# Automating Access to Data from HTML Forms

Duncan Temple Lang

March 26, 2009

Downloading data via HTML forms is tedious and error prone when done repeatedly. It is better to access the data programmatically in order to allow automated access that can be easily controlled in high-level languages. Additionally, such access allows particular data to be determined dynamically within a script and then accessed.

To provide such programmatic access, we need to emulate the HTML form access. With that interactive interface, the user completes different elements of the HTML form and submits the query to the Web server. On submission in the user's browser, the contents of the different form elements are collected and sent as a query to the target Web server. The query combines the name and value for each form element, including both visible and *hidden* fields. There are details about how the query is constructed, ecoded and sent to the Web server.

A natural equivalent interface in a progarmming language like R is to have an R function that allows the caller to specify the same information as she might on the HTML form. At its simplest (and most common), each visible HTML form element corresponds to a formal argument of the R function. For HTML elements with default values, we can use those same default values in R. Hidden HTML form elements are not included in the formal arguments but are sent as part of the query. The Submit button of the HTML form corresponds to the actual invocation of the function and so is not an explicit part of the function.

If we wish to create an R function corresponding to an HTML form, we can examine the rendered HTML form and the corresonding HTML source. We find the different form elements such as ¡input¿, ¡textarea¿, ¡select¿, ¡option¿. For each element, we take the name and define a formal argument for the R function. We also determine if the element has a default value. And for elements such as option menus, we compute the set of permissible values.

Let's look at a simple example from the UC Santa Cruz Genome web site. This has been explicitly formatted to make it easier to read and identify the different elements.

Unfortunately, much of this is formatting information rather than related to the actions of the form. The important components of the form are

1. `<form action="/cgi-bin/hgText" name="mainForm"  method="post">`
2. `<input type="hidden" name="hgsid" value="30932773">`
3. `<input type="image" border="0" name="tbDummyEnterButton" src="hgText_files/DOT.gif">`
4. `<select name="org">` and the associated ¡option¿ elements such as `<option selected="selected" value="Hur`
5. `<input type="hidden" name="phase" value="table">`

Essentially, there are two entries of interest: the two pull down menus. These are identifed by the ¡select¿ tags and are populated with choices via the nested ¡option¿ tags. As with all form elements, each ¡select¿ entry has a name. In our example, these are `org` and `db`. These become the names of the formal arguments in our R function. Since, for each of these menus, the user of the HTML form can only select a value from within its list, we know the set of possible values. For example, looking at the `db` element[1], we have three possible values: July 2003, April 2003, and Nov. 2002. Corresponding to these visible selections are the three values that will be sent as the actual selected value in the HTTP query when the form is submitted. These are "hg16", "hg15" and "hg13". From the HTML source, we see that the default value is July 2003 (i.e. hg16) as that has the additional HTML attribute **selected**. To provide an interface to this argument in R, we might define our function something like

[1]The value `db` comes from the **name** attribute of the ¡select¿ element.

```
<form action="/cgi-bin/hgText" name="mainForm" method="post">
 <table bgcolor="#cccc99" border="0" cellpadding="1" cellspacing="0">
  <tbody>
   <tr>
    <td>
     <table bgcolor="#fefdef" bordercolor="#cccc99" border="0" cellpadding="0" cellspacing
      <tbody>
       <tr>
        <td>
         <table bgcolor="#fffef3" border="0">
          <tbody>
           <tr>
            <td>
             <input type="hidden" name="hgsid" value="30932773">
             </input>
             <input type="image" border="0" name="tbDummyEnterButton" src="hgText_files/D
             </input>
             <table>
              <tbody>
               <tr>
                <td align="center" valign="baseline">
                genome
                </td>
                <td align="center" valign="baseline">
                assembly
                </td>
               </tr>
               <tr>
                <td align="center">
                 <select name="org" onchange="document.orgForm.org.value = document.mainF
                  <option selected="selected" value="Human">
                  Human
                  </option>
                  <option value="Chimp">
                  Chimp
                  </option>
                  <option value="Mouse">
                  Mouse
                  </option>
                  <option value="Rat">
                  Rat
                  </option>
                  <option value="Chicken">
                  Chicken
                  </option>
                  <option value="Fugu">
                  Fugu
                  </option>
                  <option value="Fruitfly">
                  Fruitfly
                  </option>
                  <option value="C. elegans">
                  C. elegans
                  </option>
                  <option value="C. briggsae">
                  C. briggsae
                  </option>
                  <option value="Yeast">
                  Yeast
```

```
hgText =
function(db = "hg16")
{
 if(!missing(db)) {
   dbValues =  c("hg16" = "July 2003",
                 "hg15" = "April 2003",
                 "hg13" = "Nov. 2002")


   i = pmatch(db, names(dbValues))
   if(is.na(i)) {
      i = pmatch(db, dbValues)
      if(is.na(i))
         stop("Incorrect value for db:", db, " must be one of ",
                 paste(names(dbValues), collapse= ", "))

      db = db[i]
   } else
      db = names(dbValues)[i]
 }
}
```

Obvsiously, we need to extend this to include the `org` ¡select¿ element in this function. And we need to include the hidden elements (`phase` and `hgsid`) in the form in the query and actually send the query.

We can construct such functions manually by examining the HTML source for the form. However, we can also do this programmatically by parsing the HTML content and extracting the relevant information for the different form element types. We can do this in R using the HTML parsing mechanism in the *XML* package. By providing code to process each of the different HTML form element types, we can extract the name of the form element, its default value, its possible values (i.e. options) and readily automate the task of constructing the R function to mimic the HTML form.