Matrix Plots

Matrix plots allow you to plot data as color-encoded matrices.

In [1]:
```
import seaborn as sns
%matplotlib inline
```

In [2]:
```
flights = sns.load_dataset('flights')
```

In [3]:
```
tips = sns.load_dataset('tips')
```

In [4]:
```
tips.head()
```

Out[4]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

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

Out[5]:

	year	month	passengers
0	1949	January	112
1	1949	February	118
2	1949	March	132
3	1949	April	129
4	1949	May	121

Heatmap

In order for a heatmap to work properly, your data should already be in a matrix form

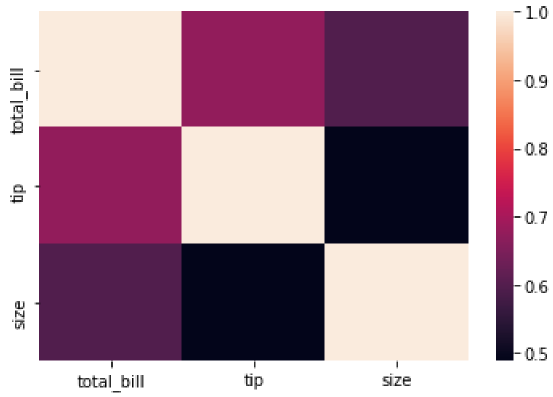
In [6]:
```
# Matrix form for correlation data
tips.corr()
```

Out[6]:

	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

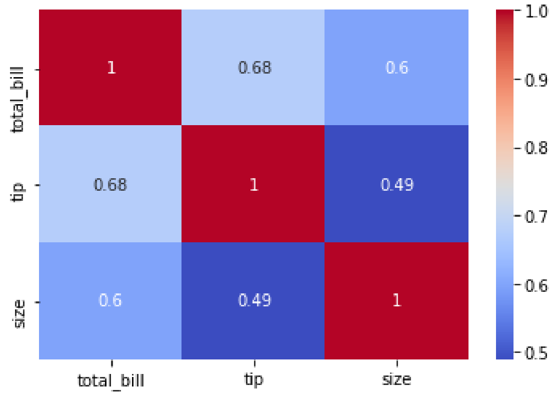
```
In [7]: sns.heatmap(tips.corr())
```

```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x1a221c26d0>
```




```
In [8]: sns.heatmap(tips.corr(),cmap='coolwarm',annot=True)
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2318d5d0>
```



Or for the flights data:

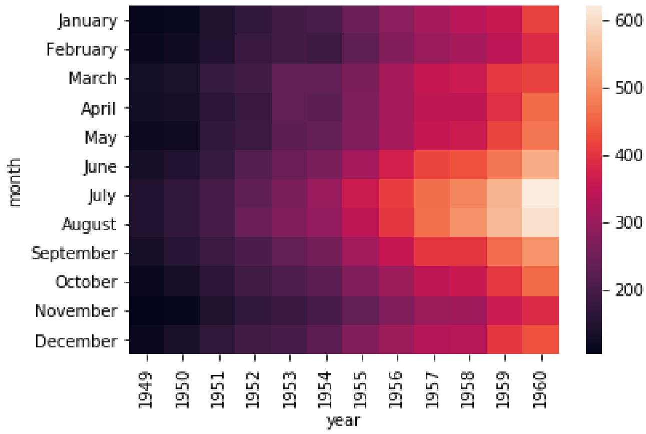
```
In [9]: flights.pivot_table(values='passengers', index='month', columns='year')
```

Out[9]:

year	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
month												
January	112	115	145	171	196	204	242	284	315	340	360	417
February	118	126	150	180	196	188	233	277	301	318	342	391
March	132	141	178	193	236	235	267	317	356	362	406	419
April	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
June	135	149	178	218	243	264	315	374	422	435	472	535
July	148	170	199	230	264	302	364	413	465	491	548	622
August	148	170	199	242	272	293	347	405	467	505	559	606
September	136	158	184	209	237	259	312	355	404	404	463	508
October	119	133	162	191	211	229	274	306	347	359	407	461
November	104	114	146	172	180	203	237	271	305	310	362	390
December	118	140	166	194	201	229	278	306	336	337	405	432

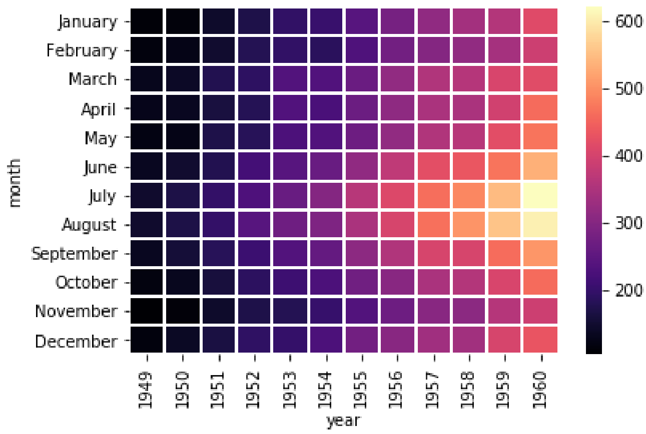
In [10]:
```python
pvflights = flights.pivot_table(values='passengers', index='month', columns='year')
sns.heatmap(pvflights)
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1a232da550>



In [11]:
```python
sns.heatmap(pvflights, cmap='magma', linecolor='white', linewidths=1)
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x1a234394d0>

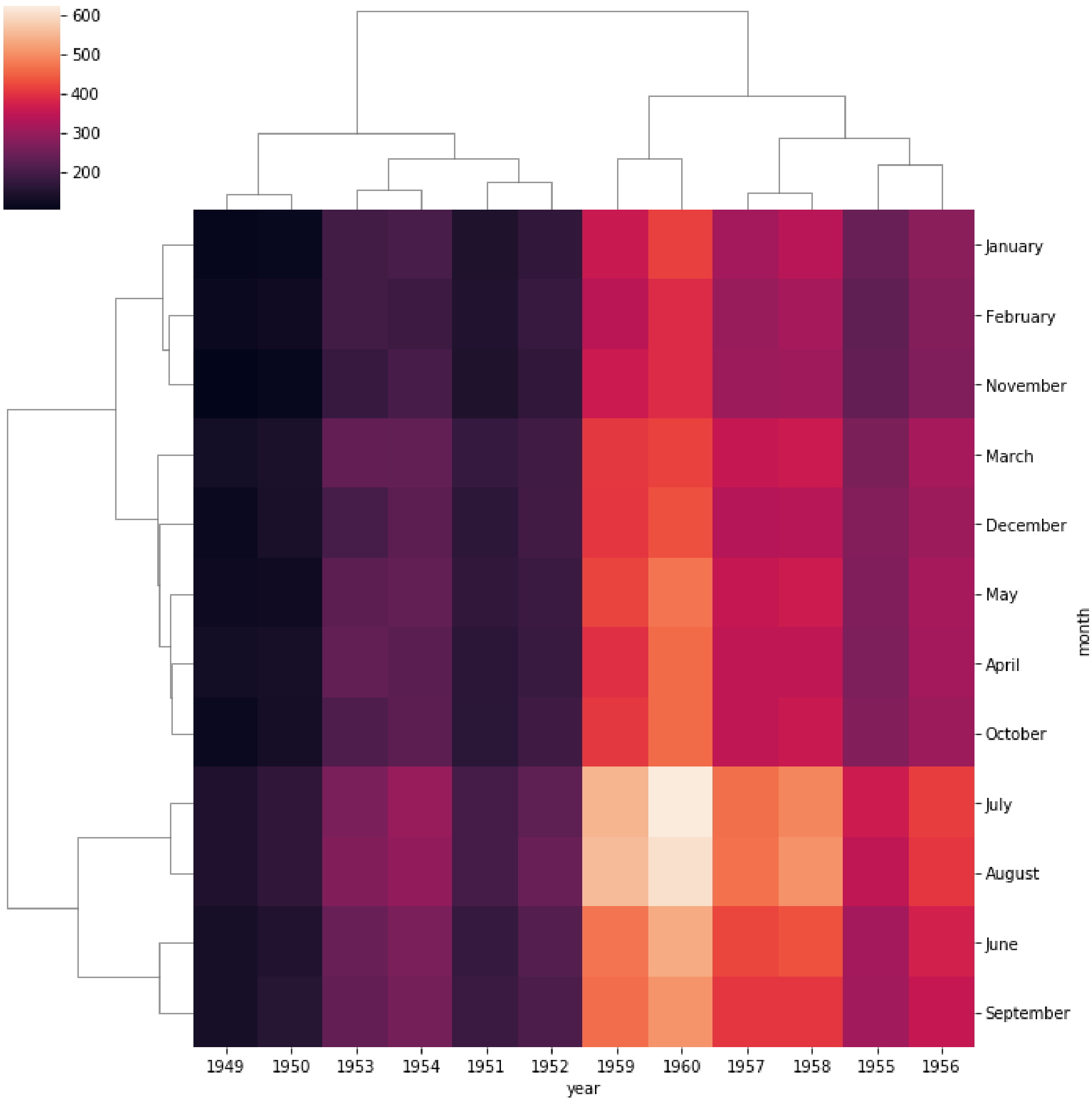


clustermap

The clustermap uses hierarchal clustering to produce a clustered version of the heatmap.

In [12]:
```
sns.clustermap(pvflights)
```

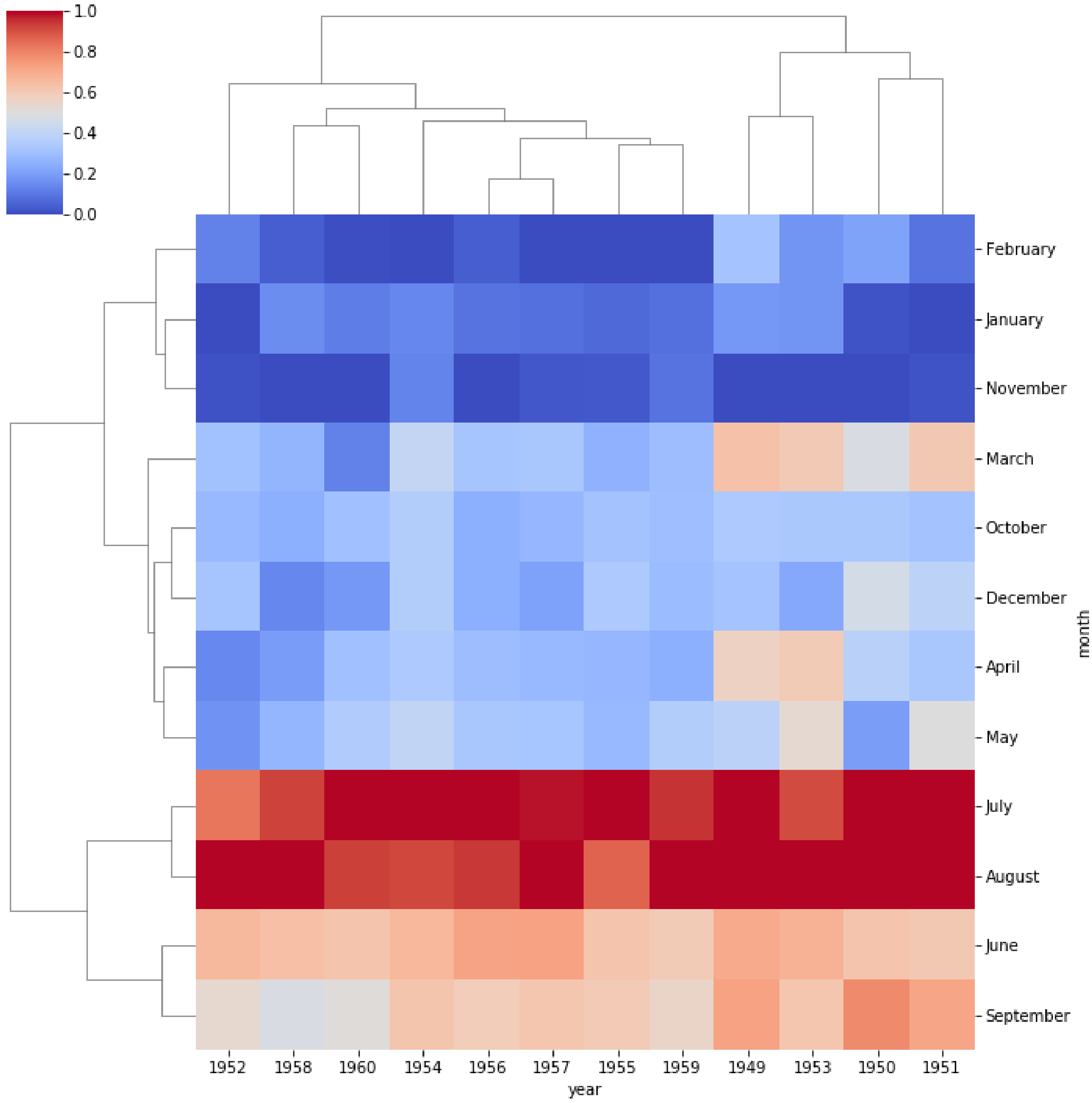
Out[12]: <seaborn.matrix.ClusterGrid at 0x1a23567cd0>



The years and months are no longer in order, but grouped by similarity in value. We can begin to infer things from this plot.

