SCR assignment 1

Assignment introduction

Deadline

Upload a report of your answers as a `YOURULCN_A1SCR2017.Rmd` file to Blackboard **before** 11.00am, October 19th 2017.

It is very important that you hand in the assignment in time: we'll be discussing the assignment, and the answers to the assignment, during the lecture of the 19th of October, so we cannot accept any submissions after 11.00 am.

You can hand in the assignment on Blackboard: go to Course Documents, Week 4, Assignment 1.

The assignment is basically about applying your basic `R` skills to solve a data cleaning problem that you might meet in your future job.

Strategy

Divide your coding problem into smaller pieces. This is especially helpful when you are stuck. Most of the times, after coding a few of these little steps, you might see how you could solve a bigger part of the coding problem.

Data cleaning

Introduction

A researcher from another department in your workplace comes to you with a request. The researcher has collected data for one of his experiments. The data are a set of measurements on blood samples taken from patients. The blood samples have been sent out to twelve different research centres in the Netherlands for processing, because this would be faster than waiting for a single center to process all the samples by themselves. Unfortunately, each of the centers uses a slightly different coding scheme for their samples.

The researcher wants to know which sample comes from which center, but does not want to code over a 1000 samples by hand. You are going to use your `R` skills to automate this, doing work that, by hand, would take a week to do, but using `R` only seconds. While you are at it, he talks to you about a few other issues you might be able to solve…

Although the data and the problem in this assignment are fictitious, the scenario is quite realistic!

First, let's do some exploratory work to see what kind of data we have.

1.1 Reading in the data

a

The researcher already tried to read in the data, the `R` object for this can be found on under the name `bloodsamples_data_bad`. And he tried to do some calculation, seen in the code below. However, the code produces an error. Explain why. Rewrite the code to make it work.

```r
load("0_data/assignment_1.Rdata")
```

```r
bloodsamples_data_bad[, 10]*10
```

b

Read in the data yourself using one of the `read.` functions (e.g `read.csv`). Tell `R`, you don't want it to interpret strings as factors. Look at the help file of the `read.` functions to see how to do this. The file to read in is called `bloodsamples_data.txt`.

c

If you've read in the data correctly, it should be equal to the `R` object `bloodsamples_data`. Check that it is. *Hint, see if you should use `all.equal` or `identical`.*

1.2 Exploratory

For the rest of the assignment, continue working with the `bloodsamples_data` object.

Before we start doing the work the researcher wants help with, we do some exploratory work to see if the data looks OK. Just because reading in the data did not fail, does not necessarily mean it worked completely correct.

a.

How many samples in total have been processed?

b.

Sometimes, missing values are coded, not with `NA`, but rather with some clearly 'impossible' value. Are there any missings or strange measurements? If so, in what rows? For now, do not look at the `sample_date`, `measurement_date` and `difference_in_days` variables.

c.

Convert any strange measurements to `NA`'s.

d.

Create an index to select in the data set only the rows with *no* NA's.

Hint: if selected correctly, you will have about 619 valid cases...

e.

Give some summary statistics for the variable `ir` (at least the mean and standard deviation). Also give these summary statistics for this variables for `males` and `females` seperately. Do you think there is any population difference between male and female values of iron concentration in the blood?

f.

The researcher wants to calculate the age of every subject in years as unit, also using the information about the months. He wants to calculate the following:

$$\frac{age_{years} \cdot 12 + age_{months}}{12}$$

He didn't have the correct data in `R` yet, so he could not test hist code, but he thinks this will work:

```
age_in_years <- data$age_years * 12 + data$age_months / 12[4]
```

Explain why the results of the `R` code above do not work.

1.3 Working with dates

Because the blood samples go 'bad' quite quickly, the centers have also included a variable that denotes the days that have passed between taking the blood sample (`sample_date`) and the date the blood sample was measured (`measurement_date`). The researcher checked a few of these by hand and thinks this calculation has not been done correctly.

a.

