Tutorial: Online two armed bandit experiment

Publishing experiments online becomes more and more common in psychology. It has the advantage of not wasting time of the participants and experimenters on recruitment and travel, and the ability to reach more diverse populations than first and second year psychology students. Building and running experiments using the browser also have the huge advantage of an ever-growing amount of tools developed by web designers and programmers, and highly active online community to provide you with tips and solutions to your frustrations. It is therefore extremely useful tool even for lab based experiments.

While there are tons of tutorials for building websites, games and other web apps, there aren't much around for creating experiments. When I started building my experiments, moving from other platforms like Cogent and Presentation I collected bits and pieces from around the web and was finally able to get a functioning web based versions of my experiments. I thought that it will be useful to share what I learned by writing a tutorial. This tutorial is not comprehensive in any way, and you will probably want to use other resources to get the more comprehensive information, but I aim at showing how to build a functioning two armed bandit experiment running on your browser.

This tutorial will focus on building the experiment in JavaScript and running it, while the next one will be about saving the data (i.e. interacting with a server and a database). We will work with these files:

- Html – the front of the project, where images will be displayed and buttons pressed.
- CSS – the code in charge of styling of the content.
- JS – javascript, the engine (brain?) running the experiment and controlling the display.

These files can be edited on any text editing software (notepad), I personally use NetBeans (https://netbeans.org/) and NotePad++ (https://notepad-plus-plus.org/), and the firebug browser plugin (http://getfirebug.com/) for debugging.

1. Setting the stage, the html file

We start with setting up the html file. Open a new file and save it as 'index.html'. It will contain in its header links to the other relevant resources of the project, i.e. the css and js files. In addition to the project's specific js and css files I use two other tools. The first is bootstrap.css (http://getbootstrap.com/css/) which includes a lot of professional looking styling code, and a great grid system to control the appearance for your experiment. The other is JQuery (https://jquery.com/) which is a shorthand version for javascript, allowing you to write complicated javascript commands in a short way. All four resources are linked in the header:

```
<head>

  <title>Experiment</title>

    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css">

    <link rel='stylesheet' type='text/css' href='TwoArmedTut.css' media = "screen"/>
```

```
        <script src="http://code.jquery.com/jquery-1.11.1.js"></script>

        <script src="TwoArmedTut.js"></script>

    </head>
```

The body of the html file includes the elements which will be displayed. In html these elements are contained within something called div which serve as a container for other elements (text, images, forms, other divs and so on). I use a main div which contain three divs: top, stage and bottom. Html elements are displayed one under the other (vertically) unless otherwise specified. Each element can be assigned a class name, which is controlled by the css code, and an id. The element id is the way to control it using javascript.

```
    <body>

   <div class ='container' id = 'Main'>

        <div class = 'DivElements' id = 'Top'></div>

        <div class ='DivElements' id = 'Stage'> </div>

        <div class ='DivElements' id = 'Bottom'> </div>

    </div>

    </body>
```

The whole html file should look like this:

```
<!DOCTYPE html>


<html>

  <head>

  <title>Experiment</title>

    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css">

    <link rel='stylesheet' type='text/css' href='TwoArmedTut.css' media = "screen"/>

        <script src="http://code.jquery.com/jquery-1.11.1.js"></script>

        <script src="TwoArmedTut.js"></script>

 </head>

 <body>
```

```
 <div class ='container' id = 'Main'>
        <div class = 'DivElements' id = 'Top'></div>
        <div class ='DivElements' id = 'Stage'> </div>
        <div class ='DivElements' id = 'Bottom'> </div>
  </div>


 </body>
</html>
```

2. The CSS file

The CSS file contain styling information for our elements. We can later control these from javascript. CSS can actually do much more, but for our modest purpose we will use relatively elementary properties.

Open a new file and name it 'TwoArmedTut.css'. Each class can have its own settings. We had different class for the main div and for the div elements:

```
.Main {
   margin-right: auto;
   margin-left: auto;
   background-color: #DDDDDD;
   font-family:Arial,Helvetica,sans-serif;
}
.DivElements {
   margin-right: auto;
   margin-left: auto;
   display: block;
  }
```

Setting the margins to auto will help in making the div appear in the middle of the window and not squashed to the left. We can also set the background color, fonts and other parameters.

3. JavaScripting – Initializing parameters

Our experiment is a two armed bandit task, in which participants have to choose between two options associated with two different reward probabilities. In this version I'll use a very simple design in which reward probabilities will be fixed across the session, and by using a binary reward (coin/nothing).

Open a new file and name it 'TwoArmedTut.js'. The first thing we will do is to create the main function which will start working as soon as the browser is open on our 'index.html' file:

```
$(document).ready(function() {

});
```

This command tells the computer to start running the function as soon as the document is ready. Functions in javascript are delineated by curly brackets, {}. We will add all the other commands and functions within the main function brackets.

We will start by initializing the variables that will be used throughout the experiment. These include the number of trials, the probability of getting a reward from the two options, the number of points the participant got, and the time the experiment starts. We will also create a unique user ID which can be used to store the participant's data (later).

```
var NumTrials = 10;
var P1 = 0.7;
var P2 = 0.3;
var SumReward = 0;
var Init = (new Date()).getTime();
var SubID = CreateCode();
```

Note that the SubID is calling a function called CreateCode. This function is not part of javascript and we will have to write it ourselves. To keep things organized I put all my 'utility functions' together towards the end of the script, just below the 'experiment functions'. Functions can get input parameters and send output parameters. Our function will create a 10 digit random number:

```
function CreateCode() {
        return Math.floor(Math.random() * 10000000000);
};
```

Debugging

Debugging and keeping track on variables may be a headache in JS. We can use two simple tools to help with finding what goes wrong in our code.

Information about the web application has a console which can be accessed by selecting 'inspect' option from the menu that pops up when you right-click on the browser window in chrome, or via the top-menus. You will be able to inspect the structure of your html page there, see which files are loaded (images, php, js and so on), the communication of information across the network to the server and see the **console**. The console is a place where javascript send information such as error messages, or costume made messages.

We can send information to the console by using the console.log() function. For example:

```
console.log('Subject ID is:'+SubID)
```

This command will display the text 'Subject ID is' followed by the value stored in the variable SubID in the console window. You can display any variable from your code there, hopefully with an informative comment, and keep track on the flow and logic of your code, and hunt down bugs.

In addition, you can use the more aggressive 'alert' command.

```
alert('Subject ID is:'+SubID)
```

This command will display the message in a pop screen in the browser, which has to be dismissed by the user. It will stop the code from running until it is dismissed. While this is a useful function, if you get into some loop of messages you will have to shut down the browser window as these messages will keep popping with no way to refresh the screen

Both options are useful for keeping track of your code and debugging.

Setting the stage

We also want to initiate some display parameters according to the participant's own screen and resolution. We will grab the window size property and use it to set the size of our html divs. We can manipulate the html divs by using their ID preceded by #. Here we will control style parameters by using the .css command and changing display properties:

```
var thisHeight = $(document).height() * 0.9;

var thisWidth = thisHeight * 4 / 3;

var DispWidth = thisHeight * 5 / 6;

var DispHeight = DispWidth / 2;

 $('#Main').css('min-height', thisHeight);

$('#Main').css('width', thisWidth);
```

You can now open the index.html file in the browser and see your stage. Try changing the window size and refreshing the page.

4. Starting with Experiment functions –information sheet, consent form and instructions.

As we are interested in running an experiment we need to start with some screens that include information about our experiment and research which are not usually found in other web apps. We will start by presenting the information sheet of the experiment, which will include your contact information and all the things your ethics committee requires. We will than have a consent form and the instructions. The way I build my experiment is that each such stage is a function, and each one calls the next function either after some time or on button press.

We will start by adding the line 'Information();' right after the last parameter initializing line. This will send the participant to the Information function. Note that as the functions are inside the main 'ready' function, all the variables declared above can be used within the functions. It is not the case for variables declared within the functions – these can be used only inside the functions and should be sent as output (or input to other functions) if we need them.

We will start by setting another utility function to create a new div and append it to some other div element already created. In our case we want to add a text div inside the 'Stage' div. The function will get the name of the parent div and the new div as inputs. It will create a new div and append it to the parent:

```
function CreateDiv(ParentID, ChildID) {
    var d = $(document.createElement('div'))
        .attr("id", ChildID);
    var container = document.getElementById(ParentID);
    d.appendTo(container);
};
```

