Homework 2: Exploring & Visualizing Data

Make sure you have seaborn and missingno installed. Run `pip3 install seaborn` and `pip3 install missingno` in your container/shell if you don't.

Setup

In this homework, we will more rigorously explore data visualization and data manipulation with a couple datasets. Please fill in the cells with `## YOUR CODE HERE` following the appropriate directions.

In [1]:
```
# removes the need to call plt.show() every time
%matplotlib inline
```

Seaborn is a powerful data visualization library built on top of matplotlib. We will be using seaborn for this homework (since it is a better tool and you should know it well). Plus seaborn comes default with *much* better aesthetics (invoked with the `set()` function call).

In [2]:
```
import missingno as msno
import seaborn as sns
sns.set()
```

Import `numpy` and `pandas` (remember to abbreviate them accordingly!)

In [3]:
```
import numpy as np
import pandas as pd
```

Getting to know a new dataset

First load the `titanic` dataset directly from seaborn. The `load_dataset` function will return a pandas dataframe.

In [4]:
```
titanic = sns.load_dataset('titanic')
```

Now use some pandas functions to get a quick overview/statistics on the dataset. Take a quick glance at the overview you create.

In [5]:
```
titanic.head()
```

Out[5]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	de
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Na
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Na
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Na

With your created overview, you should be able to answer these questions:

In [6]:
```
titanic.sort_values(by=['age'],ascending=False)
len(titanic["survived"][titanic["survived"]==1])/len(titanic["survived"])
np.mean(titanic["fare"])
```

Out[6]:
```
32.2042079685746
```

- What was the age of the oldest person on board? 80.0
- What was the survival rate of people on board? 0.3838383838383838
- What was the average fare of people on board? 32.2042079685746

By the way, for getting overviews, pandas also has a `groupby` function that is quite nice to use. example:

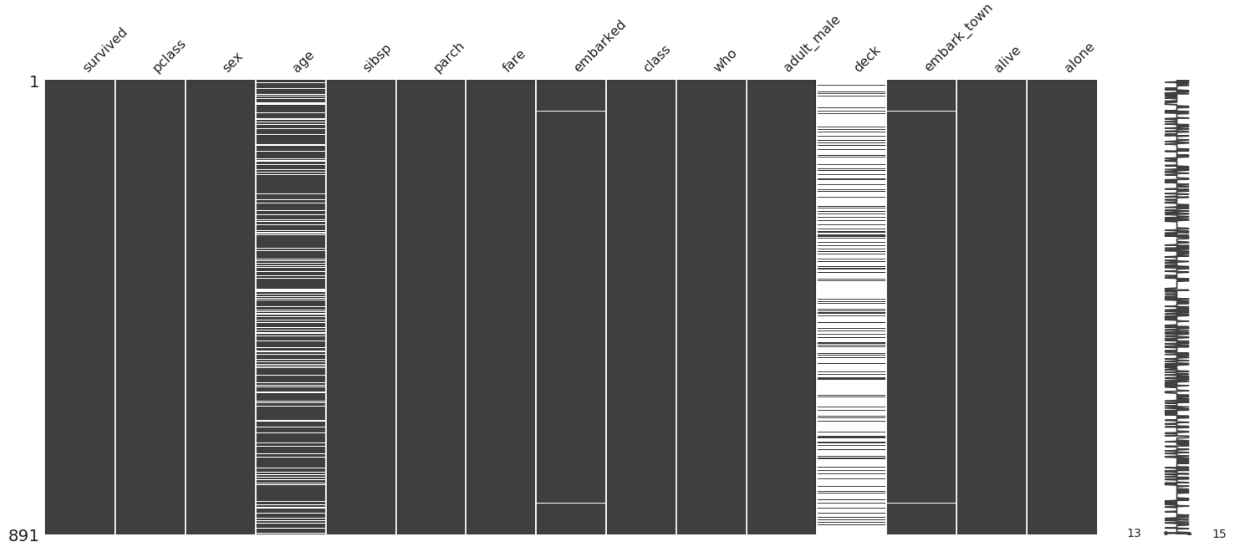
In [7]:
```
titanic.groupby(['sex','embark_town'])['survived'].mean()
```

Out[7]:
```
sex     embark_town
female  Cherbourg      0.876712
        Queenstown     0.750000
        Southampton    0.689655
male    Cherbourg      0.305263
        Queenstown     0.073171
        Southampton    0.174603
Name: survived, dtype: float64
```

Now we have an overview of our dataset. The next thing we should do is clean it - check for missing values and deal with them appropriately.

