Spark Developer Training - 3 Days

Manaranjan Pradhan
manaranjan@enablecloud.com
This notebook is given as part of Spark Training to Participants. Forwarding others is strictly prohibited.

Lab: Visualizations, Statistics & Machine Learning in Spark

Things to learn

- Integrating with matplotlib and seaborn libraries
- Creating bar plot and charts
- Creating distribution plots
- Comparing distribution plots
- Creating correlation plots
- Inferring insights using basic Statistics
- Machine learning - Predictive Analytics

In [1]:

```
sc
```

Out[1]:

```
<pyspark.context.SparkContext at 0x7f72dc234240>
```

In [2]:

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
```

Read the data from csv file

In [3]:

```
sheart = sqlContext.read.format("com.databricks.spark.csv").options(
    delimiter=',',
    header  = True ,
    inferSchema = True).load('file:///home/hadoop/lab/data/SAheart.data')
```

Dataset Description

A retrospective sample of males in a heart-disease high-risk region of the Western Cape, South Africa. There are roughly two controls per case of CHD. Many of the CHD positive men have undergone blood pressure reduction treatment and other programs to reduce their risk factors after their CHD event. In some cases the measurements were made after these treatments. These data are taken from a larger dataset, described in Rousseauw et al, 1983, South African Medical Journal.

- sbp - systolic blood pressure
- tobacco - cumulative tobacco (kg)
- ldl - low densiity lipoprotein cholesterol
- adiposity
- famhist - family history of heart disease (Present, Absent)
- typea - type-A behavior
- obesity
- alcohol - current alcohol consumption
- age - age at onset
- chd - response, coronary heart disease

This dataset is taken from http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/ (http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/)

In [4]:

```
sheart.show( 10 )
```

```
+---------+---+-------+----+---------+-------+-----+-------+-------+---
+---+
|row.names|sbp|tobacco| ldl|adiposity|famhist|typea|obesity|alcohol|age
|chd|
+---------+---+-------+----+---------+-------+-----+-------+-------+---
+---+
|        1|160|   12.0|5.73|    23.11|Present|   49|   25.3|   97.2| 52
|  1|
|        2|144|   0.01|4.41|    28.61| Absent|   55|  28.87|   2.06| 63
|  1|
|        3|118|   0.08|3.48|    32.28|Present|   52|  29.14|   3.81| 46
|  0|
|        4|170|    7.5|6.41|    38.03|Present|   51|  31.99|  24.26| 58
|  1|
|        5|134|   13.6| 3.5|    27.78|Present|   60|  25.99|  57.34| 49
|  1|
|        6|132|    6.2|6.47|    36.21|Present|   62|  30.77|  14.14| 45
|  0|
|        7|142|   4.05|3.38|     16.2| Absent|   59|  20.81|   2.62| 38
|  0|
|        8|114|   4.08|4.59|     14.6|Present|   62|  23.11|   6.72| 58
|  1|
|        9|114|    0.0|3.83|     19.4|Present|   49|  24.86|   2.49| 29
|  0|
|       10|132|    0.0| 5.8|    30.96|Present|   69|  30.11|    0.0| 53
|  1|
+---------+---+-------+----+---------+-------+-----+-------+-------+---
+---+
only showing top 10 rows
```

Findout relationship between family history and coronary heart disease

In [5]:

```
chd_count = sheart.groupBy( 'famhist', 'chd' ).count()
```

In [6]:

```
chd_count.show()
```

```
+-------+---+-----+
|famhist|chd|count|
+-------+---+-----+
| Absent|  0|  206|
| Absent|  1|   64|
|Present|  0|   96|
|Present|  1|   96|
+-------+---+-----+
```

In [7]:

```
chd_count_pd = chd_count.toPandas()
```

In [8]:

```
chd_count_pd
```

Out[8]:

	famhist	chd	count
0	Absent	0	206
1	Absent	1	64
2	Present	0	96
3	Present	1	96

In [9]:

```
import matplotlib.pyplot as plt
import seaborn as sn
%matplotlib inline
```

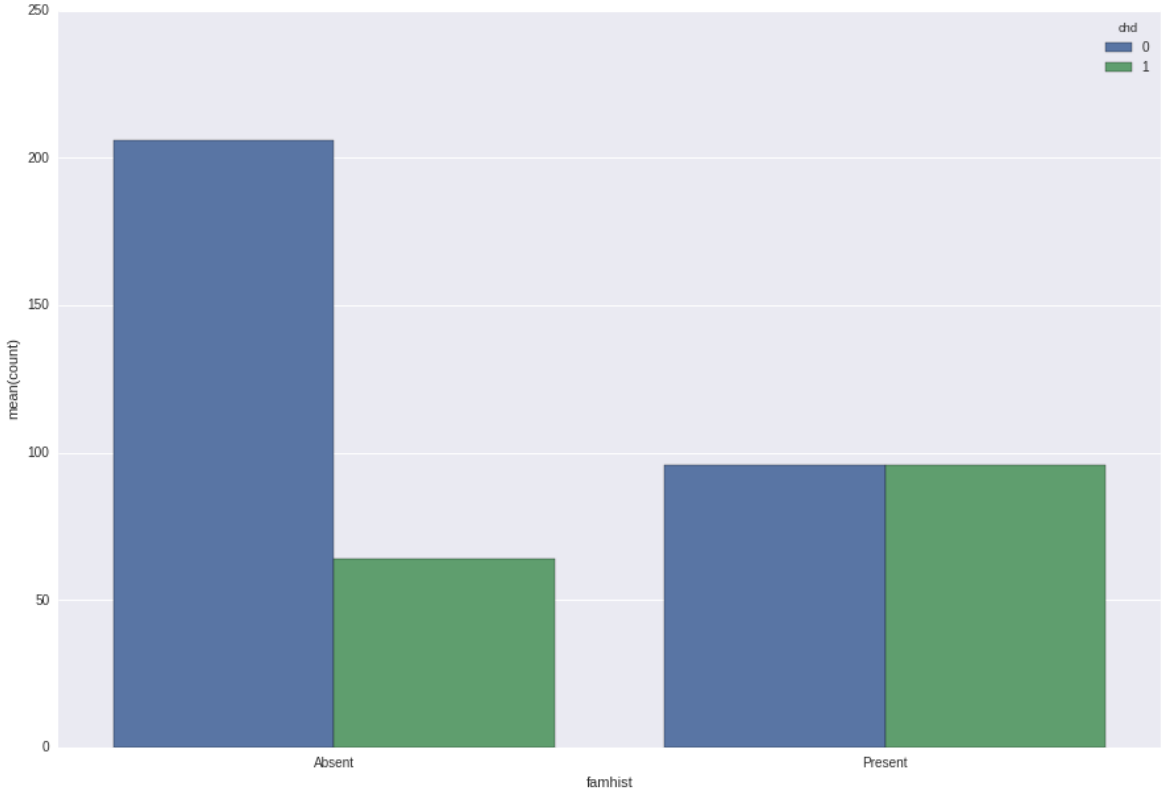
```
:0: FutureWarning: IPython widgets are experimental and may change in t
he future.
```

In [10]:

```
plt.figure(figsize=(15, 10))

sn.barplot( y='count',
            x = 'famhist',
            hue = 'chd',
            data = chd_count_pd )
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f72aab5ac50>
```



Finding relationship between alcohol consumption and coronary heart disease

In [11]:

```
chd_alcohol_pd = sheart.select( sheart['alcohol'],
                                sheart['chd'] ).toPandas()
```

In [12]:

```
chd_alcohol_pd.head()
```

Out[12]:

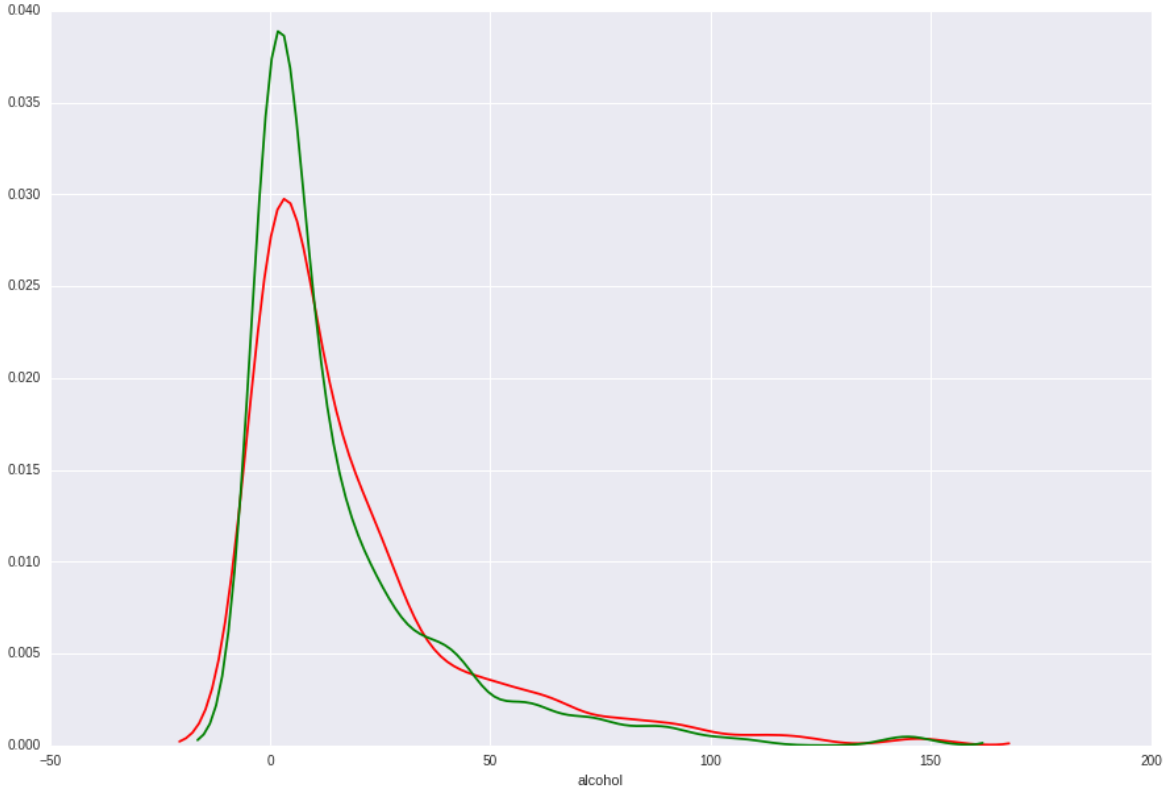
	alcohol	chd
0	97.20	1
1	2.06	1
2	3.81	0
3	24.26	1
4	57.34	1

In [13]:

```
plt.figure(figsize=(15, 10))

sn.distplot( chd_alcohol_pd[chd_alcohol_pd.chd == 1].alcohol,
             hist = False,
             color = 'r' )

sn.distplot( chd_alcohol_pd[chd_alcohol_pd.chd == 0].alcohol,
             hist = False,
             color = 'g' )
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f72aab18160>
```



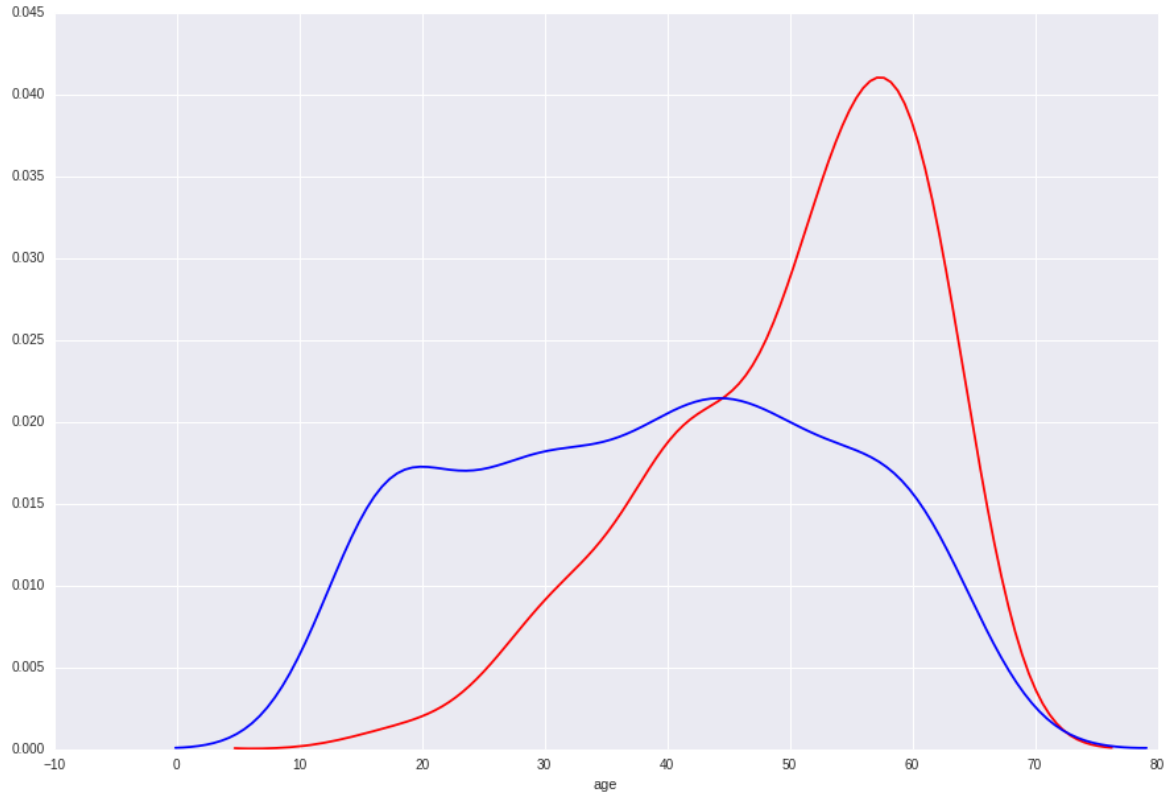
Relationship between age and coronary heart disease

In [14]:

```
chd_age_pd = sheart.select( sheart['age'], sheart['chd'] ).toPandas()

plt.figure(figsize=(15, 10))

sn.distplot( chd_age_pd[chd_age_pd.chd == 1].age,
             hist = False, color = 'r' )

sn.distplot( chd_age_pd[chd_age_pd.chd == 0].age,
             hist = False, color = 'b' )
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f72aa76e198>
```



Find correations between variables

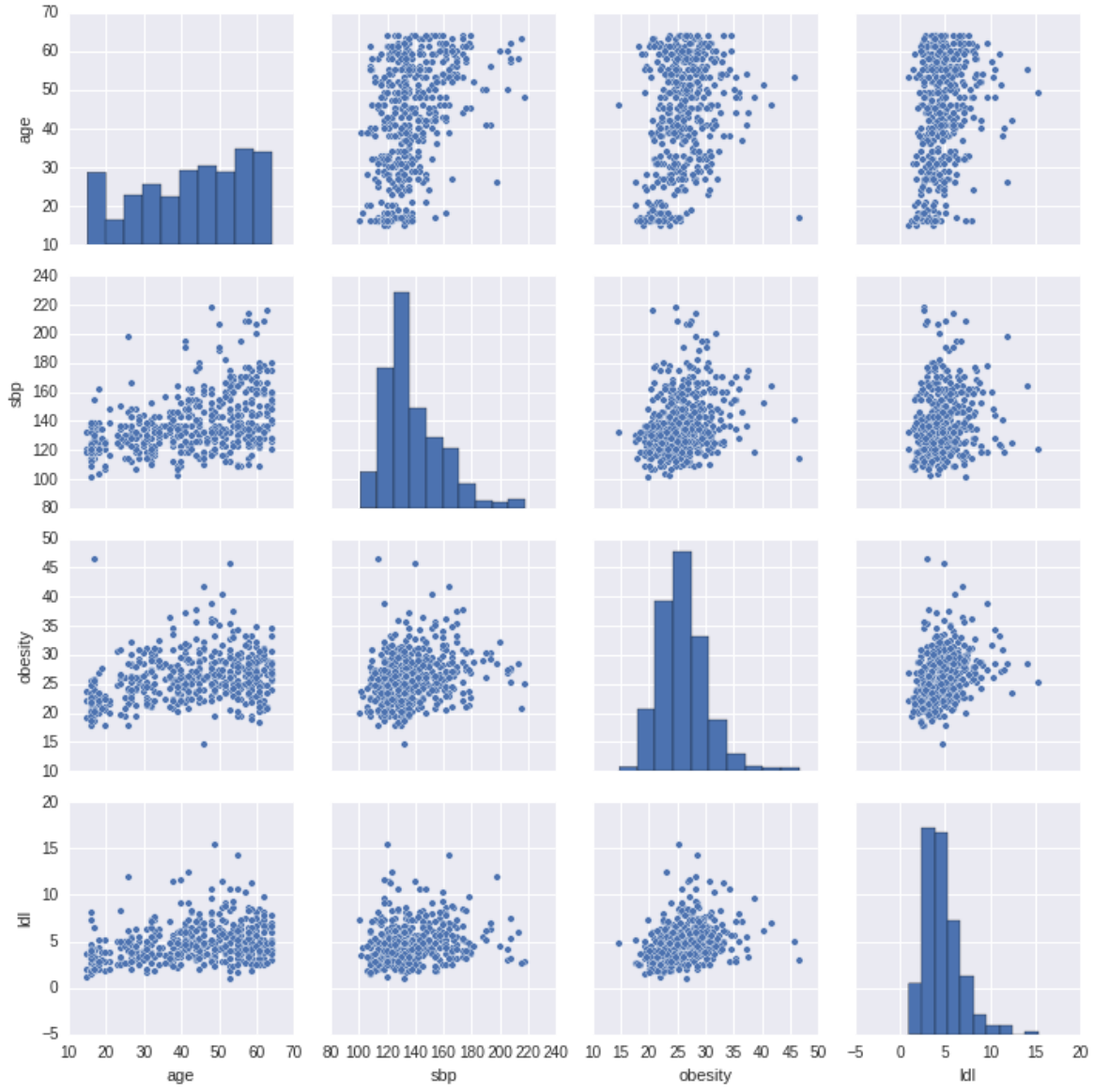
In [15]:

