Bokeh Tutorial

(http://bokeh.pydata.org/)

07. Bar and Categorical Data Plots

In [1]:

```
from bokeh.io import show, output_notebook
from bokeh.plotting import figure

output_notebook()
```

(https://bokeh.pydata.org) BokehJS 0.12.14 successfully loaded.

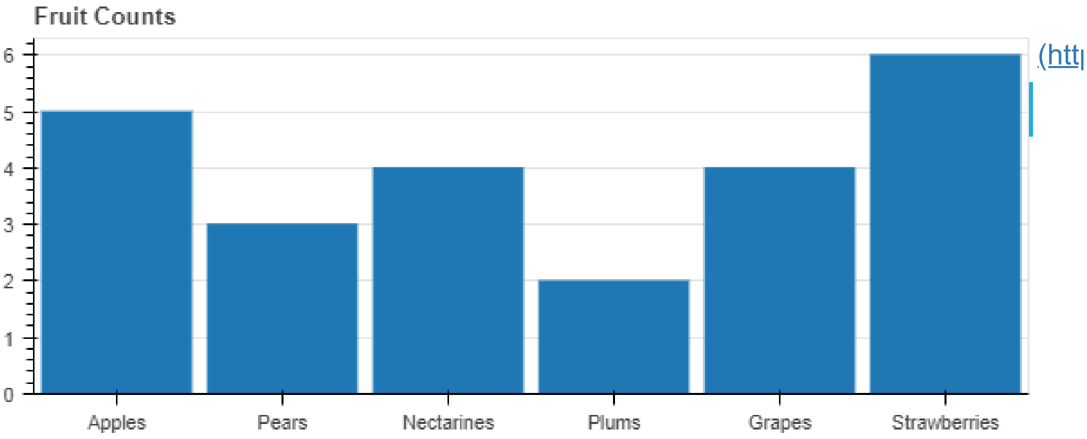
Basic Bar Charts

Bar charts are a common and important type of plot. Bokeh makes it simple to create all sorts of stacked or nested bar charts, and to deal with categorical data in general.

The example below shows a simple bar chart created using the `vbar` method for drawing vertical bars. (There is a corresponding `hbar` for horizontal bars.) We also set a few plot properties to make the chart look nicer, see chapter Styling and Theming (02 - Styling and Theming.ipynb) for information about visual properties.

In [2]:

```
# Here is a list of categorical values (or factors)
fruits = ['Apples', 'Pears', 'Nectarines', 'Plums', 'Grapes', 'Strawberries']

# Set the x_range to the list of categories above
p = figure(x_range=fruits, plot_height=250, title="Fruit Counts")

# Categorical values can also be used as coordinates
p.vbar(x=fruits, top=[5, 3, 4, 2, 4, 6], width=0.9)

# Set some properties to make the plot look better
p.xgrid.grid_line_color = None
p.y_range.start = 0

show(p)
```



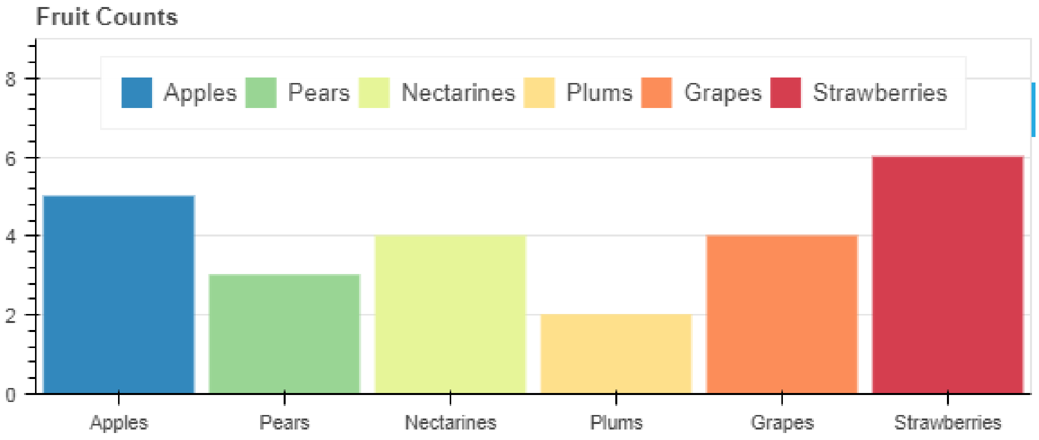
When we want to create a plot with a categorical range, we pass the ordered list of categorical values to `figure` , e.g. `x_range=['a', 'b', 'c']` . In the plot above, we passed the list of fruits as `x_range` , and we can see those refelected as the x-axis.