Many softwarepackages, including R, provide ways to calculate differences between calendar dates, for example in days. Look at the function `as.Date()`. There are many ways to use this function. Use the `as.Date()` function to calculate the difference between the day of sampling (when the blood was taken), and the day of measurement (when the blood was measured in the lab).

Hint: You will have to tell `R` how the dates are formatted: dates can be written in many ways (such as day/month/year or day-month-year) and `R` needs to know how to interpret the string correctly!

b.

Store the correctly calculated difference in days between sample and measurement time as a variable called `corrected_difference_in_days` in the dataset.

c.

Using the correct dates, create an index with `TRUE` or `FALSE` entries that can be used to select *only* the rows for which the number of days between the two dates is smaller or equal to **30**. How many samples have more than 30 days beteen time of measurement and time of sampling?

1.4 Doing the work

The assignments **a** and **b** below help you divide the task of this part of the assignment smaller pieces. Feel free to skip these if you think you have a more effective way to deal with the problem! However: make sure you use `strsplit` and `if-else` statements to show us that you know how these work!

Before you start working on this part, you might want to check the helpfile of `strsplit` and work through the examples in the helpfile.

a.

i.

Use the function `strsplit()` to split up the following string: "subject_Groningen: 1" into a *vector* of individual characters.

ii.

There is a single character that is recurring for every sample name that can be used to denote the *end* of a center location. Namely: the double colon (':')! Determine the position of the ':' character in the string you've split up using `strsplit` and `which`.

b.

We can use the colon character as a reference point. If we read characters in reverse order from this position we will get the characters that belong to the particular centers. We can use `paste` to put these letters back together, and easily check which center location is equal to the string we get from this paste operation. Unfortunately not all center locations are of equal character length. There are two simple ways to solve this, and we will do both to show that are multiple ways of solving these types of problems. Sometimes one of these comes to use quickly, and others require more thinking. Often you have to iterate on the design of your solution, because one is more elegant or faster than the other, or possibly more robust against changes in the data at a later moment. Different situations often require slightly different solutions. But the concepts are often the same.

b: The short route

Write a function called `GetLocation` that takes as argument one of the strings considered above (e.g. “subject_Groningen: 1”) and splits it at the colon (‘:’) character. Select the left part of the split and use some `if-else` statements to see to which location the string belongs to (e.g. “subject_Groningen”, belongs to Groningen).

b: The long route

i

Create a vector with all the center names.

ii

Create a `for` loop, to go through all of these centers, to split up the characters of each of the center locations, and save *only* the last character of each location in a character vector called `center_abbreviated`. I.e. suppose you have three centers called `c("Alpha", "Bravo", "Charlie")` then the resulting vector should be `c("a", "o", "e")`. Check if there are any duplicates.

iii

If done correctly you will have found that there are some duplicate entries in `center_abbreviated`: “Groningen” and “Leeuwarden” both end in the letter ‘n’. So we will need more characters to uniquely identify each center.

Create a function called `Stringtail` that takes as argument a string, such as “Groningen” (call this argument `x`), and a total number of characters (`a`) to keep. Let the output return the last `a` characters of string `x`, pasted as a single string.

iv

Like in question **4b-ii** create a character vector with center location abbreviations, but this time saving the last 2, 3, 4, 5 and 6 characters. Use `for` to loop over both the centers, and the variable number of characters to save. Save all these vectors (each vector contains abbrevations for each of the twelve centers) into a list object.

v

Check for each of the elements in your list whether the character vectors contain duplicates or not. How many characters do you need, minimally, to uniquely identify each center?

c

We will continue with the work we did in the short route, as this is probably most convenient.

Use the function `GetLocation`, you created in **b: the short route* to create a new `R` object, a character vector, called `location`. This vector should contain, for each sample, an entry equal to one of the twelve locations, depending on the location the sample was analysed at, found by looking at the variable `bloodSampleID`.

d

Add the vector `location` as a variable to the dataset.

e

Save the cleaned up dataset as an `.Rdata` file with name `cleaned_data.Rdata`.