In [13]:
```
# More options to get the information a little clearer like normalization
sns.clustermap(pvflights, cmap='coolwarm', standard_scale=1)
```

Out[13]: <seaborn.matrix.ClusterGrid at 0x1a239786d0>



Grids

Grids are general types of plots that allow you to map plot types to rows and columns of a grid, this helps you create similar plots separated by features.

In [1]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
iris = sns.load_dataset('iris')
```

In [3]:
```
iris.head()
```

Out[3]:

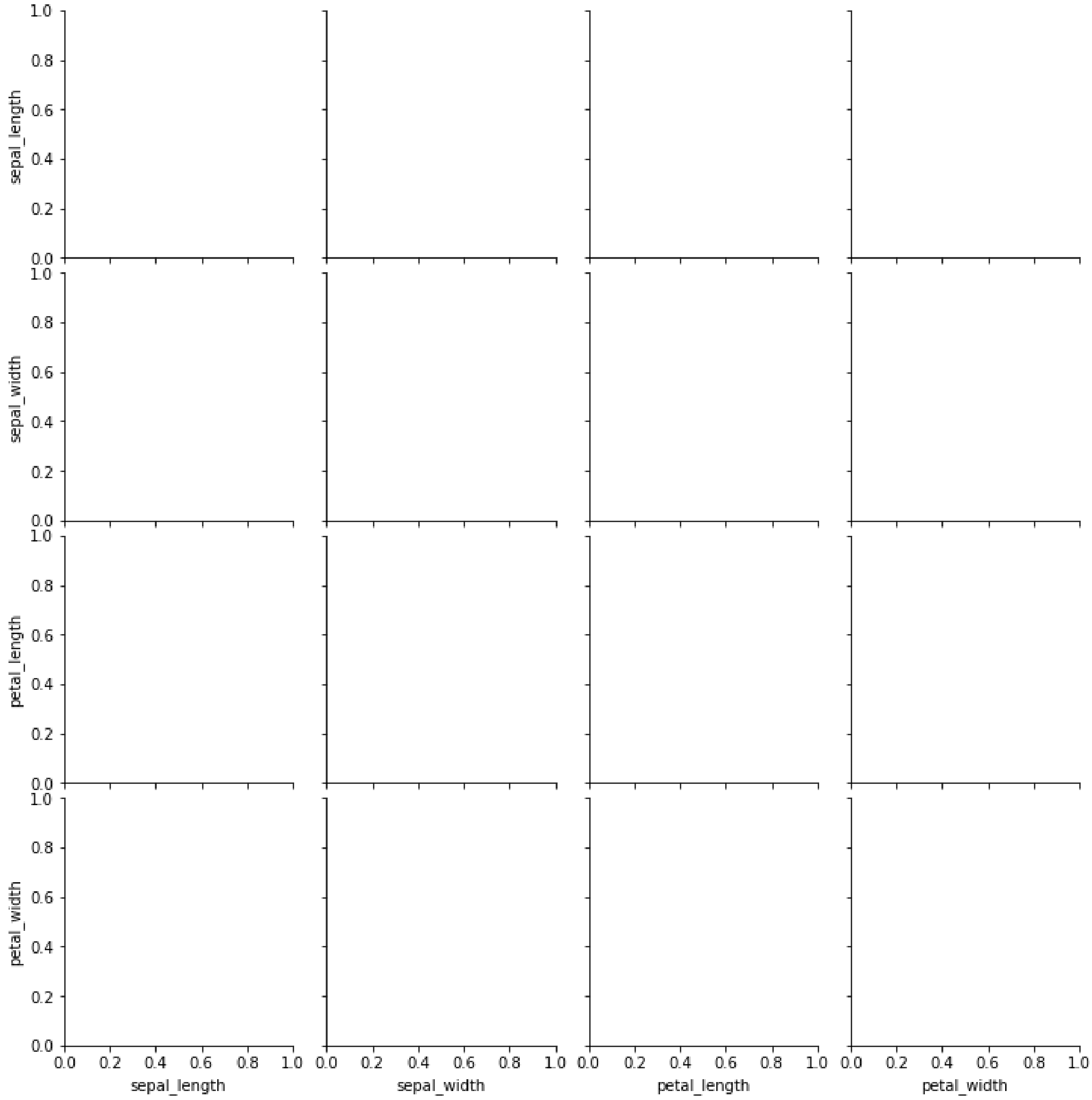
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

PairGrid

Pairgrid is a subplot grid for plotting pairwise relationships in a dataset.

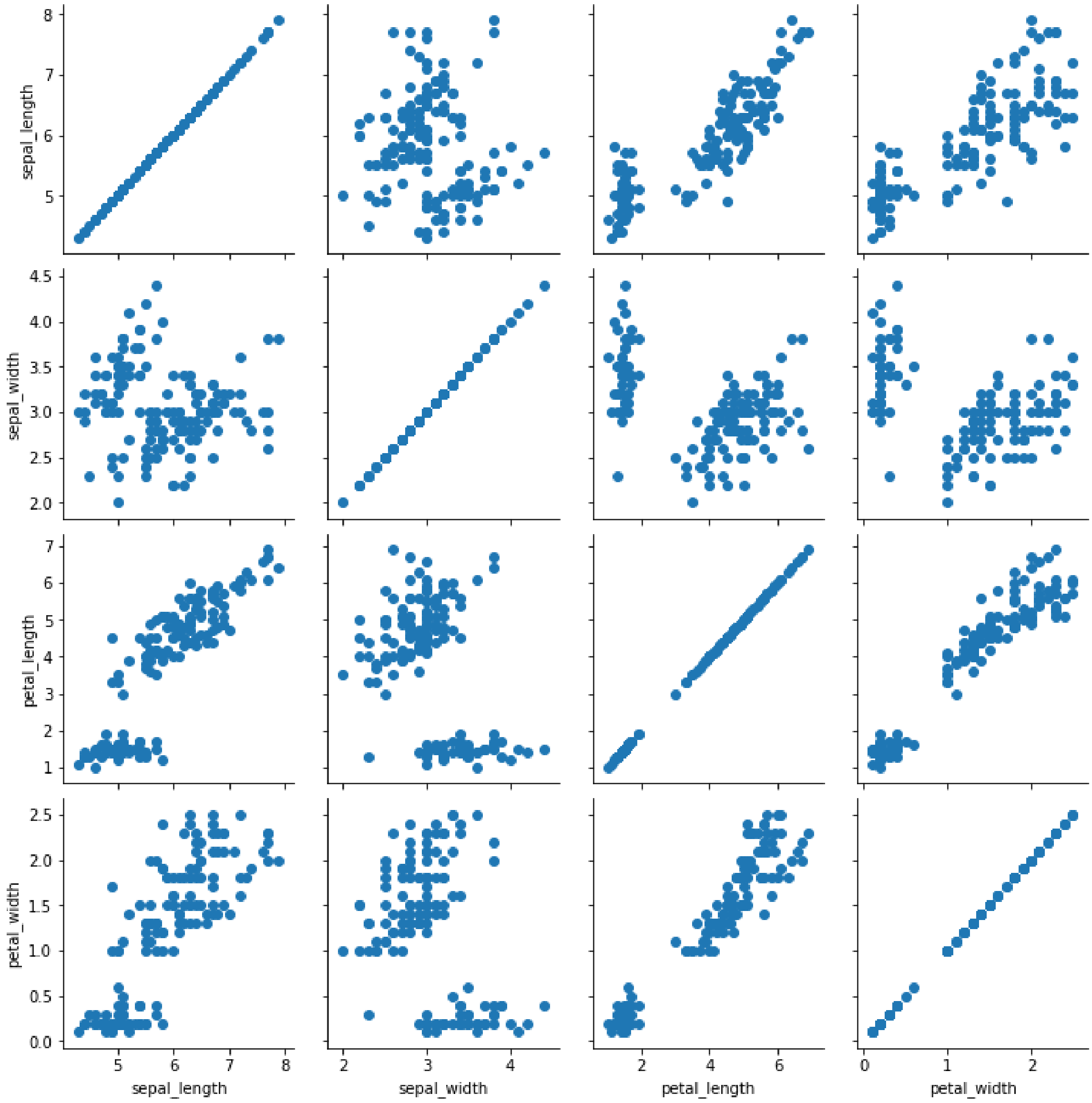
In [4]:
```
# Just the Grid
sns.PairGrid(iris)
```

Out[4]: <seaborn.axisgrid.PairGrid at 0x1a187ae650>



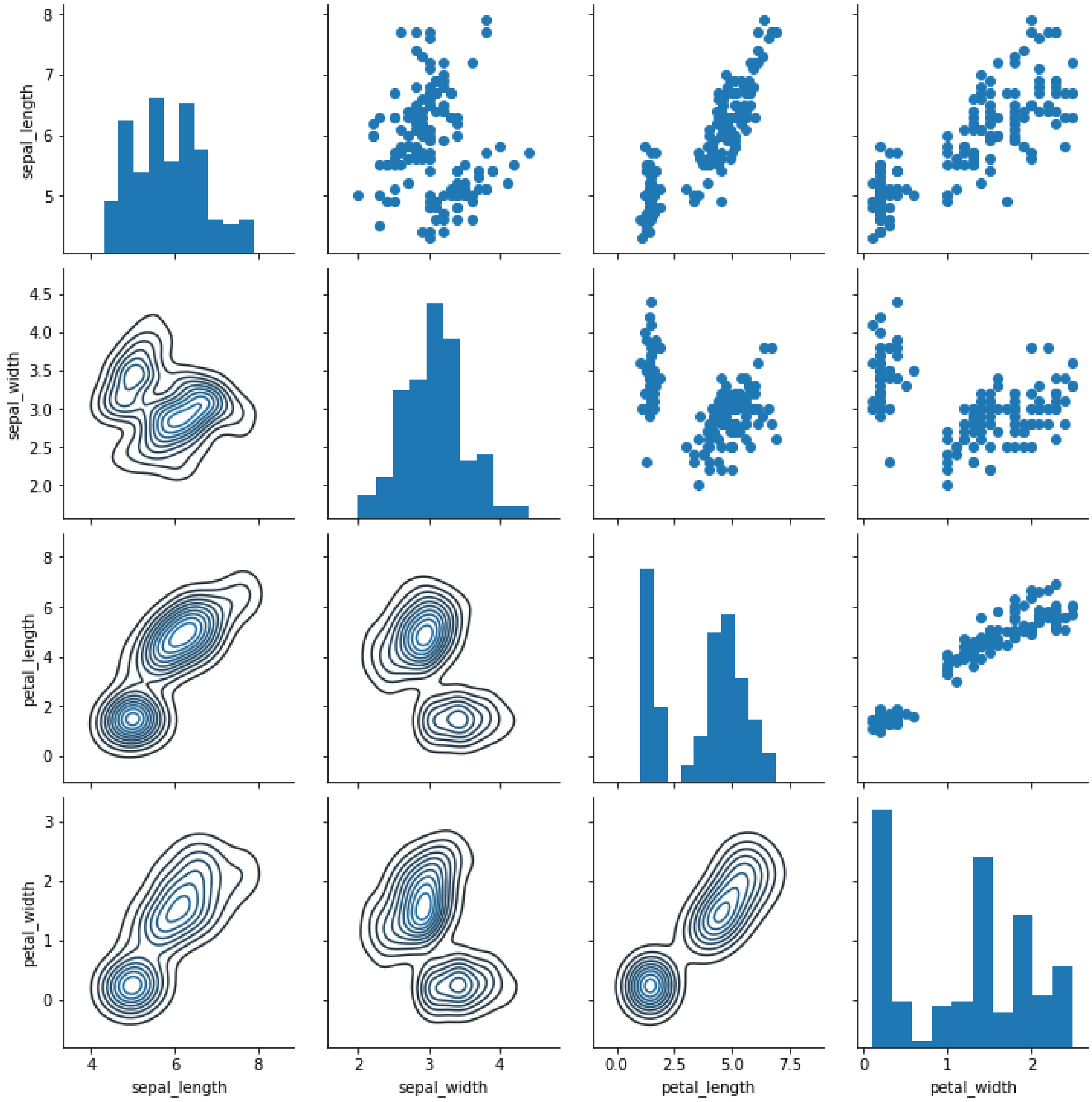
In [5]:
```
# Then you map to the grid
g = sns.PairGrid(iris)
g.map(plt.scatter)
```

Out[5]: <seaborn.axisgrid.PairGrid at 0x1a19223cd0>



In [6]:
```
# Map to upper,lower, and diagonal
g = sns.PairGrid(iris)
g.map_diag(plt.hist)
g.map_upper(plt.scatter)
g.map_lower(sns.kdeplot)
```

Out[6]: <seaborn.axisgrid.PairGrid at 0x1a19d24fd0>

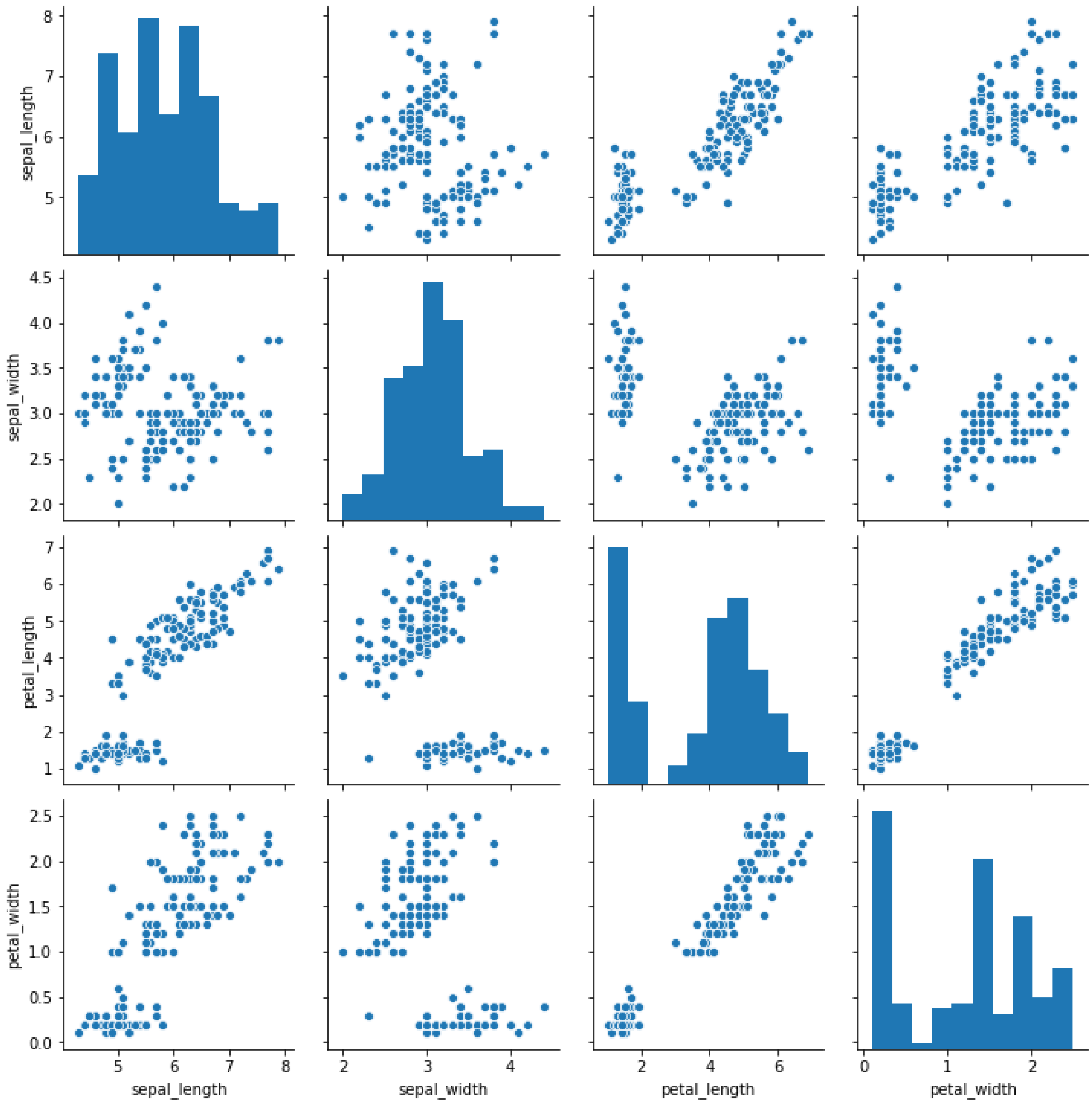


PairPlot

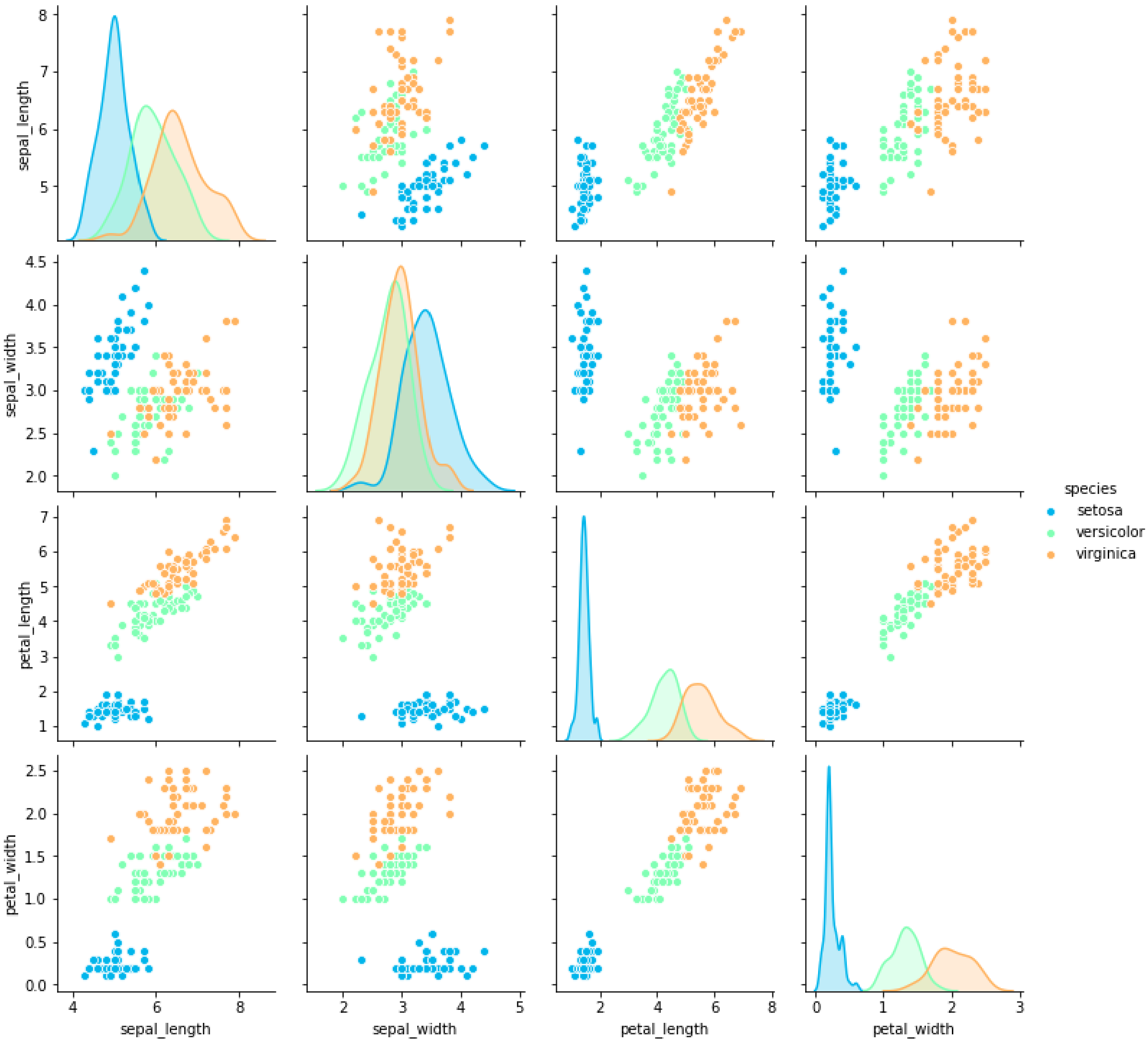
pairplot is a simpler version of PairGrid

In [7]:
```
sns.pairplot(iris)
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x1a1aa119d0>
```



In [8]:
```
sns.pairplot(iris, hue='species', palette='rainbow')
```
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x1a1b4f2110>
```



Regression Plots

lmplot allows you to display linear models, and it allows to split up those plots based off of features, as well as coloring the hue based off of features.

