In [1]:

```
import pandas as pd
import numpy as np
import plotly
plotly.tools.set_credentials_file(username='jz2673', api_key='sbHDKL7fQjBk08bBmOLG'
```

In [3]:

```
covtype = pd.read_csv('Data/covtype.csv')
```

In [22]:

```
N = 7
c = ['hsl('+str(h)+',50%'+',50%)' for h in np.linspace(0, 360, 8)]
```

Boxplot: Elevation distribution of different tree types

In [330]:

```
typename = ['Spruce/Fir','Lodgepole Pine','Ponderosa Pine','Cottonwood/Willow','Aspe
data = [{
    'name': typename[i],
    'y':covtype.Elevation[covtype.Cover_Type==(i+1)],
    'type':'box',
    'marker':{'color': c[i]}
    } for i in range(N)]

# format the layout
layout = {'xaxis': {'showgrid':False,'zeroline':False, 'tickangle':60,'showticklabel
          'yaxis': {'zeroline':False,'gridcolor':'white'},
          'paper_bgcolor': 'rgb(233,233,233)',
          'plot_bgcolor': 'rgb(233,233,233)',
          }

py.iplot(data)
```

```
//anaconda/lib/python3.5/site-packages/plotly/plotly/plotly.py:236: Us
erWarning:

Woah there! Look at all those points! Due to browser limitations, the
Plotly SVG drawing functions have a hard time graphing more than 500k
data points for line charts, or 40k points for other types of charts.
Here are some suggestions:
(1) Use the `plotly.graph_objs.Scattergl` trace object to generate a W
ebGl graph.
(2) Trying using the image API to return an image instead of a graph U
RL
(3) Use matplotlib
(4) See if you can create your visualization with fewer data points

If the visualization you're using aggregates points (e.g., box plot, h
istogram, etc.) you can disregard this warning.
```

Out[330]:

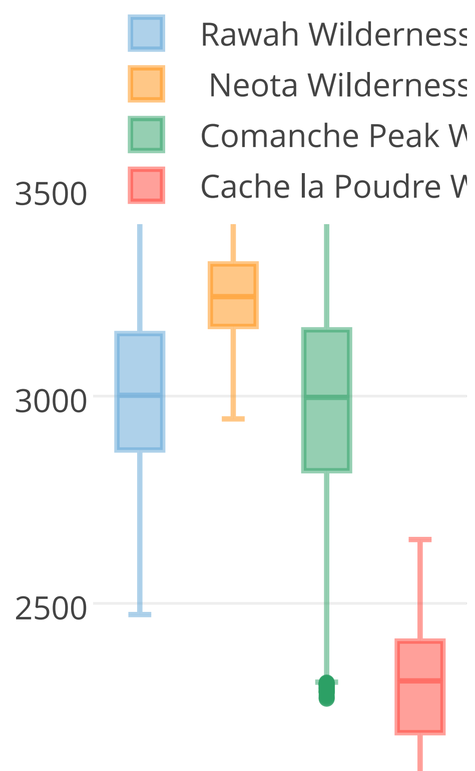
Boxplot: Elevation distribution of different Wilderness area

In [32]:

```
areaname = ['Rawah Wilderness Area','Neota Wilderness Area','Comanche Peak Wildernes
N = len(areaname)
colors = ['rgba(93, 164, 214, 0.5)', 'rgba(255, 144, 14, 0.5)', 'rgba(44, 160, 101,
data = [{
    'name': areaname[i],
    'y': covtype.Elevation[covtype['Wilderness_Area{}'.format(i+1)]==1],
    'type':'box',
    'marker':{'color': colors[i]}
    } for i in range(N)]

# format the layout
layout = {'xaxis': {'showgrid':False,'zeroline':False, 'tickangle':60,'showticklabel
          'yaxis': {'zeroline':False,'gridcolor':'white'},
          'paper_bgcolor': 'rgb(233,233,233)',
          'plot_bgcolor': 'rgb(233,233,233)',
          }

py.iplot(data)
```

```
//anaconda/lib/python3.5/site-packages/plotly/plotly/plotly.py:236: Us
erWarning:

Woah there! Look at all those points! Due to browser limitations, the
Plotly SVG drawing functions have a hard time graphing more than 500k
data points for line charts, or 40k points for other types of charts.
Here are some suggestions:
(1) Use the `plotly.graph_objs.Scattergl` trace object to generate a W
ebGl graph.
(2) Trying using the image API to return an image instead of a graph U
RL
(3) Use matplotlib
(4) See if you can create your visualization with fewer data points

If the visualization you're using aggregates points (e.g., box plot, h
istogram, etc.) you can disregard this warning.
```

```
Out[32]:
```



