Sawtooth Software Labels File Reference

The Sawtooth Software Labels File is an XML file that describes data in a Comma Separated Values (CSV) file. XML files are text files that can be easily read and modified by programs. They can also be read and modified by humans using a simple text editor or an XML viewer or editor.

Document Overview

This document is a reference that describes each of the elements in the Sawtooth Software Labels File. If you are not familiar with CSV or XML files, you may want to read Appendix C (A Brief Introduction to XML) first. If you would like to see complete examples, please see Appendix A (A Demographic Variables Example) and Appendix B (A Sample Utilities File).

Labels File Reference

We will now walk through the Sawtooth Software Labels File XML document and define each of the available tags and attributes that are used to describe data files.

Each tag will show a table of attributes and a table of sequential child elements that are supported for that tag. The child elements must appear in the order that they are shown in the table. If an element is not going to be included it can be skipped, but the other elements must be in the defined order.

XML Declaration

The first line of the Labels File is the XML declaration. Generally it will not need to change.

```
<?xml version="1.0" encoding="utf-8"?>
```

sawtoothSoftwareLayout Element

Parent Element

None. This element must be the root element.

Attributes

Attribute	Required	Data Type	Description
creator	Optional	Text	Describes who or what program created the Labels File

Sequential Child Elements

Element	Number of Occurrences
segmentationVariables	0 or 1
utilities	0 or more

The sawtoothSoftwareLayout tag is used to define the root element of the Labels File. Each Labels File must contain one and only one of these tags as shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<sawtoothSoftwareLayout>
    ...
</sawtoothSoftwareLayout>
```

All of the remaining elements and attributes will be nested within this root element. The optional creator attribute can be included for informational purposes. It is never used by Sawtooth Software products when reading the file. If a labels file is created by SSI Web, it will use this attribute to specify the version of SSI Web that created the file. When creating labels files on your own, you can just leave this attribute out.

Sawtooth Software CSV data files can potentially contain demographic variables, utilities, or both. If demographic variables are in the file they will be described in the segmentationVariables element. This element may be left out if there are no demographic variables. Zero, one, or more sets of utilities can also be contained in the data file, and therefore the labels file. Each set of utilities must be contained in a utilities element. The segmentationVariables element (if present) must be the first child element followed by any utilities elements that may exist.

Example

A sample labels file for a CSV that only contains demographic variables will look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<sawtoothSoftwareLayout>
    <segmentationVariables>
        ...
    </segmentationVariables>
        ...
</sawtoothSoftwareLayout>
```

segmentationVariables Element

Parent Element

sawtoothSoftwareLayout

Attributes

None.

Sequential Child Elements

Element	Number of Occurrences
variable	0 or more

The segmentationVariables element contains one variable element for each variable.

Example

A sample labels file that contains three demographic variables will have a segmentationVariables element that looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<sawtoothSoftwareLayout>
    <segmentationVariables>
        <variable>
            ...
        </variable>
        <variable>
            ...
        </variable>
        <variable>
            ...
        </variable>
    </segmentationVariables>
</sawtoothSoftwareLayout>
```

variable Element

Parent Element

segmentationVariables

Attributes

Attribute	Required	Data Type	Description
column	Optional	Positive integer	The CSV column number that contains this variable's data
isRespondentId	Optional	"true" or "false"	Specifies if this variable is the respondent id variable
variableType	Optional	"coded", "number", or "text"	Defines the type of the data in the CSV file
skip	Optional	"true" or "false"	If set to true, Sawtooth Software products will ignore this variable (it will not be imported)

Sequential Child Elements

Element	Number of Occurrences
variableLabel	0 or 1
valueLabel	0 or more

The variable element describes a specific variable.

The optional column attribute specifies the column number in the CSV file that contains the data for the variable. If the column attribute is not specified, then the variable elements will be matched to the CSV columns in order.

The optional isRespondentId attribute specifies if this variable is *the* respondent id variable. Only one respondent id variable is allowed per data file. If none of the variables have this attribute specified, then the first variable will be treated as the respondent id variable.

The optional variableType attribute specifies what kind of variable it is. The value of this attribute must be one of the following values:

Attribute Value	Description
coded	The variable's data are coded (i.e. categorical data)
number	The variable's data are numeric
text	The variable's data are text (i.e. open-ended responses)

The variable element can optionally contain a variableLabel element followed by zero, one, or more valueLabel elements.

Example

A sample labels file that contains a coded age variable will have a variable element that looks something like this:

