タイタニック号データセット

December, 2020

概要

図表やグラフによるデータの整理について学びながら、タイタニック号乗船者の生死を分けた要因について考察してみましょう。

目次

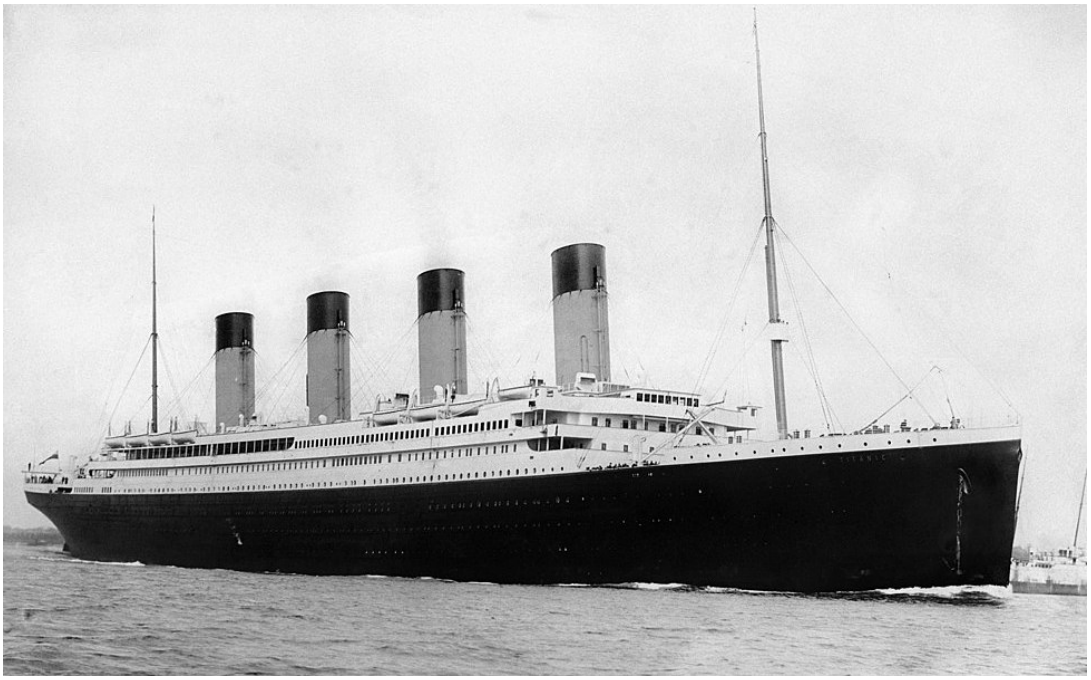
1　タイタニック号について

[1]:
```
import IPython.display
IPython.display.YouTubeVideo('CHekzSiZjrY', width=960, height=540)
```

[1]:

タイタニック (客船) について

タイタニック号沈没事故について



2　タイタニック号データセット

＊データ提供元：ヴァンダービルト大学生物統計学科 [1]、kaggle[2]、seaborn[3] など。

2.1　変数

変数名	内容	キー
survival	生存状況	0 = No, 1 = Yes
pclass	チケットクラス	1 = 一等, 2 = 二等, 3 = 3 等
sex	性別	
Age	年齢	
sibsp	同乗した兄弟や配偶者の数	
parch	同乗した親や子の数	
ticket	チッケット番号	
fare	旅客運賃	
cabin	船室番号	
embarked	乗船港	C = Cherbourg（仏）, Q = Queenstown（愛）, S = Southampton（英）

2.2　乗船港

[2]:
```
import folium

Cherbourg = [49.63, -1.62]
Queenstown = [51.851, -8.2967]
Southampton = [50.89696, -1.40416]
center = [(x+y+z)/3 for (x, y, z) in zip(Cherbourg, Queenstown, Southampton)]
# center =[45, 5]

m = folium.Map(location=center, tiles='Stamen Terrain', zoom_start=6)
folium.Marker(location=Cherbourg, popup='<b>Cherbourg</b>').add_to(m)
folium.Marker(location=Queenstown, popup='<b>Queenstown () </b>').add_to(m)
folium.Marker(location=Southampton, popup='<b> Southampton</b>').add_to(m)

m
```

[2]:
```
<folium.folium.Map at 0x7ff714800710>
```

2.3 Init

[3]:
```
import numpy as np
import pandas as pd
# pd.set_option('display.max_rows')
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style='darkgrid', font_scale = 1.8)
plt.rcParams['figure.dpi'] = 300
%matplotlib inline
```

```
Duplicate key in file PosixPath('/Users/sunggi/opt/anaconda3/lib/python3.7/site-
packages/matplotlib/mpl-data/matplotlibrc'), line 250 ('font.family:
IPAexGothic')
```

2.4 Load

[4]:
```
df = pd.read_csv("./data/train.csv")
```

[5]:
```
titanic = sns.load_dataset("titanic")
```

2.5 データの様子

[6]:
```
df.head()
```

[6]:

	PassengerId	Survived	Pclass	\
0	1	0	3	
1	2	1	1	
2	3	1	3	
3	4	1	1	

```
4            5         0       3
```

```
                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

[7]:
```
# 敬称を抽出する関数
def Title(name):
    ret = 'Other'
    target = name.split(" ")
    for i in range(len(target)):
        if "." in target[i]:
            ret = target[i]
            break
    return ret
```

[8]:
```
df['Title'] = df['Name'].apply(Title)
```

[9]:
```
df['Title'].unique()
```

[9]:
```
array(['Mr.', 'Mrs.', 'Miss.', 'Master.', 'Don.', 'Rev.', 'Dr.', 'Mme.',
       'Ms.', 'Major.', 'Lady.', 'Sir.', 'Mlle.', 'Col.', 'Capt.',
       'Countess.', 'Jonkheer.'], dtype=object)
```

[10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
```

```
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
 12  Title        891 non-null    object
dtypes: float64(2), int64(5), object(6)
memory usage: 90.6+ KB
```