In [1]:
```
import seaborn as sns
%matplotlib inline
```

In [2]:
```
tips = sns.load_dataset('tips')
```

In [3]:
```
tips.head()
```

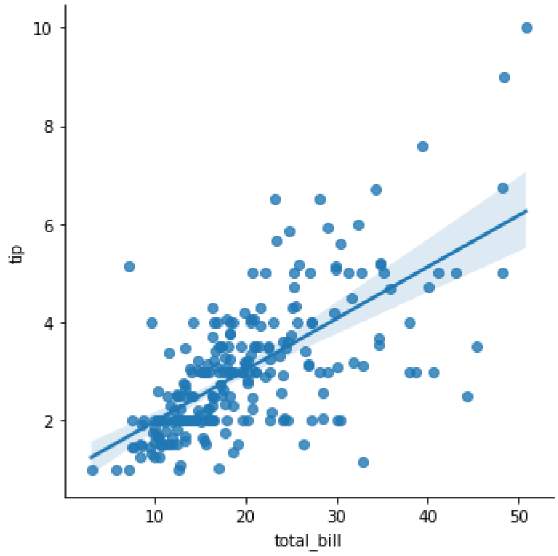
Out[3]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

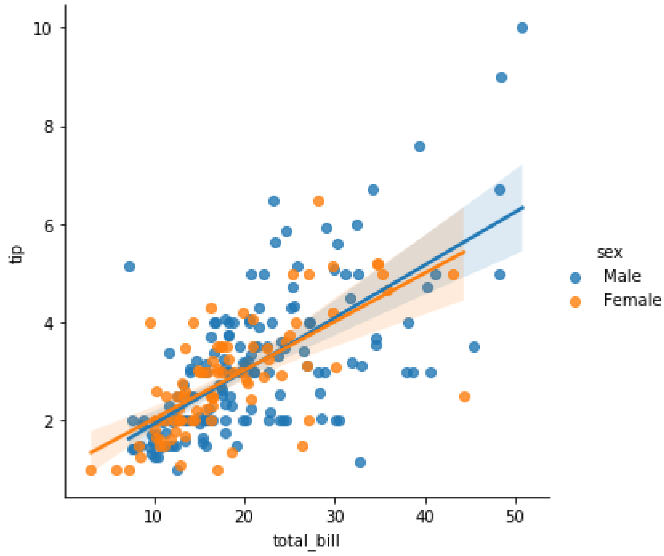
lmplot()

In [4]:
```
sns.lmplot(x='total_bill', y='tip', data=tips)
```

Out[4]: <seaborn.axisgrid.FacetGrid at 0x1a22b75150>

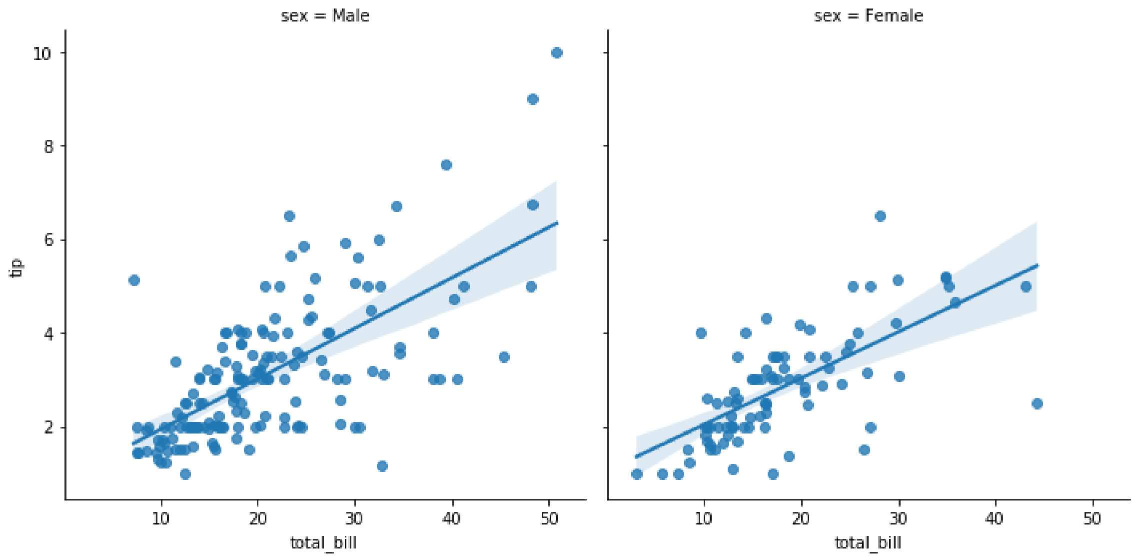


In [5]:
```
sns.lmplot(x='total_bill', y='tip', data=tips, hue='sex')
```
Out[5]: <seaborn.axisgrid.FacetGrid at 0x1a23e6c3d0>

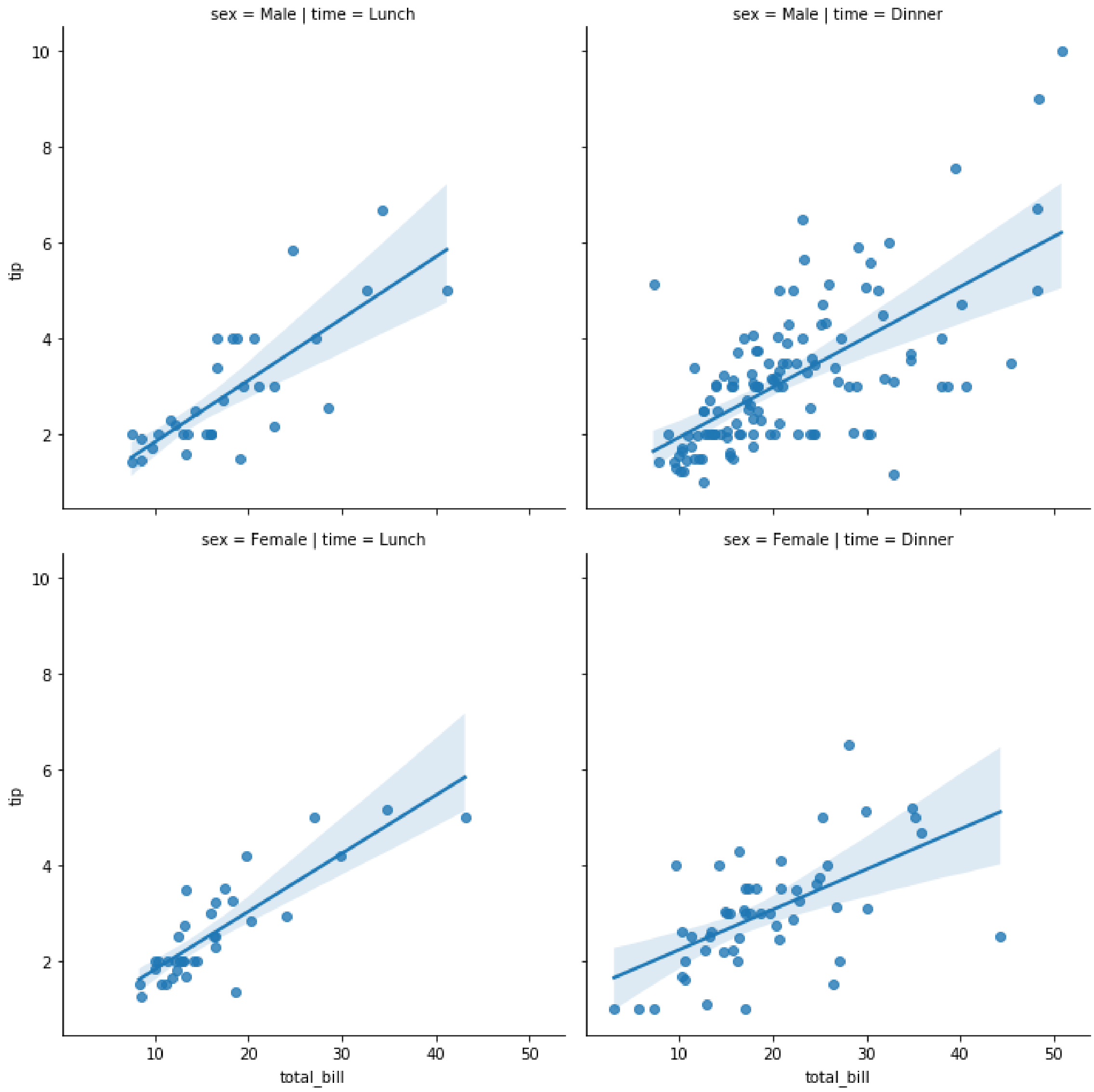


Using a Grid

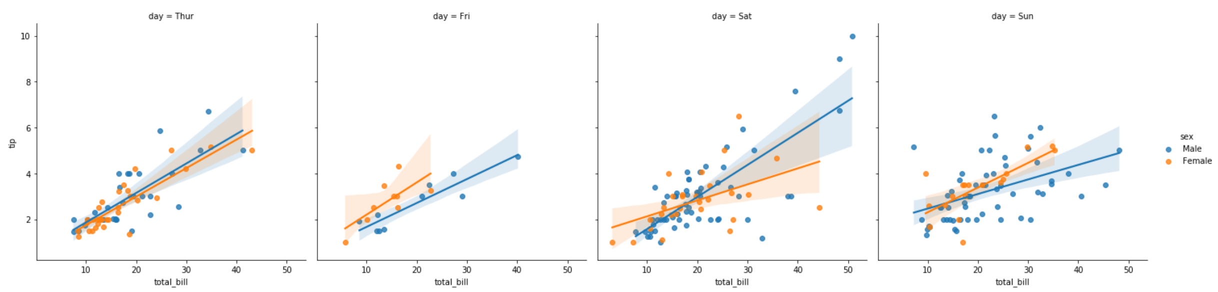
In [7]:
```
sns.lmplot(x='total_bill', y='tip', data=tips, col='sex')
```
Out[7]: <seaborn.axisgrid.FacetGrid at 0x1a2421e510>



In [8]:
```
sns.lmplot(x="total_bill", y="tip", row="sex", col="time", data=tips)
```
Out[8]: <seaborn.axisgrid.FacetGrid at 0x1a243d1610>



In [10]:
```
sns.lmplot(x='total_bill', y='tip', data=tips, col='day', hue='sex')
```
Out[10]: <seaborn.axisgrid.FacetGrid at 0x1a24f32e10>

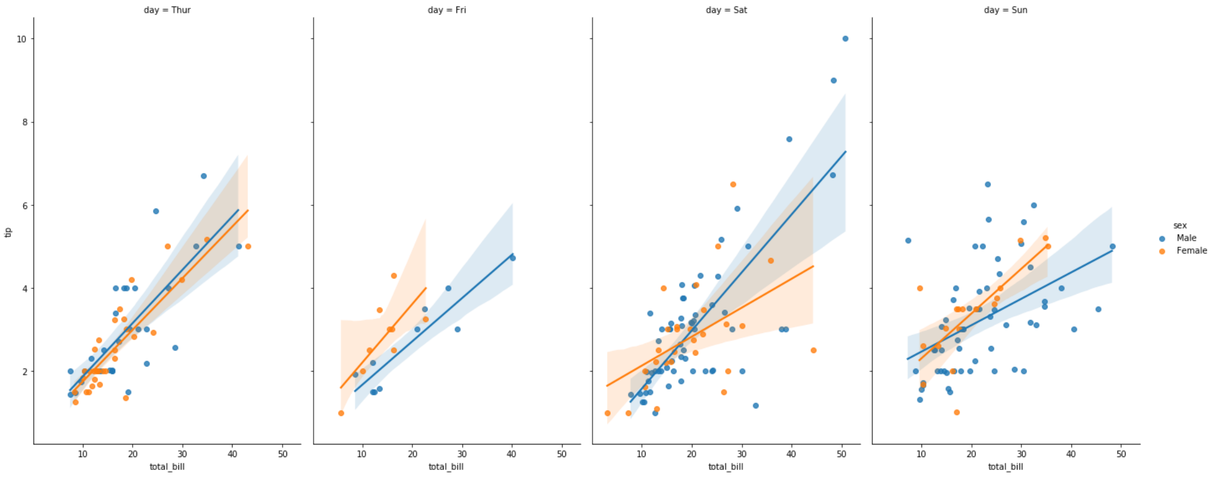


Aspect and Size

Seaborn figures can have their size and aspect ratio adjusted with the **height** and **aspect** parameters:

In [12]:
```python
sns.lmplot(x='total_bill', y='tip', data=tips, col='day', hue='sex',
           aspect=0.6, height=8)
```

Out[12]: <seaborn.axisgrid.FacetGrid at 0x1a25e66d10>



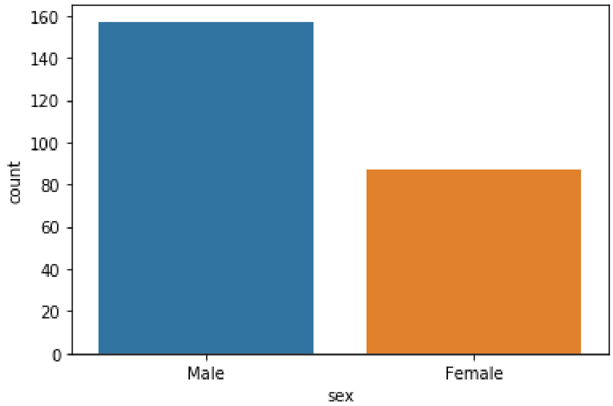
Style and Color

In [1]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
tips = sns.load_dataset('tips')
```

Styles

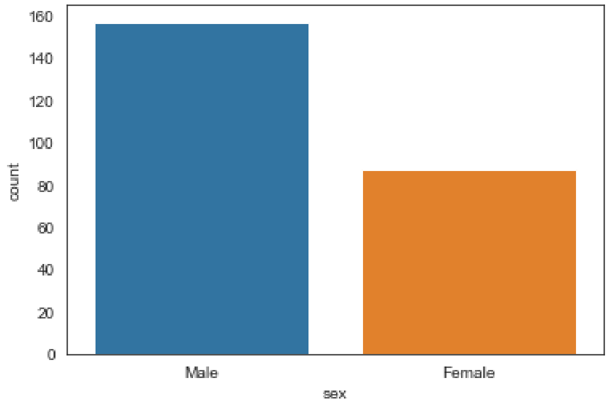
In [2]:
```
sns.countplot(x='sex', data=tips)
```

Out[2]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1a504110>



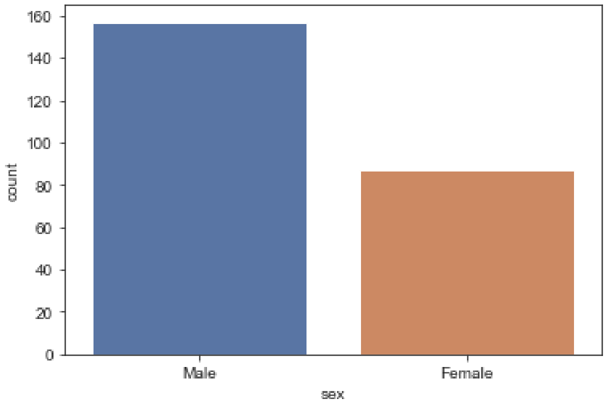
In [3]:
```
sns.set_style('white')
sns.countplot(x='sex',data=tips)
```

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1b4484d0>



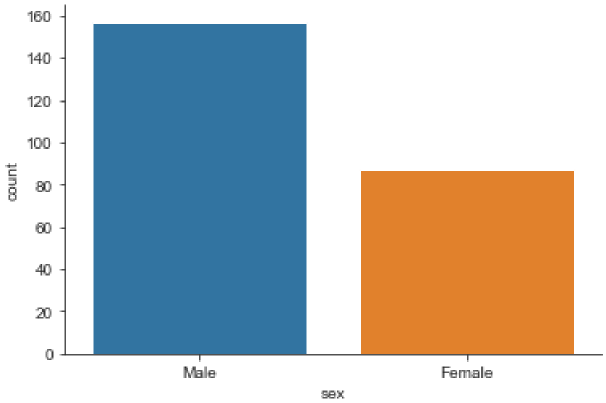
In [4]:
```
sns.set_style('ticks')
sns.countplot(x='sex', data=tips, palette='deep')
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1b559850>

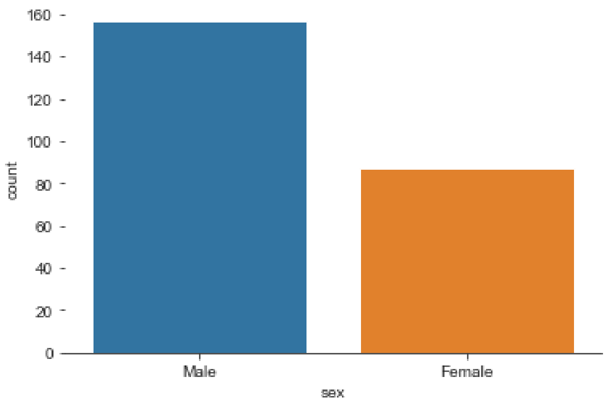


Spine Removal

In [5]:
```
sns.countplot(x='sex', data=tips)
sns.despine()
```



In [6]:
```
sns.countplot(x='sex', data=tips)
sns.despine(left=True)
```

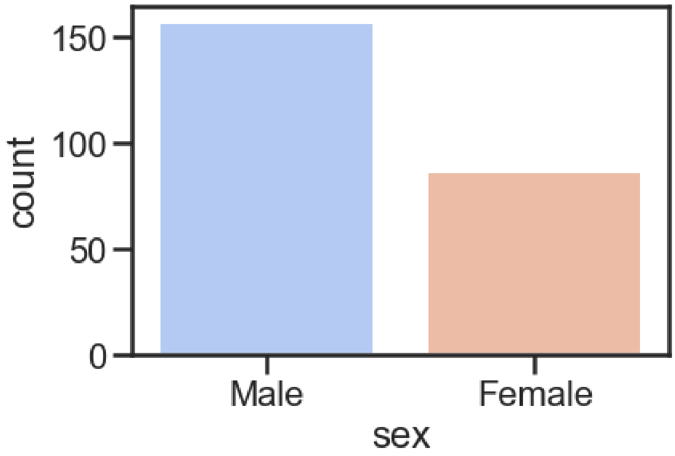


Scale and Context

The set_context() allows you to override default parameters:

```
In [10]: sns.set_context('poster', font_scale=1)
         sns.countplot(x='sex', data=tips, palette='coolwarm')
```

```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1ba9bdd0>
```



Check out the documentation page for more info on these topics: [http://seaborn.pydata.org/tutorial/aesthetics.html (http://seaborn.pydata.org/tutorial/aesthetics.html)](http://seaborn.pydata.org/tutorial/aesthetics.html)

Seaborn Exercises - Solutions

The Data

We will be working with a famous titanic data set for these exercises.

In [1]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
sns.set_style('whitegrid')
```

In [3]:
```
titanic = sns.load_dataset('titanic')
```

In [4]:
```
titanic.head()
```

Out[4]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no

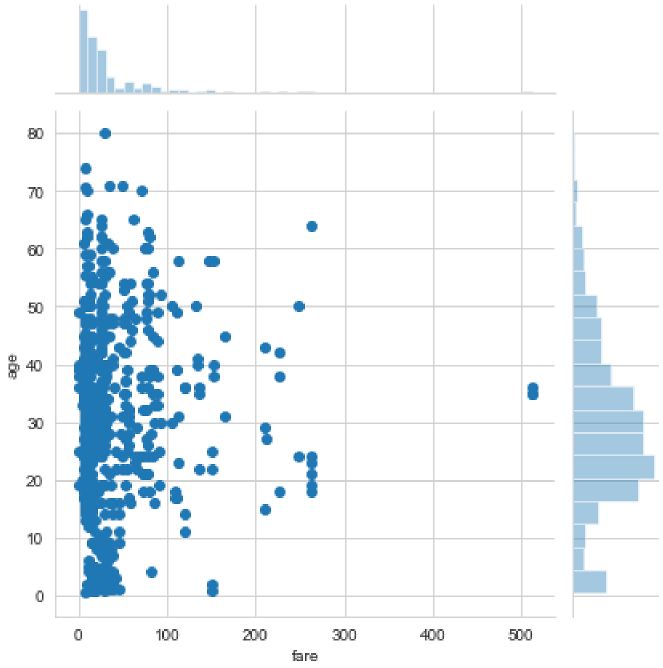
Exercises

- **Recreate the plots below using the titanic dataframe.**
- **In order not to lose the plot image, code in the cell with `# CODE HERE`**
- **The palettes are not important**

In [5]:
```
# CODE HERE
```

In [6]:
```
sns.jointplot(x='fare', y='age', data=titanic)
```

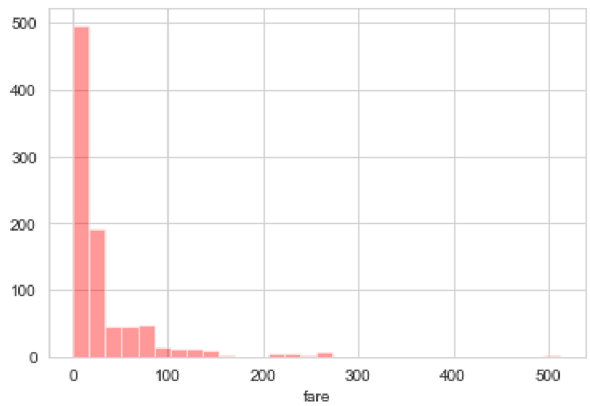
Out[6]: <seaborn.axisgrid.JointGrid at 0x1a15de5850>



In [7]:
```
# CODE HERE
```

In [8]:
```
sns.distplot(titanic['fare'], bins=30, kde=False, color='red')
```

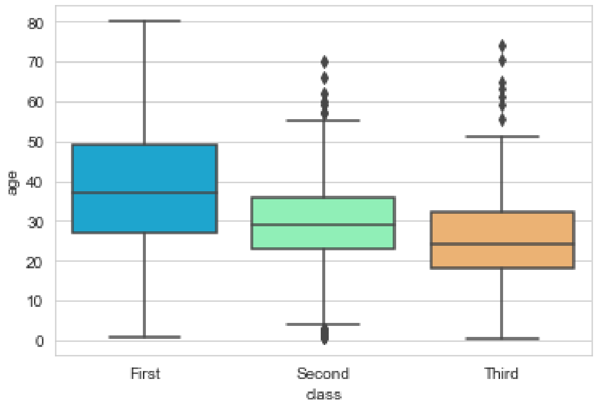
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1a16979710>



In [9]:
```
# CODE HERE
```

In [10]:
```
sns.boxplot(x='class', y='age', data=titanic, palette='rainbow')
```

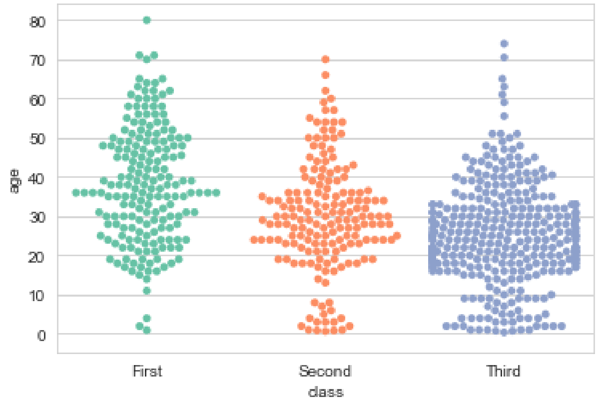
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1a169aa410>