The `vbar` glyph method takes an `x` location for the center of the bar, a `top` and `bottom` (which defaults to 0), and a `width` . When we are using a categorical range as we are here, each category implicitly has width of 1, so setting `width=0.9` as we have done here makes the bars shrink away from each other. (Another option would be to add some padding to the range.)

In [3]:
```
# Exercise: Create your own simple bar chart
```

Since `vbar` is a glyph method, we can use it with a `ColumnDataSource` just as we woudl with any other glyph. In the example below, we put the data (including color data) in a `ColumnDataSource` and use that to drive our plot. We also add a legend, see chapter Adding Annotations.ipynb (03 - Adding Annotations.ipynb) for more information about legends and other annotations.

In [3]:
```
from bokeh.models import ColumnDataSource
from bokeh.palettes import Spectral6

fruits = ['Apples', 'Pears', 'Nectarines', 'Plums', 'Grapes', 'Strawberries']
counts = [5, 3, 4, 2, 4, 6]

source = ColumnDataSource(data=dict(fruits=fruits, counts=counts, color=Spectral6))

p = figure(x_range=fruits, plot_height=250, y_range=(0, 9), title="Fruit Counts")
p.vbar(x='fruits', top='counts', width=0.9, color='color', legend="fruits", source=source)

p.xgrid.grid_line_color = None
p.legend.orientation = "horizontal"
p.legend.location = "top_center"

show(p)
```



(htt

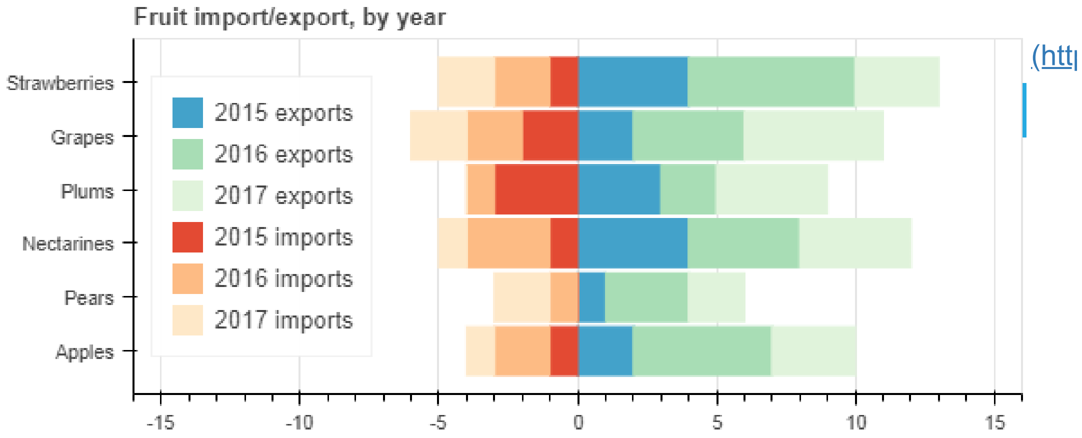
In [5]:
```
# Exercise: Create your own simple bar chart driven by a ColumnDataSource
```

Stacked Bars

It's often

In [4]:

```
from bokeh.palettes import GnBu3, OrRd3

years = ['2015', '2016', '2017']

exports = {'fruits' : fruits,
           '2015'   : [2, 1, 4, 3, 2, 4],
           '2016'   : [5, 3, 4, 2, 4, 6],
           '2017'   : [3, 2, 4, 4, 5, 3]}
imports = {'fruits' : fruits,
           '2015'   : [-1, 0, -1, -3, -2, -1],
           '2016'   : [-2, -1, -3, -1, -2, -2],
           '2017'   : [-1, -2, -1, 0, -2, -2]}

p = figure(y_range=fruits, plot_height=250, x_range=(-16, 16), title="Fruit import/export, b

p.hbar_stack(years, y='fruits', height=0.9, color=GnBu3, source=ColumnDataSource(exports),
             legend=["%s exports" % x for x in years])

p.hbar_stack(years, y='fruits', height=0.9, color=OrRd3, source=ColumnDataSource(imports),
             legend=["%s imports" % x for x in years])

p.y_range.range_padding = 0.1
p.ygrid.grid_line_color = None
p.legend.location = "center_left"

show(p)
```



