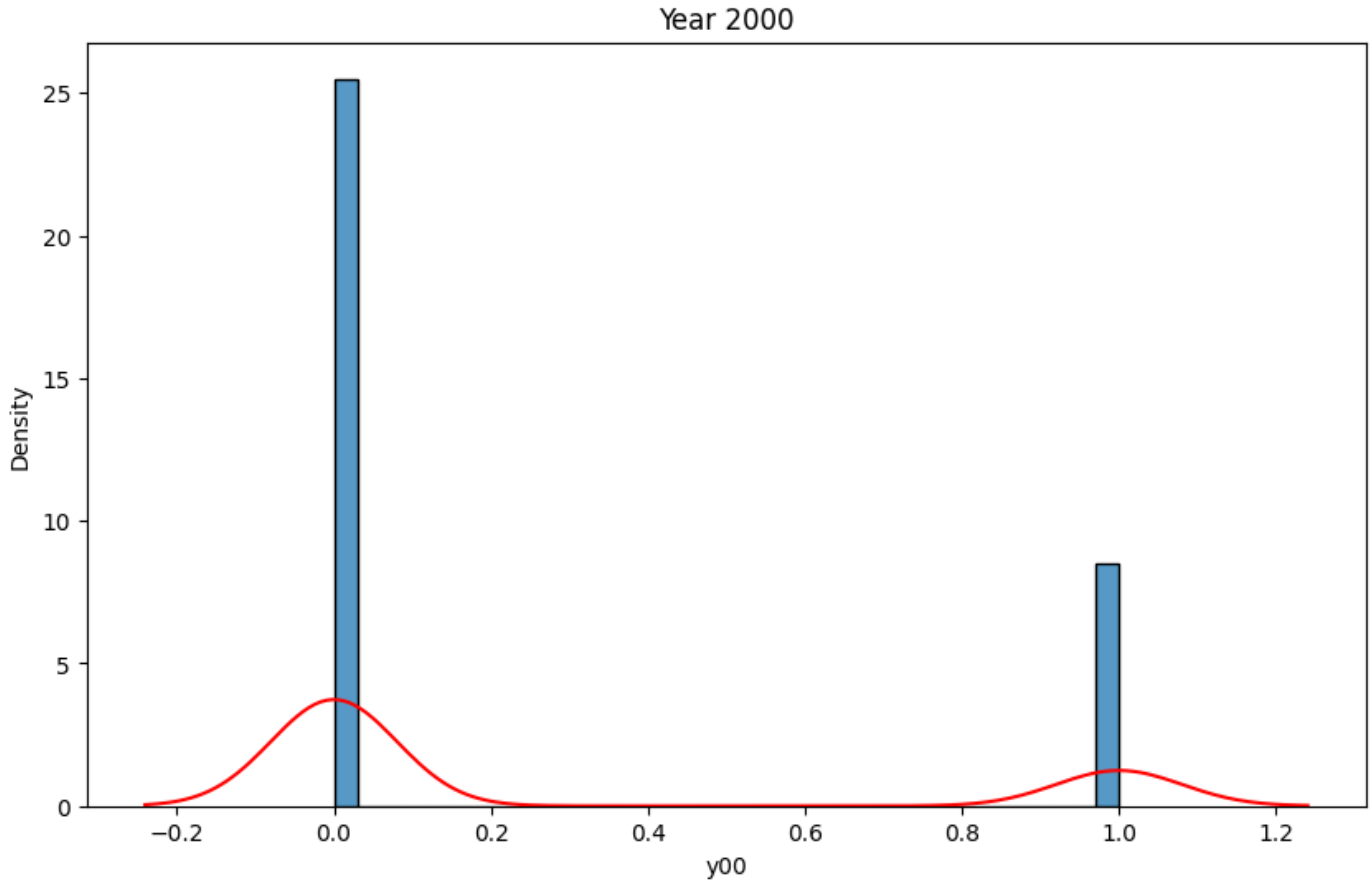

Log(Distance)
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
ldist


Passengers
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
2
4
6
8
10
lpassen


Fraction market, biggest carrier
Density
bmktshr
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0


Year 1998
Density
y98
25
20
15
10
5
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2



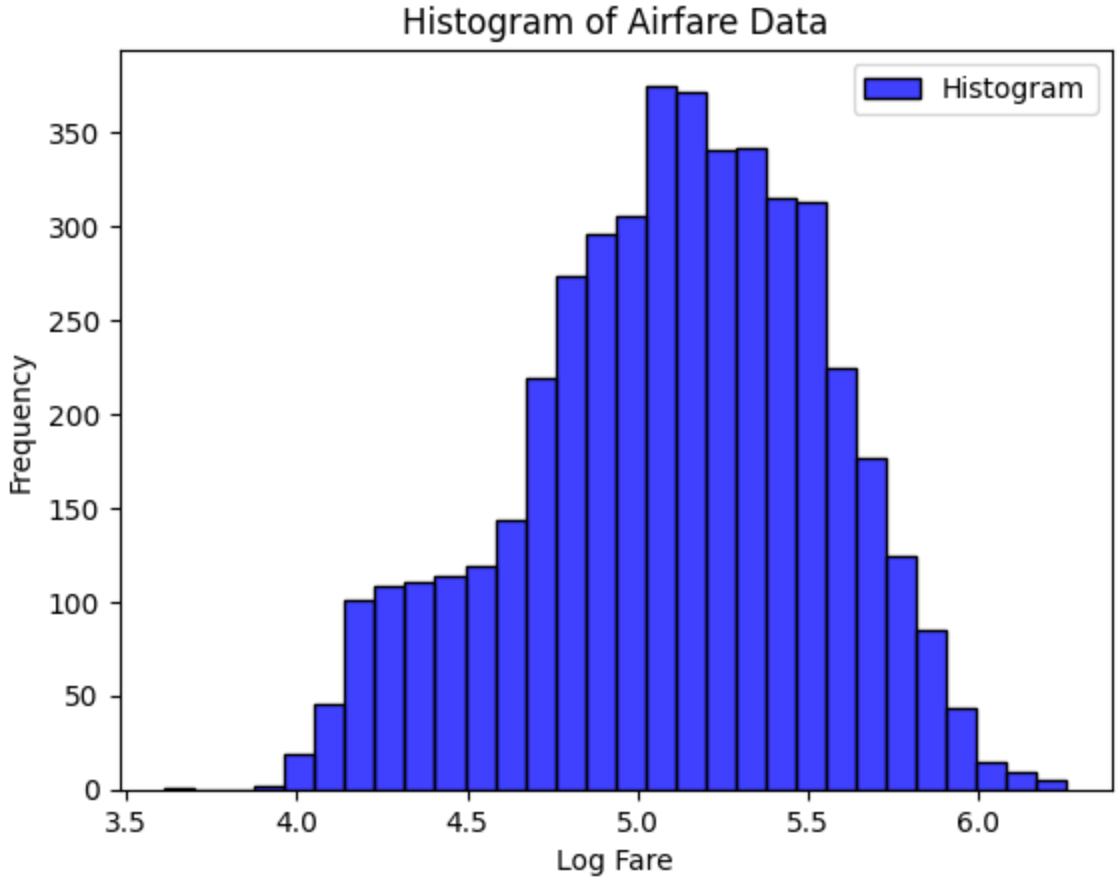


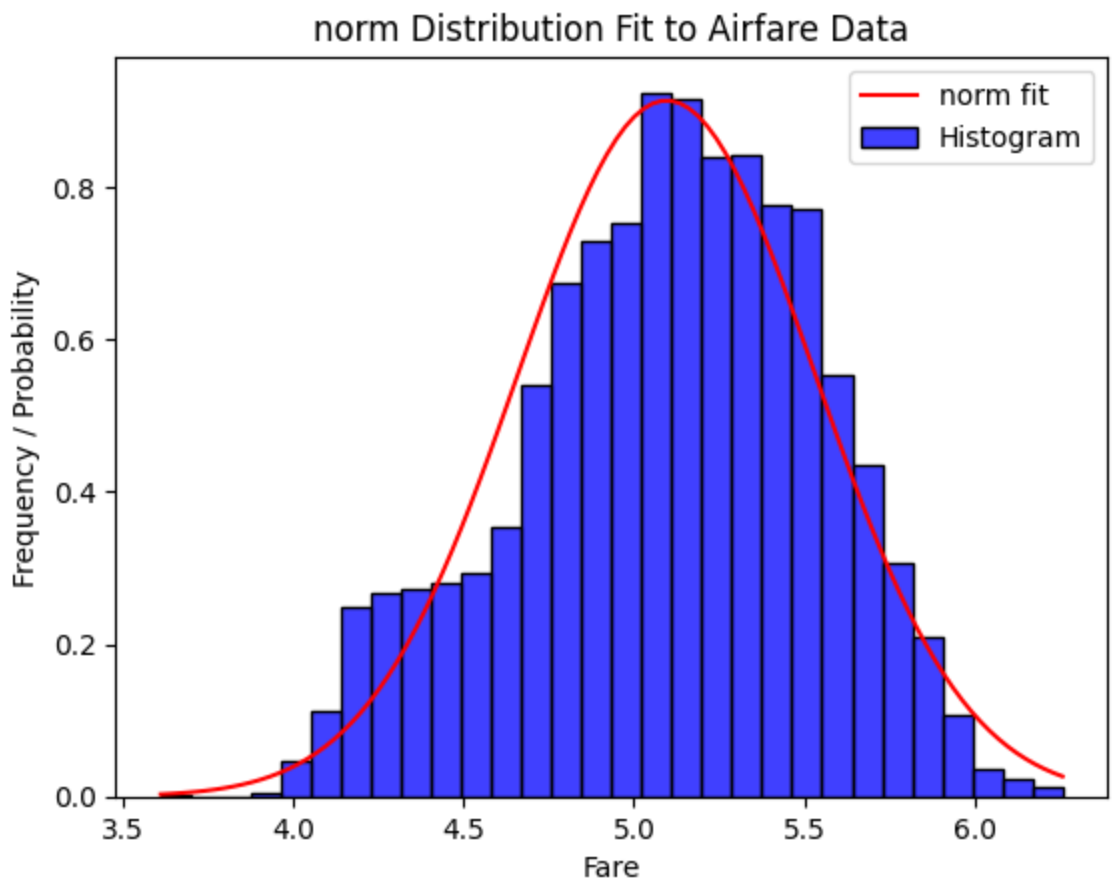
All of the histograms of the log variables (with an exception for years) appear to be normally distributed. As far as data transformation goes, there does not seem to be a need for it in these variables.

Fitted Distributions

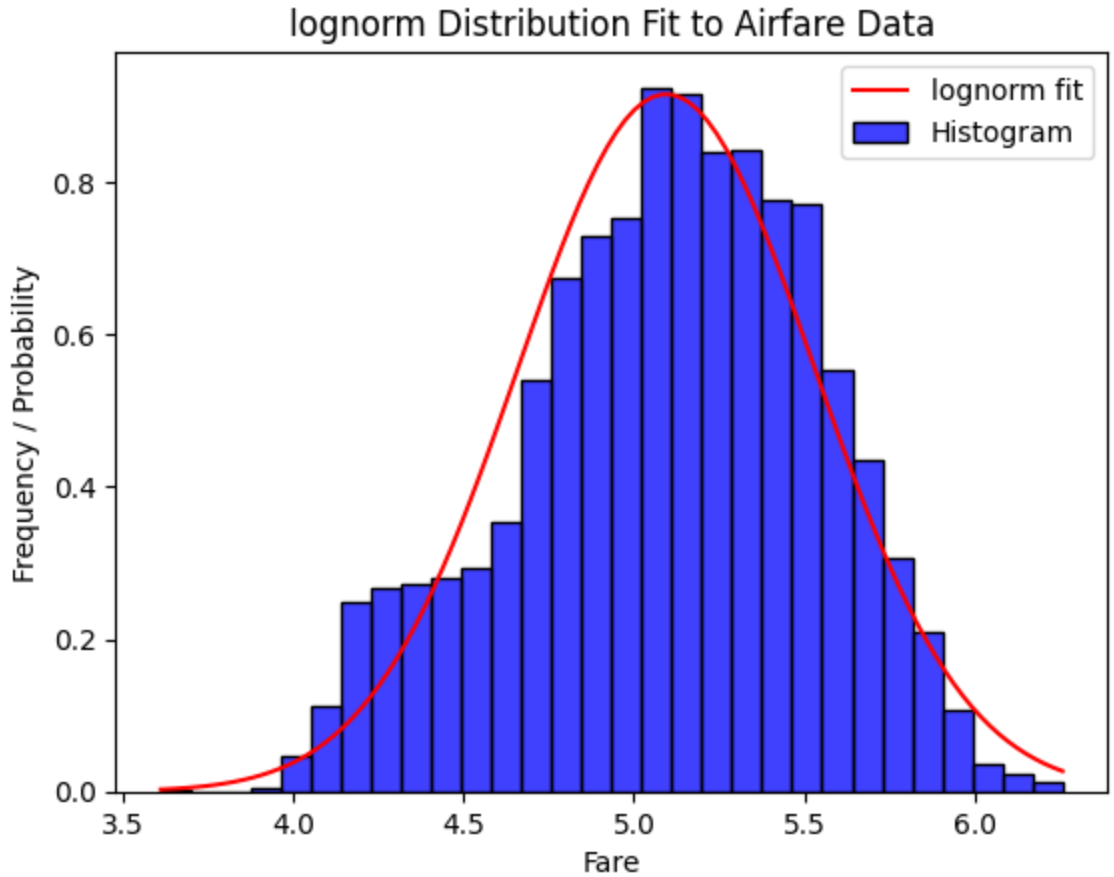
In [8]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import norm, lognorm, gamma, kstest
import seaborn as sns
import statsmodels.api as sm
import scipy