`missingno` allows us to really easily see where missing values are in our dataset. It's a simple command:

In [8]:
```
msno.matrix(titanic)
```



The white lines show us the missing data. One quick observation is the `deck` has a lot of missing data. Let's just go ahead and drop that column from the dataset since it's not relevant.

In [9]:
```
titanic.drop('deck', axis=1, inplace=True)
```

Now let's rerun the matrix and see. All that white is gone! Nice.

We still have a bunch of missing values for the age field. We can't just drop the age column since it is a pretty important datapoint. One way to deal with this is simply to just remove the records with missing information with `dropna()`, but this would end up removing out a significant amount of our data.

What do we do now? We can now explore a technique called `missing value imputation`. What this means is basically we find a reasonable way to *replace* the unknown data with workable values.

There's a lot of theory regarding how to do this properly, (for the curious look here (http://www.stat.columbia.edu/~gelman/arm/missing.pdf)). We can simply put in the average age value for the missing ages. But this really isn't so great, and would skew our stats.

If we assume that the data is missing *at random* (which actually is rarely the case and very hard to prove), we can just fit a model to predict the missing value based on the other available factors. One popular way to do this is to use KNN (where you look at the nearest datapoints to a certain point to conclude the missing value), but we can also use deep neural networks to achieve this task.

You must now make you own decision on how to deal with the missing data. You may choose any of the methods discussed above. Easiest would be to fill in with average value (but this will skew our visualizations) (if you use pandas correctly, you can do this in one line - try looking at pandas documentation!). After writing your code, verify the result by rerunning the matrix.

In [10]:
```
mn = np.mean(titanic["age"])
print(mn)
titanic[["age"]]= titanic[["age"]].fillna(value=mn)
len(titanic["age"][titanic["age"].isnull()])
```

```
29.69911764705882
```

Out[10]:
```
0
```

Intro to Seaborn

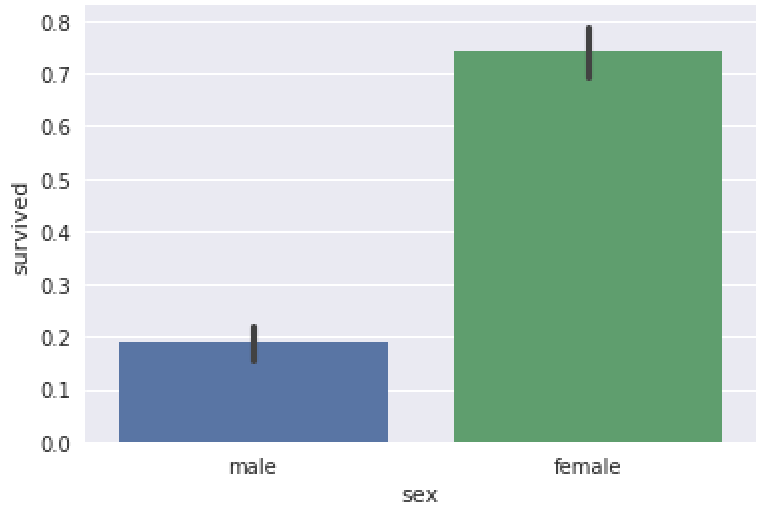
There are 2 types of data in any dataset: categorial and numerical data. We will first explore categorical data.

One really easy way to show categorical data is through bar plots. Let's explore how to make some in seaborn. We want to investigate the difference in rates at which males vs females survived the accident. Using the documentation here (https://seaborn.pydata.org/generated/seaborn.barplot.html) and example here (http://seaborn.pydata.org/examples/color_palettes.html), create a `barplot` to depict this. It should be a really simple one-liner.

We will show you how to do this so you can get an idea of how to use the API.

In [11]:
```
sns.barplot(x='sex', y='survived', data=titanic)
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f65a0029198>
```