```
In [11]: # CODE HERE
```

```
In [12]: sns.swarmplot(x='class', y='age', data=titanic, palette='Set2')
```

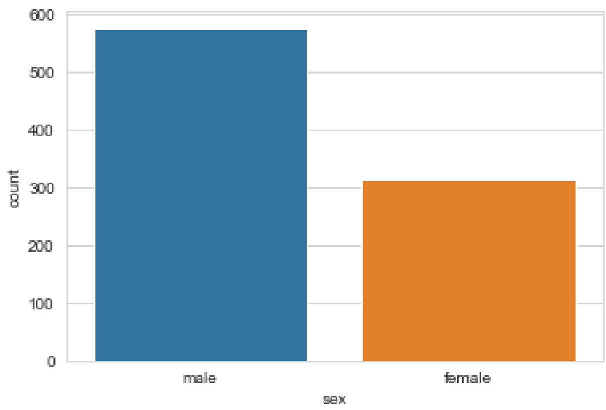
```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1a16b6f3d0>
```




```
In [13]: # CODE HERE
```

```
In [14]: sns.countplot(x='sex', data=titanic)
```

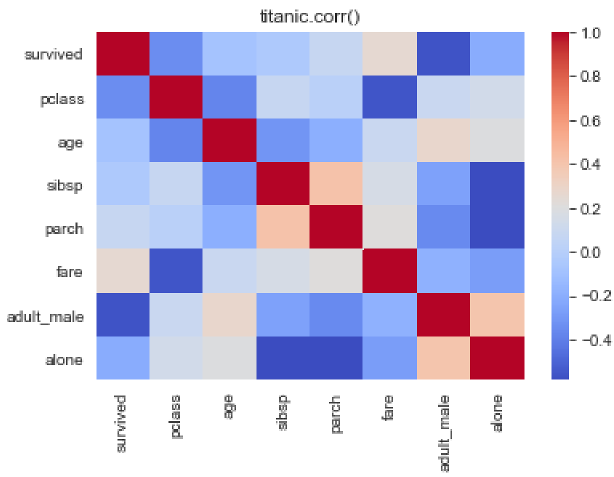
```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x1a16bcbc50>
```




```
In [15]: # CODE HERE
```

```
In [16]: sns.heatmap(titanic.corr(), cmap='coolwarm')
         plt.title('titanic.corr()')
```

```
Out[16]: Text(0.5, 1, 'titanic.corr()')
```



Pandas Built-in Data Visualization

In this lecture we will learn about pandas built-in capabilities for data visualization! It's built-off of matplotlib, but it baked into pandas for easier usage!

Imports

In [1]:
```
import numpy as np
import pandas as pd
%matplotlib inline
```

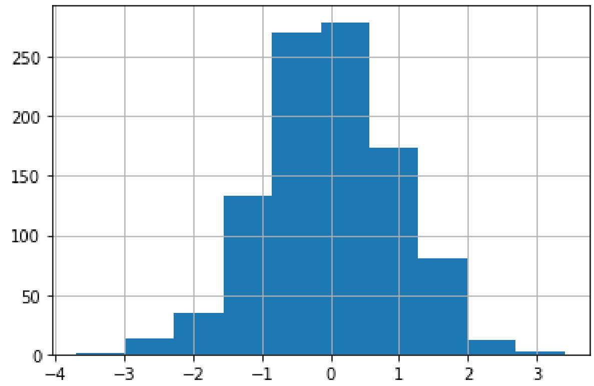
The Data

There are some fake data csv files you can read in as dataframes:

In [2]:
```
df1 = pd.read_csv('df1',index_col=0)
df2 = pd.read_csv('df2')
```

In [3]:
```
df1['A'].hist()
```

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x11ee9a5d0>

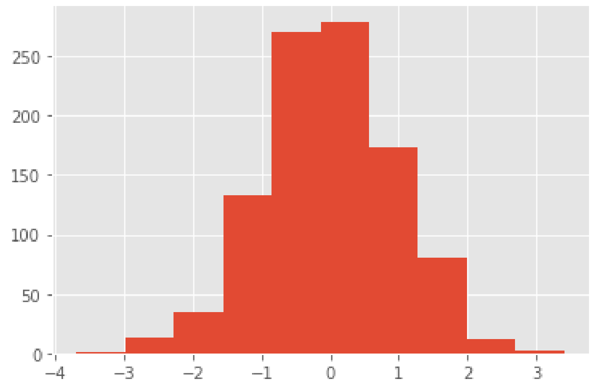


Add a style:

In [4]:
```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

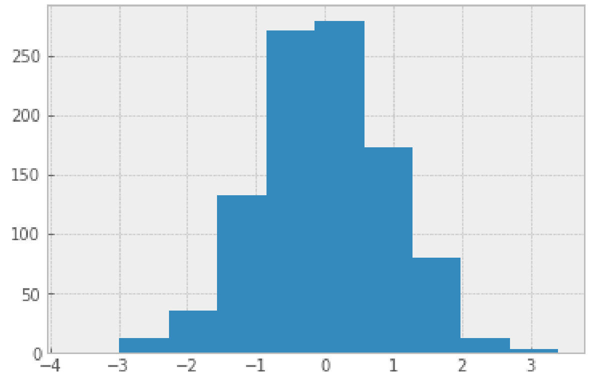
In [5]:
```
df1['A'].hist()
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x11f8cc150>



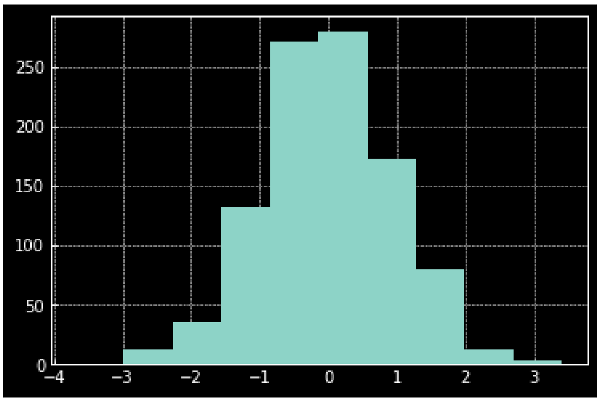
In [6]:
```
plt.style.use('bmh')
df1['A'].hist()
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x11fa16910>



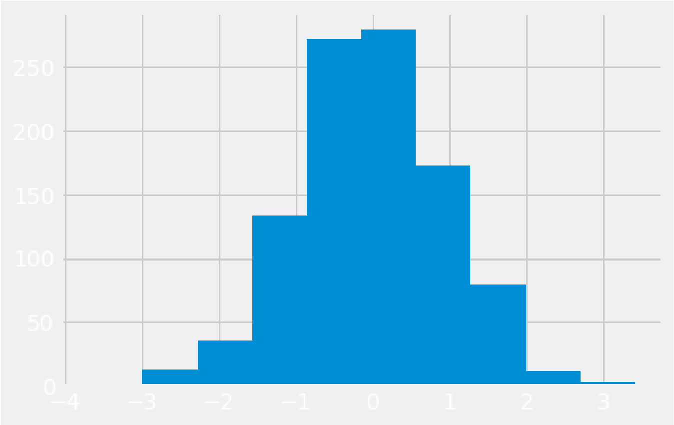
In [7]:
```
plt.style.use('dark_background')
df1['A'].hist()
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x11fb07090>



In [8]:
```
plt.style.use('fivethirtyeight')
df1['A'].hist()
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x11fab7150>



In [9]:
```
plt.style.use('ggplot')
```

Plot Types

There are several plot types built-in to pandas, most of them statistical plots by nature:

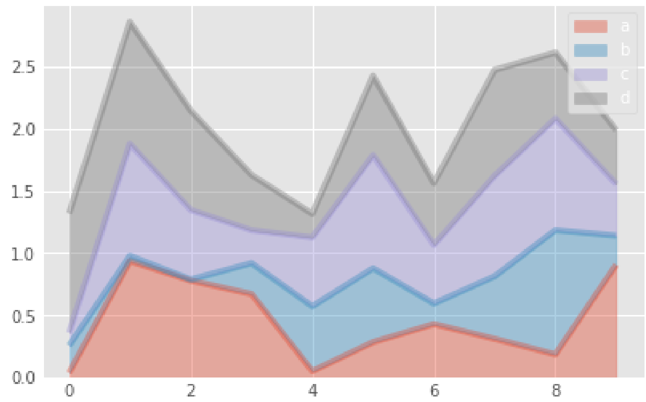
- `df.plot.area`
- `df.plot.barh`
- `df.plot.density`
- `df.plot.hist`
- `df.plot.line`
- `df.plot.scatter`
- `df.plot.bar`
- `df.plot.box`
- `df.plot.hexbin`
- `df.plot.kde`
- `df.plot.pie`

You can also just call `df.plot(kind='hist')`

Area

In [10]:
```
df2.plot.area(alpha=0.4)
```

Out[10]: `<matplotlib.axes._subplots.AxesSubplot at 0x11fceaed0>`



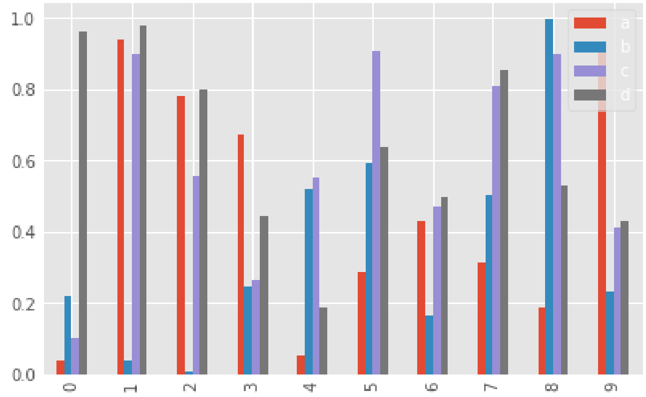
Barplots

In [11]:
```
df2.head()
```

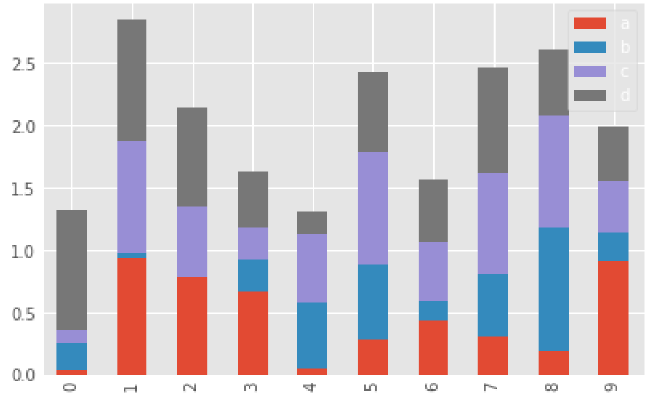
Out[11]:

	a	b	c	d
0	0.039762	0.218517	0.103423	0.957904
1	0.937288	0.041567	0.899125	0.977680
2	0.780504	0.008948	0.557808	0.797510
3	0.672717	0.247870	0.264071	0.444358
4	0.053829	0.520124	0.552264	0.190008

In [12]:
```
df2.plot.bar()
```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x11fe29690>

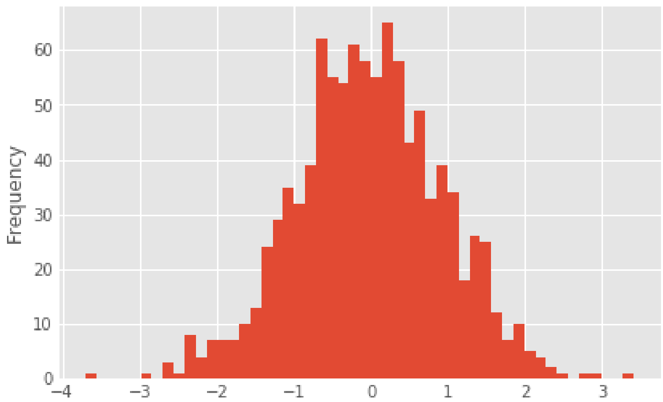


In [13]:
```
df2.plot.bar(stacked=True)
```
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x11ff28050>



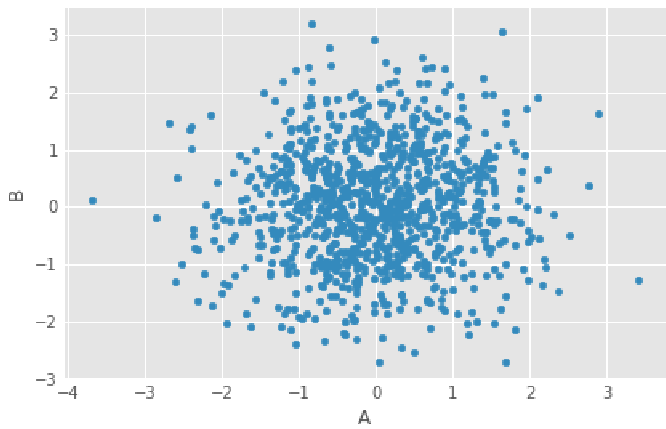
Histograms

In [14]:
```
df1['A'].plot.hist(bins=50)
```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x120033ed0>



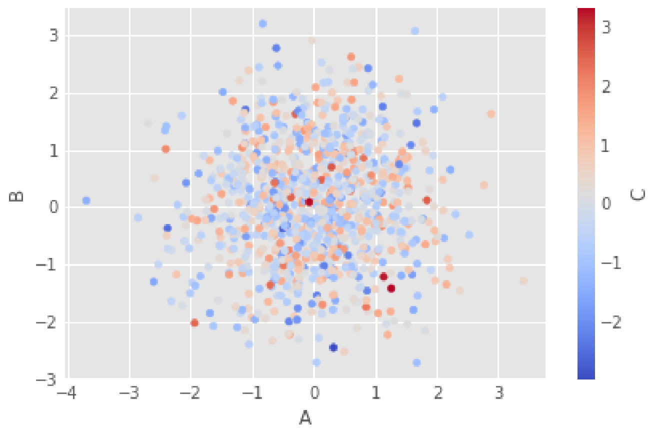
Scatter Plots

In [16]:
```
df1.plot.scatter(x='A', y='B')
```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1203d4210>



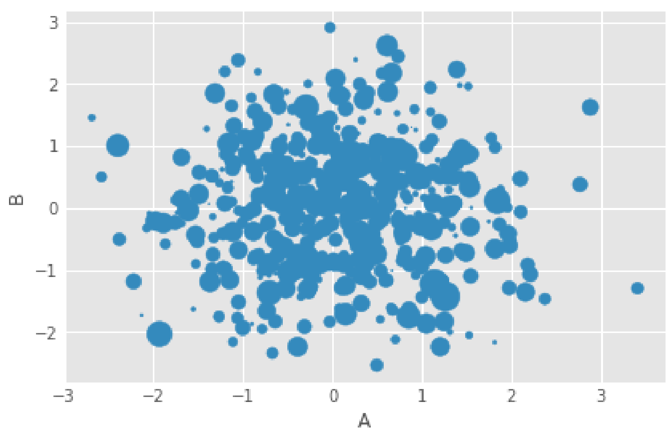
- You can use c to color based off another column value
- Use cmap to indicate colormap to use.
- For all the colormaps, check out: http://matplotlib.org/users/colormaps.html (http://matplotlib.org/users/colormaps.html)

In [17]:
```
df1.plot.scatter(x='A', y='B', c='C', cmap='coolwarm')
```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x11ff6dad0>



Or use `s` to indicate size based off another column. `s` parameter needs to be an array, not just the name of a column:

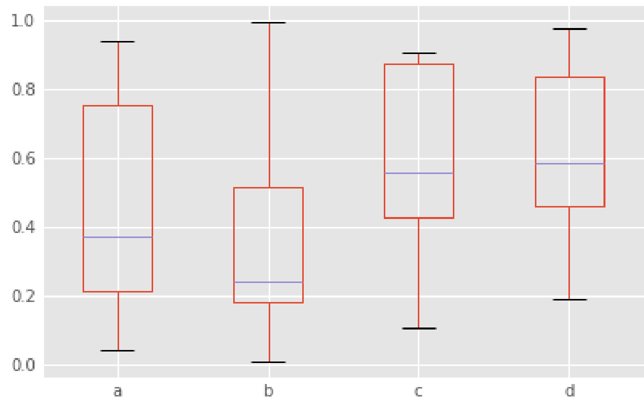
In [21]:
```
df1.plot.scatter(x='A', y='B', s=df1['C']*100.0)
```
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x1208a59d0>



BoxPlots

In [22]:
```
df2.plot.box() # Can also pass a by= argument for groupby
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x1209c0410>

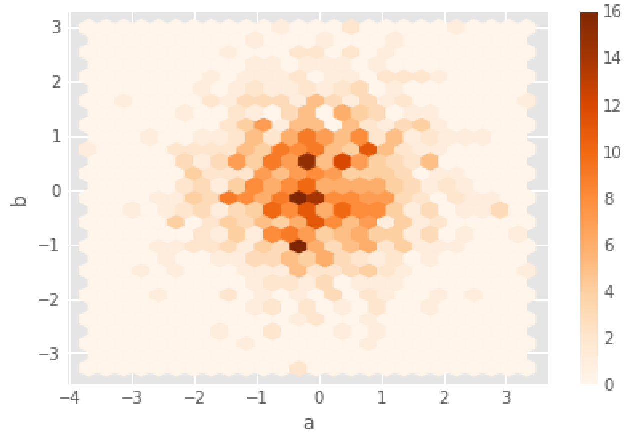


Hexagonal Bin Plot

Useful for Bivariate Data, alternative to scatterplot:

In [23]:
```
df = pd.DataFrame(np.random.randn(1000, 2), columns=['a', 'b'])
df.plot.hexbin(x='a', y='b', gridsize=25, cmap='Oranges')
```

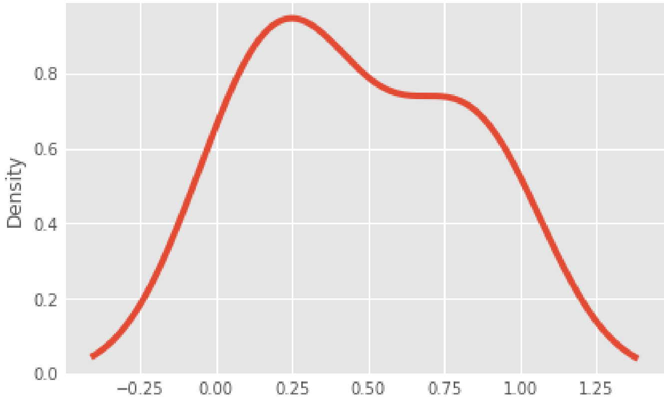
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x120aec610>



Kernel Density Estimation plot (KDE)

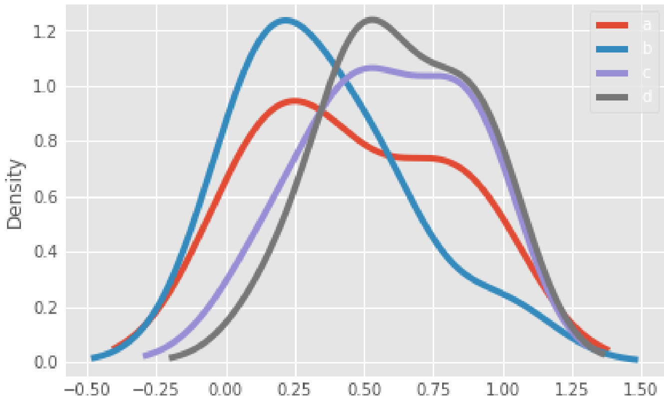
In [24]:
```
df2['a'].plot.kde()
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x120c233d0>



In [25]:
```
df2.plot.density()
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x12099cdd0>



Geographic Plotting with Geopandas

In [1]:
```
#%pip install geopandas
#%pip install descartes

import matplotlib.pyplot as plt
import geopandas as gpd
import pandas as pd

%matplotlib inline
```

In [2]:
```
# https://ourworldindata.org/obesity
df = pd.read_csv('overweight.csv')
df.head()
```

Out[2]:

	Entity	Code	Year	Share of adults that are overweight (%)
0	Afghanistan	AFG	1975	5.3
1	Afghanistan	AFG	1976	5.5
2	Afghanistan	AFG	1977	5.7
3	Afghanistan	AFG	1978	5.9
4	Afghanistan	AFG	1979	6.1

In [3]:
```
columns = list(df.columns)
columns[3] = 'Overweight'
df.columns = columns
df.head()
```

Out[3]:

	Entity	Code	Year	Overweight
0	Afghanistan	AFG	1975	5.3
1	Afghanistan	AFG	1976	5.5
2	Afghanistan	AFG	1977	5.7
3	Afghanistan	AFG	1978	5.9
4	Afghanistan	AFG	1979	6.1

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8316 entries, 0 to 8315
Data columns (total 4 columns):
Entity        8316 non-null object
Code          8022 non-null object
Year          8316 non-null int64
Overweight    8316 non-null float64
dtypes: float64(1), int64(1), object(2)
memory usage: 260.0+ KB
```

In [5]:
```python
shapefile = 'ne_110m_admin_0_countries/ne_110m_admin_0_countries.shp'
gdf = gpd.read_file(shapefile)
gdf.head(2)
```

Out[5]:

	featurecla	scalerank	LABELRANK	SOVEREIGNT	SOV_A3	ADM0_DIF	LEVEL	TYPE	ADMIN	ADM0_A3	...	NAM
0	Admin-0 country	1	6	Fiji	FJI	0	2	Sovereign country	Fiji	FJI	...	
1	Admin-0 country	1	3	United Republic of Tanzania	TZA	0	2	Sovereign country	United Republic of Tanzania	TZA	...	탄

2 rows × 95 columns

In [6]:
```python
gdf = gdf[['ADMIN', 'ADM0_A3', 'geometry']]
gdf.head()
```

Out[6]:

	ADMIN	ADM0_A3	geometry
0	Fiji	FJI	MULTIPOLYGON (((180.00000 -16.06713, 180.00000...
1	United Republic of Tanzania	TZA	POLYGON ((33.90371 -0.95000, 34.07262 -1.05982...
2	Western Sahara	SAH	POLYGON ((-8.66559 27.65643, -8.66512 27.58948...
3	Canada	CAN	MULTIPOLYGON (((-122.84000 49.00000, -122.9742...
4	United States of America	USA	MULTIPOLYGON (((-122.84000 49.00000, -120.0000...