# Extract the 'log fare' column
fare_data = data1['lfare']

# Plot the histogram of the data
sns.histplot(fare_data, bins=30, kde=False, color='blue', label='Histogram')
plt.title('Histogram of Airfare Data')
plt.xlabel('Log Fare')
plt.ylabel('Frequency')
plt.legend()
plt.show()

# Fit different distributions to the data
dist_names = ['norm', 'lognorm', 'gamma']

for dist_name in dist_names:
    # Fit the distribution to the data
    dist = getattr(scipy.stats, dist_name)
    params = dist.fit(fare_data)

    # Plot the fitted PDF
    sns.histplot(fare_data, bins=30, stat = 'density', kde=False, color='blue', label='H
    x = np.linspace(fare_data.min(), fare_data.max(), 100)
    pdf_fitted = dist.pdf(x, *params)
    plt.plot(x, pdf_fitted, 'r-', label=f'{dist_name} fit')

    plt.title(f'{dist_name} Distribution Fit to Airfare Data')
    plt.xlabel('Fare')
    plt.ylabel('Frequency / Probability')
    plt.legend()
    plt.show()

    # Perform Kolmogorov-Smirnov test
    ks_statistic, ks_p_value = kstest(fare_data, dist_name, args=params)
    print(f'{dist_name} - KS Statistic: {ks_statistic}, P-value: {ks_p_value}')
```

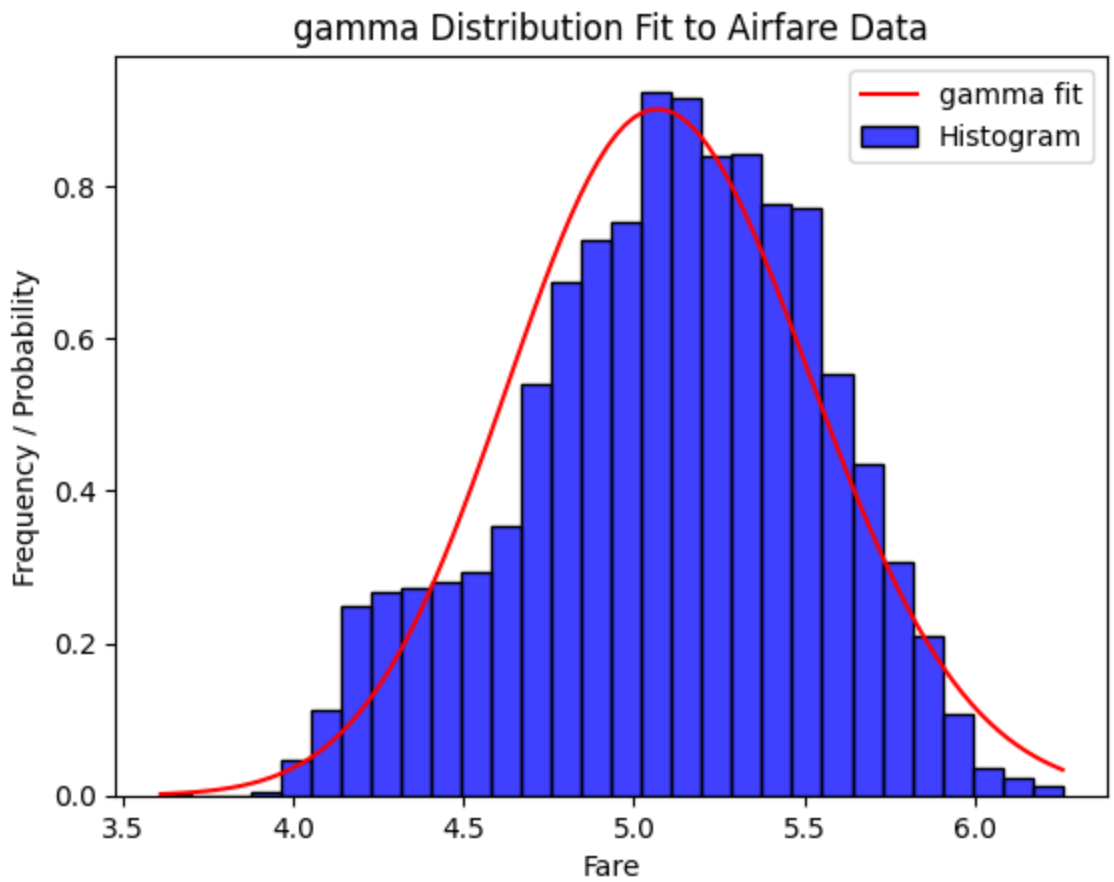





```
norm - KS Statistic: 0.03409909706720682, P-value: 4.4508820255598254e-05
```




```
lognorm - KS Statistic: 0.03424178272379186, P-value: 4.068731276071736e-05
```




```
gamma - KS Statistic: 0.040946020916200365, P-value: 3.9268519853503625e-07
```

Each of the histrograms above shows different distribution fits with the associated p-values

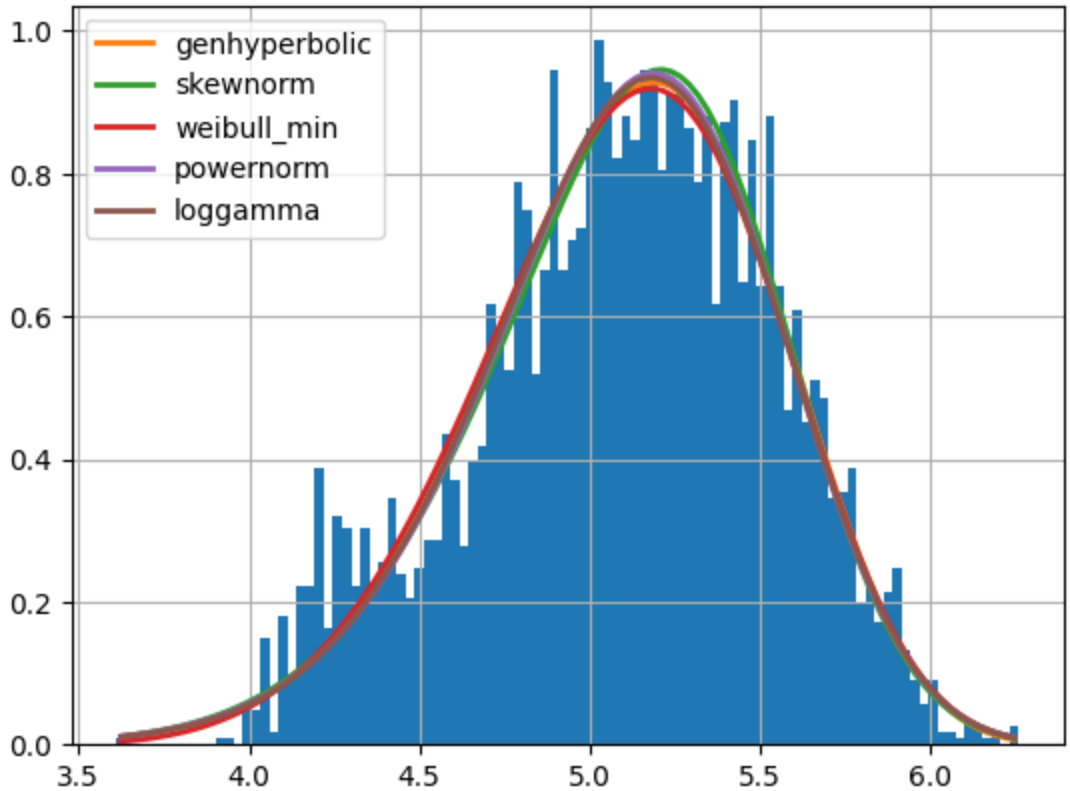
Density Distribitions - Cullen & Frey

In [7]:
```python
# Density distributions - Cullen & Frey
# lfare
from fitter import Fitter
f = Fitter(data1.lfare)
f.fit()
f.summary()
```

```
SKIPPED _fit distribution (taking more than 30 seconds)
SKIPPED johnsonsu distribution (taking more than 30 seconds)
SKIPPED kappa4 distribution (taking more than 30 seconds)
SKIPPED ksone distribution (taking more than 30 seconds)
SKIPPED levy_stable distribution (taking more than 30 seconds)
SKIPPED lognorm distribution (taking more than 30 seconds)
SKIPPED lomax distribution (taking more than 30 seconds)
SKIPPED mielke distribution (taking more than 30 seconds)
SKIPPED nakagami distribution (taking more than 30 seconds)
SKIPPED nct distribution (taking more than 30 seconds)
SKIPPED ncf distribution (taking more than 30 seconds)
SKIPPED ncx2 distribution (taking more than 30 seconds)
SKIPPED norminvgauss distribution (taking more than 30 seconds)
SKIPPED rv_continuous distribution (taking more than 30 seconds)
SKIPPED rv_histogram distribution (taking more than 30 seconds)
SKIPPED pearson3 distribution (taking more than 30 seconds)
SKIPPED powerlognorm distribution (taking more than 30 seconds)
/Users/nali/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225:
IntegrationWarning: The maximum number of subdivisions (50) has been achieved.
  If increasing the limit yields no improvement it is advised to analyze
  the integrand in order to determine the difficulties.  If the position of a
  local difficulty can be determined (singularity, discontinuity) one will
  probably gain from splitting up the interval and calling the integrator
  on the subranges.  Perhaps a special-purpose integrator should be used.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
SKIPPED rdist distribution (taking more than 30 seconds)
SKIPPED recipinvgauss distribution (taking more than 30 seconds)
SKIPPED rice distribution (taking more than 30 seconds)
/Users/nali/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225:
IntegrationWarning: The integral is probably divergent, or slowly convergent.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
SKIPPED studentized_range distribution (taking more than 30 seconds)
SKIPPED t distribution (taking more than 30 seconds)
SKIPPED trapezoid distribution (taking more than 30 seconds)
SKIPPED triang distribution (taking more than 30 seconds)
SKIPPED truncexpon distribution (taking more than 30 seconds)
SKIPPED truncnorm distribution (taking more than 30 seconds)
SKIPPED truncpareto distribution (taking more than 30 seconds)
SKIPPED truncweibull_min distribution (taking more than 30 seconds)
SKIPPED tukeylambda distribution (taking more than 30 seconds)
SKIPPED vonmises distribution (taking more than 30 seconds)
SKIPPED vonmises_line distribution (taking more than 30 seconds)
SKIPPED wrapcauchy distribution (taking more than 30 seconds)
```

Out[7]:

	sumsquare_error	aic	bic	kl_div	ks_statistic	ks_pvalue
genhyperbolic	0.680135	339.316408	371.481116	inf	0.019692	0.055876
skewnorm	0.691384	332.826878	352.125703	inf	0.018438	0.086786
weibull_min	0.693190	343.546890	362.845715	inf	0.023262	0.013612
powernorm	0.701481	332.328304	351.627129	inf	0.020300	0.044665
loggamma	0.702675	333.306686	352.605511	inf	0.020869	0.036000



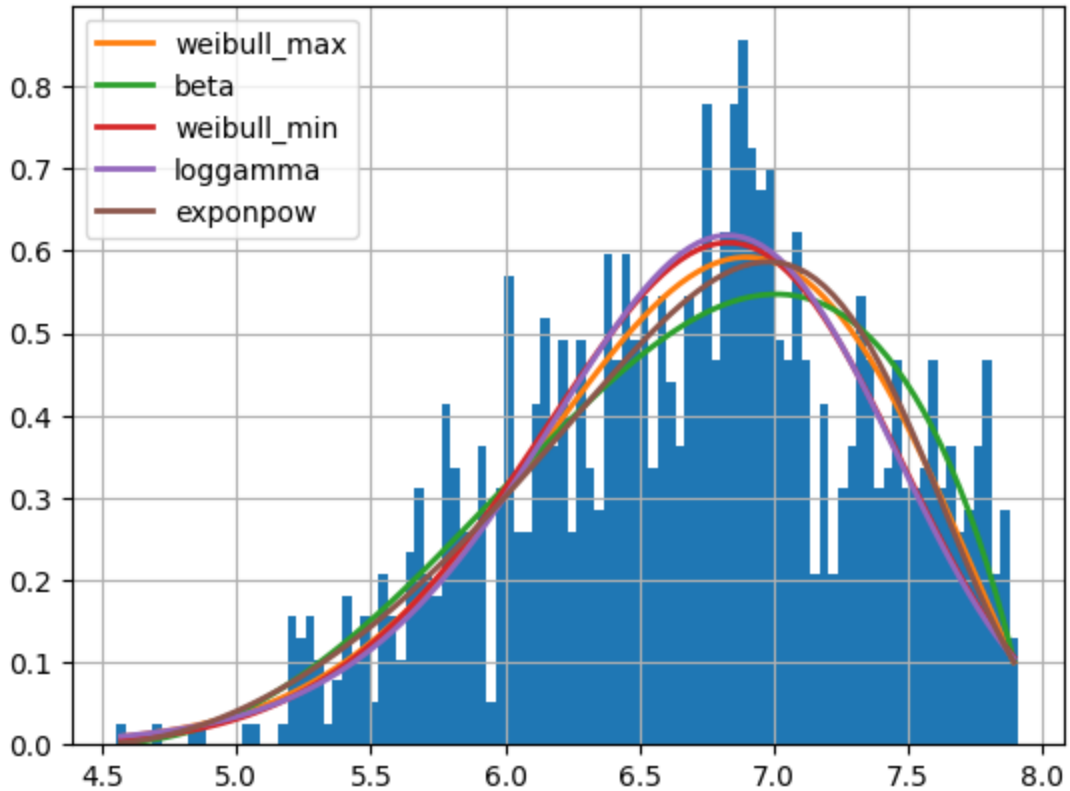
In [8]:
```
# ldist
f = Fitter(data1.ldist)
f.fit()
f.summary()
```