[11]:
```
df.describe(include='all')
```

[11]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
count	891.000000	891.000000	891.000000	891	891	714.000000	891.000000	891.000000	891	891.000000	204	889	891
unique	NaN	NaN	NaN	891	2	NaN	NaN	NaN	681	NaN	147	3	17
top	NaN	NaN	NaN	Shelley, Mrs. William (Imanita Parrish Hall)	male	NaN	NaN	NaN	347082	NaN	C23 C25 C27	S	Mr.
freq	NaN	NaN	NaN	1	577	NaN	NaN	NaN	7	NaN	4	644	517
mean	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.523008	0.381594	NaN	32.204208	NaN	NaN	NaN
std	257.353842	0.486592	0.836071	NaN	NaN	14.526497	1.102743	0.806057	NaN	49.693429	NaN	NaN	NaN
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000	0.000000	NaN	0.000000	NaN	NaN	NaN
25%	223.500000	0.000000	2.000000	NaN	NaN	20.125000	0.000000	0.000000	NaN	7.910400	NaN	NaN	NaN
50%	446.000000	0.000000	3.000000	NaN	NaN	28.000000	0.000000	0.000000	NaN	14.454200	NaN	NaN	NaN
75%	668.500000	1.000000	3.000000	NaN	NaN	38.000000	1.000000	0.000000	NaN	31.000000	NaN	NaN	NaN
max	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000	6.000000	NaN	512.329200	NaN	NaN	NaN

[12]:
```
df.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1, inplace=True)
```

[13]:
```
df[['Survived', 'Pclass']] = df[['Survived', 'Pclass']].astype(str)
```

2.6 欠損値

[14]:
```
df.isna().sum()
```

[14]:
```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
Title         0
dtype: int64
```

[15]:
```
df[df.isna().sum(axis=1)>0].head()
```

[15]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
5	0	3	male	NaN	0	0	8.4583	Q	Mr.
17	1	2	male	NaN	0	0	13.0000	S	Mr.
19	1	3	female	NaN	0	0	7.2250	C	Mrs.
26	0	3	male	NaN	0	0	7.2250	C	Mr.
28	1	3	female	NaN	0	0	7.8792	Q	Miss.

[16]:
```
df.dropna(subset=['Embarked'], inplace=True)
```

3　単変量分析

[17]:
```
print(df.columns.values)
```

```
['Survived' 'Pclass' 'Sex' 'Age' 'SibSp' 'Parch' 'Fare' 'Embarked' 'Title']
```

[18]:
```
cat = ['Survived', 'Pclass', 'Sex', 'Embarked', 'Title']
num = ['Age', 'SibSp', 'Parch', 'Fare']
print('NOE: cat=', len(cat))
print('NOE: num=', len(num))
```

```
NOE: cat= 5
NOE: num= 4
```

3.1　質的変数

3.1.1　水準数・最頻値

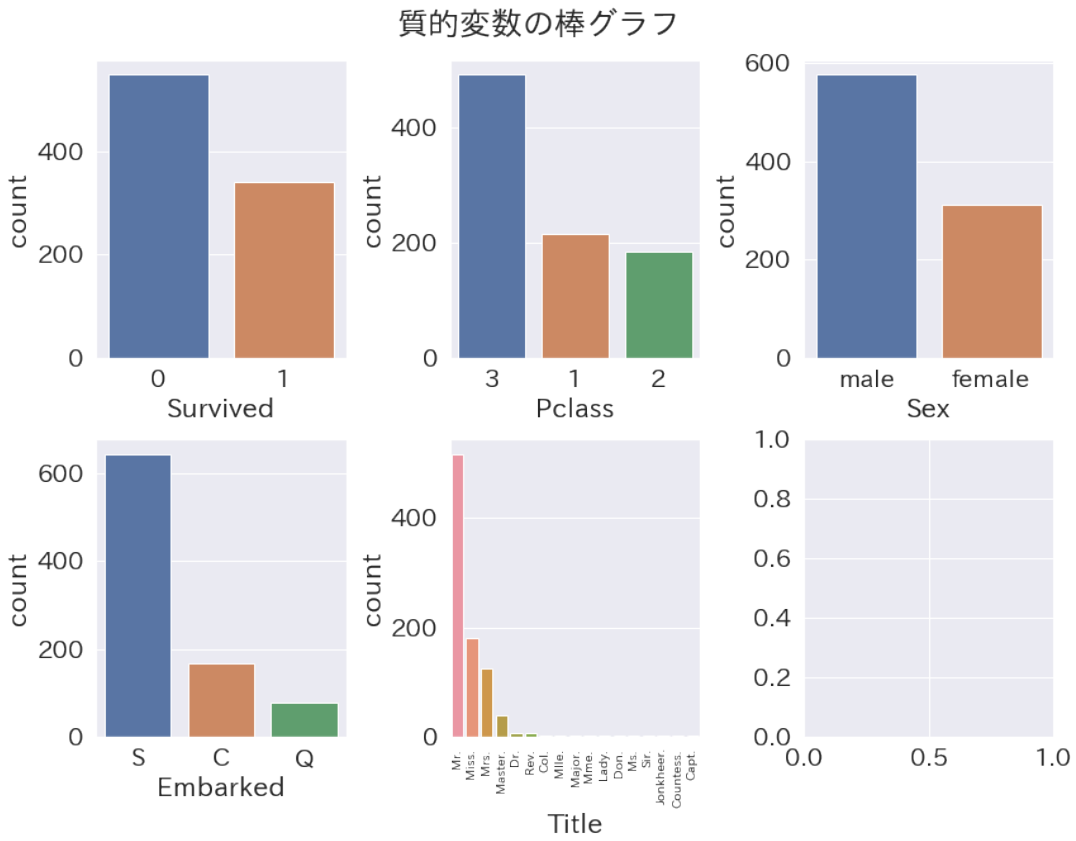
[19]:
```
df[cat].describe()
```

[19]:

	Survived	Pclass	Sex	Embarked	Title
count	889	889	889	889	889
unique	2	3	2	3	17
top	0	3	male	S	Mr.
freq	549	491	577	644	517

3.1.2　棒グラフ

[20]:
```
nor, noc = 2, 3
fig, axs = plt.subplots(nor, noc, figsize=(12, 9))
for i in range(nor):
    for j in range(noc):
        k = noc*i + j
        if k < len(cat):
            sns.countplot(x=cat[k], data=df, order=df[cat[k]].value_counts().index,␣
 ↪ax=axs[i, j])
fig.tight_layout()
fig.suptitle('質的変数の棒グラフ', fontsize=25)
fig.subplots_adjust(top=0.92)
labels = axs[1,1].get_xticklabels()
axs[1,1].set_xticklabels(labels, rotation='vertical', fontsize=10)
plt.show()
```