And now to the information function. It will start with some display settings, followed by setting the new text div. The title and information text will be stored in two variables and will be added to the text div. The title and text are styled using html commands, for example the command <h3> tells the browser that this text should be displayed as a title (but not as big as h1 title):

```
$('#Top').css('height', thisHeight / 20);
    $('#Stage').css('width', DispWidth);
    $('#Stage').css('min-height', thisHeight * 17 / 20);
    $('#Bottom').css('min-height', thisHeight / 20);
    CreateDiv('Stage', 'TextBoxDiv');
    var Title = '<H3 align = "center">Information page for participants in research studies</H3>';
    var Info = 'Here is a lot of information. It is important.';
    $('#TextBoxDiv').html(Title + Info);
```

Now we add the button to take us to the consent form. I put the button inside another div element, which is another way to create div (and other html elements) not using our previous function. I use a class for the button from 'Bootstrap.css' to make it look nicer. After appending the button to the bottom div we need to program its behaviour. This is done by setting its click properties with a function. We want to remove the text and buttons from the screen (emptying their parents' divs), and start the function 'consent'. Here is the entire 'Information' function;

```
function Information() {

    $('#Top').css('height', thisHeight / 20);

    $('#Stage').css('width', DispWidth);

    $('#Stage').css('min-height', thisHeight * 17 / 20);

    $('#Bottom').css('min-height', thisHeight / 20);

    CreateDiv('Stage', 'TextBoxDiv');

    var Title = '<H3 align = "center">Information page for participants in research studies</H3>';

    var Info = 'Here is a lot of information. It is important.';

    $('#TextBoxDiv').html(Title + Info);


    var Buttons = '<div align="center"><input align="center" type="button"  class="btn btn-default"
id="toConsent" value="Next" ></div>';

    $('#Bottom').html(Buttons);


    $('#toConsent').click(function() {

      $('#TextBoxDiv').remove();

      $('#Stage').empty();

      $('#Bottom').empty();

      Consent();

    });

  };
```

Moving on to the next section – consent form. It will be very similar to the information sheet screen but we will add some tick boxes to make sure your participant knows what he is in for. You can add other fields like name, gender, handedness and so on. The beginning of the function should not surprise you too much; we're just adding tick boxes after the information text. Ticking the boxes is

mandatory, so we will have to check that the participant ticked all the boxes when pressing the button before moving to the instructions. This is done by inserting an 'if' statement, checking that all boxes are ticked before proceeding. If no an alert window will tell you that all boxes must be ticked. If everything is ok you can proceed to page 1 of the instructions.

```
function Consent() {

    $('#Top').css('height', thisHeight / 20);

    $('#Stage').css('width', DispWidth);

    $('#Stage').css('min-height', thisHeight * 17 / 20);

    $('#Bottom').css('min-height', thisHeight / 20);

    CreateDiv('Stage', 'TextBoxDiv');

    var Title = '<H3 align = "center">Consent form for participants in research studies</H3>';

    var Info = 'Please read the following criteria and tick all boxes. <br><br>';

    var Ticks = '<input type="checkbox" name="consent" value="consent1" id= >I have read the information page<br>' +

        '<input type="checkbox" name="consent" value="consent2">I have had the opportunity to contact the researcher to ask questions and discuss the study<br>' +

        '<input type="checkbox" name="consent" value="consent3">I have received satisfactory answers to my questions or have been advised of an individual to contact for answers to pertinent questions about the research and my rights as a participant<br>' +

        '<input type="checkbox" name="consent" value="consent4">I understand that I am free to withdraw at any time, without giving a reason, and without incurring any penalty<br>' +

        '<input type="checkbox" name="consent" value="consent5">I am over 18 years of age.<br>';

    $('#TextBoxDiv').html(Title + Info + Ticks);

    var Buttons = '<div align="center"><input align="center" type="button"  class="btn btn-default" id="toInstructions" value="Next" ></div>';

    $('#Bottom').html(Buttons);

    $('#toInstructions').click(function() {

      if ($("input:checkbox:not(:checked)").length > 0) {

        alert('You must tick all check boxes to continue.');

      } else {
```

```
            $('#TextBoxDiv').remove();
            $('#Stage').empty();
            $('#Bottom').empty();
            Instructions(1);//move to first page of instructions
        };
    });
} ;
```