```
SKIPPED _fit distribution (taking more than 30 seconds)
SKIPPED alpha distribution (taking more than 30 seconds)
SKIPPED arcsine distribution (taking more than 30 seconds)
SKIPPED betaprime distribution (taking more than 30 seconds)
SKIPPED burr distribution (taking more than 30 seconds)
SKIPPED burr12 distribution (taking more than 30 seconds)
SKIPPED chi distribution (taking more than 30 seconds)
SKIPPED chi2 distribution (taking more than 30 seconds)
SKIPPED crystalball distribution (taking more than 30 seconds)
SKIPPED erlang distribution (taking more than 30 seconds)
SKIPPED exponweib distribution (taking more than 30 seconds)
SKIPPED f distribution (taking more than 30 seconds)
SKIPPED fatiguelife distribution (taking more than 30 seconds)
SKIPPED fisk distribution (taking more than 30 seconds)
SKIPPED foldcauchy distribution (taking more than 30 seconds)
SKIPPED foldnorm distribution (taking more than 30 seconds)
SKIPPED gamma distribution (taking more than 30 seconds)
SKIPPED gausshyper distribution (taking more than 30 seconds)
SKIPPED genexpon distribution (taking more than 30 seconds)
SKIPPED genextreme distribution (taking more than 30 seconds)
SKIPPED gengamma distribution (taking more than 30 seconds)
SKIPPED genhalflogistic distribution (taking more than 30 seconds)
SKIPPED genhyperbolic distribution (taking more than 30 seconds)
SKIPPED geninvgauss distribution (taking more than 30 seconds)
SKIPPED genpareto distribution (taking more than 30 seconds)
SKIPPED halfgennorm distribution (taking more than 30 seconds)
SKIPPED invgamma distribution (taking more than 30 seconds)
SKIPPED kstwo distribution (taking more than 30 seconds)
SKIPPED invgauss distribution (taking more than 30 seconds)
SKIPPED invweibull distribution (taking more than 30 seconds)
SKIPPED johnsonsb distribution (taking more than 30 seconds)
SKIPPED johnsonsu distribution (taking more than 30 seconds)
SKIPPED kappa3 distribution (taking more than 30 seconds)
```

```
SKIPPED kappa4 distribution (taking more than 30 seconds)
SKIPPED ksone distribution (taking more than 30 seconds)
SKIPPED levy distribution (taking more than 30 seconds)
SKIPPED levy_stable distribution (taking more than 30 seconds)
SKIPPED loglaplace distribution (taking more than 30 seconds)
SKIPPED lognorm distribution (taking more than 30 seconds)
SKIPPED lomax distribution (taking more than 30 seconds)
SKIPPED mielke distribution (taking more than 30 seconds)
SKIPPED nakagami distribution (taking more than 30 seconds)
SKIPPED ncf distribution (taking more than 30 seconds)
SKIPPED nct distribution (taking more than 30 seconds)
SKIPPED ncx2 distribution (taking more than 30 seconds)
SKIPPED norminvgauss distribution (taking more than 30 seconds)
SKIPPED rv_continuous distribution (taking more than 30 seconds)
SKIPPED rv_histogram distribution (taking more than 30 seconds)
SKIPPED pearson3 distribution (taking more than 30 seconds)
SKIPPED powerlognorm distribution (taking more than 30 seconds)
SKIPPED powernorm distribution (taking more than 30 seconds)
SKIPPED rdist distribution (taking more than 30 seconds)
SKIPPED recipinvgauss distribution (taking more than 30 seconds)
SKIPPED rice distribution (taking more than 30 seconds)
SKIPPED skewcauchy distribution (taking more than 30 seconds)
/Users/nali/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225:
IntegrationWarning: The maximum number of subdivisions (50) has been achieved.
  If increasing the limit yields no improvement it is advised to analyze
  the integrand in order to determine the difficulties.  If the position of a
  local difficulty can be determined (singularity, discontinuity) one will
  probably gain from splitting up the interval and calling the integrator
  on the subranges.  Perhaps a special-purpose integrator should be used.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
SKIPPED skewnorm distribution (taking more than 30 seconds)
SKIPPED studentized_range distribution (taking more than 30 seconds)
SKIPPED t distribution (taking more than 30 seconds)
SKIPPED trapezoid distribution (taking more than 30 seconds)
SKIPPED trapz distribution (taking more than 30 seconds)
SKIPPED triang distribution (taking more than 30 seconds)
SKIPPED truncexpon distribution (taking more than 30 seconds)
SKIPPED truncnorm distribution (taking more than 30 seconds)
/Users/nali/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225:
IntegrationWarning: The integral is probably divergent, or slowly convergent.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
SKIPPED truncpareto distribution (taking more than 30 seconds)
SKIPPED truncweibull_min distribution (taking more than 30 seconds)
SKIPPED tukeylambda distribution (taking more than 30 seconds)
SKIPPED vonmises distribution (taking more than 30 seconds)
SKIPPED vonmises_line distribution (taking more than 30 seconds)
SKIPPED wrapcauchy distribution (taking more than 30 seconds)
```

Out[8]:

	sumsquare_error	aic	bic	kl_div	ks_statistic	ks_pvalue
weibull_max	1.250410	340.628395	359.927220	inf	0.030386	4.033514e-04
beta	1.257455	350.626916	376.358683	inf	0.048078	1.135784e-09
weibull_min	1.263615	354.223048	373.521873	inf	0.031012	2.831761e-04
loggamma	1.292017	347.967220	367.266045	inf	0.032589	1.124952e-04
exponpow	1.298514	347.739230	367.038055	inf	0.039376	1.253485e-06



In [9]:

```
# lpassen
f = Fitter(data1.lpassen)
f.fit()
f.summary()
```

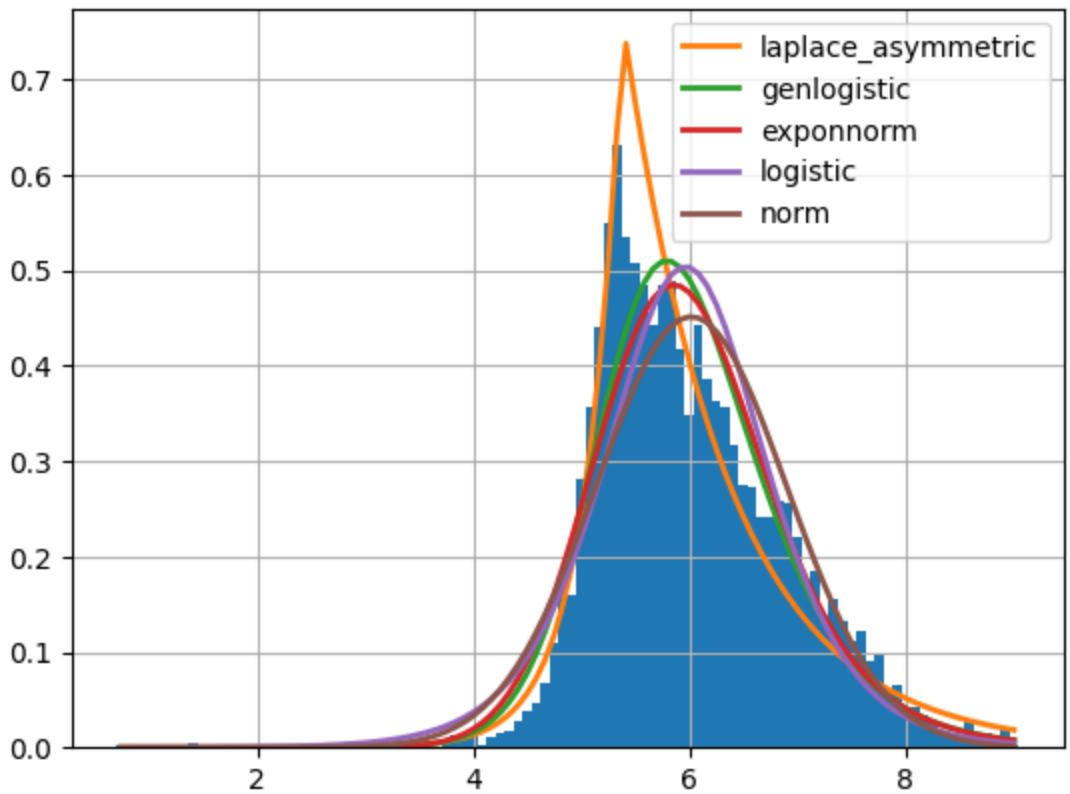
```
SKIPPED _fit distribution (taking more than 30 seconds)
SKIPPED alpha distribution (taking more than 30 seconds)
SKIPPED argus distribution (taking more than 30 seconds)
SKIPPED betaprime distribution (taking more than 30 seconds)
SKIPPED arcsine distribution (taking more than 30 seconds)
SKIPPED beta distribution (taking more than 30 seconds)
SKIPPED burr distribution (taking more than 30 seconds)
SKIPPED burr12 distribution (taking more than 30 seconds)
SKIPPED chi distribution (taking more than 30 seconds)
SKIPPED chi2 distribution (taking more than 30 seconds)
SKIPPED crystalball distribution (taking more than 30 seconds)
SKIPPED erlang distribution (taking more than 30 seconds)
SKIPPED exponpow distribution (taking more than 30 seconds)
SKIPPED exponweib distribution (taking more than 30 seconds)
SKIPPED f distribution (taking more than 30 seconds)
SKIPPED fatiguelife distribution (taking more than 30 seconds)
SKIPPED fisk distribution (taking more than 30 seconds)
SKIPPED foldcauchy distribution (taking more than 30 seconds)
SKIPPED gamma distribution (taking more than 30 seconds)
SKIPPED gausshyper distribution (taking more than 30 seconds)
SKIPPED genexpon distribution (taking more than 30 seconds)
SKIPPED genextreme distribution (taking more than 30 seconds)
SKIPPED gengamma distribution (taking more than 30 seconds)
SKIPPED genhalflogistic distribution (taking more than 30 seconds)
SKIPPED genhyperbolic distribution (taking more than 30 seconds)
SKIPPED geninvgauss distribution (taking more than 30 seconds)
SKIPPED genpareto distribution (taking more than 30 seconds)
SKIPPED gibrat distribution (taking more than 30 seconds)
SKIPPED gilbrat distribution (taking more than 30 seconds)
SKIPPED gompertz distribution (taking more than 30 seconds)
SKIPPED halfcauchy distribution (taking more than 30 seconds)
SKIPPED halfgennorm distribution (taking more than 30 seconds)
SKIPPED halflogistic distribution (taking more than 30 seconds)
```