3.2 量的変数

3.2.1 平均・標準偏差・５数要約

[21]:
```
round(df[num].describe())
```

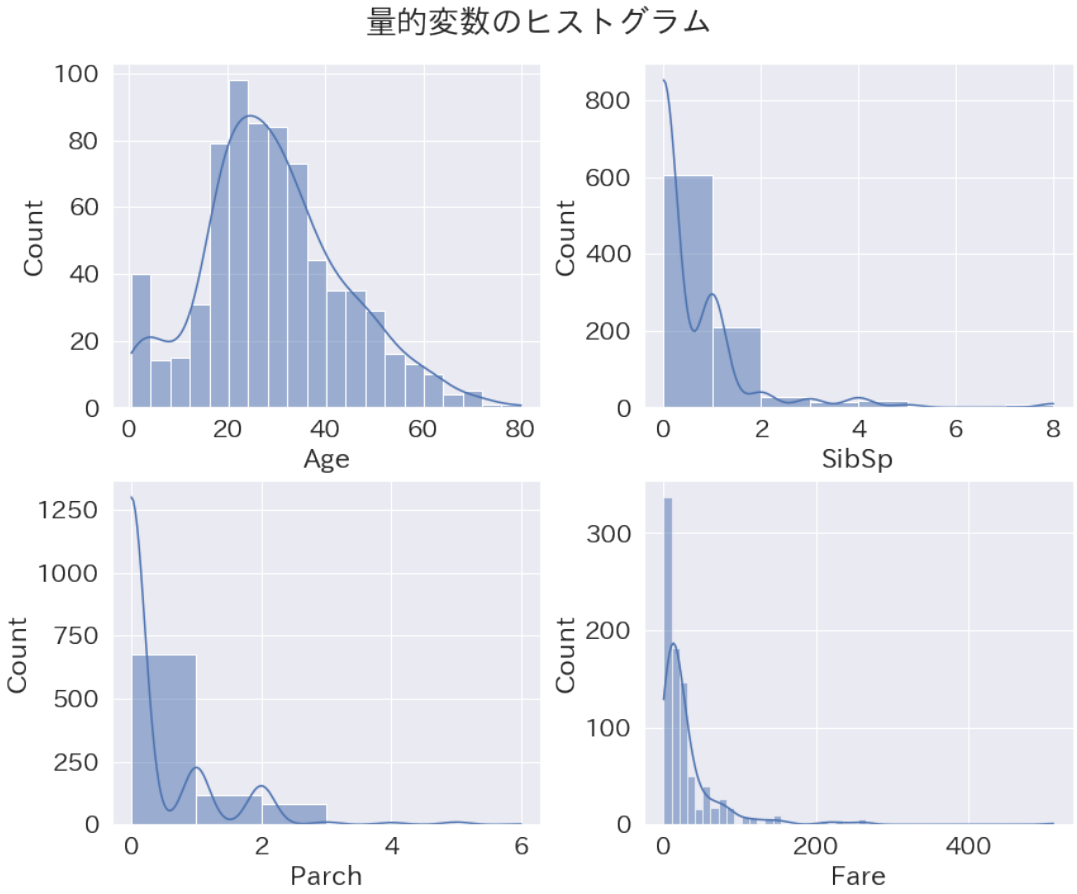
[21]:

	Age	SibSp	Parch	Fare
count	712.0	889.0	889.0	889.0
mean	30.0	1.0	0.0	32.0
std	14.0	1.0	1.0	50.0
min	0.0	0.0	0.0	0.0
25%	20.0	0.0	0.0	8.0
50%	28.0	0.0	0.0	14.0
75%	38.0	1.0	0.0	31.0
max	80.0	8.0	6.0	512.0

3.2.2 ヒストグラム

[22]:
```
nor, noc = 2, 2
fig, axs = plt.subplots(nor, noc, figsize=(12, 10))
for i in range(nor):
    for j in range(noc):
        k = noc*i + j
        if k < len(num):
```

```
            sns.histplot(data=df, x=num[k], kde=True, bins=[20,8,6,50][k], ax=axs[i, j])
fig.tight_layout()
fig.suptitle('量的変数のヒストグラム', fontsize=25)
fig.subplots_adjust(top=0.92)
plt.show()
```