Our instructions will have two pages, each with some instructions and an image. Images will be stored in an 'images' folder. After the instructions the participant will see a countdown and will start the experiment proper. The instruction function is very similar to the consent and information functions, but get the page number as input. It also has an additional 'back' button, which will be hidden from the first page, and a 'to experiment' button which will be hidden from all pages but the last one. We will assign functionality to all these buttons in the same way as before, and will control their visibility. Finally, we will have a countdown before the experiment start. We will use the javascript function 'setTimeout' which runs a function after a predefined time interval, which is very useful when you want to show something for a short while or automatically move to the next function.

```
function Instructions(PageNum) {
    $('#Top').css('height', thisHeight / 20);
    $('#Stage').css('width', DispWidth);
    $('#Stage').css('min-height', thisHeight * 17 / 20);
    $('#Bottom').css('min-height', thisHeight / 20);
    var NumPages = 2;//number of pages
    var PicHeight = DispWidth / 2;
    CreateDiv('Stage', 'TextBoxDiv');
    var Title = '<H2 align = "center">Instructions</H2>';
    switch (PageNum) {
        case 1:
            var Info = '<H3>In this experiment you have to collect as many coins as possible, hidden behind two doors. \n\
On each trial you have to choose between the doors.<br> Each door has some probability of getting a coin. \n\
```

```
You chose the door by clicking on it with your mouse.  <br>There are' + NumTrials + ' trials in this experiment.</h3><br><br>';

            break;

        case 2:

            var Info = '<H3>After each decision you will see the outcome – coin or nothing. You will then continue directly\n\

 to the next trial. At the end you will see how many coins you earned.<br>Good luck!</h3><br><br>';

            break;

        default:

            var Info;

            break;

    }  ;

    var ThisImage = '<div align = "center"><img src="images/Inst' + PageNum + '.png" alt="house" height="' + PicHeight + '" align="center"><br><br></div>';

    $('#TextBoxDiv').html(Title + Info + ThisImage);

    var Buttons = '<div align="center"><input align="center" type="button"  class="btn btn-default" id="Back" value="Back" >\n\

<input align="center" type="button"  class="btn btn-default" id="Next" value="Next" >\n\

<input align="center" type="button"  class="btn btn-default" id="Start" value="Start!" ></div>';

    $('#Bottom').html(Buttons);

   if (PageNum === 1) {

        $('#Back').hide();

    }  ;

    if (PageNum === NumPages) {

        $('#Next').hide();

    }   ;

    if (PageNum < NumPages) {

        $('#Start').hide();
```

```
} ;
$('#Back').click(function() {
    $('#TextBoxDiv').remove();
    $('#Stage').empty();
    $('#Bottom').empty();
    Instructions(PageNum - 1);
});
$('#Next').click(function() {
    $('#TextBoxDiv').remove();
    $('#Stage').empty();
    $('#Bottom').empty();
    Instructions(PageNum + 1);
});
$('#Start').click(function() {
    $('#TextBoxDiv').remove();
    $('#Stage').empty();
    $('#Bottom').empty();
    setTimeout(function() {
        $('#Stage').html('<H1 align = "center">Ready</H1>');
        setTimeout(function() {
            $('#Stage').html('<H1 align = "center">Steady</H1>');
            setTimeout(function() {
                $('#Stage').html('<H1 align = "center">Go!</H1>');
                setTimeout(function() {
                    Options(1);//Start with the first trial
                }, 1000);
            }, 1000);
```

```
        }, 1000);
      }, 10);
    });
  }  ;
```

5. The two armed bandit functions – decision and reward

Now we will program the two functions that consist of our experiment. The decision function and reward function will repeat for the number of times we specified, and at the end will go to the final function with the results.

In the 'options' function we need to present two images on the screen, one slightly to the left and the other to the right. In order to make sure that the participant is not biased to one side we will switch the location of the doors from time to time. To set the images horizontally aligned I am using the grid functionality from 'bootstrap.css'. They divide the screen to 12 columns, and you can specify which columns your div element will use. We will leave one column empty, and then use 3 columns for the left door, 4 columns for the reward (leave them empty for now), 3 for the right door and 1 empty. The doors themselves will be the buttons, as we can assign click functionality to div elements. Once a door is clicked we will mark it with a border (outline) and move to the reward function without erasing the current images, sending the trial number and choice as inputs.

```
function Options(TrialNum) {
    $('#Top').css('height', thisHeight / 20);
    $('#Stage').css('width', DispWidth);
    $('#Stage').css('min-height', thisHeight * 17 / 20);
    $('#Bottom').css('min-height', thisHeight / 20);
    CreateDiv('Stage', 'TextBoxDiv');
    var Title = '<div id = "Title"><H2 align = "center">Choose a door:</H2></div>';
    var Door1 = '<img id = "Door1" src="images/Door1.png" class="img-responsive center-block" >';
    var Door2 = '<img id = "Door2" src="images/Door2.png" class="img-responsive center-block" >';
    var RandPosition = Math.random();
    if (RandPosition < 0.5) {
      var Images = '<div class="row">  <div class="col-md-1"></div>  <div class="col-md-3">' +
Door1 + '</div><div id = "Middle" class="col-md-4"></div><div class="col-md-3">' + Door2 +
'</div><div class="col-md-1"></div></div>';
    } else {
```

```
        var Images = '<div class="row">  <div class="col-md-1"> </div> <div class="col-md-3">' +
Door2 + '</div><div  id = "Middle"  class="col-md-4"></div><div class="col-md-3">' + Door1 +
'</div><div class="col-md-1"></div></div>';

    }

    $('#TextBoxDiv').html(Title + Images);

    $('#Door1').click(function() {

      $(this).css({"border-color": "#CCFF33",

        "border-width": "3px",

        "border-style": "solid"});

      Reward(TrialNum, 1);

    });

    $('#Door2').click(function() {

      $(this).css({"border-color": "#CCFF33",

        "border-width": "3px",

        "border-style": "solid"});

     Reward(TrialNum, 2);

    });

  }
```

The reward function will determine the outcome of the choice according to the reward probabilities we set in the beginning. It will then display the outcome. We will use the empty space between the doors to display either a coin or a frowny. This empty space in the middle was assigned the id 'Middle' in the previous function, so we can change it directly, as was the title. After the outcome display we will update the overall reward collected by the participant, and determine if he reached the end of the experiment. If the trial number is equal to the number of trials in the experiment we will call the 'end' function. Otherwise we will update the trial number and call the 'options' function again. We will use the setTimeOut function twice, once to give the participant time to see the outcome (and the choice), and again to have a fadeout effect of the screen. The fadeout is important as the doors change locations occasionally, and it is likely to miss that if they just jump (try it if you want). The fadeout also makes a nicer transition between trials, giving a better sense of moving from one trial to the other.

```
function Reward(TrialNum, Choice) {

    $('#Title').empty();

    var ThisReward = 0;
```

```
        var RandomNum = Math.random();
        if (Choice === 1) {//Door1
            if (RandomNum < P1) {
                ThisReward = 1;
            }
        } else {//Door2
            if (RandomNum < P2) {
                ThisReward = 1;
            }
        }
        if (ThisReward === 1) {//Coin
            $('#Title').html('<H2 align = "center">You got a coin!!</H2>');
            $('#Middle').html('<img id = "Door1" src="images/coin.png" class="img-responsive center-
block" >');
            SumReward = SumReward + 1;
            if (TrialNum + 1 < NumTrials) {
                setTimeout(function() {
                    $('#TextBoxDiv').fadeOut(500);
                    setTimeout(function() {
                        $('#Stage').empty();
                        $('#Bottom').empty();
                        Options(TrialNum + 1);
                    }, 500);
                }, 1500);
            } else {
                $('#TextBoxDiv').remove();
                $('#Stage').empty();
                $('#Bottom').empty();
```

```
            End();
        }
    } else {//no coin
        $('#Title').html('<H2 align = "center">You got nothing...</H2>');
        $('#Middle').html('<img id = "Door1" src="images/frowny.png" class="img-responsive center-block" >');
        if (TrialNum + 1 < NumTrials) {
            setTimeout(function() {
                $('#TextBoxDiv').fadeOut(500);
                setTimeout(function() {
                    $('#Stage').empty();
                    $('#Bottom').empty();
                    Options(TrialNum + 1);
                }, 500);
            }, 1500);
        } else {
            $('#TextBoxDiv').remove();
            $('#Stage').empty();
            $('#Bottom').empty();
            End();
        }
    }
}
```

The end function is very similar to the information function, but will include the number of coins collected by the participant and no buttons:

```
function End() {
    $('#Top').css('height', thisHeight / 20);
    $('#Stage').css('width', DispWidth);
```

```
    $('#Stage').css('min-height', thisHeight * 17 / 20);

    $('#Bottom').css('min-height', thisHeight / 20);

    CreateDiv('Stage', 'TextBoxDiv');

    var Title = '<H2 align = "center">You have finished the experiment!<br> <br> You earned
'+SumReward+' coins!<br><br> Thanks for participating!</H2>';

    $('#TextBoxDiv').html(Title );

  } ;
```

6. That's it, or is it?

You should now have a fully functional experiment to run on the browser. You can play with all the elements you saw here, change the reward to continuous one or change probability of reward randomly as the experiment proceeds. You can have a look at the bootstrap.css documentation, or get better acquainted with html css and javascript using online documentaions and tutorials (I highly recommend the codecademy ones in http://www.codecademy.com/).