```
SKIPPED invgamma distribution (taking more than 30 seconds)
SKIPPED kstwo distribution (taking more than 30 seconds)
SKIPPED invgauss distribution (taking more than 30 seconds)
SKIPPED invweibull distribution (taking more than 30 seconds)
SKIPPED johnsonsb distribution (taking more than 30 seconds)
SKIPPED johnsonsu distribution (taking more than 30 seconds)
SKIPPED kappa3 distribution (taking more than 30 seconds)
SKIPPED kappa4 distribution (taking more than 30 seconds)
SKIPPED ksone distribution (taking more than 30 seconds)
SKIPPED levy_stable distribution (taking more than 30 seconds)
SKIPPED loggamma distribution (taking more than 30 seconds)
SKIPPED loglaplace distribution (taking more than 30 seconds)
SKIPPED lognorm distribution (taking more than 30 seconds)
SKIPPED loguniform distribution (taking more than 30 seconds)
SKIPPED lomax distribution (taking more than 30 seconds)
SKIPPED mielke distribution (taking more than 30 seconds)
SKIPPED nakagami distribution (taking more than 30 seconds)
SKIPPED ncf distribution (taking more than 30 seconds)
SKIPPED nct distribution (taking more than 30 seconds)
SKIPPED ncx2 distribution (taking more than 30 seconds)
SKIPPED norminvgauss distribution (taking more than 30 seconds)
SKIPPED rv_continuous distribution (taking more than 30 seconds)
SKIPPED rv_histogram distribution (taking more than 30 seconds)
SKIPPED pearson3 distribution (taking more than 30 seconds)
SKIPPED powerlognorm distribution (taking more than 30 seconds)
SKIPPED powernorm distribution (taking more than 30 seconds)
SKIPPED rdist distribution (taking more than 30 seconds)
SKIPPED recipinvgauss distribution (taking more than 30 seconds)
SKIPPED reciprocal distribution (taking more than 30 seconds)
SKIPPED rice distribution (taking more than 30 seconds)
SKIPPED semicircular distribution (taking more than 30 seconds)
SKIPPED skewcauchy distribution (taking more than 30 seconds)
SKIPPED skewnorm distribution (taking more than 30 seconds)
SKIPPED studentized_range distribution (taking more than 30 seconds)
/Users/nali/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225:
IntegrationWarning: The maximum number of subdivisions (50) has been achieved.
  If increasing the limit yields no improvement it is advised to analyze
  the integrand in order to determine the difficulties.  If the position of a
  local difficulty can be determined (singularity, discontinuity) one will
  probably gain from splitting up the interval and calling the integrator
  on the subranges.  Perhaps a special-purpose integrator should be used.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
SKIPPED t distribution (taking more than 30 seconds)
SKIPPED trapezoid distribution (taking more than 30 seconds)
SKIPPED trapz distribution (taking more than 30 seconds)
SKIPPED triang distribution (taking more than 30 seconds)
SKIPPED truncexpon distribution (taking more than 30 seconds)
/Users/nali/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225:
IntegrationWarning: The integral is probably divergent, or slowly convergent.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
SKIPPED truncnorm distribution (taking more than 30 seconds)
SKIPPED truncpareto distribution (taking more than 30 seconds)
SKIPPED truncweibull_min distribution (taking more than 30 seconds)
SKIPPED tukeylambda distribution (taking more than 30 seconds)
SKIPPED vonmises distribution (taking more than 30 seconds)
SKIPPED vonmises_line distribution (taking more than 30 seconds)
SKIPPED wald distribution (taking more than 30 seconds)
SKIPPED weibull_max distribution (taking more than 30 seconds)
SKIPPED weibull_min distribution (taking more than 30 seconds)
SKIPPED wrapcauchy distribution (taking more than 30 seconds)
```

Out[9]:

	sumsquare_error	aic	bic	kl_div	ks_statistic	ks_pvalue
laplace_asymmetric	0.187106	996.081913	1015.380738	inf	0.059915	8.751419e-15
genlogistic	0.192727	1117.078942	1136.377767	inf	0.038571	2.234721e-06

exponnorm	0.225678	1364.462117	1383.760942	inf	0.040884	4.114174e-07
logistic	0.352344	802.386543	815.252426	inf	0.052125	2.712414e-11
norm	0.356135	1075.805307	1088.671190	inf	0.064410	5.083307e-17



In [10]:

```
# bmktshr
f = Fitter(data1.bmktshr)
f.fit()
f.summary()
```

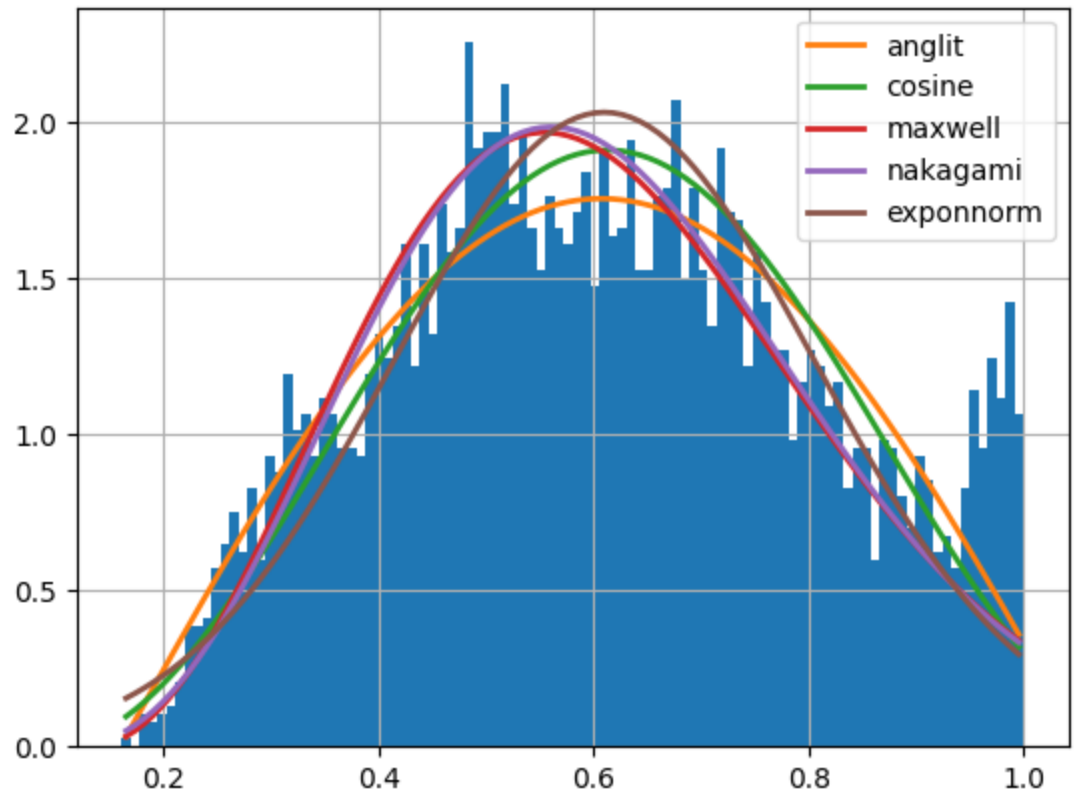
```
SKIPPED _fit distribution (taking more than 30 seconds)
SKIPPED alpha distribution (taking more than 30 seconds)
SKIPPED arcsine distribution (taking more than 30 seconds)
SKIPPED betaprime distribution (taking more than 30 seconds)
SKIPPED bradford distribution (taking more than 30 seconds)
SKIPPED beta distribution (taking more than 30 seconds)
SKIPPED argus distribution (taking more than 30 seconds)
SKIPPED burr distribution (taking more than 30 seconds)
SKIPPED burr12 distribution (taking more than 30 seconds)
SKIPPED chi distribution (taking more than 30 seconds)
SKIPPED chi2 distribution (taking more than 30 seconds)
SKIPPED crystalball distribution (taking more than 30 seconds)
SKIPPED dgamma distribution (taking more than 30 seconds)
SKIPPED erlang distribution (taking more than 30 seconds)
SKIPPED exponpow distribution (taking more than 30 seconds)
SKIPPED exponweib distribution (taking more than 30 seconds)
SKIPPED f distribution (taking more than 30 seconds)
SKIPPED fatiguelife distribution (taking more than 30 seconds)
SKIPPED fisk distribution (taking more than 30 seconds)
SKIPPED foldcauchy distribution (taking more than 30 seconds)
SKIPPED foldnorm distribution (taking more than 30 seconds)
SKIPPED gamma distribution (taking more than 30 seconds)
SKIPPED gausshyper distribution (taking more than 30 seconds)
SKIPPED genexpon distribution (taking more than 30 seconds)
SKIPPED genextreme distribution (taking more than 30 seconds)
SKIPPED gengamma distribution (taking more than 30 seconds)
SKIPPED genhalflogistic distribution (taking more than 30 seconds)
SKIPPED genhyperbolic distribution (taking more than 30 seconds)
```

```
SKIPPED geninvgauss distribution (taking more than 30 seconds)
SKIPPED genlogistic distribution (taking more than 30 seconds)
SKIPPED gennorm distribution (taking more than 30 seconds)
SKIPPED genpareto distribution (taking more than 30 seconds)
SKIPPED gibrat distribution (taking more than 30 seconds)
SKIPPED gilbrat distribution (taking more than 30 seconds)
SKIPPED gompertz distribution (taking more than 30 seconds)
SKIPPED halfcauchy distribution (taking more than 30 seconds)
SKIPPED halfgennorm distribution (taking more than 30 seconds)
SKIPPED halflogistic distribution (taking more than 30 seconds)
SKIPPED invgamma distribution (taking more than 30 seconds)
SKIPPED invgauss distribution (taking more than 30 seconds)
SKIPPED invweibull distribution (taking more than 30 seconds)
SKIPPED johnsonsb distribution (taking more than 30 seconds)
SKIPPED johnsonsu distribution (taking more than 30 seconds)
SKIPPED kappa3 distribution (taking more than 30 seconds)
SKIPPED kappa4 distribution (taking more than 30 seconds)
SKIPPED ksone distribution (taking more than 30 seconds)
SKIPPED levy distribution (taking more than 30 seconds)
SKIPPED levy_stable distribution (taking more than 30 seconds)
SKIPPED loggamma distribution (taking more than 30 seconds)
SKIPPED loglaplace distribution (taking more than 30 seconds)
SKIPPED lognorm distribution (taking more than 30 seconds)
SKIPPED loguniform distribution (taking more than 30 seconds)
SKIPPED lomax distribution (taking more than 30 seconds)
SKIPPED mielke distribution (taking more than 30 seconds)
SKIPPED ncf distribution (taking more than 30 seconds)
SKIPPED nct distribution (taking more than 30 seconds)
SKIPPED ncx2 distribution (taking more than 30 seconds)
SKIPPED norminvgauss distribution (taking more than 30 seconds)
SKIPPED rv_continuous distribution (taking more than 30 seconds)
SKIPPED rv_histogram distribution (taking more than 30 seconds)
SKIPPED pearson3 distribution (taking more than 30 seconds)
SKIPPED powerlognorm distribution (taking more than 30 seconds)
SKIPPED powernorm distribution (taking more than 30 seconds)
SKIPPED rdist distribution (taking more than 30 seconds)
SKIPPED recipinvgauss distribution (taking more than 30 seconds)
SKIPPED reciprocal distribution (taking more than 30 seconds)
SKIPPED rice distribution (taking more than 30 seconds)
/Users/nali/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225:
IntegrationWarning: The maximum number of subdivisions (50) has been achieved.
  If increasing the limit yields no improvement it is advised to analyze
  the integrand in order to determine the difficulties.  If the position of a
  local difficulty can be determined (singularity, discontinuity) one will
  probably gain from splitting up the interval and calling the integrator
  on the subranges.  Perhaps a special-purpose integrator should be used.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
SKIPPED skewcauchy distribution (taking more than 30 seconds)
SKIPPED skewnorm distribution (taking more than 30 seconds)
SKIPPED studentized_range distribution (taking more than 30 seconds)
SKIPPED t distribution (taking more than 30 seconds)
SKIPPED trapezoid distribution (taking more than 30 seconds)
/Users/nali/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225:
IntegrationWarning: The integral is probably divergent, or slowly convergent.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
SKIPPED trapz distribution (taking more than 30 seconds)
SKIPPED triang distribution (taking more than 30 seconds)
SKIPPED truncexpon distribution (taking more than 30 seconds)
SKIPPED truncnorm distribution (taking more than 30 seconds)
SKIPPED truncpareto distribution (taking more than 30 seconds)
SKIPPED truncweibull_min distribution (taking more than 30 seconds)
SKIPPED tukeylambda distribution (taking more than 30 seconds)
SKIPPED vonmises distribution (taking more than 30 seconds)
SKIPPED vonmises_line distribution (taking more than 30 seconds)
SKIPPED weibull_max distribution (taking more than 30 seconds)
```

```
SKIPPED weibull_min distribution (taking more than 30 seconds)
SKIPPED wrapcauchy distribution (taking more than 30 seconds)
```

Out[10]:

	sumsquare_error	aic	bic	kl_div	ks_statistic	ks_pvalue
anglit	6.541320	2.867332	15.733215	inf	0.026590	0.002954
cosine	7.608589	10.351127	23.217010	inf	0.032799	0.000099
maxwell	7.616502	25.297561	38.163445	inf	0.037406	0.000005
nakagami	7.671420	25.449703	44.748528	inf	0.035680	0.000016
exponnorm	9.192739	18.862068	38.160892	inf	0.035847	0.000014



In [11]:

```
#ldistsq
f = Fitter(data1.ldistsq)
f.fit()
f.summary()
```

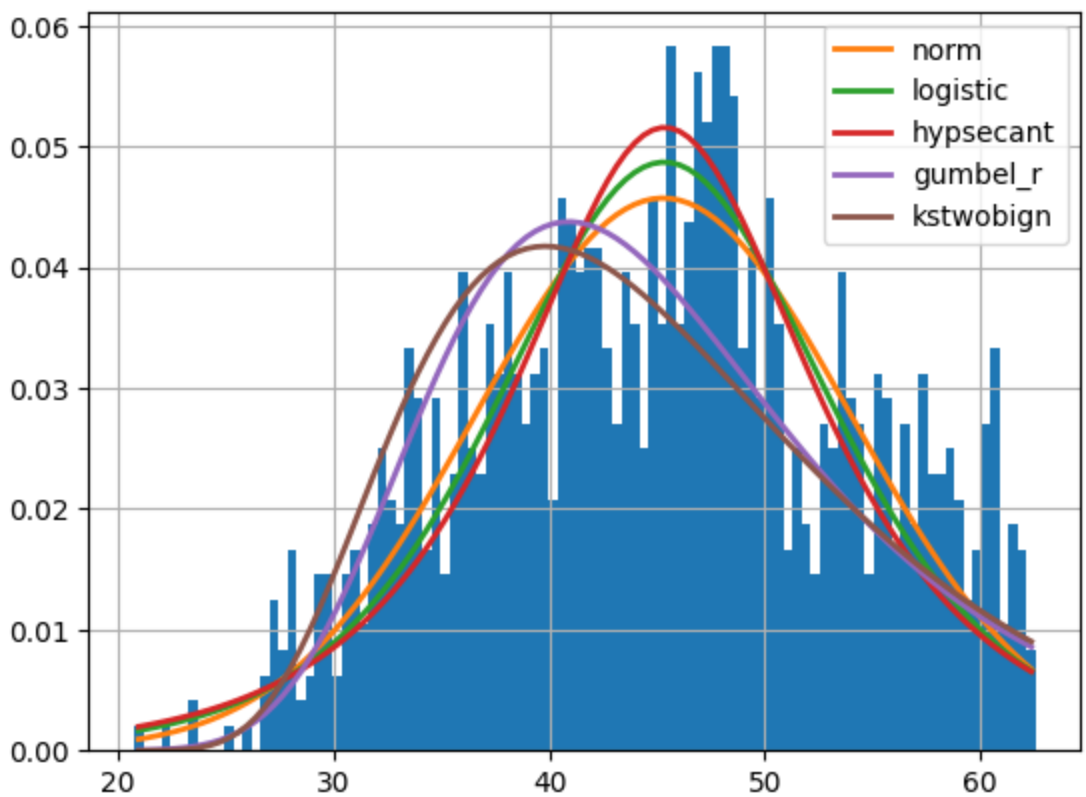
```
SKIPPED _fit distribution (taking more than 30 seconds)
SKIPPED alpha distribution (taking more than 30 seconds)
SKIPPED beta distribution (taking more than 30 seconds)
SKIPPED anglit distribution (taking more than 30 seconds)
SKIPPED betaprime distribution (taking more than 30 seconds)
SKIPPED arcsine distribution (taking more than 30 seconds)
SKIPPED burr distribution (taking more than 30 seconds)
SKIPPED argus distribution (taking more than 30 seconds)
SKIPPED bradford distribution (taking more than 30 seconds)
SKIPPED burr12 distribution (taking more than 30 seconds)
SKIPPED chi distribution (taking more than 30 seconds)
SKIPPED chi2 distribution (taking more than 30 seconds)
SKIPPED cosine distribution (taking more than 30 seconds)
SKIPPED crystalball distribution (taking more than 30 seconds)
SKIPPED dgamma distribution (taking more than 30 seconds)
SKIPPED dweibull distribution (taking more than 30 seconds)
SKIPPED erlang distribution (taking more than 30 seconds)
SKIPPED exponnorm distribution (taking more than 30 seconds)
SKIPPED exponpow distribution (taking more than 30 seconds)
SKIPPED exponweib distribution (taking more than 30 seconds)
```

```
SKIPPED f distribution (taking more than 30 seconds)
SKIPPED fatiguelife distribution (taking more than 30 seconds)
SKIPPED fisk distribution (taking more than 30 seconds)
SKIPPED foldcauchy distribution (taking more than 30 seconds)
SKIPPED foldnorm distribution (taking more than 30 seconds)
SKIPPED gamma distribution (taking more than 30 seconds)
SKIPPED gausshyper distribution (taking more than 30 seconds)
SKIPPED genexpon distribution (taking more than 30 seconds)
SKIPPED genextreme distribution (taking more than 30 seconds)
SKIPPED gengamma distribution (taking more than 30 seconds)
SKIPPED genhalflogistic distribution (taking more than 30 seconds)
SKIPPED genhyperbolic distribution (taking more than 30 seconds)
SKIPPED geninvgauss distribution (taking more than 30 seconds)
SKIPPED genlogistic distribution (taking more than 30 seconds)
SKIPPED gennorm distribution (taking more than 30 seconds)
SKIPPED genpareto distribution (taking more than 30 seconds)
SKIPPED gibrat distribution (taking more than 30 seconds)
SKIPPED gilbrat distribution (taking more than 30 seconds)
SKIPPED gompertz distribution (taking more than 30 seconds)
SKIPPED halfcauchy distribution (taking more than 30 seconds)
SKIPPED halfgennorm distribution (taking more than 30 seconds)
SKIPPED halflogistic distribution (taking more than 30 seconds)
SKIPPED halfnorm distribution (taking more than 30 seconds)
SKIPPED invgamma distribution (taking more than 30 seconds)
SKIPPED kstwo distribution (taking more than 30 seconds)
SKIPPED invgauss distribution (taking more than 30 seconds)
SKIPPED invweibull distribution (taking more than 30 seconds)
SKIPPED johnsonsb distribution (taking more than 30 seconds)
SKIPPED johnsonsu distribution (taking more than 30 seconds)
SKIPPED kappa3 distribution (taking more than 30 seconds)
SKIPPED kappa4 distribution (taking more than 30 seconds)
SKIPPED ksone distribution (taking more than 30 seconds)
SKIPPED laplace_asymmetric distribution (taking more than 30 seconds)
SKIPPED levy distribution (taking more than 30 seconds)
SKIPPED levy_l distribution (taking more than 30 seconds)
SKIPPED levy_stable distribution (taking more than 30 seconds)
SKIPPED loggamma distribution (taking more than 30 seconds)
SKIPPED loglaplace distribution (taking more than 30 seconds)
SKIPPED loguniform distribution (taking more than 30 seconds)
SKIPPED lognorm distribution (taking more than 30 seconds)
SKIPPED lomax distribution (taking more than 30 seconds)
SKIPPED maxwell distribution (taking more than 30 seconds)
SKIPPED mielke distribution (taking more than 30 seconds)
SKIPPED nakagami distribution (taking more than 30 seconds)
SKIPPED ncf distribution (taking more than 30 seconds)
SKIPPED nct distribution (taking more than 30 seconds)
SKIPPED ncx2 distribution (taking more than 30 seconds)
SKIPPED norminvgauss distribution (taking more than 30 seconds)
SKIPPED rv_continuous distribution (taking more than 30 seconds)
SKIPPED rv_histogram distribution (taking more than 30 seconds)
SKIPPED pearson3 distribution (taking more than 30 seconds)
SKIPPED powerlognorm distribution (taking more than 30 seconds)
SKIPPED powernorm distribution (taking more than 30 seconds)
SKIPPED rdist distribution (taking more than 30 seconds)
SKIPPED recipinvgauss distribution (taking more than 30 seconds)
SKIPPED reciprocal distribution (taking more than 30 seconds)
SKIPPED rice distribution (taking more than 30 seconds)
SKIPPED semicircular distribution (taking more than 30 seconds)
SKIPPED skewcauchy distribution (taking more than 30 seconds)
/Users/nali/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225:
IntegrationWarning: The maximum number of subdivisions (50) has been achieved.
  If increasing the limit yields no improvement it is advised to analyze
  the integrand in order to determine the difficulties.  If the position of a
  local difficulty can be determined (singularity, discontinuity) one will
  probably gain from splitting up the interval and calling the integrator
  on the subranges.  Perhaps a special-purpose integrator should be used.
```

```
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
SKIPPED skewnorm distribution (taking more than 30 seconds)
SKIPPED studentized_range distribution (taking more than 30 seconds)
SKIPPED t distribution (taking more than 30 seconds)
SKIPPED trapezoid distribution (taking more than 30 seconds)
SKIPPED trapz distribution (taking more than 30 seconds)
SKIPPED triang distribution (taking more than 30 seconds)
/Users/nali/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225:
IntegrationWarning: The integral is probably divergent, or slowly convergent.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
SKIPPED truncexpon distribution (taking more than 30 seconds)
SKIPPED truncnorm distribution (taking more than 30 seconds)
SKIPPED truncpareto distribution (taking more than 30 seconds)
SKIPPED truncweibull_min distribution (taking more than 30 seconds)
SKIPPED tukeylambda distribution (taking more than 30 seconds)
SKIPPED vonmises distribution (taking more than 30 seconds)
SKIPPED vonmises_line distribution (taking more than 30 seconds)
SKIPPED wald distribution (taking more than 30 seconds)
SKIPPED weibull_max distribution (taking more than 30 seconds)
SKIPPED weibull_min distribution (taking more than 30 seconds)
SKIPPED wrapcauchy distribution (taking more than 30 seconds)
```

Out[11]:

	sumsquare_error	aic	bic	kl_div	ks_statistic	ks_pvalue
norm	0.007127	829.775274	842.641157	inf	0.032744	1.024475e-04
logistic	0.007939	829.767760	842.633643	inf	0.037049	6.448147e-06
hypsecant	0.008748	831.954408	844.820291	inf	0.042937	8.438754e-08
gumbel_r	0.009618	886.893555	899.759438	inf	0.075548	2.929742e-23
kstwobign	0.010261	908.068471	920.934354	inf	0.090369	4.130194e-33



In [12]:

```python
# year
f = Fitter(data1.year)
f.fit()
f.summary()
```

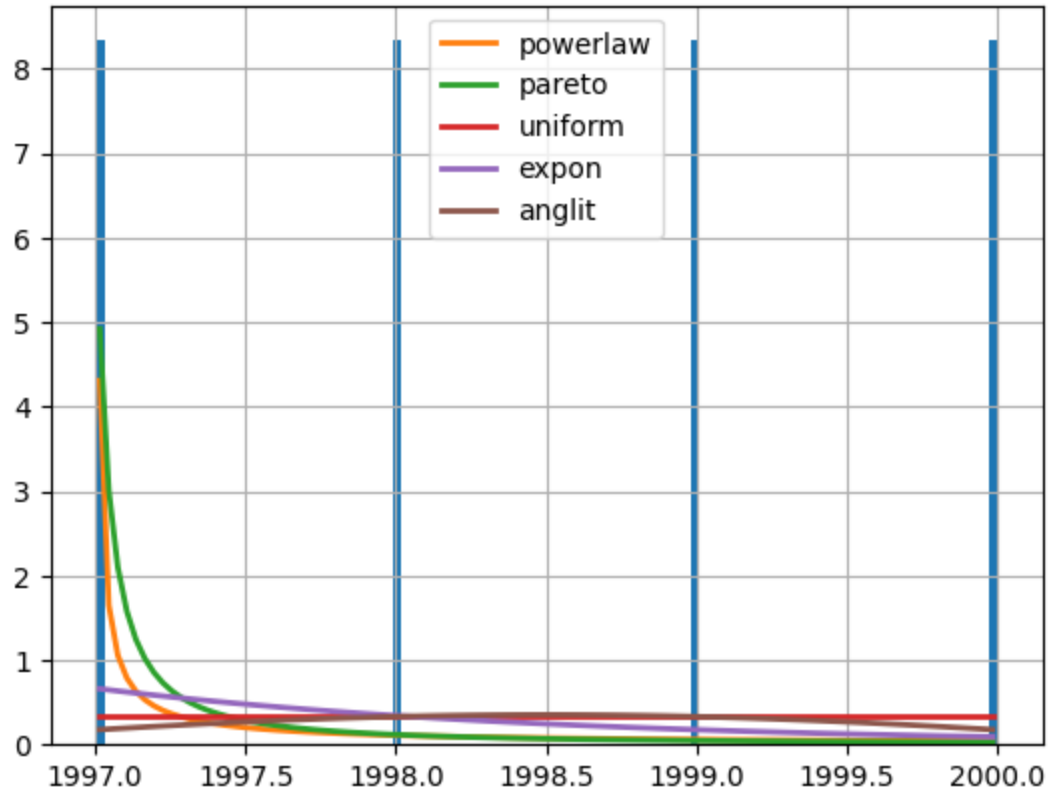
```
SKIPPED _fit distribution (taking more than 30 seconds)
```