```
chd_pair_pd = sheart.select( sheart['age'],
                             sheart['sbp'],
                             sheart['obesity'],
                             sheart['ldl'] ).toPandas()

plt.figure(figsize=(15, 10))

sn.pairplot( chd_pair_pd )
```

Out[15]:

<seaborn.axisgrid.PairGrid at 0x7f72aa9f0e10>

<matplotlib.figure.Figure at 0x7f72aa9f0cc0>



Find relationship between obesity and ldl

In [16]:

```
plt.figure(figsize=(15, 10))

sn.regplot(y="ldl", x="obesity", data=chd_pair_pd)
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f72aa198eb8>
```

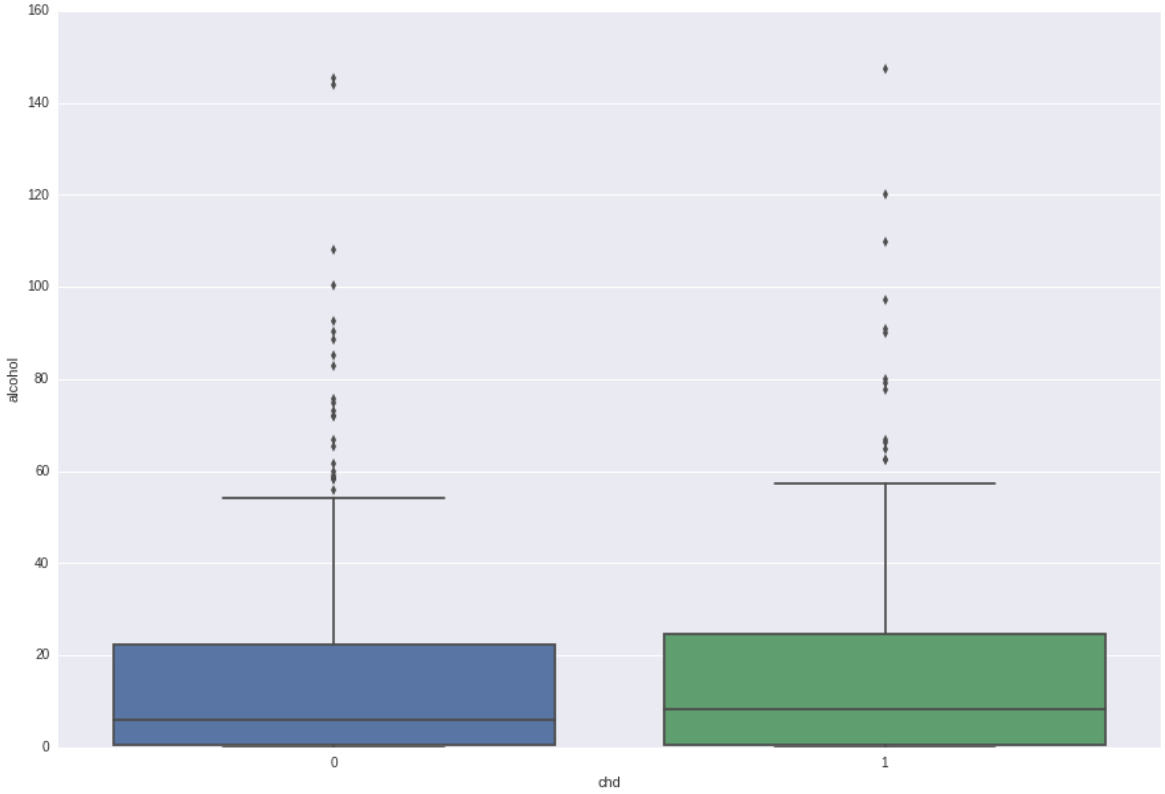
Drawing boxplots to understand distributions

In [17]:

```
chd_alh_tob_pd = sheart.select( sheart['alcohol'],
                                sheart['tobacco'],
                                sheart['chd'] ).toPandas()
plt.figure(figsize=(15, 10))
sn.boxplot(y="alcohol", x="chd", data=chd_alh_tob_pd)
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f72aa2d8470>
```

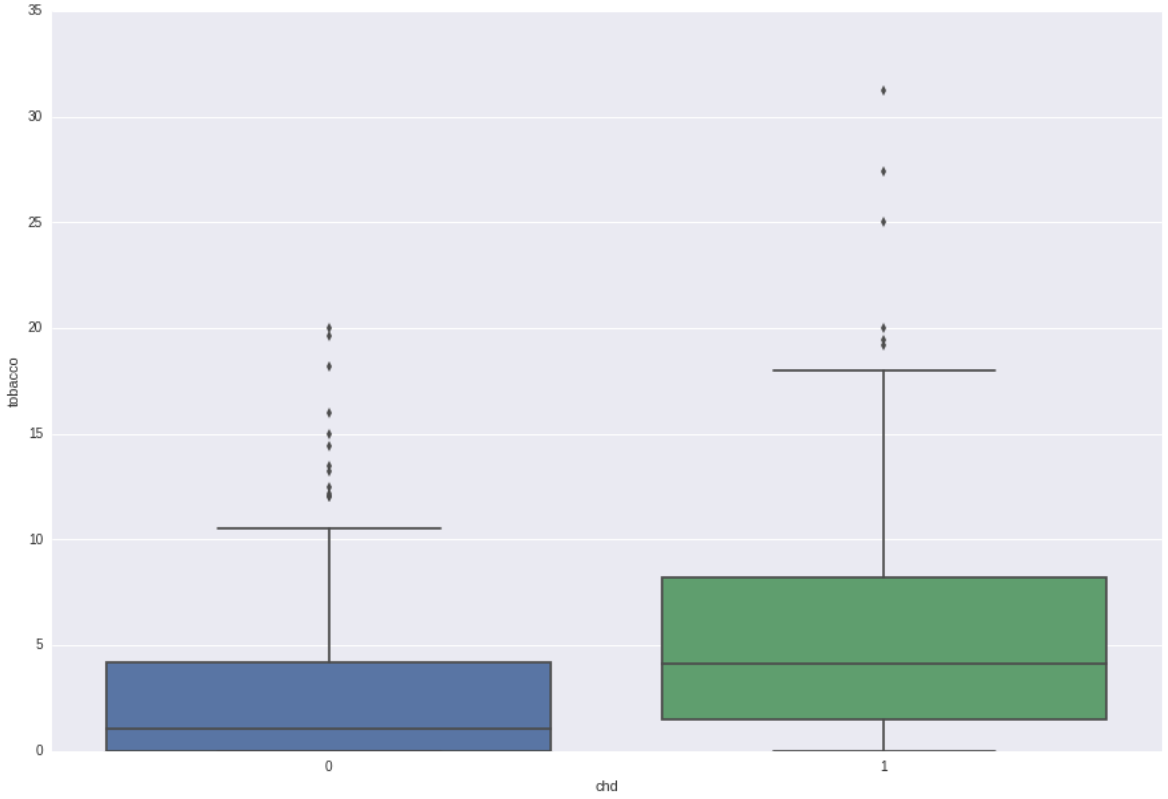


In [18]:

```
chd_alh_tob_pd = sheart.select( sheart['alcohol'],
                                sheart['tobacco'],
                                sheart['chd'] ).toPandas()
plt.figure(figsize=(15, 10))
sn.boxplot(y="tobacco", x="chd", data=chd_alh_tob_pd)
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f72a803f9e8>
```



Calculating basic statistics

In [19]:

```
from pyspark.mllib.stat import Statistics
from pyspark.mllib.linalg import Vectors
from pyspark.mllib.regression import LabeledPoint
```

In [20]:

```
def getVector( rec ):
        return Vectors.dense(rec.alcohol,
                             rec.tobacco,
                             rec.age,
                             rec.obesity,
                             rec.ldl)
```

In [21]:

```
chd_vec = sheart.map( lambda rec: getVector( rec ) )
```

In [22]:

```
summary = Statistics.colStats( chd_vec )
```

In [23]:

```
summary.mean()
```

Out[23]:

```
array([ 17.04439394,   3.63564935,  42.81601732,  26.04411255,   4.7403
2468])
```

In [24]:

```
summary.variance()
```

Out[24]:

```
array([ 599.32223466,   21.09587018,  213.4216084 ,   17.75510105,
          4.28866475])
```

Calcuating correlations

In [25]:

```
import numpy as np
np.sqrt( summary.variance() )
```

Out[25]:

```
array([ 24.48105869,   4.59302408,  14.60895644,   4.21368023,   2.0709
0916])
```

In [26]:

```
seriesX = sheart.select( sheart["obesity"] )
seriesY = sheart.select( sheart["ldl"] )
```

In [27]:

```
seriesX
```

Out[27]:

```
DataFrame[obesity: double]
```

In [28]:

```
correlation = Statistics.corr(chd_vec, method="pearson")
```

In [29]:

```
correlation
```

Out[29]:

```
array([[ 1.        ,  0.20081339,  0.10112465,  0.05161957, -0.0334034
],
       [ 0.20081339,  1.        ,  0.45033016,  0.12452941,  0.1589054
6],
       [ 0.10112465,  0.45033016,  1.        ,  0.29177713,  0.3117992
3],
       [ 0.05161957,  0.12452941,  0.29177713,  1.        ,  0.3305058
6],
       [-0.0334034 ,  0.15890546,  0.31179923,  0.33050586,  1.
]])
```

Creating vectors to represent multidimensional data

- Using only continuous variables
- Buidling predictive model using only alcohol, tobacoo consumption, age, obesity and ldl

In [30]:

```
def parsePoint(rec):
    return LabeledPoint( rec.chd, Vectors.dense(rec.alcohol,
                                                rec.tobacco,
                                                rec.age,
                                                rec.obesity,
                                                rec.ldl))
```

In [31]:

```
chd_lp = sheart.map( lambda rec: parsePoint( rec ) )
```

In [32]:

```
chd_lp.take( 10 )
```

Out[32]:

```
[LabeledPoint(1.0, [97.2,12.0,52.0,25.3,5.73]),
 LabeledPoint(1.0, [2.06,0.01,63.0,28.87,4.41]),
 LabeledPoint(0.0, [3.81,0.08,46.0,29.14,3.48]),
 LabeledPoint(1.0, [24.26,7.5,58.0,31.99,6.41]),
 LabeledPoint(1.0, [57.34,13.6,49.0,25.99,3.5]),
 LabeledPoint(0.0, [14.14,6.2,45.0,30.77,6.47]),
 LabeledPoint(0.0, [2.62,4.05,38.0,20.81,3.38]),
 LabeledPoint(1.0, [6.72,4.08,58.0,23.11,4.59]),
 LabeledPoint(0.0, [2.49,0.0,29.0,24.86,3.83]),
 LabeledPoint(1.0, [0.0,0.0,53.0,30.11,5.8])]
```

Building a predictive model using Logistic Regression

In [33]:

```
from pyspark.mllib.classification import LogisticRegressionWithLBFGS,    \
                                         LogisticRegressionModel
```

In [34]:

```
model = LogisticRegressionWithLBFGS.train( chd_lp )
```

Making predictions using the predictive model

In [35]:

```
labelsAndPreds = chd_lp.map(lambda lp: ( lp.label,
                                         float(model.predict(lp.features))))
```

In [36]:

```
labelsAndPreds.take( 10 )
```

Out[36]:

```
[(1.0, 1.0),
 (1.0, 0.0),
 (0.0, 0.0),
 (1.0, 0.0),
 (1.0, 1.0),
 (0.0, 0.0),
 (0.0, 0.0),
 (1.0, 1.0),
 (0.0, 0.0),
 (1.0, 0.0)]
```

In [37]:

```
total_count = labelsAndPreds.count()
success_count = labelsAndPreds.filter(lambda rec: rec[0] == rec[1]).count()
```

Calculating the accuracy of the model

In [38]:

```
print("Successful prediction percentage: " +
      str( round( success_count / total_count, 2 ) ) )
```

```
Successful prediction percentage: 0.71
```

Adding more variables to the model prediction

- Adding a catogorical variable - family history
- Adiposity

- sbp
- typea

In [39]:

```
from pyspark.ml.feature import OneHotEncoder, StringIndexer
```

Encoding the categorical variable

In [40]:

```
month_stringIndexer = StringIndexer(inputCol="famhist",
                                    outputCol="famhistIndex")

month_model = month_stringIndexer.fit(sheart)

month_indexed = month_model.transform(sheart)

month_encoder = OneHotEncoder(dropLast=False,
                              inputCol="famhistIndex",
                              outputCol="famhistVec")

traindata_final = month_encoder.transform(month_indexed)
```

In [41]:

```
traindata_final.show( 5 )
```

```
+---------+---+-------+----+---------+-------+-----+-------+-------+---
+---+------------+-------------+
|row.names|sbp|tobacco| ldl|adiposity|famhist|typea|obesity|alcohol|age
|chd|famhistIndex|   famhistVec|
+---------+---+-------+----+---------+-------+-----+-------+-------+---
+---+------------+-------------+
|        1|160|   12.0|5.73|    23.11|Present|   49|   25.3|   97.2| 52
|  1|         1.0|(2,[1],[1.0])|
|        2|144|   0.01|4.41|    28.61| Absent|   55|  28.87|   2.06| 63
|  1|         0.0|(2,[0],[1.0])|
|        3|118|   0.08|3.48|    32.28|Present|   52|  29.14|   3.81| 46
|  0|         1.0|(2,[1],[1.0])|
|        4|170|    7.5|6.41|    38.03|Present|   51|  31.99|  24.26| 58
|  1|         1.0|(2,[1],[1.0])|
|        5|134|   13.6| 3.5|    27.78|Present|   60|  25.99|  57.34| 49
|  1|         1.0|(2,[1],[1.0])|
+---------+---+-------+----+---------+-------+-----+-------+-------+---
+---+------------+-------------+
only showing top 5 rows
```

Creating the vectors again

In [42]:

```
def parseNewPoint(rec):
    return LabeledPoint( rec.chd,
            Vectors.dense(tuple( [rec.sbp,
                                  rec.tobacco,
                                  rec.ldl,
                                  rec.adiposity,
                                  rec.typea,
                                  rec.obesity,
                                  rec.alcohol,
                                  rec.age] +
                                  rec.famhistVec.toArray().tolist() ) ) )
```

In [43]:

```
chd_lp_new = traindata_final.map( lambda rec: parseNewPoint( rec ) )
```

In [44]:

```
chd_lp_new.take( 5 )
```

Out[44]:

```
[LabeledPoint(1.0, [160.0,12.0,5.73,23.11,49.0,25.3,97.2,52.0,0.0,1.
0]),
 LabeledPoint(1.0, [144.0,0.01,4.41,28.61,55.0,28.87,2.06,63.0,1.0,0.
0]),
 LabeledPoint(0.0, [118.0,0.08,3.48,32.28,52.0,29.14,3.81,46.0,0.0,1.
0]),
 LabeledPoint(1.0, [170.0,7.5,6.41,38.03,51.0,31.99,24.26,58.0,0.0,1.
0]),
 LabeledPoint(1.0, [134.0,13.6,3.5,27.78,60.0,25.99,57.34,49.0,0.0,1.
0])]
```

Building the model and making predictions

In [45]:

```
model = LogisticRegressionWithLBFGS.train( chd_lp_new )

labelsAndPreds_new = chd_lp_new.map(lambda lp: ( lp.label,
                                                 float(model.predict(lp.features))))

success_count_new = labelsAndPreds_new.filter(lambda rec:
                                              rec[0] == rec[1]).count()

print("Successful prediction percentage: " +
      str( round( success_count_new / total_count, 2 ) ) )
```

```
Successful prediction percentage: 0.74
```

Make a note of lessons you learnt in the lab