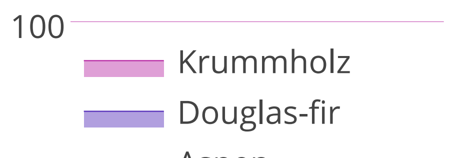
EDIT CHART

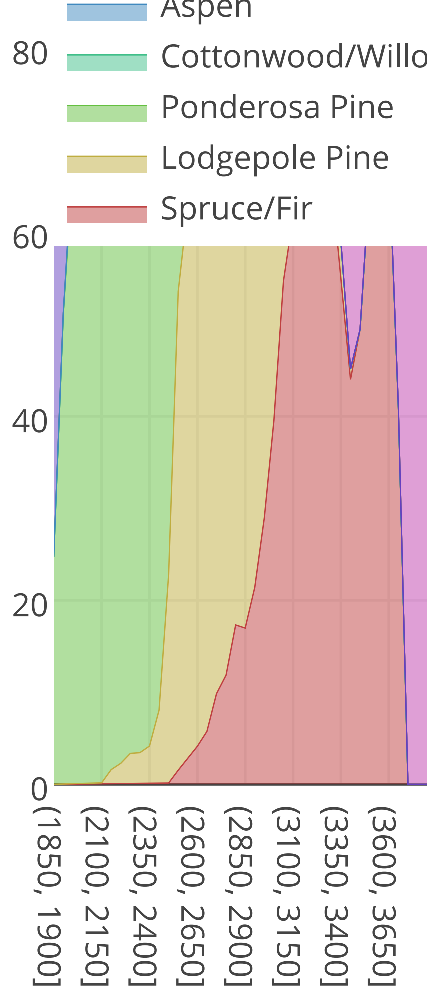
For each wilderness area, how are cover types distributed

In [132]:

```
N = len(typename)
type_by_area = covtype.groupby('Cover_Type').sum().iloc[:,10:14]
yyy = np.round_(np.array(type_by_area)/(np.array(type_by_area).sum(0))*100,1)
yyyy = np.cumsum(np.array(type_by_area)/(np.array(type_by_area).sum(0)),axis=0)
data = [{
        'x': areaname,
        'y': yyyy[i],
        'text':['{}%'.format(x) for x in list(yyy[i])],
        'name':typename[i],
        'mode':'lines',
        'line':{'width':0.5,'color':c[i]},
        'fill':'tonexty'} for i in range(N)]

layout = go.Layout(
    showlegend=True,
    xaxis=dict(
        type='category',
    ),
    yaxis=dict(
        type='linear',
        range=[0, 1],
        dtick=0.2
    )
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='tacked-area-plot-hover')
```

Out[132]:





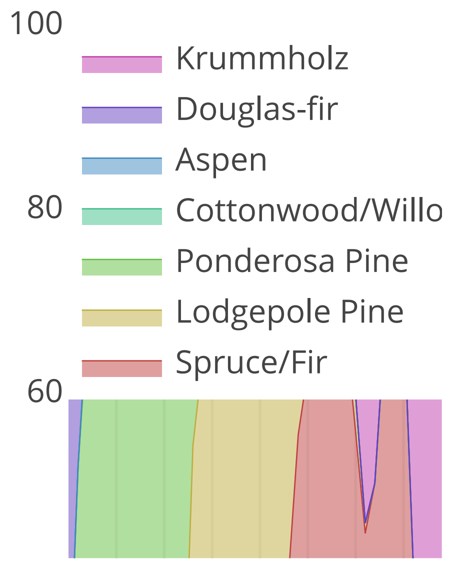
EDIT CHART

By Elevation

```
In [225]:
```

```
area_by_elevation = covtype.groupby(pd.cut(covtype['Elevation'], np.arange(1850,390
area_by_elevation.shape
# N = len(typename)
# type_by_area = covtype.groupby('Cover_Type').sum().iloc[:,10:14]
abe_value= np.round_(np.array(area_by_elevation)/(np.array(area_by_elevation).sum(1
abe_accu = np.cumsum(abe_value,axis=1)
data = [{
        'x': area_by_elevation.index,
        'y': abe_accu.T[i],
        'text':['{}%'.format(x) for x in list(abe_value.T[i])],
        'name':areaname[i],
        'mode':'lines',
        'line':{'width':0.5,'color':colors[i]},
        'fill':'tonexty'} for i in range(4)]

layout = go.Layout(
    showlegend=True,
    xaxis=dict(
        type='category',
    ),
    yaxis=dict(
        type='linear',
        range=[0, 100],
        dtick=20,
        ticksuffix='%'
    )
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='tacked-area-plot-hover')
```