```
SKIPPED argus distribution (taking more than 30 seconds)
SKIPPED alpha distribution (taking more than 30 seconds)
SKIPPED beta distribution (taking more than 30 seconds)
SKIPPED arcsine distribution (taking more than 30 seconds)
SKIPPED burr distribution (taking more than 30 seconds)
SKIPPED bradford distribution (taking more than 30 seconds)
SKIPPED betaprime distribution (taking more than 30 seconds)
SKIPPED burr12 distribution (taking more than 30 seconds)
SKIPPED chi distribution (taking more than 30 seconds)
SKIPPED chi2 distribution (taking more than 30 seconds)
SKIPPED cosine distribution (taking more than 30 seconds)
SKIPPED crystalball distribution (taking more than 30 seconds)
SKIPPED dweibull distribution (taking more than 30 seconds)
SKIPPED dgamma distribution (taking more than 30 seconds)
SKIPPED erlang distribution (taking more than 30 seconds)
SKIPPED exponnorm distribution (taking more than 30 seconds)
SKIPPED exponpow distribution (taking more than 30 seconds)
SKIPPED exponweib distribution (taking more than 30 seconds)
SKIPPED f distribution (taking more than 30 seconds)
SKIPPED fatiguelife distribution (taking more than 30 seconds)
SKIPPED fisk distribution (taking more than 30 seconds)
SKIPPED foldcauchy distribution (taking more than 30 seconds)
SKIPPED foldnorm distribution (taking more than 30 seconds)
SKIPPED geninvgauss distribution (taking more than 30 seconds)
SKIPPED gamma distribution (taking more than 30 seconds)
SKIPPED gausshyper distribution (taking more than 30 seconds)
SKIPPED genexpon distribution (taking more than 30 seconds)
SKIPPED genextreme distribution (taking more than 30 seconds)
SKIPPED gengamma distribution (taking more than 30 seconds)
SKIPPED genhalflogistic distribution (taking more than 30 seconds)
SKIPPED genhyperbolic distribution (taking more than 30 seconds)
SKIPPED genlogistic distribution (taking more than 30 seconds)
SKIPPED gennorm distribution (taking more than 30 seconds)
SKIPPED genpareto distribution (taking more than 30 seconds)
SKIPPED gibrat distribution (taking more than 30 seconds)
SKIPPED gilbrat distribution (taking more than 30 seconds)
SKIPPED gompertz distribution (taking more than 30 seconds)
SKIPPED halfcauchy distribution (taking more than 30 seconds)
SKIPPED halfgennorm distribution (taking more than 30 seconds)
SKIPPED halflogistic distribution (taking more than 30 seconds)
SKIPPED halfnorm distribution (taking more than 30 seconds)
SKIPPED invgamma distribution (taking more than 30 seconds)
SKIPPED invgauss distribution (taking more than 30 seconds)
SKIPPED invweibull distribution (taking more than 30 seconds)
SKIPPED kstwo distribution (taking more than 30 seconds)
SKIPPED johnsonsb distribution (taking more than 30 seconds)
SKIPPED johnsonsu distribution (taking more than 30 seconds)
SKIPPED kappa3 distribution (taking more than 30 seconds)
SKIPPED kappa4 distribution (taking more than 30 seconds)
SKIPPED ksone distribution (taking more than 30 seconds)
SKIPPED laplace_asymmetric distribution (taking more than 30 seconds)
SKIPPED levy distribution (taking more than 30 seconds)
SKIPPED levy_l distribution (taking more than 30 seconds)
SKIPPED levy_stable distribution (taking more than 30 seconds)
SKIPPED loggamma distribution (taking more than 30 seconds)
SKIPPED loglaplace distribution (taking more than 30 seconds)
SKIPPED lognorm distribution (taking more than 30 seconds)
SKIPPED lomax distribution (taking more than 30 seconds)
SKIPPED mielke distribution (taking more than 30 seconds)
SKIPPED moyal distribution (taking more than 30 seconds)
SKIPPED nakagami distribution (taking more than 30 seconds)
SKIPPED ncf distribution (taking more than 30 seconds)
SKIPPED nct distribution (taking more than 30 seconds)
SKIPPED ncx2 distribution (taking more than 30 seconds)
SKIPPED norminvgauss distribution (taking more than 30 seconds)
SKIPPED rv_continuous distribution (taking more than 30 seconds)
```

```
SKIPPED rv_histogram distribution (taking more than 30 seconds)
SKIPPED pearson3 distribution (taking more than 30 seconds)
SKIPPED powerlognorm distribution (taking more than 30 seconds)
SKIPPED powernorm distribution (taking more than 30 seconds)
SKIPPED rdist distribution (taking more than 30 seconds)
SKIPPED recipinvgauss distribution (taking more than 30 seconds)
SKIPPED rice distribution (taking more than 30 seconds)
SKIPPED semicircular distribution (taking more than 30 seconds)
SKIPPED skewcauchy distribution (taking more than 30 seconds)
SKIPPED skewnorm distribution (taking more than 30 seconds)
/Users/nali/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225:
IntegrationWarning: The maximum number of subdivisions (50) has been achieved.
  If increasing the limit yields no improvement it is advised to analyze
  the integrand in order to determine the difficulties.  If the position of a
  local difficulty can be determined (singularity, discontinuity) one will
  probably gain from splitting up the interval and calling the integrator
  on the subranges.  Perhaps a special-purpose integrator should be used.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
SKIPPED studentized_range distribution (taking more than 30 seconds)
SKIPPED t distribution (taking more than 30 seconds)
SKIPPED trapezoid distribution (taking more than 30 seconds)
SKIPPED trapz distribution (taking more than 30 seconds)
SKIPPED triang distribution (taking more than 30 seconds)
SKIPPED truncexpon distribution (taking more than 30 seconds)
SKIPPED truncnorm distribution (taking more than 30 seconds)
SKIPPED truncpareto distribution (taking more than 30 seconds)
/Users/nali/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225:
IntegrationWarning: The integral is probably divergent, or slowly convergent.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
SKIPPED truncweibull_min distribution (taking more than 30 seconds)
SKIPPED tukeylambda distribution (taking more than 30 seconds)
SKIPPED vonmises distribution (taking more than 30 seconds)
SKIPPED vonmises_line distribution (taking more than 30 seconds)
SKIPPED wald distribution (taking more than 30 seconds)
SKIPPED weibull_max distribution (taking more than 30 seconds)
SKIPPED weibull_min distribution (taking more than 30 seconds)
SKIPPED wrapcauchy distribution (taking more than 30 seconds)
```

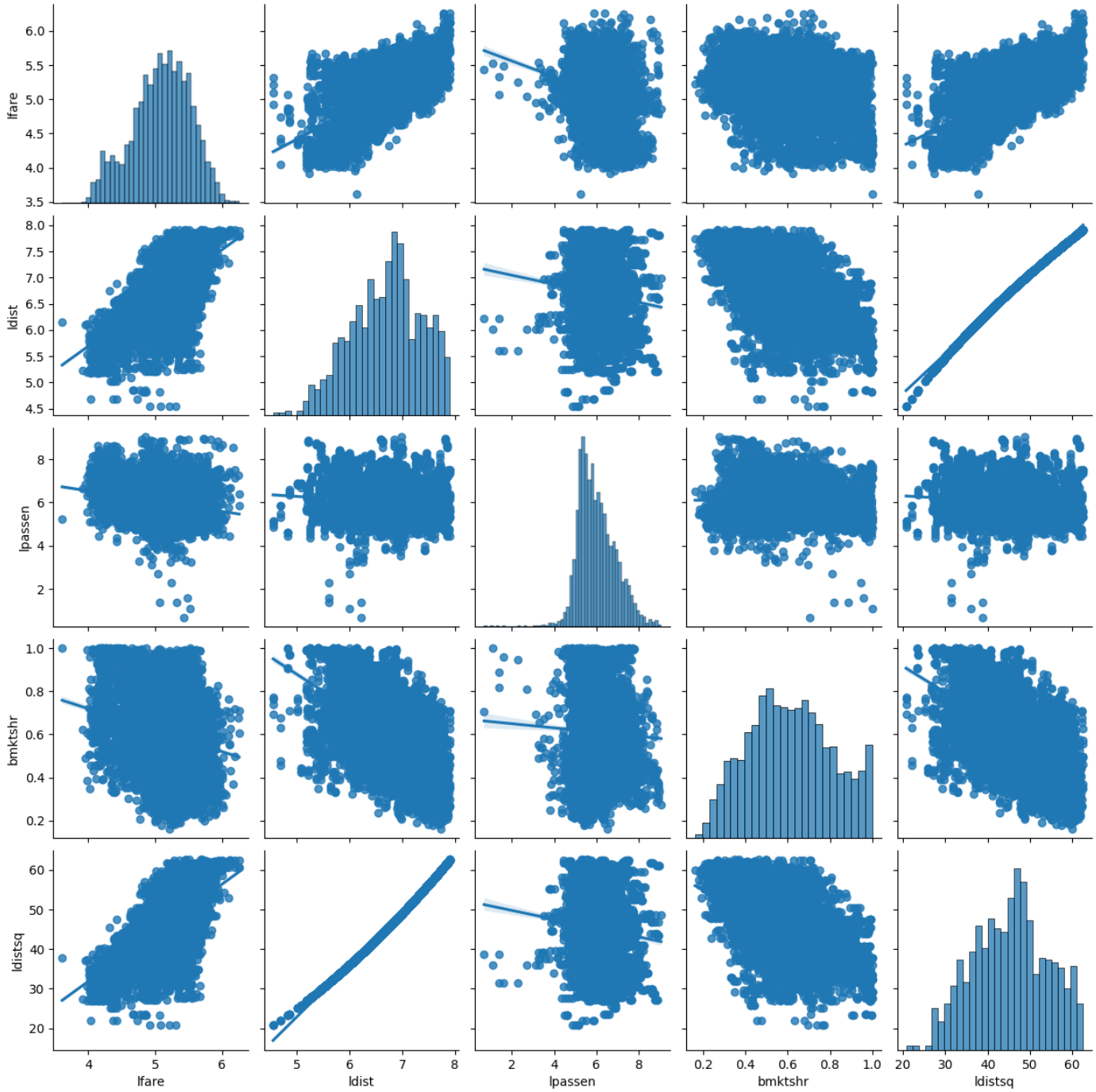
Out[12]:

	sumsquare_error	aic	bic	kl_div	ks_statistic	ks_pvalue
powerlaw	227.829124	465.650481	484.949306	inf	0.620610	0.000000e+00
pareto	239.161834	472.208403	491.507228	inf	0.422214	0.000000e+00
uniform	266.666667	223.722458	236.588341	inf	0.250000	1.417125e-253
expon	267.540976	285.093022	297.958905	inf	0.250000	1.417125e-253
anglit	269.578684	255.290069	268.155952	inf	0.185355	1.082356e-138



Scatter Plots - Pair plots

In [6]:
```
# Pair Plot
airfare_1 = data1[['lfare','ldist','lpassen','bmktshr', 'ldistsq']]
sns.pairplot(airfare_1, kind = 'reg')
plt.show()
```



From the above pair plots the graphs seem to show a linear relationship. The diagonal subplots show histograms that applear to have a normal distribution.

Correlation plot

In [7]:
```
# Correlation Plot
airfare_corr = data1[['lfare','ldist', 'lpassen', 'bmktshr', 'ldistsq','y98', 'y99', 'y0

