Problem Set 5

CSCI E-5a: Programming in R

```r
load( "Problem Set 5 R Objects.Rdata" )
```

Problem 1: Report Missing Values

In this problem, we will examine the vector `problem.1.data` for missing data.

Part (a)

Does the vector contain any `NA` values? Write R code to determine this, and then report your answer using one or two sentences of text.

Solution

Part (b)

How many elements of `problem.1.data` have the value `NA`? Report your result using a `cat()` statement.

Solution

Part (c)

What proportion of the elements of `problem.1.data` have the value `NA`? Report your result using a `cat()` statement, displaying this value using 2 decimal places.

Solution

Part (d)

Determine which locations of `problem.1.data` contain `NA` values, and save these locations in a variable. Then use a `for` loop to print out a simple report of these values:

- Each location should be reported on a single line.
- Enumerate the items, so that each line is numbered.
- Make sure that you have correct spacing in your answer.

Solution

End of problem 1

Problem 2: Report and Replace -9 Values

Some people use the numeric value -9 to represent missing data instead of NA. I think this is a bad idea, but people do it nonethless. The best practice here is that if you encounter a dataset that uses -9 values to represent missing data you should convert these to NA values, and in this problem we're going to explore this idea.

Part (a): Sample mean before cleaning

To see why it's a bad idea to represent missing data by using -9 instead of NA, let's start by calculating the sample mean of problem.2.data. Report your result using a cat() statement, displaying this value with 2 decimal places.

Solution

Part (b): Detecting -9 values

Does the vector problem.2.data contain any -9 values? Write R code to determine this, and then report your answer using one or two sentences of text.

Solution

Part (c): Counting -9 elements

How many elements of problem.2.data have the value -9? (Hint: use a vectorized comparison operation.) Report your result using a cat() statement.

Solution

Part (d): Proportion of elements equal to -9

What proportion of the elements of problem.2.data have the value -9? Report your result using a cat() statement, displaying this value using 2 decimal places.

Solution

Part (e): Locations of -9 elements

Determine which locations of problem.2.data contain -9 values, and save these locations in a variable. Then use a for loop to print out a simple report of these values:

- Each location should be reported on a single line.
- Enumerate the items, so that each line is numbered.
- Make sure that you have correct spacing in your answer.

Solution

Part (f): Replacing -9 with `NA`

In Lecture 3, Module 3, Section 3, there was a line of code that showed how to assign a single value to multiple locations by using a vector of positive integers.

Using this technique, along with the vector of locations from part (e), assign the value `NA` to the locations in `problem.2.data` that currently have a -9 value.

You only need one line of code to do this.

There's nothing to report here, but write your code cleanly so that the TAs can understand it.

Solution

Part (g): Sample mean after cleaning

Calculate the sample mean of the non-missing values of the corrected version of `problem.2.data` that you created in part (f). Report your result using a `cat()` statement, displaying this value with 2 decimal places.

Compare your answer for this part with your answer for part (a). Now do you see why it's a bad idea to use -9 to represent a missing value?

Solution

End of problem 2

Problem 3: Report and Replace Outliers

Part (a): Stripchart

Construct a stripchart of the values in `problem.3.data`. Do you think that there are any outliers in this data? Explain your answer with one or two sentences.

Solution

Part (b): Histogram

Construct a histogram of the values in `problem.3.data`. Do you think that there are any outliers in this data? Explain your answer with one or two sentences.

Solution

Part (c): Assigning `NA`

For this problem, we're going to assign the value `NA` to the outliers. The challenge here is to specify the outliers. I'm going to suggest that you select a threshold value, and anything above that threshold value is considered an outlier. You'll have a lot of flexibility as to how you define the threshold, but choose something reasonable.

Use the `which()` function with a vectorized comparison operation to determine the exact locations of the outliers in `problem.3.data`, and then use positive integer indexing to assign `NA` to these values.

You can do this with one line of code, so if you're writing lots of code, then you're doing too much work.

Solution

Part (d): Stripchart

Construct a stripchart of the values in `problem.3.data` with the outliers removed. Do you think that there are any outliers in this data? Explain your answer with one or two sentences.

Solution

Part (e): Histogram

Construct a histogram of the values in your cleaned version of `problem.3.data`. Then superimpose the best-fitting normal density curve on the data.

There's a subtle point here: when we fixed the outliers, we changed their values to `NA`. But when we fit the normal density curve, we calculate the sample mean and sample standard deviation. What do we have to do to calculate the sample mean and sample standard deviation when there are `NA` values in the data?

Solution

End of problem 3

Problem 4: Grocery Store Sales

Let's return to our grocery store example.

The prices per box for each brand of breakfast cereal are:

Brand	Price
SBZ	2.99
KYM	3.49
HKT	7.99

Here we have a sequence of transactions:

Transaction	Brand	Number of Boxes
1	KYM	2
2	SBZ	4
3	SBZ	3
4	HKT	1
5	SBZ	3
6	KYM	2
7	SBZ	5

Our goal in this problem is to calculate the total amount in sales for Sugar Bomz (SBZ).

Part (a)

Store the data on the brand information and the number of boxes sold into vectors. Also construct a lookup vector based on the pricing table.

There's nothing to report here, but write your code clearly so the TAs can understand what you're doing.

Solution

Part (b)

Use the pricing lookup vector from part (a) to convert the brand vector into a vector of prices per box. Display this vector of prices per box directly.