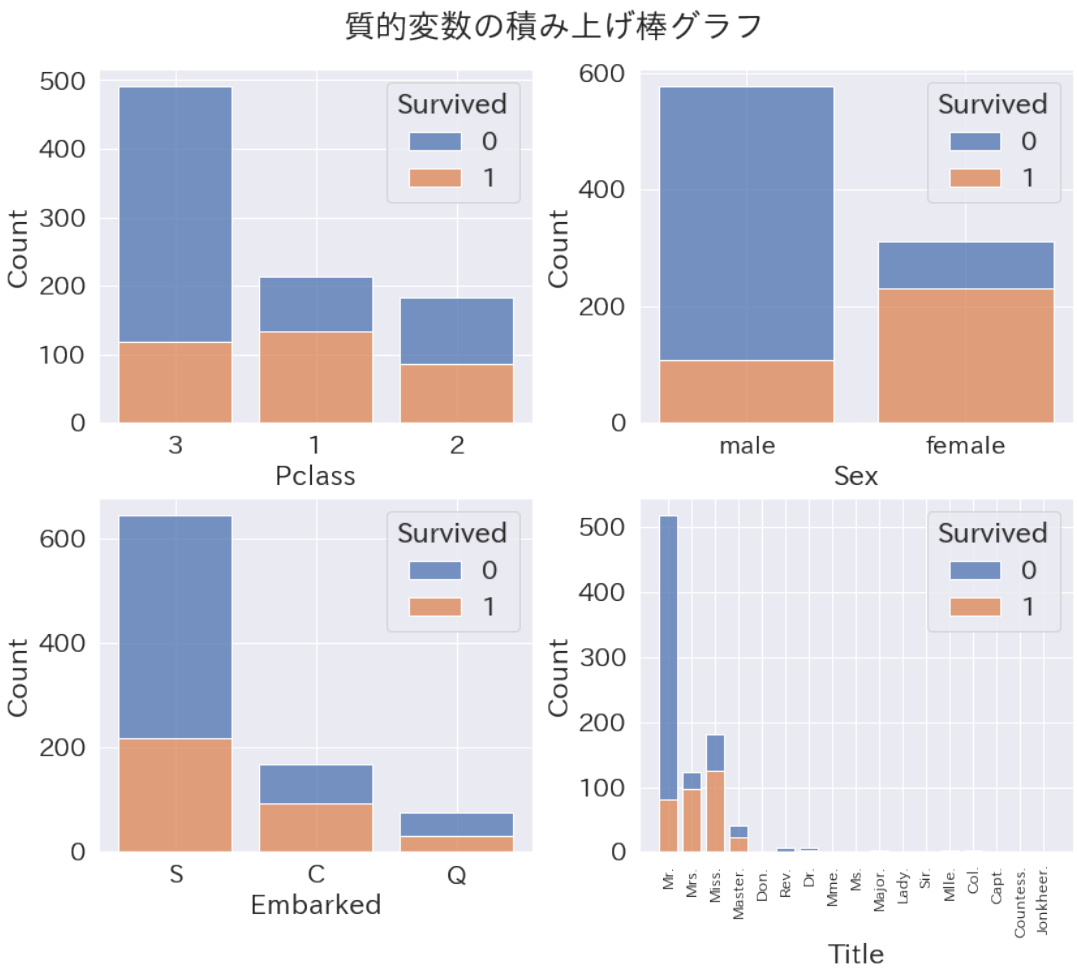


4　多変量分析

4.1　質的変数

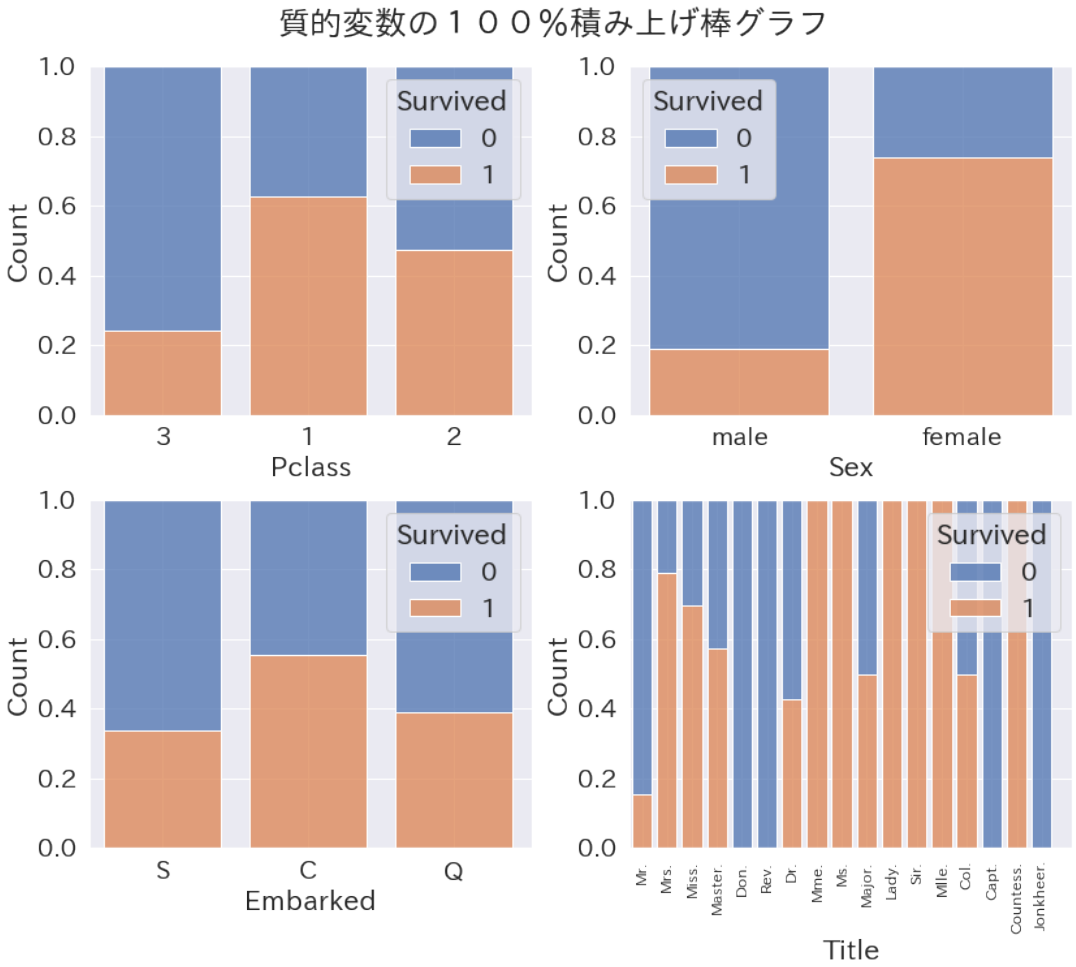
4.1.1　積み上げ棒グラフ

[23]:
```
nor, noc = 2, 2
fig, axs = plt.subplots(nor, noc, figsize=(12, 10))
for i in range(nor):
    for j in range(noc):
        k = noc*i + j
        sns.histplot(data=df, x=cat[k+1], hue='Survived', multiple='stack', shrink=.8,␣
↪ax=axs[i, j])
fig.tight_layout()
fig.suptitle('質的変数の積み上げ棒グラフ', fontsize=25)
fig.subplots_adjust(top=0.92)
labels = axs[1,1].get_xticklabels()
plt.setp(labels, rotation=90, fontsize=12)
plt.show()
```



4.1.2 100 %積み上げ棒グラフ

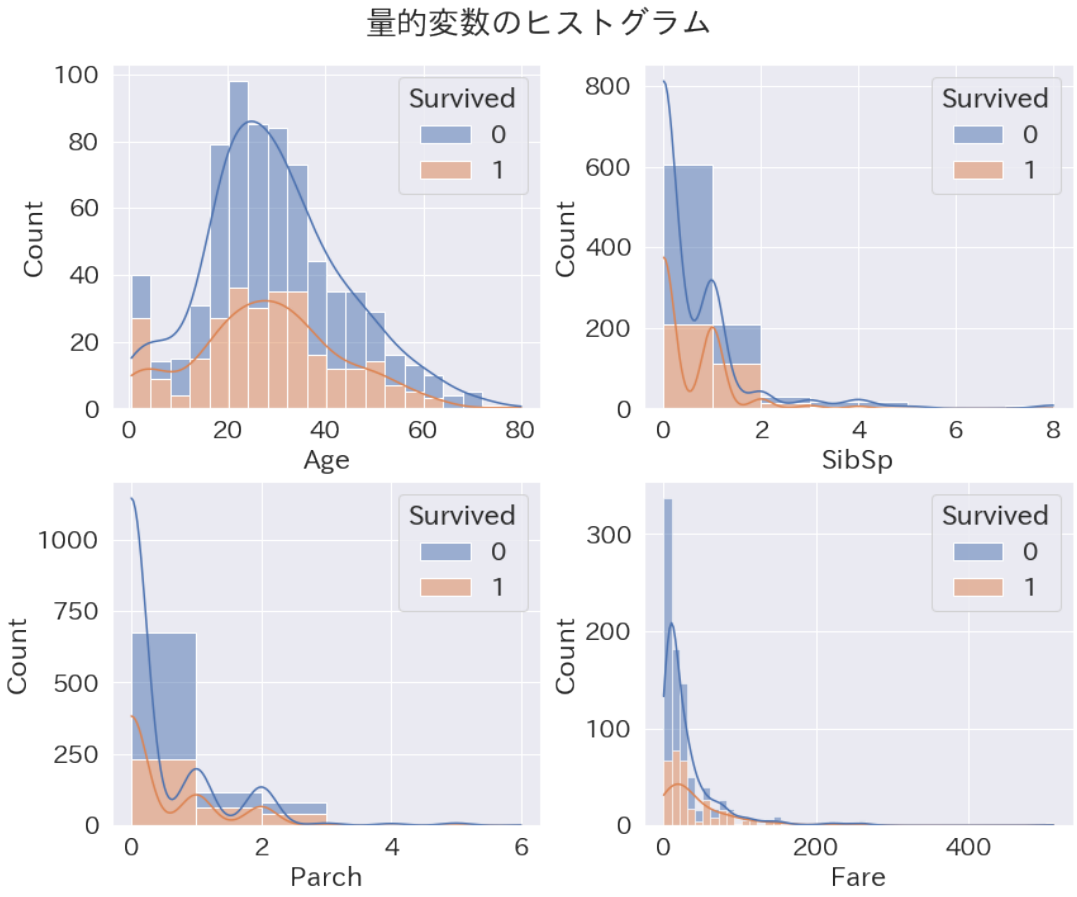
[24]:
```
nor, noc = 2, 2
fig, axs = plt.subplots(nor, noc, figsize=(12, 10))
for i in range(nor):
    for j in range(noc):
        k = noc*i + j
        sns.histplot(data=df, x=cat[k+1], hue='Survived', multiple='fill', shrink=.8,␣
↪ax=axs[i, j])
fig.tight_layout()
fig.suptitle('質的変数の１００%積み上げ棒グラフ', fontsize=25)
fig.subplots_adjust(top=0.92)
labels = axs[1,1].get_xticklabels()
plt.setp(labels, rotation=90, fontsize=12)
plt.show()
```