Notice we also added some padding *around* the categorical range (e.g. at both ends of the axis) by specifying

```
p.y_range.range_padding = 0.1
```

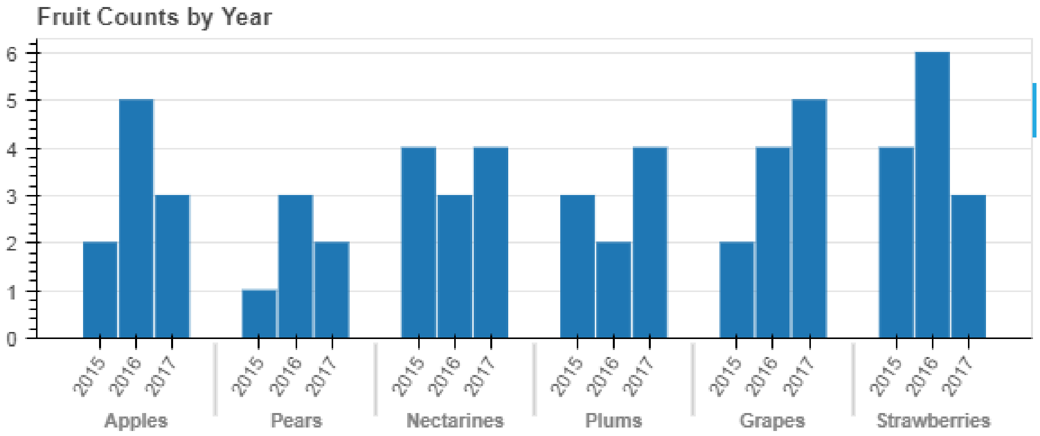
In [7]:

```
# Create a stacked bar chart with a single call to vbar_stack
```

Grouped Bar Charts

In [5]:

```
from bokeh.models import FactorRange

fruits = ['Apples', 'Pears', 'Nectarines', 'Plums', 'Grapes', 'Strawberries']
years = ['2015', '2016', '2017']

data = {'fruits' : fruits,
        '2015'   : [2, 1, 4, 3, 2, 4],
        '2016'   : [5, 3, 3, 2, 4, 6],
        '2017'   : [3, 2, 4, 4, 5, 3]}

# this creates [ ("Apples", "2015"), ("Apples", "2016"), ("Apples", "2017"), ("Pears", "2015
x = [ (fruit, year) for fruit in fruits for year in years ]
counts = sum(zip(data['2015'], data['2016'], data['2017']), ()) # like an hstack

source = ColumnDataSource(data=dict(x=x, counts=counts))

p = figure(x_range=FactorRange(*x), plot_height=250, title="Fruit Counts by Year")

p.vbar(x='x', top='counts', width=0.9, source=source)

p.y_range.start = 0
p.x_range.range_padding = 0.1
p.xaxis.major_label_orientation = 1
p.xgrid.grid_line_color = None

show(p)
```



(htt

In [9]:

```
# Exercise: Make the chart above have a different color for each year by adding colors to th
```

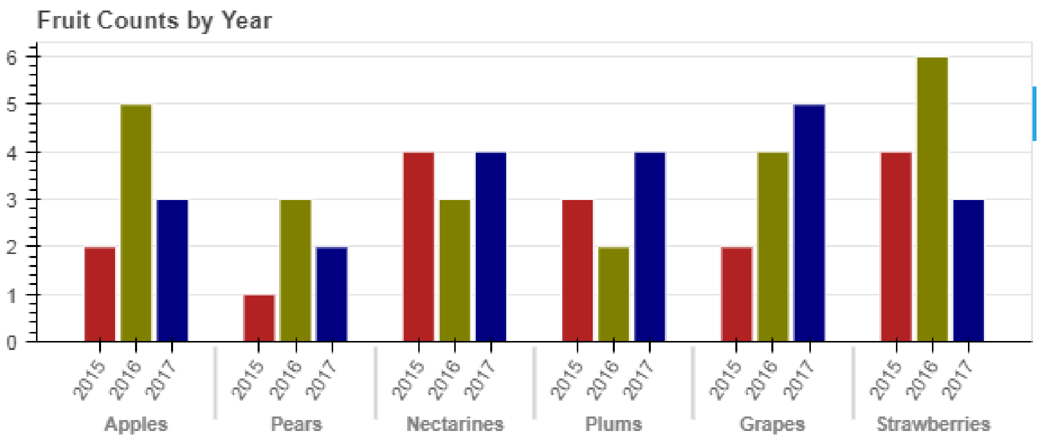
Another way we can set the color of the bars is to use a transorm. We first saw some transforms in previous chapter Data Sources and Transformations (04 - Data Sources and Transformations.ipynb). Here we use a new one `factor_cmap` that accepts a the name of a column to use for colormapping, as well as the palette and factors that define the color mapping.