```
...
```

```
    <segmentationVariables>
        ...
        <variable column="3" variableType="coded">
            <variableLabel>...</variableLabel>
            <valueLabel>...</valueLabel>
        </variable>
        ...
    </segmentationVariables>
...
```

variableLabel Element

Parent Element
[variable]

Attributes
None.

Sequential Child Elements
None.

A `variableLabel` element contains the variable’s label.

Example
A sample labels file that contains an age variable could define the variable’s label like this:

```
...
    <variable>
        ...
        <variableLabel>Respondent’s Age</variableLabel>
        ...
    </variable>
...
```

valueLabel Element

Parent Element

variable

Attributes

None.

Sequential Child Elements

Element	Number of Occurrences
integerValue	Exactly once
label	Exactly once

The valueLabel element associates a label with a value for a coded variable.

Example

A sample labels file that has a coded age variable with two different age categories would have valueLabel elements like this:

```
...
    <variable>
        ...
        <valueLabel>
            ...
        </valueLabel>
        <valueLabel>
            ...
        </valueLabel>
    </variable>
...
```

integerValue Element

Parent Element
`valueLabel`

Attributes
None.

Sequential Child Elements
None.

The `integerValue` element contains an integer that represents a coded value.

Example
A sample labels file that contains a coded variable, age, with a coded value of 1 for respondents under the age of 50 would contain a `valueLabel` element similar to this:

```
...
    <valueLabel>
        <integerValue>1</integerValue>
        ...
    </valueLabel>
...
```

label Element

Parent Elements

`valueLabel`, `fit`, `attribute`, `level`, and `none`

Attributes

None.

Sequential Child Elements

None.

The `label` element contains the text of a label. This element is used in many different elements throughout the Labels File (See the Parent Elements section above).

Example

A sample labels file that contains a coded variable, age, with a coded value of 1 for the respondents under the age of 50 would contain a `label` element similar to this:

```
...
    <valueLabel>
        ...
        <label>Under 50 Years Old</label>
    </valueLabel>
...
```

utilities Element

Parent Element

sawtoothSoftwareLayout

Attributes

None.

Sequential Child Elements

Element	Number of Occurrences
runName	Exactly once
fit	0 or 1
attribute	0 or more
interaction	0 or more
none	0 or more

The `utilities` element describes a set of utilities. It must contain a `runName` element.

Example

A sample labels file that contains one set of utilities about Hotels with a fit statistic and utilities for three attributes (with no "interaction" or "none" utilities) will have a `utilities` element that looks similar to this:

```
<?xml version="1.0" encoding="utf-8"?>
<sawtoothSoftwareLayout>
    <utilities>
        <runName>
            Hotel Utilities
        </runName>
        <fit>
            ...
        </fit>
        <attribute>
            ...
        </attribute>
        <attribute>
            ...
        </attribute>
        <attribute>
            ...
        </attribute>
    </utilities>
</sawtoothSoftwareLayout>
```

runName Element

Parent Element
utilities

Attributes
None.

Sequential Child Elements
None.

The runName element can contain a user-friendly name for the set of utilities. Sawtooth Software products that support multiple sets of utilities will use this name to help differentiate them.

Example
A sample labels file that contains utilities about Hotels may have a runName element like this:

```
...
    <utilities>
        <runName>
            Hotel Utilities
        </runName>
        ...
    </utilities>
...
```

fit Element

Parent Element

utilities

Attributes

Attribute	Required	Data Type	Description
column	Required	Positive integer	The CSV column number that contains the fit statistic for this set of utilities

Sequential Child Elements

Element	Number of Occurrences
label	0 or 1

If a set of utilities has a fit statistic, then the fit element must be included. Its required column attribute tells which column number contains the fit data. The optional label element defines a label to be used for the fit.

Example

If a data file had a set of utilities with a fit statistic in the fifth column and if you wanted to label it with "RLH," then the following elements would be used in the Labels File:

```
...
    <utilities>
        <runName>
            My Utilities
        </runName>
        <fit column="5">
            <label>RLH</label>
        </fit>
        ...
    </utilities>
...
```

attribute Element

Parent Element
utilities

Attributes

Attribute	Required	Data Type	Description
number	Required	Positive integer	The attribute number. All attributes in the utilities must be numbered from one to the number of attributes.
coding	Required	“partworth”, “linear”, or “excluded”	This describes how the attribute was coded. Attributes coded as “excluded” will be ignored by Sawtooth Software products.
attributeType	Optional	“number” or “text”	This attribute describes whether the attribute has continuous values (number) or not (text).
column	Optional	Positive integer	The column number that contains the utility value for this attribute (linear attributes only).

Sequential Child Elements

Element	Number of Occurrences
label	0 or 1
level	0 or more

The attribute element defines each of the attributes in the project. The number attribute defines the order of the attributes within the project. All attribute elements must be numbered from one to the number of attributes in the project. If the attribute is linear, then it will only have one column of data in the CSV file and that column number must be specified using the column attribute. If the attribute is partworth, then the column attribute should *not* be used. Partworth attributes will have a column with utility values for each level and therefore a level element must be defined for each level of the attribute. If an attribute has continuous values (which allows users to interpolate between level values), then the attributeType attribute should be “number.”

Example
A Labels File for a data set with a linear price attribute and a partworth shape attribute would look something like this:

```
...
<utilities>
    <runName>
        Shape Utilities
    </runName>
    <attribute number="1" coding="linear" attributeType="number" column="2">
        <label>Price</label>
    </attribute>
    <attribute number="2" coding="partworth">
        <label>Shape</label>
        <level>
```

```
            ...
        </level>
        <level>
            ...
        </level>
        <level>
            ...
        </level>
    </attribute>
    ...
</utilities>
...
```

level Element

Parent Element

attribute

Attributes

Attribute	Required	Data Type	Description
number	Required	Positive integer	This is the level number. All levels in each attribute must be numbered from one to the number of levels in that attribute.
column	Optional	Positive integer	This is the column number that contains the utility value for this level.

Sequential Child Elements

Element	Number of Occurrences
label	0 or 1
value	0 or 1

Each level of an attribute has a `level` element. It has a `number` attribute to define the order of the levels within the attribute as well as an optional `column` attribute that specifies which column contains the utility values for this level in the data file.

The `level` element may also optionally contain a `label` element to specify a user friendly label for the level and/or a `value` element to specify a value for this level to be used in simulations.

Example

A shape attribute with three levels may look similar to this in the labels file:

```
...
<attribute number="1" coding="partworth" attributeType="text">
    <label>
        Shape
    </label>
    <level number="1" column="2">
        <label>Circle</label>
    </level>
    <level number="2" column="3">
        <label>Triangle</label>
    </level>
    <level number="3" column="4">
        <label>Square</label>
    </level>
</attribute>
...
```

value Element

Parent Element

level

Attributes

None.

Sequential Child Elements

None.

The value element contains a decimal number that represents the value of a level. This value is used by simulators.

Example

A partworth price attribute may include level values like this so that the user can enter actual prices (that are interpolated) when defining products in a Sawtooth Software simulator. Its labels file would look like the following example. (Level values do not necessarily have to match the label of the level, but they should represent ascending or descending continuous values if you are going to allow interpolation in simulators for the attribute.)