4.2 量的変数

4.2.1 ヒストグラム

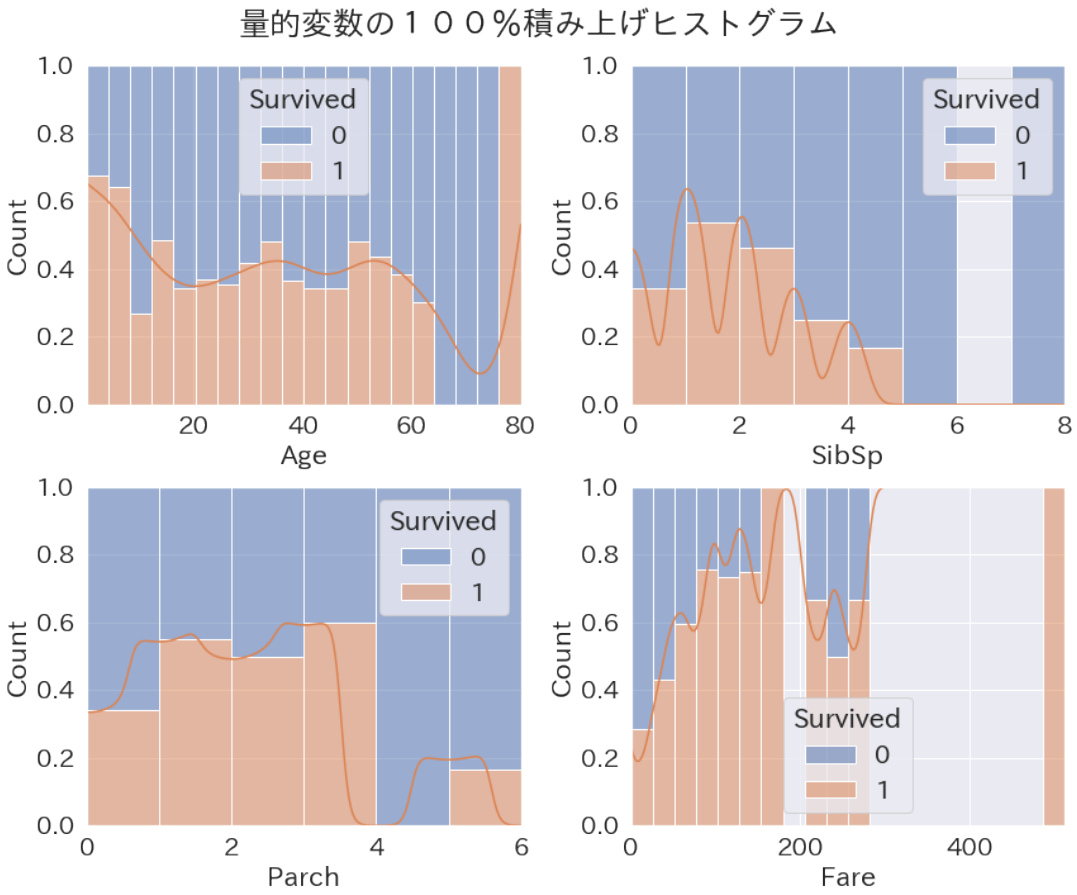
[25]:
```
nor, noc = 2, 2
fig, axs = plt.subplots(nor, noc, figsize=(12, 10))
for i in range(nor):
    for j in range(noc):
        k = noc*i + j
        sns.histplot(data=df, x=num[k], hue='Survived', multiple='stack', kde=True,␣
↪bins=[20,8,6,50][k], ax=axs[i, j])
fig.tight_layout()
fig.suptitle('量的変数のヒストグラム', fontsize=25)
fig.subplots_adjust(top=0.92)
plt.show()
```