Solution

Part (c)

Use a vectorized operation with the number of boxes sold vector from part (a) and the price per box vector from part (b) to construct a vector consisting of the sales amount for each transaction. Display this vector of sales amounts for each transaction directly.

Solution

Part (d)

In this lecture, we saw how to use values in one vector to select values from another vector.

In this part, we're going to use the values in the brand vector from part (a) to select values from the total sales amount vector we constructed in part (c).

In particular, use logical indexing to select the values in the total sales amount vector from part (c) corresponding to a Sugar Bomz (SBZ) sale.

Display this vector of total Sugar Bomz sales amounts directly.

Solution

Part (e)

Calculate the sum of the vector of total Sugar Bomz sales amounts. Report your result using a `cat()` statement, displaying this value to 2 decimal places.

Solution

End of problem 4

Problem 5: Two-Tone Stripchart

We can use logical indexing techniques to create an informative data visualization that I call a "two-tone" stripchart.

The idea of a two-tone stripchart is that we have a threshold value, and we want to emphasize or highlight the values that are greater than this threshold value.

For this problem, we will use a threshold value of 200:

```
threshold <- 200
```

The data for this problem is contained in the vector `exercise.5.data`.

Part (a): Low-pass filtering

Use logical indexing to select all the values in `exercise.5.data` that are less than or equal to the threshold value. Store these values in a variable, and report the first 5 elements using a `cat()` statment, displaying these values using 2 decimal places.

When we select values that are less than or equal to a cutoff value, this is called "low-pass filtering", because we are removing high values from the data but allowing the low values to pass through.

Solution

Part (b): High-pass filtering

Use logical indexing to select all the values in `exercise.5.data` that are strictly greater than the threshold value. Store these values in a variable, and report the first 4 elements using a `cat()` statment, displaying these values using 2 decimal places.

When we select values that are strictly greater than a cutoff value, this is called "high-pass filtering", because we are removing low values from the data but allowing the high values to pass through.

Solution

Part (c): Two-tone stripchart

Now we're going to construct a stripchart for this data, where the data below the threshold is displayed using one color and the data above the threshold above the threshold is displayed using a different color.

- First, use the low values from part (a) to construct a stripchart, including titles and using jitter. You should explicitly select the point shape, size, and color. Finally, specify the range of the x-values to be from 0 to 400.
- Next, we're going to add another stripchart to this graph. Construct another stripchart, this time using the high values from part (b). You don't have to specify the titles or the range of the x values, because those have already been determined. You *do* have to use jitter for these points, and explicitly specify the shape, size, and color of these points. Use a different color from what you used for the low values. Finally, include `add = TRUE`, just like with the `curve()` function.
- Finally, draw a vertical line with an x-value equal to the threshold.

All of these graphing actions have to occur within the same code chunk.

If you do this properly, you should end up with a stripchart where all the points below the threshold value have one color, and all the points above the threshold value have another color, and we can easily see the threshold value because it's represented by the vertical line.

Solution

End of problem 5

Problem 6: Final Grades

We finally have the tools to calculate a final course score, given the raw scores. In this problem, we'll go from the initial raw scores all the way to the final course score, using R techniques from every lecture so far.

Five students have these final raw scores:

Status	Problem Sets	Midterm	Comprehensive Assessment
Graduate	58	74	74
Undergraduate	65	65	58
Graduate	64	76	74
Graduate	60	66	72
Undergraduate	65	57	61

For this course:

- A raw score of 68 points on the problem sets is equivalent to a standardized score of 100.
- A raw score of 80 points on the Midterm Assessment is equivalent to a standardized score of 100.
- A raw score of 80 points on the Comprehensive Assessment is equivalent to a standardized score of 100.

Part (a): Preliminary Score 1

Using vectorized operations, calculate the Preliminary Score 1 for all 5 students. Report this vector using a `cat()` statement, displaying these values with 2 decimal places.

Solution

Part (b): Preliminary Score 2

Using vectorized operations, calculate the Preliminary Score 2 for all 5 students. Report this vector using a `cat()` statement, displaying these values with 2 decimal places.

Solution

Part (c): Graduate Final Course Score

Recall that the graduate final course score is calculated as the maximum of the Preliminary Scores 1 and 2.

Using the vectors of preliminary scores that you created in parts (a) and (b), construct a vector of the graduate final course scores for the five students. You'll have to figure out how to take the maximum of the two scores for each student, but here are two suggestions:

- You could write a `for` loop.
- You could use the `ifelse()` function.

It's up to you.

Report this vector of graduate course scores using a `cat()` statement, displaying these values with 2 decimal places.

Solution

Part (d): Scaling factor

Recall that I multiply the graduate course by a scaling factor, depending on the student's registration status:

Status	Scaling Factor
Graduate	1
Undergraduate	4/3

Create a lookup vector for this scaling factor. Then use this lookup vector to convert the registration status to a numeric vector of scaling factors. Report this vector of scaling factors using a `cat()` statement, displaying the values with 2 decimal places.

Solution

Part (e): Final course score

Using a vectorized operation, multiply the values in the vector of graduate course scores from part (c) by the vector of scaling factors from part (d). This will be a vector consisting of the final course scores. Report this vector using a `cat()` statement, displaying the values with 2 decimal places.

Solution

End of Problem 6