```
...
<attribute number="1" coding="partworth" attributeType="number">
    <label>
        Price
    </label>
    <level number="1" column="2">
        <label>$1.99</label>
        <value>1.99</value>
    </level>
    <level number="2" column="3">
        <label>$2.99</label>
        <value>2.99</value>
    </level>
    <level number="3" column="4">
        <label>$3.99</label>
        <value>3.99</value>
    </level>
</attribute>
...
```

interaction Element

Parent Element

utilities

Attributes

Attribute	Required	Data Type	Description
attribute1	Required	Positive integer	This is the number of the first attribute that is involved in this interaction.
attribute2	Required	Positive integer	This is the number of the second attribute that is involved in this interaction.

Sequential Child Elements

Element	Number of Occurrences
term	1 or more

An interaction results in a column of utilities for each of the level pairs of the two attributes involved in the interaction. Each of these columns of utilities for the level pairs needs to be represented by a term element. The attribute numbers in the attribute1 and attribute2 attributes need to be the same numbers as the number attribute for the respective attributes (defined in attribute elements).

Example

A sample interaction between two attributes (shape and size) each with two levels (circle and square, and small and large respectively) would look similar to this in the labels file:

```
...
<interaction attribute1="1" attribute2="2">
    <term attribute1level="1" attribute2level="1" column="8"/>
    <term attribute1level="1" attribute2level="2" column="9"/>
    <term attribute1level="2" attribute2level="1" column="10"/>
    <term attribute1level="2" attribute2level="2" column="11"/>
</interaction>
...
```

term Element

Parent Element

interaction

Attributes

Attribute	Required	Data Type	Description
`attribute1level`	Required	Positive integer	This is the level number of the first attribute involved in this interaction that is represented by this term.
`attribute2level`	Required	Positive integer	This is the level number of the second attribute involved in this interaction that is represented by this term.
`column`	Required	Positive integer	This is the column number of the column in the data file that contains the utility data for this attribute level pair.

Sequential Child Elements

None.

An interaction results in a utility value for each level pair of the two attributes involved. For example, an interaction between a size attribute with two levels (small and large) and a shape attribute with two levels (circle and triangle) will have four columns (one for small circle, one for large circle, one for small triangle, and one for large triangle). Each of these level pairs is represented in the labels file with a `term` element.

Example

A sample interaction between two attributes (shape and size) each with two levels (circle and square, and small and large respectively) would look similar to this in the labels file:

```
...
<interaction attribute1="1" attribute2="2">
    <term attribute1level="1" attribute2level="1" column="8"/>
    <term attribute1level="1" attribute2level="2" column="9"/>
    <term attribute1level="2" attribute2level="1" column="10"/>
    <term attribute1level="2" attribute2level="2" column="11"/>
</interaction>
...
```

none Element

Parent Element

utilities

Attributes

Attribute	Required	Data Type	Description
column	Required	Positive integer	This is the column number of the column in the data file that contains the utility data for this none item.

Sequential Child Elements

Element	Number of Occurrences
label	0 or 1

A set of utilities may have any number of none items. A none item is typically a “I would not choose any of these products” option used in conjoint studies. But it can also represent any other single level attribute that is in the project.

Example

A study with a none option may have the following in the labels file:

```
...
<utilities>
    ...
    <none column="8">
        <label>
            My None Option
        </label>
    </none>
</utilities>
...
```

Appendix A

A Demographic Variables Example (no Utilities)

The following is an example of a small CSV data file followed by a Labels File that correctly describes the data file.

CSV File

```
Respondent Number,Gender,Age,Name
1,2,34,"Rachel"
2,1,35,"Michael"
3,3,42,"Shannon"
```

Labels File