4.2.2 100 %積み上げヒストグラム

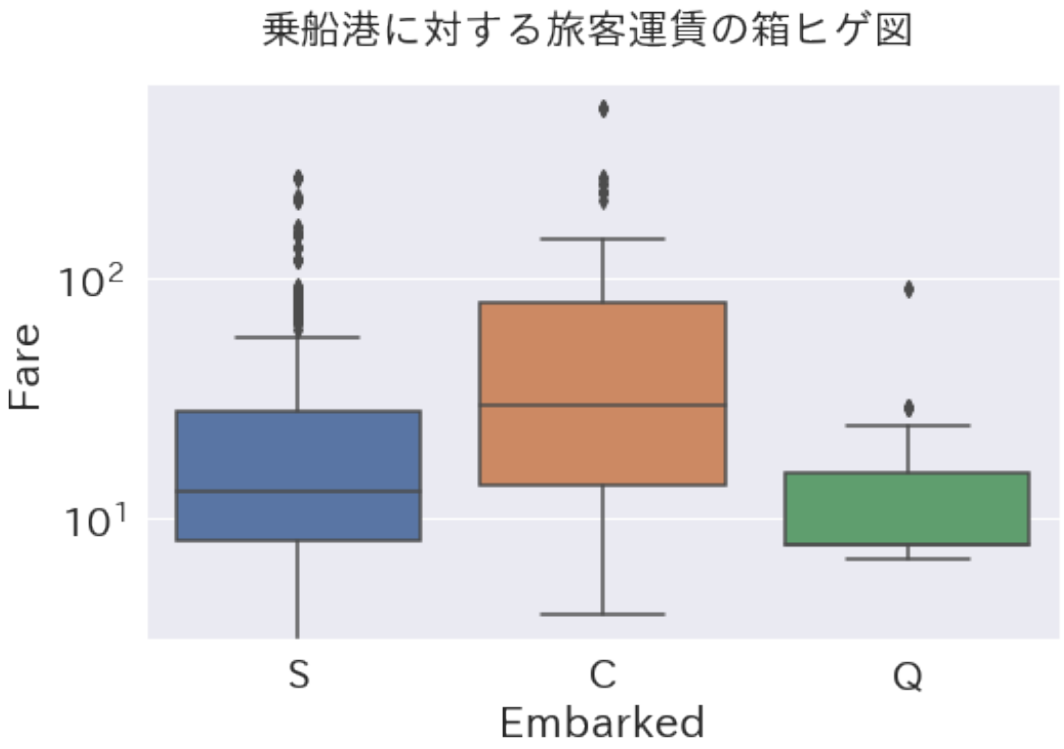
[26]:
```
nor, noc = 2, 2
fig, axs = plt.subplots(nor, noc, figsize=(12, 10))
for i in range(nor):
    for j in range(noc):
```

```
        k = noc*i + j
        sns.histplot(data=df, x=num[k], hue='Survived', multiple='fill',␣
↪bins=[20,8,6,20][k], kde=True,ax=axs[i, j])
fig.tight_layout()
fig.suptitle('量的変数の１００%積み上げヒストグラム', fontsize=25)
fig.subplots_adjust(top=0.92)
plt.show()
```

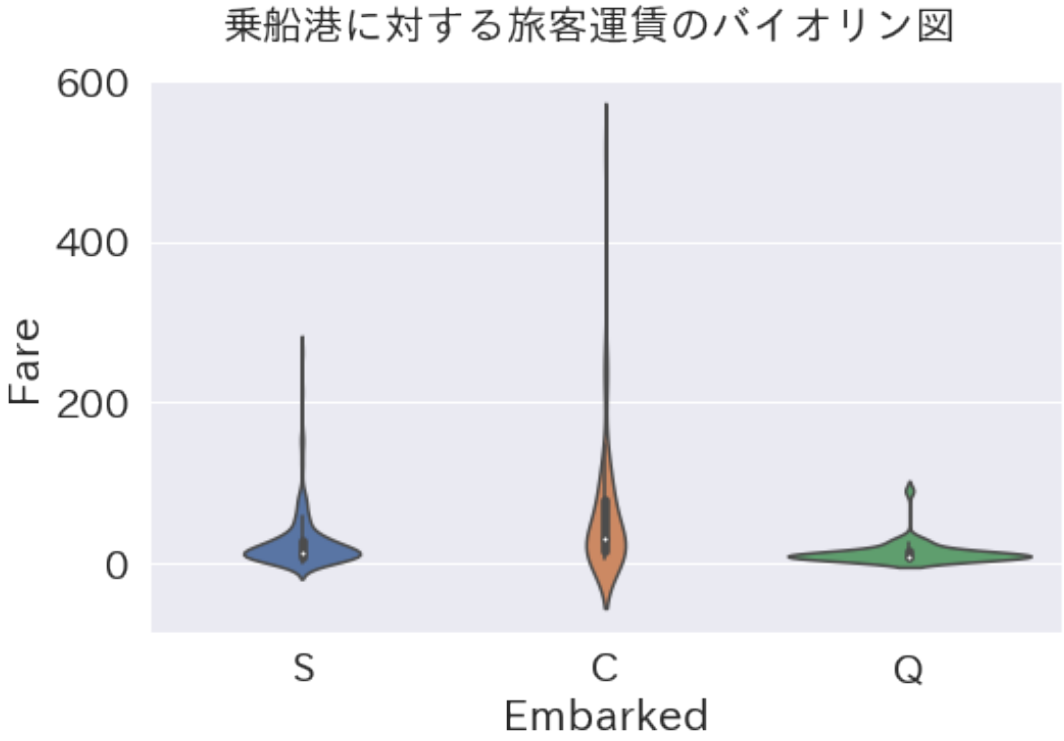


4.2.3 箱ヒゲ図とバイオリン図

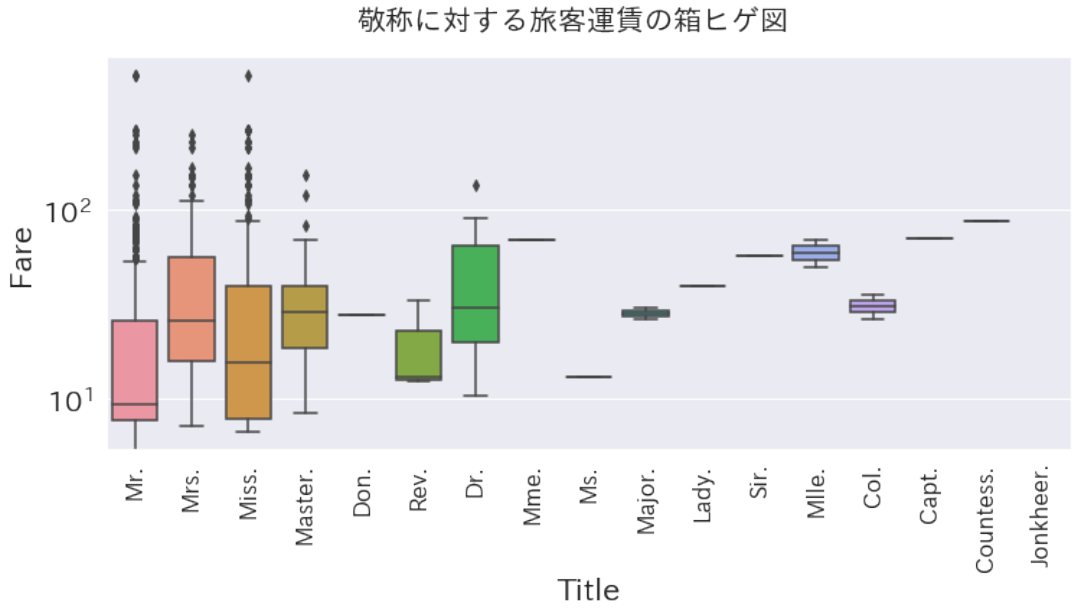
[34]:
```
fig, ax = plt.subplots(figsize=(8, 5))
sns.boxplot(x='Embarked', y='Fare', data=df, ax=ax)
ax.set_yscale("log")
plt.suptitle('乗船港に対する旅客運賃の箱ヒゲ図', fontsize=20)
plt.show()
```