plt.figure(figsize = (13,7))
data = airfare_corr
c = data.corr()
sns.heatmap(c,cmap = "Blues", annot = True, square = True)
plt.show()
```

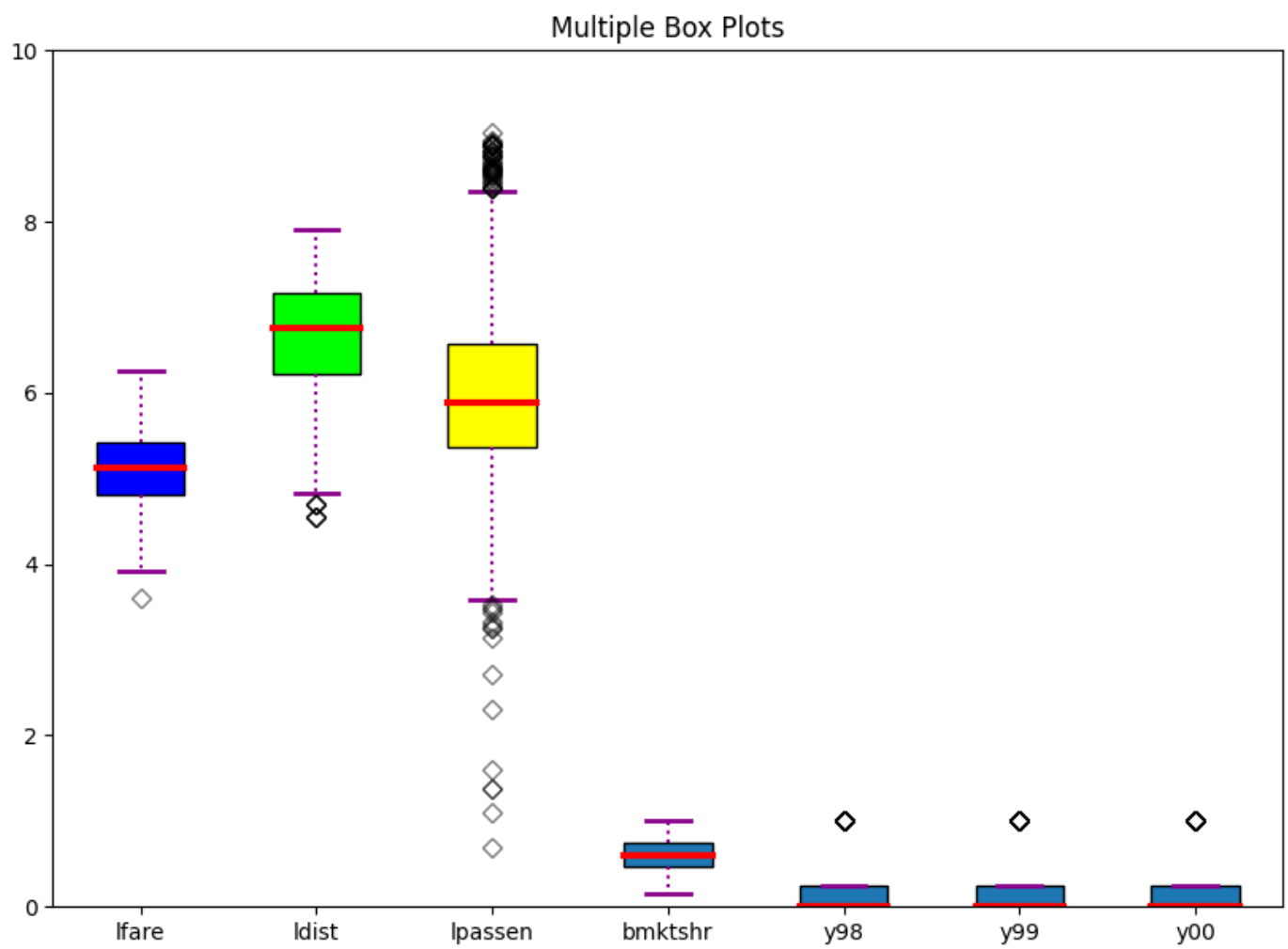
From the plot we see that the dependent variable lfare and the predictor variable ldistsq have the highest positive correlation. The predictors lfare and ldist. the predictors bmktshr and ldist have a negative correlation. Overall, we observe a lot of negative correlations compared to positive ones partly because we are using the log versions or transformed versions of the variables.

Boxplots

In [8]:

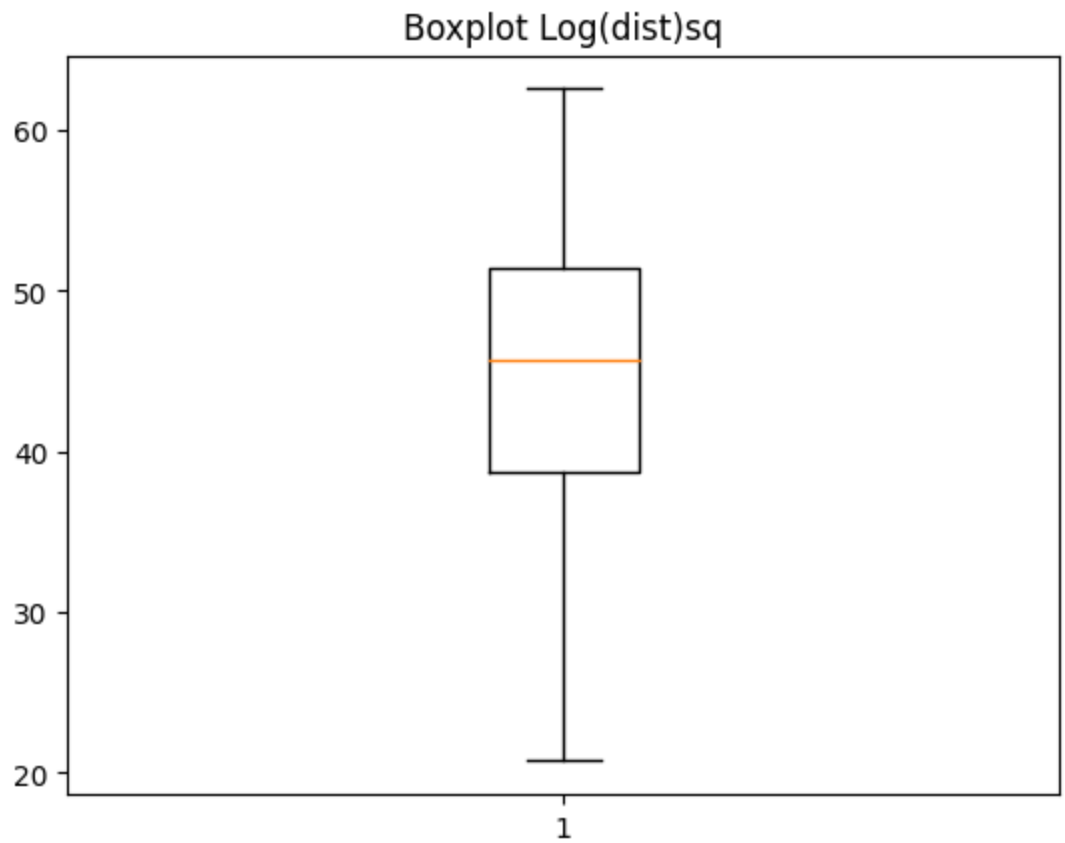
```
# Select the columns you want to create boxplots for
sel_cols = ['lfare', 'ldist', 'lpassen', 'bmktshr', 'y98', 'y99','y00']  # replace with
data_to_plot = [data1[column] for column in sel_cols]

# Create a figure instance
fig = plt.figure(figsize = (10, 7))

# Create an axes instance
ax = fig.add_subplot(111)

# Create the boxplot
bp = ax.boxplot(data_to_plot, patch_artist = True)

colors = ['#0000FF', '#00FF00', '#FFFF00']  # replace with your colors

for patch, color in zip(bp['boxes'], colors):
    patch.set_facecolor(color)

# Changing color and line width of whiskers
```

```
for whisker in bp['whiskers']:
    whisker.set(color ='#8B008B',
                linewidth = 1.5,
                linestyle =":")

# Changing color and line width of caps
for cap in bp['caps']:
    cap.set(color ='#8B008B',
            linewidth = 2)

# Changing color and line width of medians
for median in bp['medians']:
    median.set(color ='red',
               linewidth = 3)

# Changing style of fliers
for flier in bp['fliers']:
    flier.set(marker ='D',
              color ='#e7298a',
              alpha = 0.5)

# x-axis labels
ax.set_xticklabels(sel_cols)

# Adding title
plt.title("Multiple Box Plots")

# Removing top axes and right axes ticks
ax.get_xaxis().tick_bottom()
ax.get_yaxis().tick_left()
plt.ylim(0, 10)
# show plot
plt.show()
```



The variable lpassen has the most outliers. We will keep the outliers because removing them from our dataset might distort our findings.

In [9]:
```
# Boxplot for ldistsq
plt.boxplot(data1.ldistsq)
plt.title("Boxplot Log(dist)sq")
plt.show()
```



1. Fit the three models below, and identify which model is your preferred one and why. Make sure to include effects plots, statistical diagnostics, etc., to support your conclusion, and to comment on your findings.
   - Pooled
   - Fixed Effects
   - Random Effects

In [10]:
```
import wooldridge as woo

air_fare = woo.dataWoo('airfare')

# print relevant dimensions for panel:
N = air_fare.shape[0]
T = air_fare['year'].drop_duplicates().shape[0]
n = air_fare['id'].drop_duplicates().shape[0]
print(f'N: {N}\n')
print(f'T: {T}\n')
print(f'n: {n}\n')

# check non-varying variables

# (I) across time and within individuals by calculating individual
# specific variances for each variable:
isv_id = (air_fare.groupby('id').var() == 0)  # True, if variance is zero
```

```
# choose variables where all grouped variances are zero:
noVar_id = isv_id.all(axis=0)  # which cols are completely True
print(f'isv_id.columns[noVar_id]: \n{isv_id.columns[noVar_id]}\n')

# (II) across individuals within one point in time for each variable:
isv_t = (air_fare.groupby('year').var() == 0)
noVar_t = isv_t.all(axis=0)
print(f'isv_t.columns[noVar_t]: \n{isv_t.columns[noVar_t]}\n')
```

```
N: 4596

T: 4

n: 1149

isv_id.columns[noVar_id]:
Index(['dist', 'ldist', 'ldistsq'], dtype='object')

isv_t.columns[noVar_t]:
Index(['y98', 'y99', 'y00'], dtype='object')
```

The variables dist, ldist, and ldistsq do not change over time. These will not be included in the Fixed effects model.

Pooled, Fixed, and Randome Effects

possible exam questions

In [11]:

```
air_fare = woo.dataWoo('airfare')
# estimate different models:
air_fare = air_fare.set_index(['id', 'year'], drop = False)

# Pooled Model
reg_ols = plm.PooledOLS.from_formula(
    formula ='lfare ~ ldist + lpassen + bmktshr + ldistsq +'
             'C(year)', data = air_fare)
results_ols = reg_ols.fit()
# Random Effects
reg_re = plm.RandomEffects.from_formula(
    formula='lfare ~ ldist + lpassen + bmktshr + ldistsq +'
             'C(year)', data = air_fare)
results_re = reg_re.fit()

# Fixed Effects
reg_fe = plm.PanelOLS.from_formula(
    formula='lfare ~ lpassen + bmktshr +'
             'C(year) + EntityEffects', data = air_fare)
results_fe = reg_fe.fit()

# print results:
theta_hat = results_re.theta.iloc[0, 0]
print(f'theta_hat: {theta_hat}\n')

# print results: Pooled Model
table_ols = pd.DataFrame({'b': round(results_ols.params, 4),
                          'se': round(results_ols.std_errors, 4),
                          't': round(results_ols.tstats, 4),
                          'pval': round(results_ols.pvalues, 4)})
print(f'table_ols: \n{table_ols}\n')

# Print results: Random Effects
table_re = pd.DataFrame({'b': round(results_re.params, 4),
                         'se': round(results_re.std_errors, 4),
                         't': round(results_re.tstats, 4),
                         'pval': round(results_re.pvalues, 4)})
```

setting indices for panel data

fixed effects is different than pooled and random

add EntityEffects to regression for random effects fit

```
print(f'table_re: \n{table_re}\n')

# Print results: Fixed Effects
table_fe = pd.DataFrame({'b': round(results_fe.params, 4),
                         'se': round(results_fe.std_errors, 4),
                         't': round(results_fe.tstats, 4),
                         'pval': round(results_fe.pvalues, 4)})
print(f'table_fe: \n{table_fe}\n')
```

```
theta_hat: 0.8739296806509438

table_ols:
                      b      se        t  pval
ldist           -0.9722  0.1257  -7.7343   0.0
lpassen         -0.0787  0.0056 -14.0866   0.0
bmktshr          0.3098  0.0297  10.4450   0.0
ldistsq          0.1068  0.0095  11.2142   0.0
C(year)[T.1997]  7.0104  0.4158  16.8613   0.0
C(year)[T.1998]  7.0334  0.4158  16.9158   0.0
C(year)[T.1999]  7.0527  0.4158  16.9614   0.0
C(year)[T.2000]  7.1163  0.4158  17.1137   0.0

table_re:
                      b      se        t  pval
ldist           -1.1437  0.2560  -4.4670   0.0
lpassen         -0.2785  0.0064 -43.6418   0.0
bmktshr          0.1583  0.0220   7.1956   0.0
ldistsq          0.1156  0.0194   5.9764   0.0
C(year)[T.1997]  9.0465  0.8437  10.7220   0.0
C(year)[T.1998]  9.0746  0.8437  10.7551   0.0
C(year)[T.1999]  9.1001  0.8438  10.7850   0.0
C(year)[T.2000]  9.1676  0.8438  10.8649   0.0