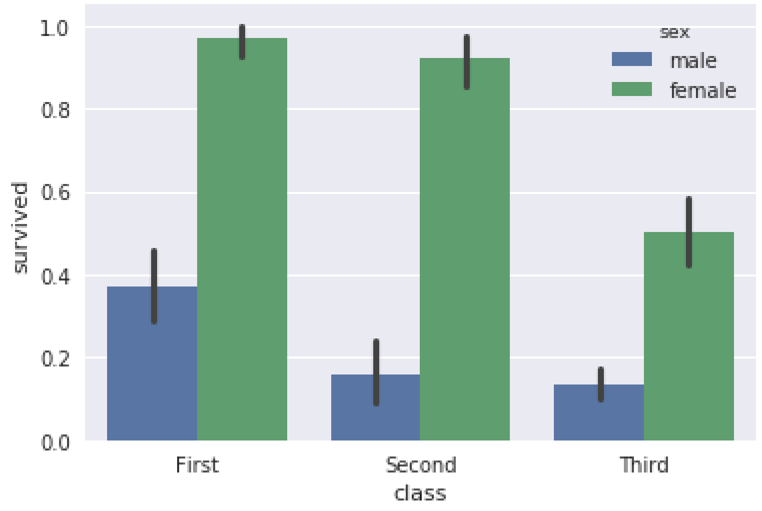


Notice how it was so easy to create the plot! You simply passed in the entire dataset, and just specified the x and y fields that you wanted exposed for the barplot. Behind the scenes seaborn ignored `NaN` values for you and automatically calculated the survival rate to plot. Also, that black tick is a 95% confidence interval that seaborn plots.

So we see that females were much more likely to make it out alive. What other factors do you think could have an impact on surival rate? Plot a couple more barplots below. Make sure to use *categorical* values, not something numerical like age or fare.

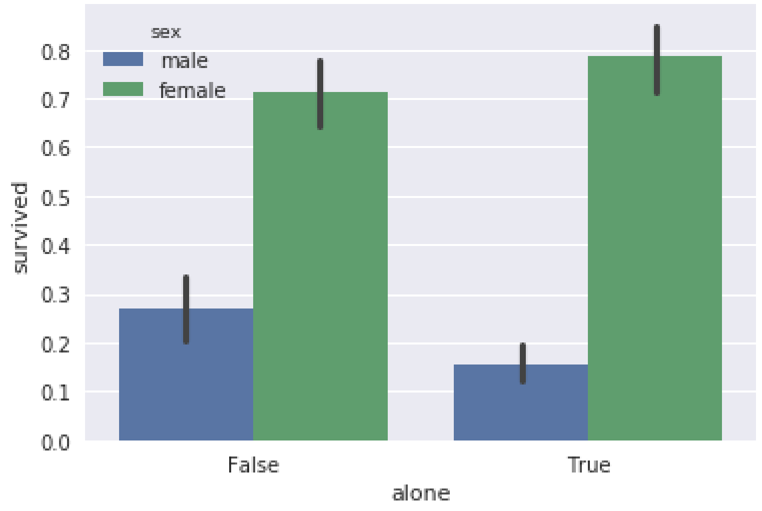
```
In [12]: sns.barplot(x="class",y="survived",data=titanic,hue='sex')
```

```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7f659889ea90>
```



```
In [13]: sns.barplot(x="alone",y="survived",data=titanic,hue='sex')
```

```
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6598dbf9b0>
```



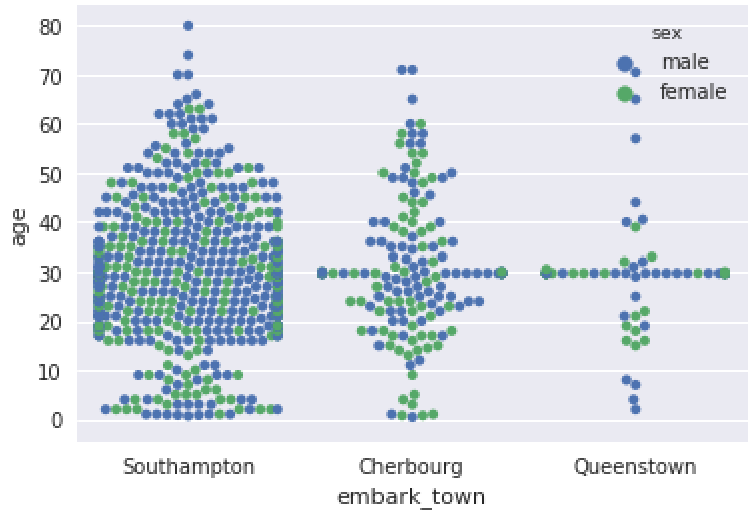
What if we wanted to add a further sex breakdown for the categories chosen above? Go back and add a `hue='sex'` parameter for the couple plots you just created, and seaborn will split each bar into a male/female comparison.

Now we want to compare the embarking town vs the age of the individuals. We don't simply want to use a barplot, since that will just give the average age; rather, we would like more insight into the relative and numeric *distribution* of ages.