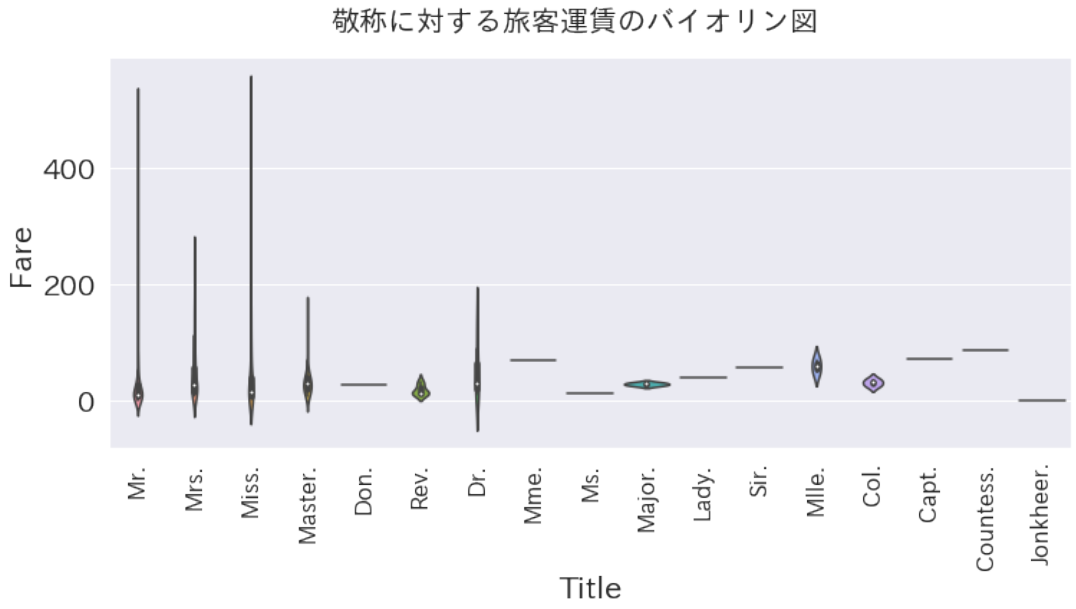
[35]:
```
fig, ax = plt.subplots(figsize=(8, 5))
sns.violinplot(x='Embarked', y='Fare', data=df, ax=ax)
plt.suptitle(' 乗船港に対する旅客運賃のバイオリン図', fontsize=20)
plt.show()
```



[36]:
```
fig, ax = plt.subplots(figsize=(12, 5))
sns.boxplot(x='Title', y='Fare', data=df, ax=ax)
labels = ax.get_xticklabels()
ax.set_xticklabels(labels, rotation='vertical', fontsize=16)
ax.set_yscale("log")
plt.suptitle('敬称に対する旅客運賃の箱ヒゲ図', fontsize=20)
plt.show()
```

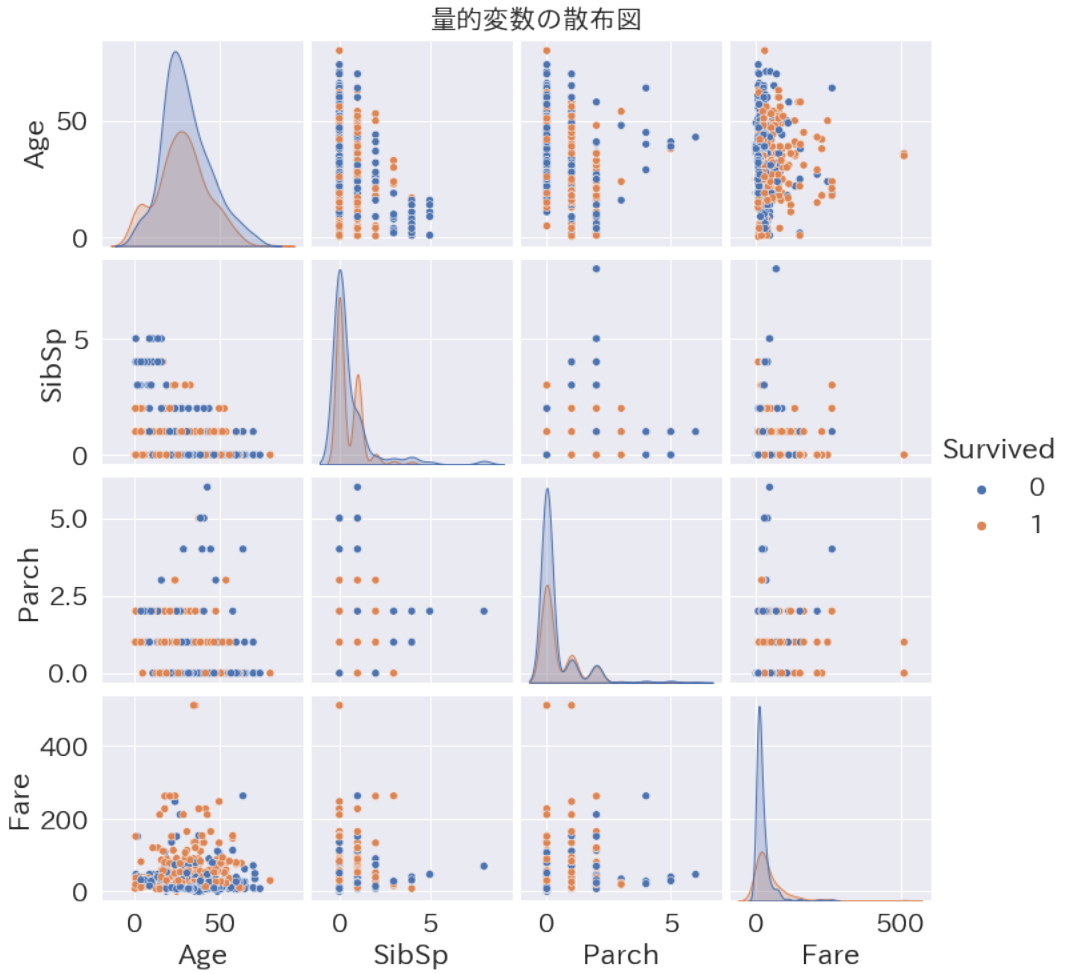


[37]:
```
fig, ax = plt.subplots(figsize=(12, 5))
sns.violinplot(x='Title', y='Fare', data=df, ax=ax)
labels = ax.get_xticklabels()
ax.set_xticklabels(labels, rotation='vertical', fontsize=16)
#ax.set_yscale("log")
plt.suptitle('敬称に対する旅客運賃のバイオリン図', fontsize=20)
plt.show()
```