Additionally we can configure it to map just the sub-factors if desired. For instance in this case we don't want shade each `(fruit, year)` pair differently. Instead, we want to only shade based on the `year`. So we pass `start=1` and `end=2` to specify the slice range of each factor to use when colormapping. Then we pass the result as the `fill_color` value:

```
    fill_color=factor_cmap('x', palette=['firebrick', 'olive', 'navy'], factors=yea
rs, start=1, end=2))
```

to have the colors be applied automatically based on the underlying data.

In [6]:
```
from bokeh.transform import factor_cmap

p = figure(x_range=FactorRange(*x), plot_height=250, title="Fruit Counts by Year")

p.vbar(x='x', top='counts', width=0.9, source=source, line_color="white",

       # use the palette to colormap based on the the x[1:2] values
       fill_color=factor_cmap('x', palette=['firebrick', 'olive', 'navy'], factors=years, st

p.y_range.start = 0
p.x_range.range_padding = 0.1
p.xaxis.major_label_orientation = 1
p.xgrid.grid_line_color = None

show(p)
```



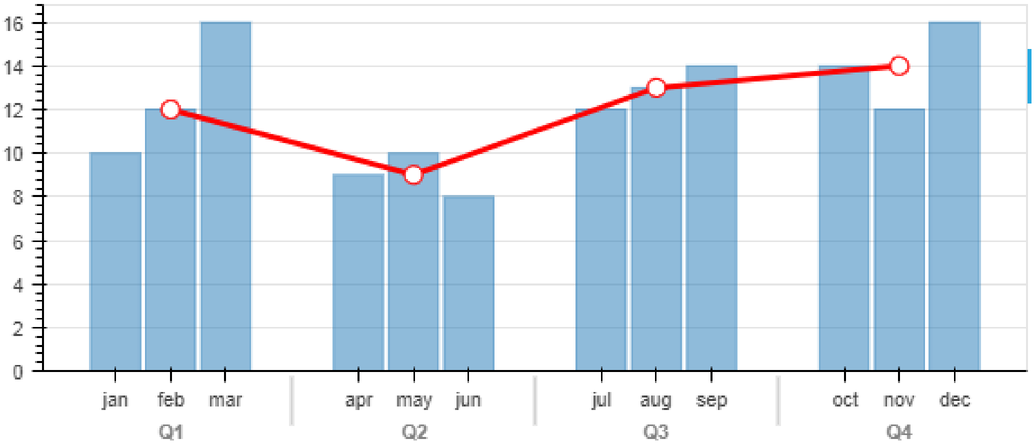
(htt

It is also possible to achieve grouped bar plots using another technique called "visual dodge". That would be useful e.g. if you only wanted to have the axis labeled by fruit type, and not include the years on the axis. This tutorial does not cover that technique but you can find information in the User's Guide (http://bokeh.pydata.org/en/dev/docs/user_guide/categorical.html#visual-dodge).