A good tool to help us here is `swarmplot` (https://seaborn.pydata.org/generated/seaborn.swarmplot.html). Use this function to view `embark_town` vs `age`, again using `sex` as the `hue`.

In [14]:
```
sns.swarmplot(x="embark_town",y="age",hue="sex",data=titanic)
```

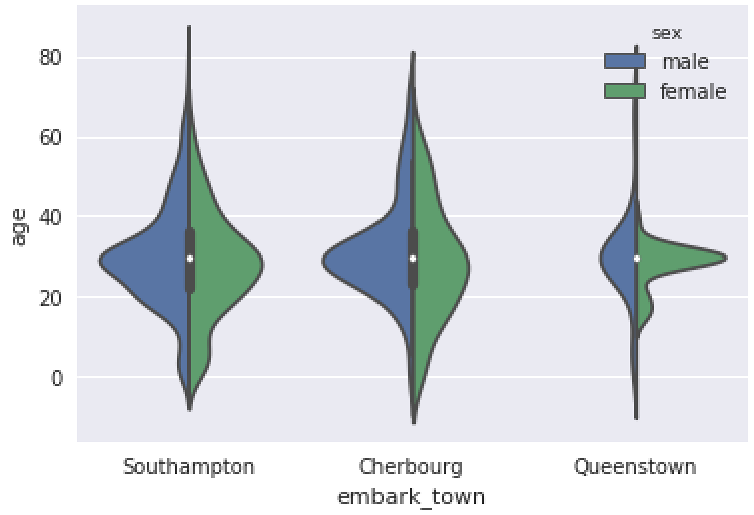
Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6598d29358>
```



Cool! This gives us much more information. What if we didn't care about the number of individuals in each category at all, but rather just wanted to see the *distribution* in each category? `violinplot` (https://seaborn.pydata.org/generated/seaborn.violinplot.html) plots a density distribution. Plot that. Keep the hue.

In [15]:
```
nplot(x=titanic["embark_town"],y=titanic["age"],data=titanic,hue="sex",split=True
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6598d29d30>
```



Go back and clean up the violinplot by adding `split='True'` parameter.

Now take a few seconds to look at the graphs you've created of this data. What are some observations? Jot a couple down here.

- Women survived at a far higher rate than men, regardless of class or alone status.
- The most people came from Southampton and most of those people ranged from 20-40 years of age.
- More men than women came from Southampton and Cherbourg, but the opposite was the case for Queenstown.

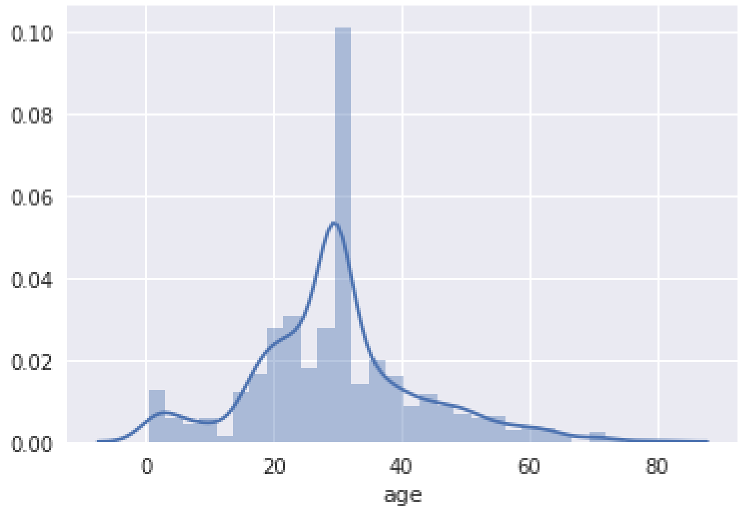