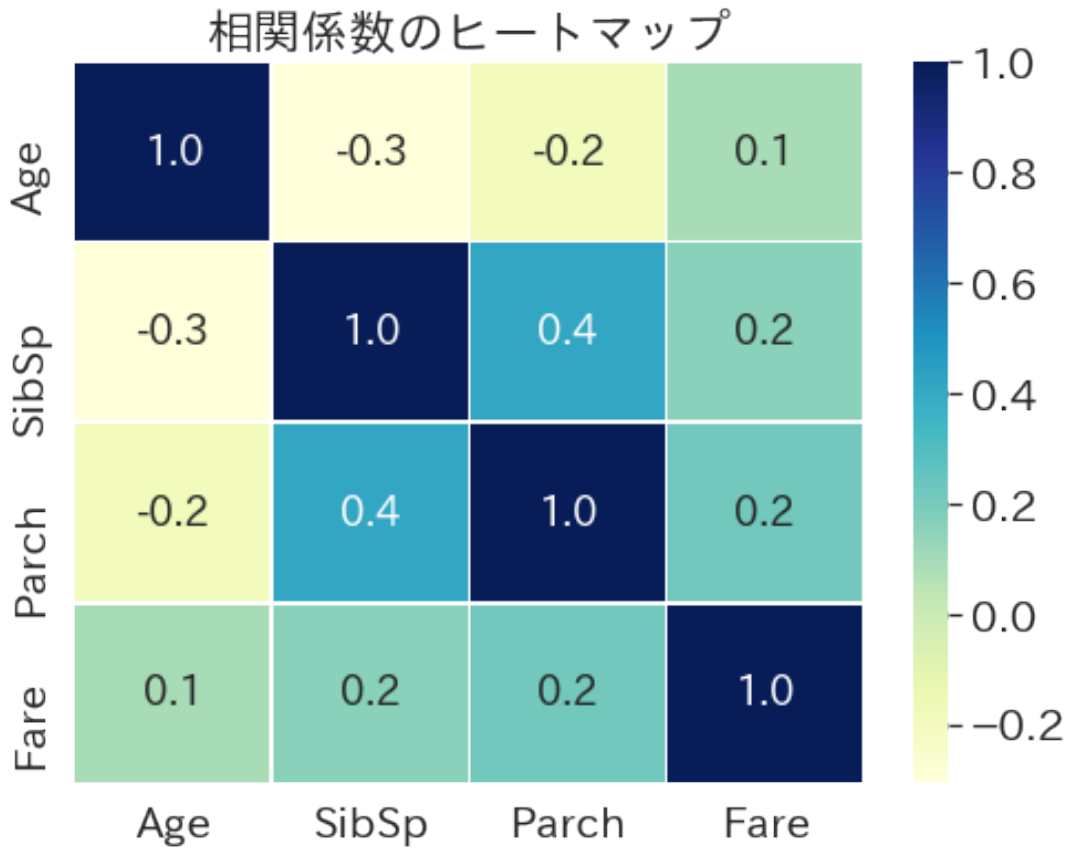
4.2.4 散布図

[31]:
```
sns.pairplot(df, hue="Survived", height=2.5)
plt.suptitle(' 量的変数の散布図', fontsize=20, y=1.02)
plt.show()
```



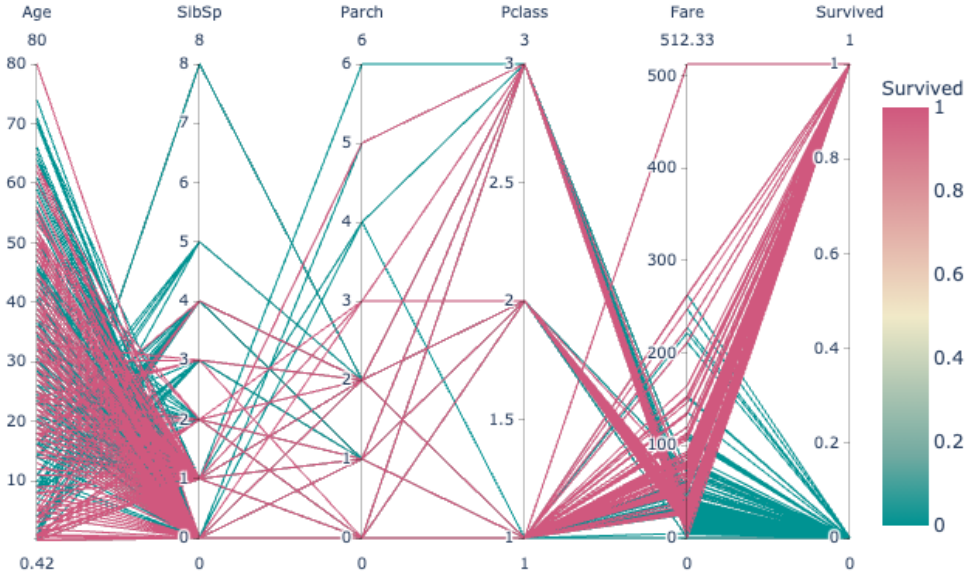
4.2.5 相関係数のヒートマップ

[32]:
```
corr = df.corr()
plt.subplots(figsize=(8, 6))
sns.heatmap(corr, linewidth=.5, annot=True, annot_kws={"size": 18}, cmap='YlGnBu', fmt='.
↪1f')
plt.title('相関係数のヒートマップ')
plt.show()
```



4.2.6 並行座標プロット

[33]:
```
df[['Survived', 'Pclass']] = df[['Survived', 'Pclass']].astype(int)
import plotly.express as px
fig = px.parallel_coordinates(df, color="Survived",
                              dimensions=['Age', 'SibSp', 'Parch', 'Pclass', 'Fare',␣
↪'Survived'],
                              color_continuous_scale=px.colors.diverging.Tealrose,
                              color_continuous_midpoint=0.5)
fig.show()
```


Age
SibSp
Parch
Pclass
Fare
Survived
80
8
6
3
512.33
1
0.42
0
0
1
0
0
Survived

参考文献

[1] Vanderbilt Biostatistics Datasets
https://hbiostat.org/data/

[2] kaggle: Titanic - Machine Learning from Disaster
https://www.kaggle.com/c/titanic

[3] seaborn
https://seaborn.pydata.org/index.html