```
<?xml version="1.0" encoding="utf-8"?>
<sawtoothSoftwareLayout>
    <segmentationVariables>
        <variable isRespondentId="true"/>
        <variable variableType="coded">
            <variableLabel>Respondent’s Gender</variableLabel>
            <valueLabel>
                <integerValue>1</integerValue>
                <label>Male</label>
            </valueLabel>
            <valueLabel>
                <integerValue>2</integerValue>
                <label>Female</label>
            </valueLabel>
            <valueLabel>
                <integerValue>3</integerValue>
                <label>Unspecified</label>
            </valueLabel>
        </variable>
        <variable variableType="number">
            <variableLabel>Respondent’s Age</variableLabel>
        </variable>
        <variable variableType="text">
            <variableLabel>Respondent’s First Name</variableLabel>
        </variable>
    </segmentationVariables>
</sawtoothSoftwareLayout>
```

In this example, there are four columns in the CSV data file. Correspondingly, there are also four `variable` elements in the labels file. In this example we chose to not include the `column` attribute in the `variable` elements. By leaving this out, the first `variable` element will map to column one in the CSV data file, the next `variable` element will map to column two in the CSV data file, etc.

Also note that for the first variable, we only specified that it was the respondent id column. Since it does not have any child elements we chose to use the XML short hand notation for an empty element.

By not specifying the `variableLabel` for this first variable, the value in the CSV header row, “Respondent Number,” will be used for this variable’s label.

The gender variable is a coded variable and therefore it contains one `valueLabel` element for each of its coded values. Each of these elements defines a value label for the specific coded value (1 is “Male”, 2 is “Female”, and 3 is “Unspecified”).

Appendix B

A Sample Utilities File

The following example is a data file that contains utilities for three attributes and a “none” option. The first two attributes are partworth attributes each with three and two levels respectively. The third attribute is a linear attribute that has four levels.

CSV File

```
Resp Id,B A,B B,B C,L P,H P,Price,NONE
1001,1.526,-3.264,0.507,1.29,-1.247,0.123,5.041
1003,5.369,3.113,-3.364,-8.651,5.981,1.333,-2.166
1005,1.727,1.131,0.437,-7.5,3.924,-0.998,-1.266
1012,2.943,2.394,1.368,-4.684,3.98,-1.446,-0.657
1014,10.618,0.453,-1.922,-2.298,1.612,-0.212,4.49
1018,2.57,2.303,0.742,-8.769,7.52,-2.351,-1.018
1022,5.499,1.737,0.568,-8.566,7.365,-1.228,-0.929
1024,4.509,1.948,-1.147,-6.758,3.923,-1.83,2.009
1029,-1.168,2.688,-0.34,-10.401,7.519,-1.229,-0.003
1035,4.471,0.029,1.574,-4.185,3.941,-2.115,0.391
1052,0.087,-1.652,1.185,-5.191,2.843,-2.025,1.42
1058,4.167,1.058,0.463,-7.764,6.017,-1.512,3.435
```

Labels File

```xml
<?xml version="1.0" encoding="utf-8"?>
<sawtoothSoftwareLayout>
    <segmentationVariables>
        <variable isRespondentId="true" column="1">
            <variableLabel>Respondent Id</variableLabel>
        </variable>
    </segmentationVariables>
    <utilities>
        <runName>CBC/HB Calculated Utilites for PC Project</runName>
        <attribute number="1" coding="partworth">
            <label>PC Brand</label>
            <level number="1" column="2">
                <label>Brand A</label>
            </level>
            <level number="2" column="3">
                <label>Brand B</label>
            </level>
            <level number="3" column="4">
                <label>Brand C</label>
            </level>
        </attribute>
        <attribute number="2" coding="partworth">
            <label>PC Performance</label>
            <level number="1" column="5">
                <label>Below Average Performance</label>
            </level>
            <level number="2" column="6">
                <label>Above Average Performance</label>
            </level>
        </attribute>
        <attribute number="3" coding="linear" column="7">
            <label>Price</label>
            <level number="1">
                <label>$799.99 USD</label>
                <value>799.99</value>
            </level>
            <level number="2">
                <label>$999.99 USD</label>
                <value>999.99</value>
            </level>
            <level number="3">
                <label>$1199.99 USD</label>
                <value>1199.99</value>
            </level>
            <level number="4">
                <label>$1399.99 USD</label>
                <value>1399.99</value>
            </level>
        </attribute>
        <none column="8">
            <label>None</label>
        </none>
    </utilities>
</sawtoothSoftwareLayout>
```

Appendix C

A Brief Introduction to XML

The CSV File

The comma-separated values (CSV) file format is a very common file format for storing tabular data. Due to its popularity and support, it is a great choice for transferring data between applications. We like the CSV file format and are changing our applications to use this format more.

One drawback to the CSV file format, however, is that it does not contain very much metadata (data about the data). It can have a header row that will give a little more information about the data in the associated column, but that is all that can be easily included. Unfortunately, we frequently need much more information about the data than this simple header can provide. For example, consider the following CSV data file:

```
Respondent Number,Age
1,1
2,4
3,2
```

Just by looking at this file we can only guess what the data mean. The second column tells us that the data represent the age of the respondents, but what exactly do the numbers in the column mean? Is this data about very young respondents (between one and four years old)? Or are these coded values where the numbers represent an age range (1 represents respondents between 18 and 30 years old, for example).

To fill the need of describing data, Sawtooth Software has created a companion file called the "Labels File." This file is an XML file that more completely describes the data that are contained in the CSV file.

XML

There are many resources on the Internet and in books to learn more about XML. One good one to start with is w3schools:

http://w3schools.com/xml/default.asp

We will not give a complete coverage of XML in this document. But we will give a simple introduction here to get you started with the Labels File. First, we need to define a few XML terms.

XML Declaration

The first line in the Labels File is the XML declaration. It defines the XML version (1.0) and the encoding used. It is strongly recommended that the UTF-8 encoding be used. This encoding will support almost any language and enables multilingual data. So, the declaration used by the Labels File looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
```

XML Tags

An **XML tag** is text contained in "<" and ">" characters. For example:

```
<variable>
```

is a “variable” tag. Each tag in XML has to have an **opening tag** as well as a **closing tag**. The above example was an opening variable tag. Its closing tag is:

```
</variable>
```

To make the closing tag all we do is add the “/” character after the “<“ character.

XML tags are case-sensitive. This means that the opening and closing tags must all be exactly the same *including* case.

XML Element

An **XML Element** is everything from (including) the element’s start tag to (including) the element’s end tag. An element can contain other elements, text, attributes (defined below), or a mix of these. For example, I could have an XML document that describes my home library:

```
<?xml version="1.0" encoding="utf-8"?>
<library>
    <book>
        <title>Getting Started with Conjoint Analysis</title>
        <author>Bryan K. Orme</author>
    </book>
    <book>
        <title>The Sawtooth Software Market Simulator</title>
        <author>Sawtooth Software, Inc.</author>
    </book>
</library>
```

In this example, the library element contains two book elements. Each book element contains a title and an author element that each contain text with the title and author respectively.

Each XML document must have one and only one **root element**. The root element is the main element in the XML document. All other content of the XML document must be inside the root element. In the above example, the library element was the root element.

The white space (space, tab and new line characters) between tags are ignored in XML. Therefore the following snippet of XML:

```
<?xml version="1.0" encoding="utf-8"?>
<library><book><title>Getting Started with Conjoint
Analysis</title><author>Bryan K. Orme</author></book><book><title>The
Sawtooth Software Market Simulator</title><author>Sawtooth Software,
Inc.</author></book></library>
```

is essentially identical to the previous library example. Throughout this document the extra white space characters will be used to make the XML easier to read. If you are creating or modifying XML we recommend that you format the file similarly so that you can more easily match up the opening and closing tags and make sure that each element is nested correctly within its parent element.

XML elements form a tree starting with the root element. Each element that is nested immediately within the root element will be the root's **child elements**. The child elements of the root element will have the root element as their **parent element**. Therefore, from the library example above, the book element has two child elements (title and author) and its parent element is library.