As I mentioned, data is categorical or numeric. We already started getting into numerical data with the swarmplot and violinplot. We will now explore a couple more examples.

Let's look at the distribution of ages. Use `displot` (https://seaborn.pydata.org/generated/seaborn.distplot.html) to make a histogram of just the ages.

If you did your missing value imputation by average value, your results will look very skewed. This is why we don't normally just fill in an average. As a quick fix for now, though, you can filter out the age values that equal the mean before passing it in to `displot`. Do this.

In [16]:
```
mean = np.mean(titanic["age"])
data = titanic["age"][titanic["age"] != mean]
sns.distplot(data)
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6598e73a20>
```

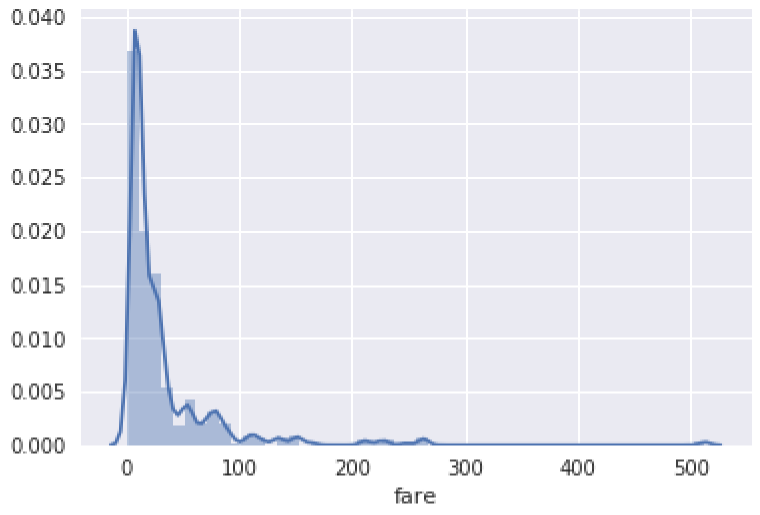


A histogram can nicely represent numerical data by breaking up numerical ranges into chunks so that it is easier to visualize. As you might notice from above, seaborn also automatically plots a gaussian kernel density estimate.

Do the same thing for fares - do you notice something odd about that histogram? What does that skew mean?

In [17]:
```
mean = np.mean(titanic["fare"])
data = titanic["fare"][titanic["fare"] != mean]
sns.distplot(data)
#The skew means that most passengers paid very little.
```

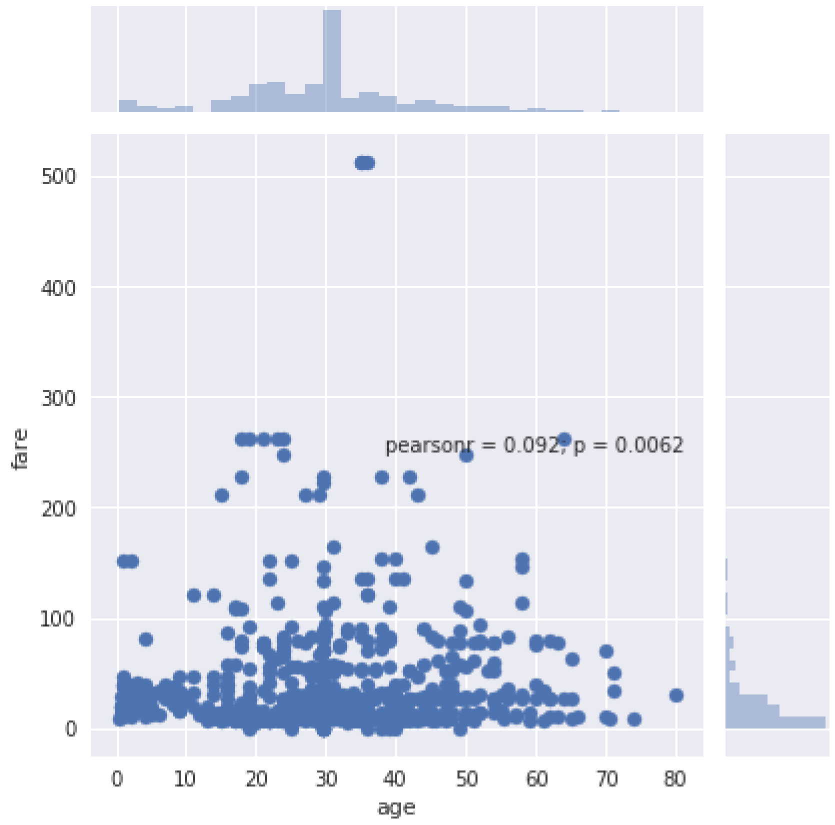
Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6598c09828>
```