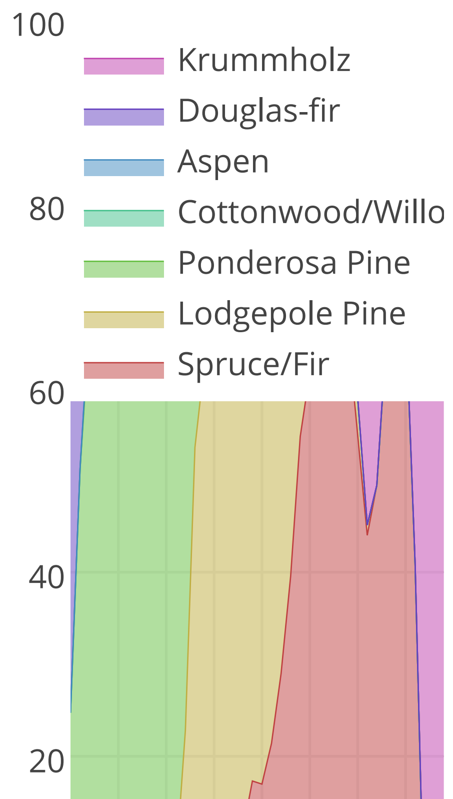
```
Out[225]:
```

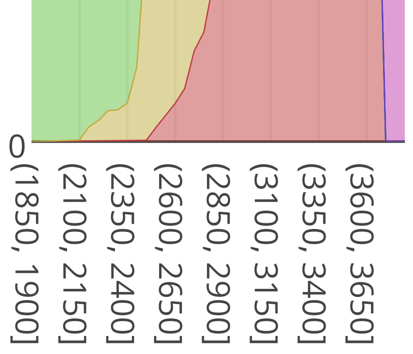


```
In [230]:
```

```
covtype_dum = pd.concat([pd.get_dummies(covtype['Cover_Type']),covtype], axis=1)

type_by_elevation = covtype_dum .groupby([pd.cut(covtype['Elevation'], np.arange(18
tbe_value= np.round_(np.array(type_by_elevation)/(np.array(type_by_elevation).sum(1
tbe_accu = np.cumsum(tbe_value,axis=1)
data = [{
        'x': type_by_elevation.index,
        'y': tbe_accu.T[i],
        'text':['{}%'.format(x) for x in list(tbe_value.T[i])],
        'hoverinfo':'x+text+name',
        'name':typename[i],
        'mode':'lines',
        'line':{'width':0.5,'color':c[i]},
        'fill':'tonexty'} for i in range(7)]

layout = go.Layout(
    showlegend=True,
    xaxis=dict(
        type='category',
    ),
    yaxis=dict(
        type='linear',
        range=[0, 100],
        dtick=20
    )
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='tacked-area-plot-hover')
```

Out[230]:





Soil Information got from : https://archive.ics.uci.edu/ml/machine-learning-databases/covtype/covtype.info (https://archive.ics.uci.edu/ml/machine-learning-databases/covtype/covtype.info)

In [285]:

```
soil = pd.read_csv('Data/covtype_add.csv')

soil.columns=['soil','code','desc']
soil['climatic_zone'] = soil.code.astype('str').str[0]
soil['geologic zones'] =  soil.code.astype('str').str[1]

covtype['soil']=np.argmax(np.array(covtype.iloc[:,14:54]),1)+1

soil_d = pd.concat([soil.drop(['climatic_zone','geologic zones','desc','code'],1),pd

covtype2 = pd.merge(covtype,soil_d,on='soil',how='left')
covtype2 = covtype2.drop(['soil'],1)
```

In [329]:

```
from sklearn.ensemble import RandomForestClassifier
df = covtype2
df['is_train'] = np.random.uniform(0, 1, len(df)) <= .75
df.Cover_Type = pd.Categorical(df.Cover_Type)
train, test = df[df['is_train']==True], df[df['is_train']==False]

features = df.drop('Cover_Type',1).columns
clf = RandomForestClassifier()
clf.fit(train[features], train['Cover_Type'])
preds =clf.predict(test[features])
# pd.crosstab(test['Cover_Type'], preds, rownames=['actual'], colnames=['preds'])




features0 = df.columns[:54]
clf0 = RandomForestClassifier()
clf0.fit(train[features0], train['Cover_Type'])
preds0 =clf0.predict(test[features0])

features1 = df.drop(['Soil_Type1', 'Soil_Type2', 'Soil_Type3', 'Soil_Type4', 'Soil_
       'Soil_Type6', 'Soil_Type7', 'Soil_Type8', 'Soil_Type9', 'Soil_Type10',
       'Soil_Type11', 'Soil_Type12', 'Soil_Type13', 'Soil_Type14',
       'Soil_Type15', 'Soil_Type16', 'Soil_Type17', 'Soil_Type18',
       'Soil_Type19', 'Soil_Type20', 'Soil_Type21', 'Soil_Type22',
       'Soil_Type23', 'Soil_Type24', 'Soil_Type25', 'Soil_Type26',
       'Soil_Type27', 'Soil_Type28', 'Soil_Type29', 'Soil_Type30',
       'Soil_Type31', 'Soil_Type32', 'Soil_Type33', 'Soil_Type34',
       'Soil_Type35', 'Soil_Type36', 'Soil_Type37', 'Soil_Type38',
       'Soil_Type39', 'Soil_Type40','Cover_Type'],1).columns
clf1 = RandomForestClassifier()
clf1.fit(train[features1], train['Cover_Type'])
preds1 =clf1.predict(test[features1])

[accuracy_score(test['Cover_Type'],preds0),accuracy_score(test['Cover_Type'],preds),
```