In [7]:
```python
df_2016 = df[df['Year'] == 2016]
```

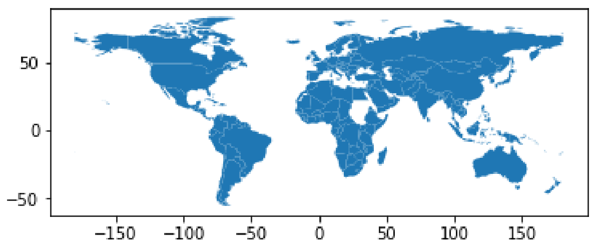
In [8]:
```python
merged = gdf.merge(df_2016, left_on='ADM0_A3', right_on='Code')
merged.head()
```

Out[8]:

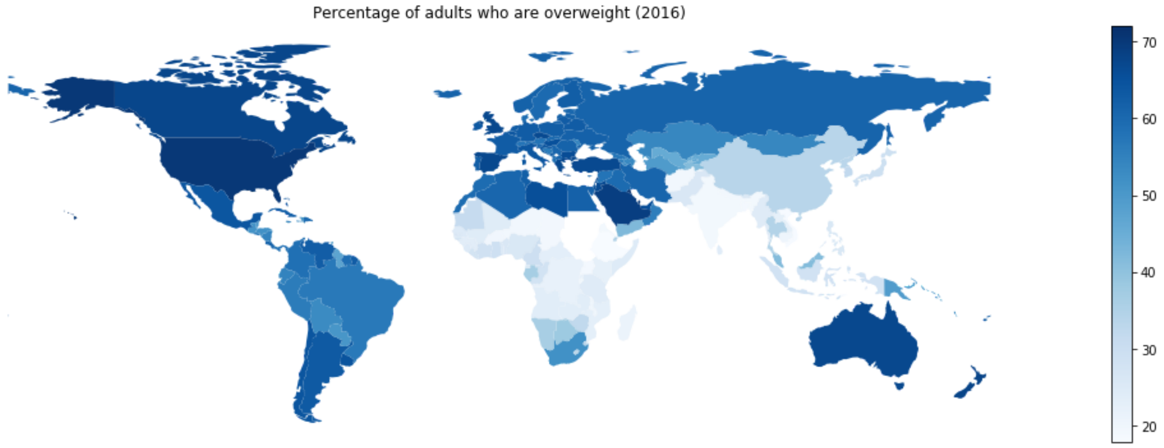
	ADMIN	ADM0_A3	geometry	Entity	Code	Year	Overweight
0	Fiji	FJI	MULTIPOLYGON (((180.00000 -16.06713, 180.00000...	Fiji	FJI	2016	63.4
1	United Republic of Tanzania	TZA	POLYGON ((33.90371 -0.95000, 34.07262 -1.05982...	Tanzania	TZA	2016	24.5
2	Canada	CAN	MULTIPOLYGON (((-122.84000 49.00000, -122.9742...	Canada	CAN	2016	67.5
3	United States of America	USA	MULTIPOLYGON (((-122.84000 49.00000, -120.0000...	United States	USA	2016	70.2
4	Kazakhstan	KAZ	POLYGON ((87.35997 49.21498, 86.59878 48.54918...	Kazakhstan	KAZ	2016	53.9

In [9]:
```python
merged.plot()
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1199ea1d0>



In [10]:
```python
ax = merged.plot(column='Overweight', cmap='Blues', figsize=(20,6), legend=True)
ax.set_title('Percentage of adults who are overweight (2016)')
ax.set_axis_off()
```



London

In [11]:
```python
# https://data.london.gov.uk/dataset/london-borough-profiles
london_data_url = "https://data.london.gov.uk/download/london-borough-profiles/c1693b82-68b1-44ee-beb2-3decf17dc1f8/london-borough-profiles.csv"
london = pd.read_csv(london_data_url, encoding='latin')
london.head()
```

Out[11]:

	Code	Area_name	Inner/_Outer_London	GLA_Population_Estimate_2017	GLA_Household_Estimate_2017	Inland_Area
0	E09000001	City of London	Inner London	8800	5326	
1	E09000002	Barking and Dagenham	Outer London	209000	78188	
2	E09000003	Barnet	Outer London	389600	151423	
3	E09000004	Bexley	Outer London	244300	97736	
4	E09000005	Brent	Outer London	332100	121048	

5 rows × 84 columns

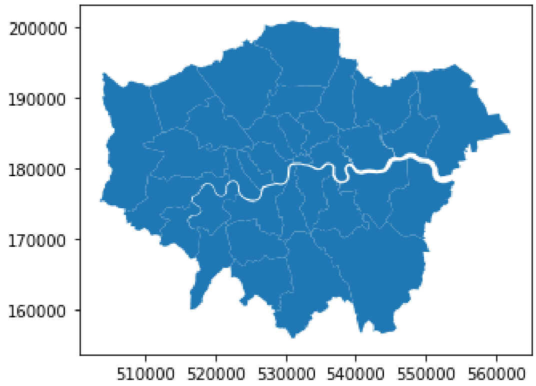
In [12]:
```python
# https://data.london.gov.uk/dataset/statistical-gis-boundary-files-london

london_shapefile = "statistical-gis-boundaries-london/ESRI/London_Borough_Excluding_MHW.shp"
london_gdf = gpd.read_file(london_shapefile)
london_gdf.head()
```

Out[12]:

	NAME	GSS_CODE	HECTARES	NONLD_AREA	ONS_INNER	SUB_2009	SUB_2006	geometry
0	Kingston upon Thames	E09000021	3726.117	0.000	F	None	None	POLYGON ((516401.600 160201.800, 516407.300 16...
1	Croydon	E09000008	8649.441	0.000	F	None	None	POLYGON ((535009.200 159504.700, 535005.500 15...
2	Bromley	E09000006	15013.487	0.000	F	None	None	POLYGON ((540373.600 157530.400, 540361.200 15...
3	Hounslow	E09000018	5658.541	60.755	F	None	None	POLYGON ((521975.800 178100.000, 521967.700 17...
4	Ealing	E09000009	5554.428	0.000	F	None	None	POLYGON ((510253.500 182881.600, 510249.900 18...

In [13]:
```
london_gdf.plot()
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1199e5650>



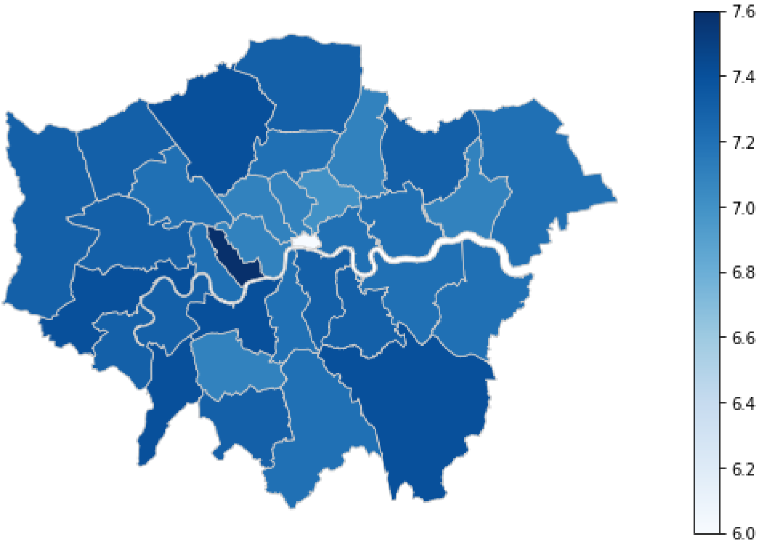
In [14]:
```
merged = london_gdf.merge(london, left_on='GSS_CODE', right_on='Code')
merged.head(2)
```

Out[14]:

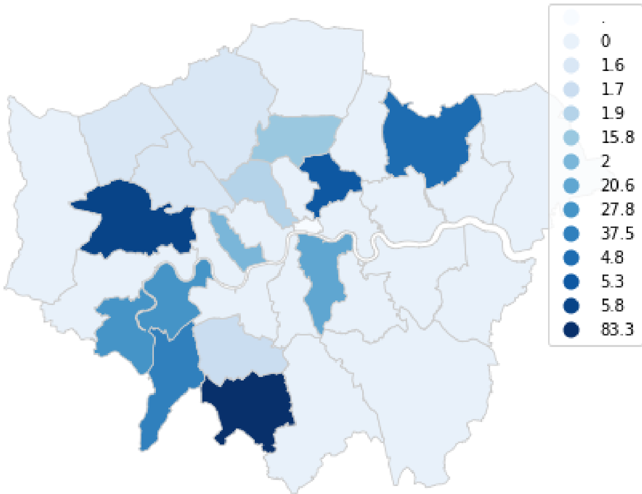
	NAME	GSS_CODE	HECTARES	NONLD_AREA	ONS_INNER	SUB_2009	SUB_2006	geometry	Code	Area_nan
0	Kingston upon Thames	E09000021	3726.117	0.0	F	None	None	POLYGON ((516401.600 160201.800, 516407.300 16...	E09000021	Kingst upo Tham
1	Croydon	E09000008	8649.441	0.0	F	None	None	POLYGON ((535009.200 159504.700, 535005.500 15...	E09000008	Croyd

2 rows × 92 columns

In [15]:
```
variable = 'Happiness_score_2011-14_(out_of_10)'
ax = merged.plot(column=variable, cmap='Blues', figsize=(10, 6),
                 linewidth=0.8, edgecolor='0.8', legend=True)
ax.set_axis_off()
```

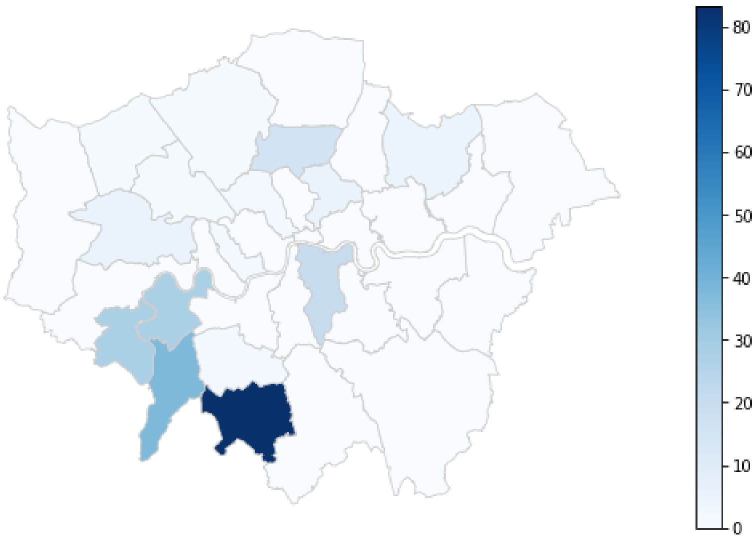


In [16]:
```
variable = 'Proportion_of_seats_won_by_Lib_Dems_in_2014_election'
ax = merged.plot(column=variable, cmap='Blues', figsize=(10, 6),
                 linewidth=0.8, edgecolor='0.8', legend=True)
ax.set_axis_off()
```



In [17]:
```
def to_float(x):
    try:
        return float(x)
    except:
        return 0.0

variable = 'Proportion_of_seats_won_by_Labour_in_2014_election'
merged[variable] = merged[variable].apply(to_float)
variable = 'Proportion_of_seats_won_by_Lib_Dems_in_2014_election'
merged[variable] = merged[variable].apply(to_float)
variable = 'Proportion_of_seats_won_by_Conservatives_in_2014_election'
merged[variable] = merged[variable].apply(to_float)
```

In [18]:
```
variable = 'Proportion_of_seats_won_by_Lib_Dems_in_2014_election'
ax = merged.plot(column=variable, cmap='Blues', figsize=(10, 6),
                 linewidth=0.8, edgecolor='0.8', legend=True)
ax.set_axis_off()
```



In [19]:

```
fig, axes = plt.subplots(ncols=3, figsize=(20,5))

variable = 'Proportion_of_seats_won_by_Labour_in_2014_election'
ax = merged.plot(column=variable, ax=axes[0], cmap='Reds',
                 linewidth=0.8, edgecolor='0.8', legend=True)
ax.set_axis_off()

variable = 'Proportion_of_seats_won_by_Lib_Dems_in_2014_election'
ax = merged.plot(column=variable, ax=axes[1], cmap='Greens',
                 linewidth=0.8, edgecolor='0.8', legend=True)
ax.set_axis_off()

variable = 'Proportion_of_seats_won_by_Conservatives_in_2014_election'
ax = merged.plot(column=variable, ax=axes[2], cmap='Blues',
                 linewidth=0.8, edgecolor='0.8', legend=True)
ax.set_axis_off()
```