Now, using the `jointplot` (https://seaborn.pydata.org/generated/seaborn.jointplot.html#seaborn.jointplot) function, make a scatterplot of the `age` and `fare` variables to see if there is any relationship between the two.

In [18]:
```
sns.jointplot(x="age",y="fare",data=titanic)
#I don't see any relationship. Most people, regardless of age, paid less than 100.
```

Out[18]:
```
<seaborn.axisgrid.JointGrid at 0x7f65a000dbe0>
```



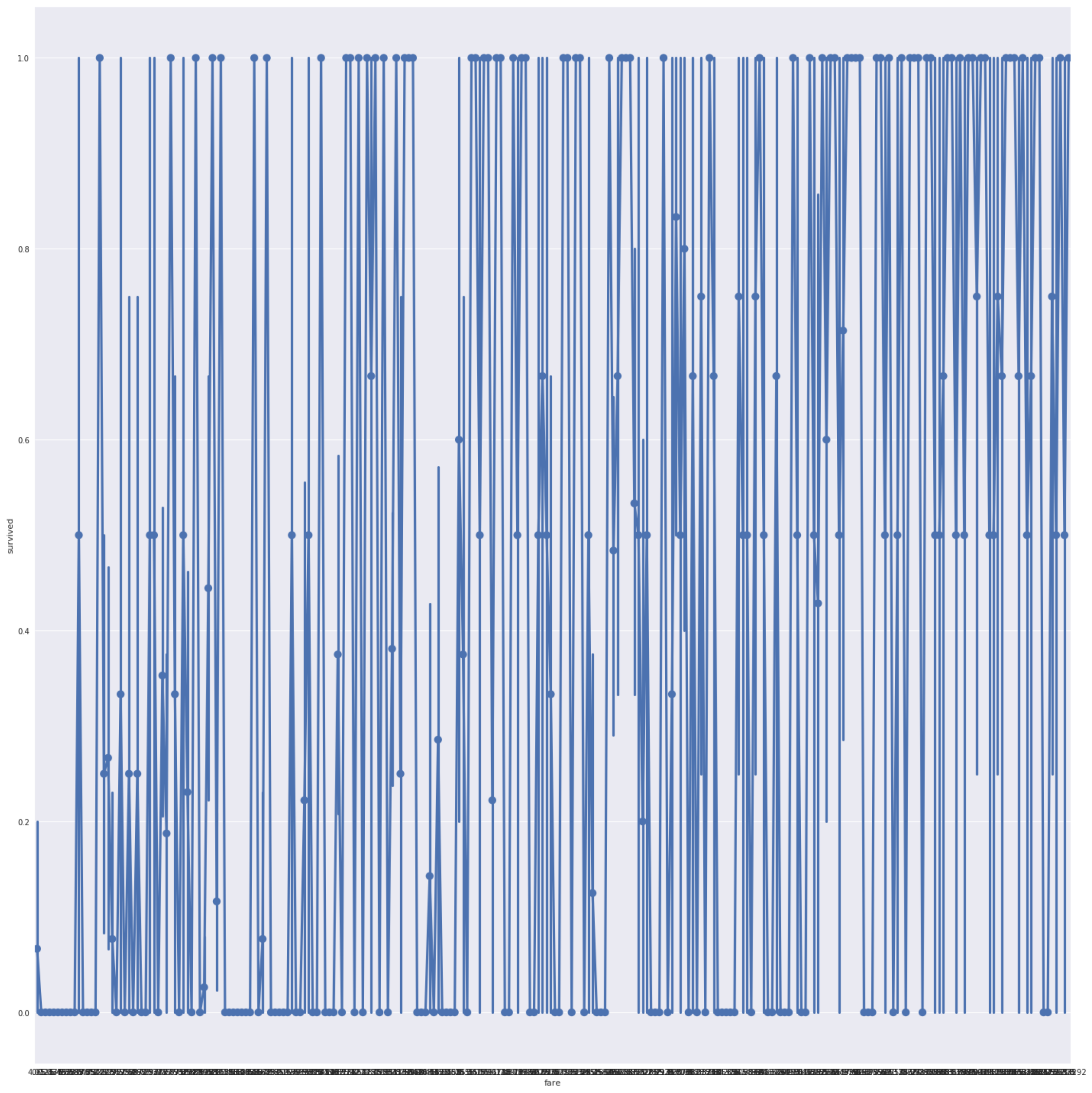
Scatterplots allow one to easily see trends/coorelations in data. As you can see here, there seems to be very little correlation. Also observe that seaborn automatically plots histograms.

Now, use a seaborn function we haven't used yet to plot something. The API (http://seaborn.pydata.org/api.html) has a list of all the methods.

In [ ]:

```
In [19]: sns.factorplot(x="fare",y="survived",data=titanic,size=20)
```

```
Out[19]: <seaborn.axisgrid.FacetGrid at 0x7f6598ac12b0>
```



```
In [ ]:
```

```
In [ ]:
```