table_fe:
                      b      se         t  pval
lpassen         -0.3696  0.0073  -50.9469   0.0
bmktshr          0.1500  0.0222    6.7547   0.0
C(year)[T.1997]  7.1735  0.0457  156.9219   0.0
C(year)[T.1998]  7.2033  0.0459  156.9231   0.0
C(year)[T.1999]  7.2326  0.0461  156.9267   0.0
C(year)[T.2000]  7.3021  0.0462  157.9874   0.0
```

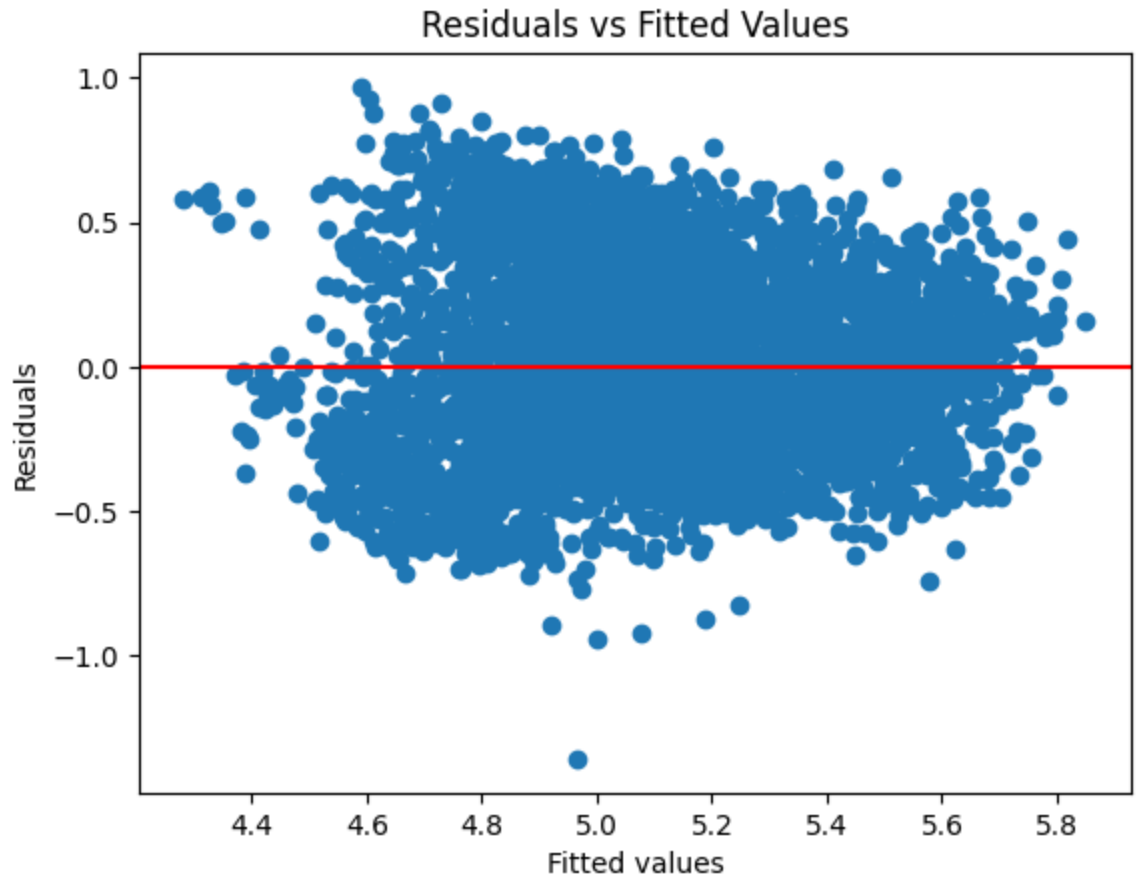
The p-values of the three models are very low. An analysis of which model is suitable will be conducted further down the line.

Effects Plots

We will start this section by looking at plots of residuals versus fitted values of the three models, Pooled, Random Effects, and Fixed Effects models.

Pooled model

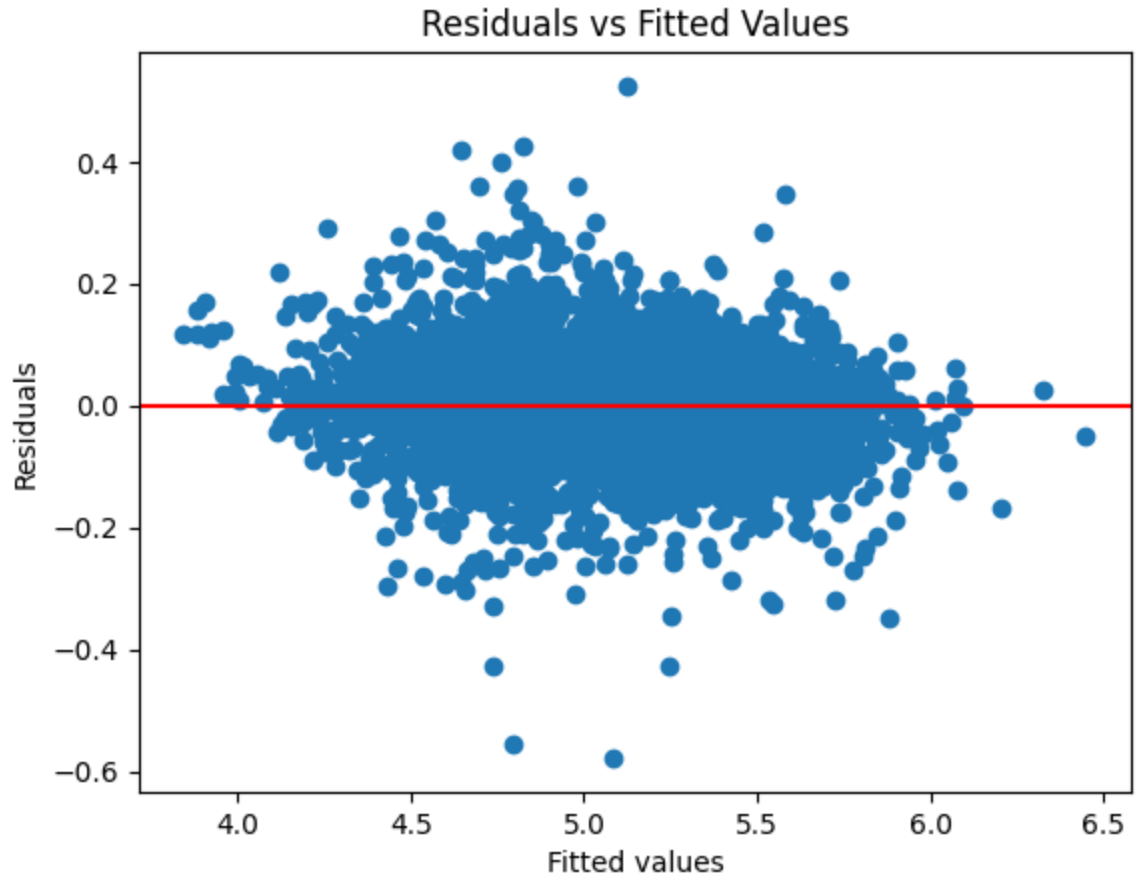
In [13]:
```
# 'results_ols' is the results object from the Pooled OLS model
fitted_values = results_ols.predict().fitted_values
residuals = results_ols.resids

plt.scatter(fitted_values, residuals)
plt.xlabel('Fitted values')
plt.axhline(color = 'red')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted Values')
plt.show()
```



This model shows a slight pattern, an indication that the model is not adequately capturing the data. Lets take a look at the other other models.

Random Effects

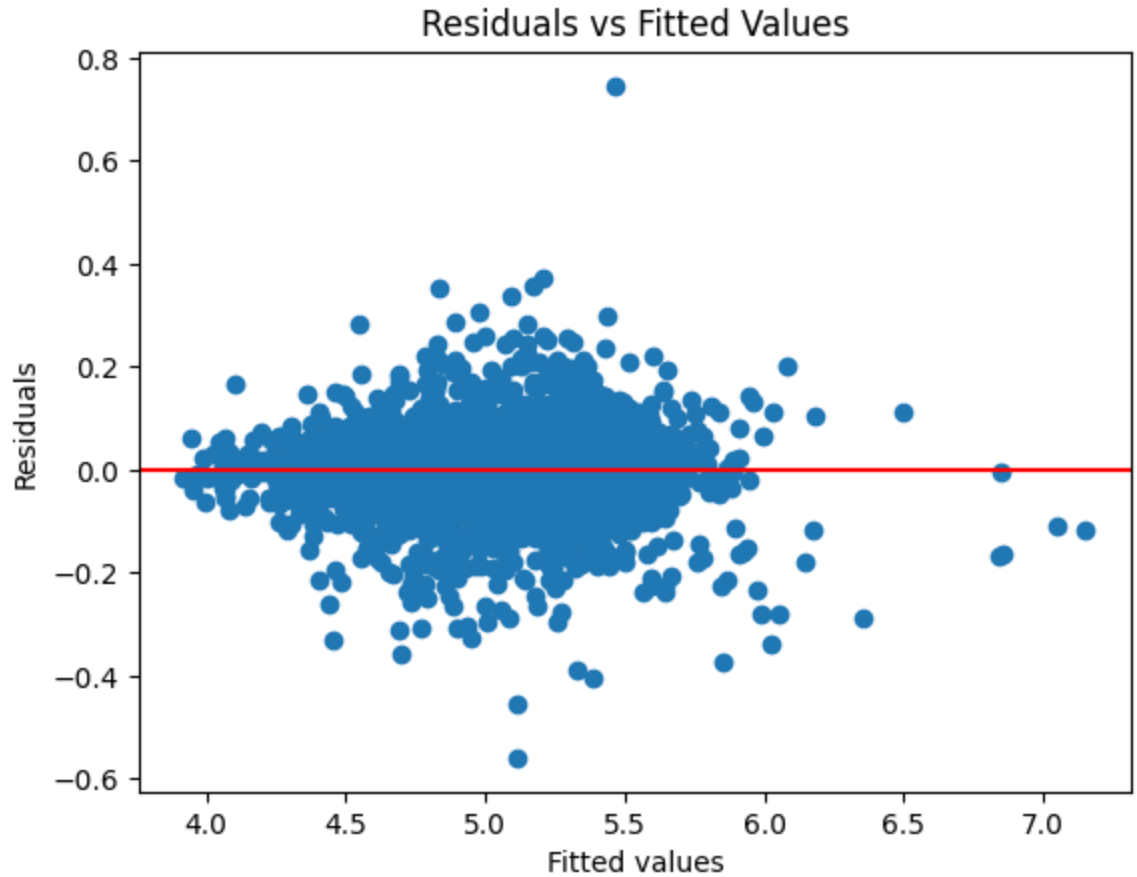
In [15]:
```
# 'results_re' is the results object from the Random Effects model
fitted_values = results_re.predict().fitted_values
residuals = results_re.resids

plt.scatter(fitted_values, residuals)
plt.xlabel('Fitted values')
plt.axhline(color = 'red')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted Values')
plt.show()
```



This Random Effects plot shows most of the datapoints centered around 0 with a few other individual data points lying not too close to zero.

Fixed Effects Model

In [14]:
```
# 'results_ee' is the results object from the Fixed Effects model
fitted_values = results_fe.predict().fitted_values
residuals = results_fe.resids

plt.scatter(fitted_values, residuals)
plt.xlabel('Fitted values')
plt.axhline(color = 'red')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted Values')
plt.show()
```



Given the results of the residuals vs the fitted values plots from the three models it is not very easy to decipher which model is the best. A Hausman Test will be able to settle the questione of which model is the best.

Hausman Test

In [15]:
```
import wooldridge as woo
import numpy as np
import linearmodels as plm
import scipy.stats as stats

air_fare = woo.dataWoo('airfare')
air_fare = air_fare.set_index(['id', 'year'], drop=False)

# estimation of FE and RE:
reg_fe = plm.PanelOLS.from_formula(formula='lfare ~ lpassen + bmktshr +'
             'C(year) + EntityEffects', data = air_fare)
results_fe = reg_fe.fit()
b_fe = results_fe.params
b_fe_cov = results_fe.cov

reg_re = plm.RandomEffects.from_formula(
    formula='lfare ~ ldist + lpassen + bmktshr + ldistsq +'
             'C(year)', data = air_fare)
results_re = reg_re.fit()
b_re = results_re.params
b_re_cov = results_re.cov

# Hausman test of FE vs. RE
# (I) find overlapping coefficients:
common_coef = set(results_fe.params.index).intersection(results_re.params.index)
common_coef = list(common_coef)
# (II) calculate differences between FE and RE:
b_diff = np.array(results_fe.params[common_coef] - results_re.params[common_coef])
```

```
df = len(b_diff)
b_diff.reshape((df, 1))
b_cov_diff = np.array(b_fe_cov.loc[common_coef, common_coef] -
                      b_re_cov.loc[common_coef, common_coef])
b_cov_diff.reshape((df, df))

# (III) calculate test statistic:
stat = abs(np.transpose(b_diff) @ np.linalg.inv(b_cov_diff) @ b_diff)
pval = 1 - stats.chi2.cdf(stat, df)

print(f'stat: {stat}\n')
print(f'pval: {pval}\n')
```

```
stat: 783.6521399346065
```

→ use fixed effects model

```
pval: 0.0
```

From the results of the Hausman Test, since Random Effects falls under HO, We reject the null hypothesis since the p-value is less than alpha 0.05 and conclude that the Fixed Effects model is the best model.

just remember that high Hausman test → use FE model

In [16]:
```
results_fe
```

Out[16]:

PanelOLS Estimation Summary

Dep. Variable:	lfare	**R-squared:**	0.5070
Estimator:	PanelOLS	**R-squared (Between):**	-0.2776
No. Observations:	4596	**R-squared (Within):**	0.5070
Date:	Wed, Dec 06 2023	**R-squared (Overall):**	-0.2371
Time:	14:52:10	**Log-likelihood**	5726.4
Cov. Estimator:	Unadjusted		
		F-statistic:	707.96
Entities:	1149	**P-value**	0.0000
Avg Obs:	4.0000	**Distribution:**	F(5,3442)
Min Obs:	4.0000		
Max Obs:	4.0000	**F-statistic (robust):**	2.864e+04
		P-value	0.0000
Time periods:	4	**Distribution:**	F(5,3442)
Avg Obs:	1149.0		
Min Obs:	1149.0		
Max Obs:	1149.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
lpassen	-0.3696	0.0073	-50.947	0.0000	-0.3838	-0.3554
bmktshr	0.1500	0.0222	6.7547	0.0000	0.1065	0.1936
C(year)[T.1997]	7.1735	0.0457	156.92	0.0000	7.0839	7.2631
C(year)[T.1998]	7.2033	0.0459	156.92	0.0000	7.1133	7.2933
C(year)[T.1999]	7.2326	0.0461	156.93	0.0000	7.1422	7.3230

C(year)[T.2000]	7.3021	0.0462	157.99	0.0000	7.2114	7.3927

F-test for Poolability: 101.22
P-value: 0.0000
Distribution: F(1148,3442)

Included effects: Entity
id: 0x15c6f7950

In conclusion, in this analysis, we chose the airfare dataset from the wooldridge package to see what happens with the log of average fare of a ticket in relation to logs of distance, log of the average number of passengers and the market share of the flier or airline. The best fitting model, after the hausman test, is the Fixed Effects model. In looking at the time invariant variables we observe a minor change in the lfare over the years (T = 4). Considering the airline/flier market share, one would expect the price of a ticket for airlines with larger market shares to be lower compared to those fliers with smaller market share. In a nutshell, a 1% increase in the predictor lpassen is associated with a -0.3696% change in the outcome variable lfare.

In [ ]:

```
Rows: 1,000
Columns: 21
$ Checking_status       <chr> "<0", "0<=X<200", "no checking", "<0", "<0", "no checking", "0<=X<200", "no checking", "0<=X<200", "no checking", "0<=X
```