Mixing Categorical Levels

In [7]:

```
factors = [("Q1", "jan"), ("Q1", "feb"), ("Q1", "mar"),
           ("Q2", "apr"), ("Q2", "may"), ("Q2", "jun"),
           ("Q3", "jul"), ("Q3", "aug"), ("Q3", "sep"),
           ("Q4", "oct"), ("Q4", "nov"), ("Q4", "dec")]

p = figure(x_range=FactorRange(*factors), plot_height=250)

x = [ 10, 12, 16, 9, 10, 8, 12, 13, 14, 14, 12, 16 ]
p.vbar(x=factors, top=x, width=0.9, alpha=0.5)

qs, aves = ["Q1", "Q2", "Q3", "Q4"], [12, 9, 13, 14]
p.line(x=qs, y=aves, color="red", line_width=3)
p.circle(x=qs, y=aves, line_color="red", fill_color="white", size=10)

p.y_range.start = 0
p.x_range.range_padding = 0.1
p.xgrid.grid_line_color = None

show(p)
```



(htt

Using Pandas GroupBy

In [8]:

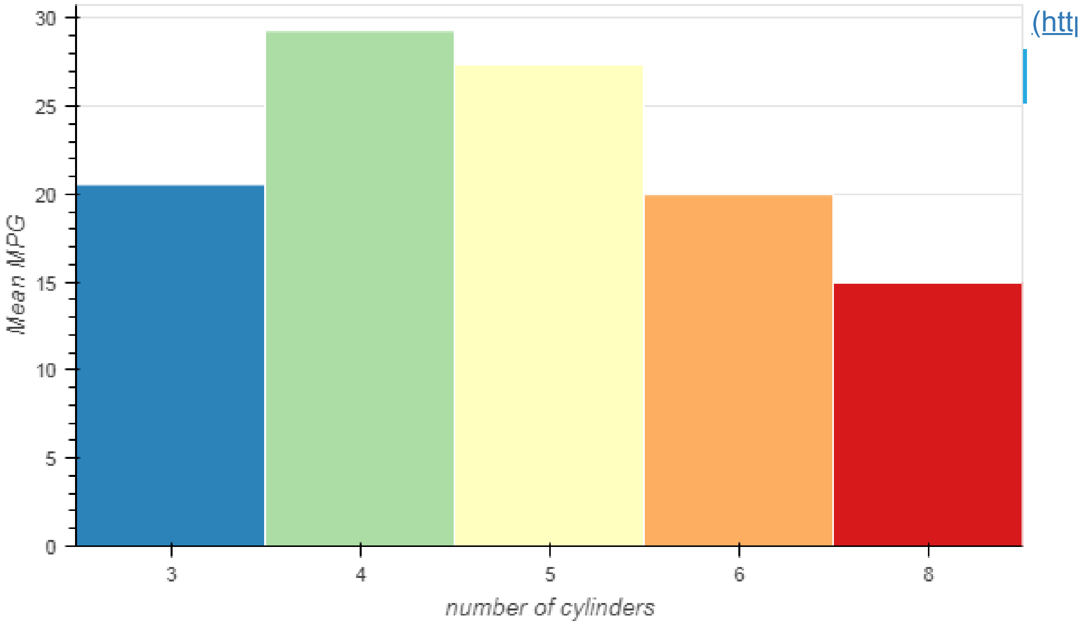
```
from bokeh.sampledata.autompg import autompg_clean as df

df.cyl = df.cyl.astype(str)
df.head()
```

Out[8]:

	mpg	cyl	displ	hp	weight	accel	yr	origin	name	mfr
0	18.0	8	307.0	130	3504	12.0	70	North America	chevrolet chevelle malibu	chevrolet
1	15.0	8	350.0	165	3693	11.5	70	North America	buick skylark 320	buick
2	18.0	8	318.0	150	3436	11.0	70	North America	plymouth satellite	plymouth
3	16.0	8	304.0	150	3433	12.0	70	North America	amc rebel sst	amc
4	17.0	8	302.0	140	3449	10.5	70	North America	ford torino	ford

In [9]:

```
from bokeh.palettes import Spectral5


group = df.groupby(('cyl'))

source = ColumnDataSource(group)
cyl_cmap = factor_cmap('cyl', palette=Spectral5, factors=sorted(df.cyl.unique()))

p = figure(plot_height=350, x_range=group)
p.vbar(x='cyl', top='mpg_mean', width=1, line_color="white",
       fill_color=cyl_cmap, source=source)

p.xgrid.grid_line_color = None
p.xaxis.axis_label = "number of cylinders"
p.yaxis.axis_label = "Mean MPG"
p.y_range.start = 0

show(p)
```



In [14]:

```
# Exercise: Use the same dataset to make a similar plot of mean horsepower (hp) by origin
```

Catgorical Scatterplots