Out[329]:

```
[0.9413368774058491 5, 0.93863061996019803, 0.93968420110316142]
```

In [324]:

```
df.drop(['Soil_Type1', 'Soil_Type2', 'Soil_Type3', 'Soil_Type4', 'Soil_Type5',
       'Soil_Type6', 'Soil_Type7', 'Soil_Type8', 'Soil_Type9', 'Soil_Type10',
       'Soil_Type11', 'Soil_Type12', 'Soil_Type13', 'Soil_Type14',
       'Soil_Type15', 'Soil_Type16', 'Soil_Type17', 'Soil_Type18',
       'Soil_Type19', 'Soil_Type20', 'Soil_Type21', 'Soil_Type22',
       'Soil_Type23', 'Soil_Type24', 'Soil_Type25', 'Soil_Type26',
       'Soil_Type27', 'Soil_Type28', 'Soil_Type29', 'Soil_Type30',
       'Soil_Type31', 'Soil_Type32', 'Soil_Type33', 'Soil_Type34',
       'Soil_Type35', 'Soil_Type36', 'Soil_Type37', 'Soil_Type38',
       'Soil_Type39', 'Soil_Type40','Cover_Type'],1)
```

Out[324]:

```
Index(['Elevation', 'Aspect', 'Slope', 'Horizontal_Distance_To_Hydrolo
gy',
       'Vertical_Distance_To_Hydrology', 'Horizontal_Distance_To_Roadw
ays',
       'Hillshade_9am', 'Hillshade_Noon', 'Hillshade_3pm',
       'Horizontal_Distance_To_Fire_Points', 'Wilderness_Area1',
       'Wilderness_Area2', 'Wilderness_Area3', 'Wilderness_Area4',
       'Soil_Type1', 'Soil_Type2', 'Soil_Type3', 'Soil_Type4', 'Soil_T
ype5',
       'Soil_Type6', 'Soil_Type7', 'Soil_Type8', 'Soil_Type9', 'Soil_T
ype10',
       'Soil_Type11', 'Soil_Type12', 'Soil_Type13', 'Soil_Type14',
       'Soil_Type15', 'Soil_Type16', 'Soil_Type17', 'Soil_Type18',
       'Soil_Type19', 'Soil_Type20', 'Soil_Type21', 'Soil_Type22',
       'Soil_Type23', 'Soil_Type24', 'Soil_Type25', 'Soil_Type26',
       'Soil_Type27', 'Soil_Type28', 'Soil_Type29', 'Soil_Type30',
       'Soil_Type31', 'Soil_Type32', 'Soil_Type33', 'Soil_Type34',
       'Soil_Type35', 'Soil_Type36', 'Soil_Type37', 'Soil_Type38',
       'Soil_Type39', 'Soil_Type40', 'Cover_Type', 'climatic_zone_2',
       'climatic_zone_3', 'climatic_zone_4', 'climatic_zone_5',
       'climatic_zone_6', 'climatic_zone_7', 'climatic_zone_8',
       'geologic zones_1', 'geologic zones_2', 'geologic zones_5',
       'geologic zones_7', 'is_train'],
      dtype='object')
```

In [318]:

```
from sklearn.metrics import accuracy_score
accuracy_score(test['Cover_Type'],preds)
```

Out[318]:

```
0.93893434719122792
```

In [319]:

```
features0 = df.columns[:54]
clf0 = RandomForestClassifier()
clf0.fit(train[features0], train['Cover_Type'])

preds0 =clf0.predict(test[features0])

pd.crosstab(test['Cover_Type'], preds, rownames=['actual'], colnames=['preds'])
```

Out[319]:

preds	1	2	3	4	5	6	7
actual							
1	50028	2821	1	0	24	9	118
2	2913	67415	136	1	103	97	19
3	2	205	8463	37	6	187	0
4	0	0	113	532	0	19	0
5	63	582	32	0	1729	8	0
6	5	156	464	14	10	3704	0
7	344	43	0	0	0	0	4785

In [320]:

```
accuracy_score(test['Cover_Type'],preds)
```

Out[320]:

```
0.9412348127944587
```

In [ ]: