|  |  |
| --- | --- |
| **Please follow the instructions carefully. Please save all the code and written answers in the same document and submit to Avenue as a PDF file when finished. Please try to work on the assignment independently before you collaborate with other students. It is critical to you learning that you seriously attempt on your own as much as possible. We will discuss any questions you have about the assignment in class.**  **Learning objectives:**   1. **More on R data structures** 2. **Descriptive statistics** 3. **Basic plotting** 4. **T-tests (weighted and unweighted)** | |
| **Q1. Describe what is happening in the code to the right.** |  |
| In the code to the right, we’re creating a list. A list is a useful data structure that can contain other types of data structures of mixed ty1pes. Here we have a list with three vectors: a numeric vector of length 10, a character vector of length 5, and a logical vector of length 2. |  |
| The first element in the list is the numeric vector, and we can access it with the code to the right |  |
| To access an element in this vector, we can access it using the code to the right. In this example, we are accessing the 1st element of the vector that is the first element in the list |  |
| We can use conditional indexing here as well. In this example, we are going to access the element in the numeric vector that has a corresponding index in the logical vector that is true. *Note that the which() function returns the index rather than the value* |  |
| As we did in the last assignment, we are going to use the curl package to import a file from the Internet. | |
| First we need to ensure you have the required package installed. Type and run the code to the right. The install.packages() function will first install the package if required, and the library() function will load it for use in this RStudio session. |  |
| Now write the code below to assign an internet address to a variable called url:    (hint: ensure that the ‘Od’ is the capital letter ‘O’, not the number ‘0’) | |
| The final step to loading the data is to type and run the code to the right. The curl() function fetches a file as specified by the url variable which is then read by the read.csv() function.r |  |
| For those having difficulty installing packages, the easiest workaround is to download the file directly to your computer  Step 1, copy and paste this into your web browser (preferably Chrome):  https://datazone.healthgeomatics.com/Hawaii\_CO2.csv  Step 2. Save the file to your hard drive  Step 3. Import into R:  data <- read.csv("<the location of your file and the file name>")  Remember, **for Windows users the path to the file must have double slashes** (e.g., "E:\\Temp\\Hawaii\_CO2.csv") | |
| You should now see a file (named ‘data’) that has three columns: CO2 (which is a monthly average CO2 concentration in parts per million), Year (which is the year of measurement) and Month (which is the month of measurement). There are 161 measurements in this file and the data are currently in order going from May 1974 to September 1987. |  |
| Write code that:  creates two new files called ‘pre’ and ‘post’ such that all data prior to 1980 are in ‘pre’ and all data 1980 and after are in ‘post’ | |
| The file called ‘data’, is a dataframe object. To refer to a column in a data frame, we can use a ‘$’ prefix. For example, to print out all values of CO2 in the console, type and run the code to the right. |  |
| Let’s calculate some summary statistics for the CO2 column. We’ll do this by putting the CO2 column into a summary() function. Type and run the code to the right. Note that there are 6 items returned from this function. The third item is the median and the fourth is the mean. |  |
| Now we are going to store the mean monthly C02 for these data in a new variable called avg\_m. Type and run the code to the right.  Note that the use of the [4] immediately following the function is an easy way of extracting the fourth element in the vector returned from the summary() function. |  |
| **Q2. Do some research online and find another function that can be used to calculate some simple statistics for the CO2 variable. Don’t install any new packages to accomplish this. Write the code to use that function.** | |
| We can create a histogram of these data using the hist() function. Type and run the code to the right. |  |
| We’re going to create a new variable in the ‘data’ dataframe that is the difference between the monthly CO2 average and the mean over the series (which we calculated earlier). Type and run the code to the right. |  |
| If you open up the dataframe ‘data’ you’ll see that there is now a new column called diff. Negative values are monthly CO2 measurements that are below the average, and positive values are CO2 measurements that are above the average. |  |
| We can plot the measurements using the plot() function. For now, we will plot CO2 on the y-axis and a number (going from 1 to 161) on the x axis. The first argument in the plot function is the x axis, and the second argument is the y-axis. The number of items in the x-axis and y-axis must be equal. The code to the right uses a vector (1,2,3...161) as the x-axis, and the monthly average CO2 measurements as the y-axis. Type and run the code to the right.  Note that the code data$CO2 is how we access the column called CO2. The ‘$’ allows us to retrieve a specific column in *dataframe* called data |  |
| The resulting plot should look something like what you see to the right. This shows CO2 concentration over time over the 161 measurements. Note that the axes don’t have suitable labels and the figure is untitled. |  |
| Recreate the plot with a title and labels by running the code below. Note that I have broken the plot function across lines for readability. | |
|  | |
| **Q3. Write your own code to create a plot in which the Y axis is the ‘diff’ variable you created a few steps ago. Ensure that the y axis is labeled "CO2 (ppm difference from series average)"** | |
| Now we are going to aggregate the data by year--so each monthly average is aggregated to a yearly value. However we are first going to exclude 1974 and 1987 since these are incomplete years. Type and run the code below to include only data between 1975 and 1986.d | |
|  | |
| As in the example above, using the ‘$’ allows us to directly access a column in the dataframe called ‘data’. In the code, we are keeping all rows in which the year is greater than or equal to 1975 and less than or equal to 1986. The result is now that data has 144 rows/records.  Now that we have the data from 1975 to 1986 we can take yearly averages using the aggregate() function. Type and run the code below | |
|  | |
| We have created a new dataframe called ‘agg’. The problem is that the columns in this dataframe are chosen by default and are also lam. We can fix that with the names() function. With this function, we assign names using a vector of column names we choose. Type and run the code to the right. |  |
| We can plot data from this new data frame using the plot function. Note that the X axis is now Year and the Y axis is CO2. | |
|  | |
| If you look carefully at the plot of CO2 concentrations by month you’ll notice that there is a seasonal cycle. At this step we aare going to determine which month of the year has the highest mean CO2 concentration. One way to do this is to aggregate on month and plot out the data. |  |
| Type and run the code below to aggregate the data by month. | |
|  | |
| As in the previous example, the columns in the new dataframe ‘agg2’ need to be renamed. Type and run the code to the right. |  |
| Now we can plot the data. This time we are going to plot the data as bars rather than points. This is because the monthly CO2 data are aggregated, axaand do not represent a continuous series. Type and run the code below. | |
|  | |
| This graph is not very illustrative, unfortunately, since the bars range from 0 to the maximum and all values are fairly close to each other. Moreover, there are no labels on the x-axis. Copy and paste the code below (rather than typing...to save time) and then run it.  Note the following: First, the ylim argument **sets limits on the y axis specified by the values in the vector c(300,500)**. Second, the **names.arg() function in barplot() assigns names to the x axis labels**, and the **xpd=FALSE argument ensures that the bars stop at the bottom of the y-axis**. |  |
| barplot(agg2$CO2, main="Monthly CO2 concentrations",  xlab="Month (1975 to 1986 aggregated)",  ylab="CO2 (ppm)",  ylim=c(300,350),xpd=FALSE,  names.arg=c("J","F","M","A","M","J","J","A","S","O","N","D")) | |
| From this graph we can better see monthly variations in CO2 concentration. |  |
| Now we are going to create a new dataframe (called ‘data2’) with only data from **April and October**. Type and run the code below. | |
| Note that the **‘|’ (called a ‘pipe’)** can be used as ‘OR” in R. The code inside the square brackets is saying ‘return all the rows in which Month = April or Month= October. Note that there is nothing to the right of the comma, which indicates that the expression should return all columns. | |
| Copy and paste the code below (rather than typing...to save time) and then run it. | |
| barplot(data2$CO2, main="Monthly CO2 concentrations",  xlab="April and October (1975 to 1986 aggregated)",  ylab="CO2 (ppm)",  ylim=c(300,350),xpd=FALSE,  col=rep(c("black","white"),12))  legend("topleft",  legend = c("April","October"),  fill = c("black", "white")) | |
| The resulting code should show the seasonal variation between April and October in the series.  Note that the legend() function has to be run following the plotting function (in this case, barplot()) in order to appear on the plot you see. Also note that the x-axis should have year labels as well, but this is a bit more tricky than just using the **names.arg()** function, so I didn’t do this in order to save from adding more code above. **As an extra challenge (if you’re up to it!) see if you can figure out a way to add the labels to the plot.** |  |
| If you look at the graph above you’ll notice that there is not much change in the month to month variations in average CO2 concentration over time. This suggests that while CO2 concentration in the atmosphere changes, the seasonal differences seem to remain relatively constant.  **Q4. Do you think a bar plot of CO2 measurements is the best way of representing this information? What do you think is a better way of visually representing mean monthly CO2 concentrations? Explain your answer (two to three sentences)** | |
| **Q5. The data in this series goes until September 1987. If you had to make predictions about the CO2 concentrations in October, November and December, what would you do? Come up with some code that makes a prediction of the remainder of 1987 CO2 concentrations. *The predictions do not have to be perfect, and you don’t need to do any fancy modeling at this stage—keep the procedure as simple as you want provided that they generate a prediction better than random chance*. Explain your reasoning. Importantly, do not import or use any new libraries for this analysis; only use base R functions.** | |
| **Ensure that all the code above is saved to a document.**  **The data are monthly average temperature readings taken at Jan Mayen Island (located in the North Atlantic). There is a *Temperature* (C) variable, a *Year* variable and a *Month* Variable. To import the data into your R session, you must change the url variable to the following:**  "<https://datazone.healthgeomatics.com//JanMayen.csv>" and then import the data the same way as you did above.  **Once the data are imported, you must write code that does the following:**  **Q6. Keep data only from 1940 to 2001**  **Q7. Calculate summary statistics for the Temperature variable using the summary() function**  **Q8. Create a histogram of Temperature**  **Q9. Plot the December average temperatures from 1940 to 2001, and ensure that the plot is properly labeled. Ensure that temperature is on the y axis and year is on the x axis.  *Don’t worry about the tick mark labels for the years on the x-axis***  **Q10. Calculate the annual average temperatures (by aggregating on *year*) from 1940 to 2001 and put the result in a dataframe called *agg***  **Q11. Plot the average temperatures by year on a plot.** | |
| Download the following file to your computer:  <https://drive.google.com/file/d/1xnYbUxVIHMcJm7rBHoQI-lf6CU0zIx1Y/view?usp=sharing>  It’s a csv file, so once it’s downloaded import it using read.csv(). Ensure the data are in a data frame, and that you called the data frame ‘df’ | |
| These data are part of a survey on internet use in the US. There are two survey variables: age and internet use, along with a state identifier, weight and respondent ID. Internet use is a categorical variable with 4 possible values: 1 **(use internet), 2 (don’t use internet**), 8 and 9 (non responses). Age has valid numeric values between 18 and 98.  **Q12. Write code to delete any records with age equal to 99 or internet use values equal to 8 or 9.** | |
| The question we’re going to try to answer is: is there a difference in age between people who use and don’t use the internet? Our **null hypothesis** is that the mean age is the same for internet users and non-users. We’re going to assume the following:   1. That responses are independent of one another 2. The response data are *roughly* normally distributed (they probably aren’t…)   Given these assumptions, we can test this hypothesis in R. | |
| Write and run the code to the right. This particular code uses a binary variable (designating group membership) on the right and the continuous numeric variable on the left. There are other ways to do this analysis that are worth experimenting with if your data are structured in different ways. |  |
| The results of this test are reported in the console. We can see that the p-value is very small. This tells us that there is a very very small chance that we would have seen a difference in means this large if the populations of internet users and non-users were the same average age. In other words, **these data are not consistent with the null hypothesis that users are the same age.** |  |
| In surveys, **weights** are often used to address non-randomness in the sampling methodology. This dataset has a weight variable, so we should probably use a **weighted t-test**. There is a library for weighted t-tests. |  |
| We need to organize the data differently in order to use this test. We will have to **subset** the dataset into internet users and non-users. Write and run the code to the right |  |
| Now we can run the test. Write and run the code to the right. Here we are specifying the data (the first two arguments) the weights (the second two arguments) and then using the same data argument to indicate that the data sets are not the same in length. |  |
| Looking at the p-value, we draw the same conclusion; the difference in means is too large to be due to chance alone. Informally, we might argue that these data suggest that **there is a difference in the ages of internet users and non-users.** |  |
| **Q13. Find some data online and use the t-test() function in R. Provide a brief description of your reasoning and interpretation of the results of your analysis.**  **Save all the code and written answers (as appropriate) for the questions above into a document. Do not include output from the code, and ensure that the document is nice, clean and readable. Save the document in a pdf file.**  **This assignment is due *before* next class. Please submit on avenue before next week’s class.** | |