Empty XML Elements

If an XML Element is empty (it contains no text, or child elements), then it can be represented as:

```
<book></book>
```

It can also be represented with the following short hand notation:

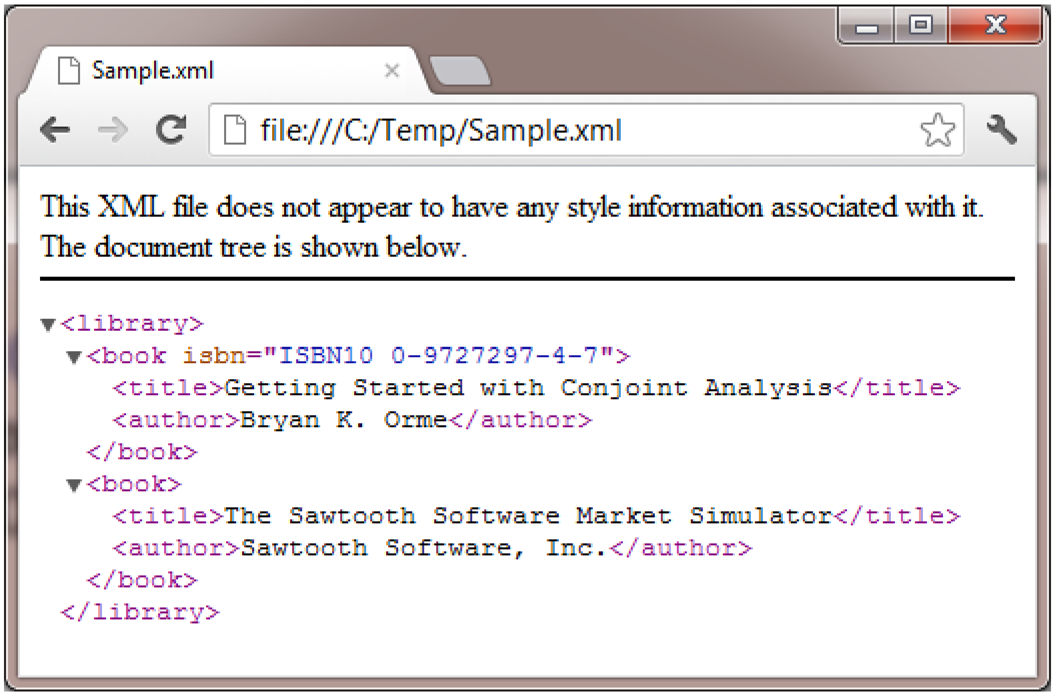
```
<book/>
```

XML Attribute

An **XML Attribute** is contained within the opening tag and provides additional information about the element. XML Attributes are key-value pairs where the key is case-sensitive and the value must be in quote characters. The key and value are separated by an "=" character. For example, I could have included the ISBN number of each of my books that have one using an "isbn" attribute:

```
<?xml version="1.0" encoding="utf-8"?>
<library>
    <book isbn="ISBN10 0-9727297-4-7">
        <title>Getting Started with Conjoint Analysis</title>
        <author>Bryan K. Orme</author>
    </book>
    <book>
        <title>The Sawtooth Software Market Simulator</title>
        <author>Sawtooth Software, Inc.</author>
    </book>
</library>
```

You can open an XML document using your browser to see the elements displayed as a tree (see Figure 1).



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<library>
  ▼<book isbn="ISBN10 0-9727297-4-7">
     <title>Getting Started with Conjoint Analysis</title>
     <author>Bryan K. Orme</author>
   </book>
  ▼<book>
     <title>The Sawtooth Software Market Simulator</title>
     <author>Sawtooth Software, Inc.</author>
   </book>
 </library>
```

Figure 1. *Sample XML document open in a Chrome Browser*

If there are any errors in the XML document, then the browser will display an error message. Each element can be collapsed or expanded in the browser by clicking on the "▼" or "▶" character next to each element (some browsers use a "-" or a "+" character instead of the triangle characters).