However, so far we did not discuss the most crucial thing for experiments: data. As this involve slightly more complicated settings and other programing languages we will leave this to the next tutorial.

Hope that you find this helpful; I will appreciate any comments you may have!

Sending stuff to the database – under construction

In the javascript code I use ajax, which is a way to send and receive information from a php file. Here I am sending two variables, subject's ID (string) and DataA (integer) from javascript to PhP. If it succeeds we can go ahead and run the experiment, if not we will run the function DontRunExperiment, and if there is some error we alert the user:

```
$.ajax({
        type: 'POST',
        data: {ID: ID, DataToSend: Data},
        async: false,
        url: 'InsertData.php',
        dataType: 'json',
        success: function(r) {
           if (r[0]> 0) {
              DontRunExperiment ();
           } else {
              RunExperiment();
           }
           ;
        }, error: function(XMLHttpRequest, textStatus, errorThrown) {
           alert("Status: " + textStatus);
           alert("Error: " + errorThrown);
        }
     });
```

Note that you can run connect the database only from php script running in the same domain (in the same server) – for security reason the database and the php scripts have to be in the same server. The PhP script ('InsertData.php'):

```
<?php
$database="databasename";
$host="hostadress";
$user="user";
$password="password";

$db = new mysqli($host, $user, $password, $database);
if (mysqli_connect_errno()) {
  printf("DB error: %s", mysqli_connect_error());
  exit();
}
//for security reasons we remove slashes from the inputs
$ID = stripslashes(htmlspecialchars($_POST['ID']));
$DATA = stripslashes(htmlspecialchars($_POST['DataToSend']));

$stmt = $db->prepare("INSERT INTO table_name VALUE(?,?,NOW())");//I also insert the time
$stmt->bind_param("si", $ID,$DATA );//s=string, i=integer
$stmt->execute();
$err = $stmt->errno ;
$data[] = array(
    'ErrorNo' => $err,
```

```
   );
$stmt->close();
 $db->close();
echo json_encode($data);
 ?>
```

This php file assumes that there is a table in the database with three columns, one for ID which is string, one for data which is integer and one for time. Here is an sql script to create such a table:

```
CREATE TABLE ` table_name` (
  `ID` varchar(20) NOT NULL,
  `Data` int(11) NOT NULL,
  `time` double NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

And finally, here is a php file that can be run independently, whice displays all the subjects ID instances in table_name, and a link to another php file (below) to extract all the data for a specific subject and save it to a csv file:

Present_Subjects.php:

```
<?php

$host=" hostadress ";
$uname="user";
$pass="password";
$database = " databasename ";

$connection=mysql_connect($host,$uname,$pass);

echo mysql_error();

$selectdb=mysql_select_db($database) or
die("Database could not be selected");

 $data = mysql_query("SELECT * FROM table_name")
 or die(mysql_error());
 Print "<table border cellpadding=3>";
 while($info = mysql_fetch_array( $data ))
 {
 Print "<tr>Experiment";
 Print "<th>ID:</th> <td>".$info['ID'] . "</td> ";
 Print "<th>Time:</th> <td>".$info['time'] . " </td>";
 Print " <td><a href=Export_Table.php?SubCode=".$info['ID'] .">Export Self</a></td></tr>";
}
 Print "</table>";
mysql_close($connection);
?>
```

Export_Table.php:

```
<?php
$host=" hostadress ";
$uname="user";
$pass="password";
$database = " databasename ";

$connection=mysql_connect($host,$uname,$pass);

echo mysql_error();

$selectdb=mysql_select_db($database) or
die("Database could not be selected");

$Subject = $_GET['SubCode'];
$result = mysql_query("SELECT  * FROM table_name WHERE ID= '$Subject' ");
$pasajeros = '';

if ($result) {
   while ($row = mysql_fetch_array($result)) {
      $pasajeros .= $row["DATA"] . ",".$row["Time"] ."\r\n"; //note the comma here
   }
}
$filename = "pasajeros_" . date("Y-m-d_H-i");
header("Content-type: application/vnd.ms-excel");
header("Content-disposition: csv" . date("Y-m-d") . ".csv");
header("Content-disposition: filename=ExperimentName-" . $Subject . ".csv");
mysql_close($connection);
echo $pasajeros;
exit();
?>
```