read.xlx

by Antonio Fasano

There is a new function in the `read.*` family, `read.xlx`, which can read Excel xlsx workbook sheets into R data frames. Some features are:

- It can import all, one, or a selection of sheets, where specific sheets are requested by means of their name.
- Instead of importing all the sheets' cells, it can import only those comprised in a named range.
- It can distinguish between cells formatted as numbers, percent, text and dates,
- Date cells are recognised whatever the language locale.
- Blank (visual) lines are detected and automatically removed from the data frame, unless you want to keep them.

- The filter is not based on any external engine and does not requires Excel to be installed at all. It's pure R code, so you can read xlsx files on Linux systems.

Synopsis

```
read.xlx(
    file, sheets=NULL, header.sheets=FALSE, header.ranges=FALSE, ranges=NULL,
    keepblanks=FALSE, allchars=FALSE, general='character', info)
```

file path to xlsx file.
sheets character vector with sheet names to read or NULL to read all.
header.sheets TRUE if, for all sheets, the first row is a header line to be used for column names. It can also be a logical vector whose values are TRUE (FALSE) for each sheet with (without) header.
header.ranges TRUE if, for all named ranges in `range`, the first row is a header line to be used for column names. It can also be a logical vector whose values are TRUE (FALSE) for each named range with (without) header.
ranges character vector with sheet names to read or NULL to not read use them.
keepblanks if true do not import rows or columns having only empty cells.
allchars If false do not infer cell style, but use always R character class.
general map Excel general format to 'character' or 'numeric'.

info : list of: sheets' name; vector whose names are ranges and values their sheets; vector whose names are ranges and values their references.

simplify : remove enclosing list for a single item.

Details

If `header.sheets` is a logical vector, its length should match the length of the workbook sheets, including empty sheets, or the length of `sheets` if this argument is not NULL. Similarly, if `header.ranges` is a logical vector, its length should match the length of `range`.

`info` can be used only with `file`. The elements name of the list returned are: wbsheets, rgsheets, rgrefs.

Setup

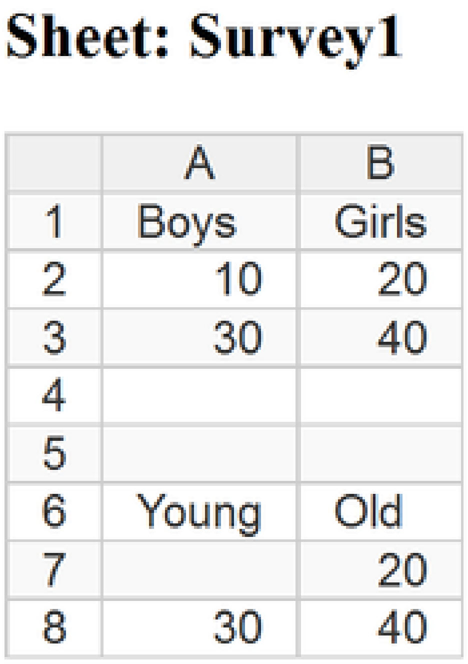
Currently the script is not in package form so, after downloading it, just source the R source with:

```
source('path\to\readxlx.r')
```

You are done.

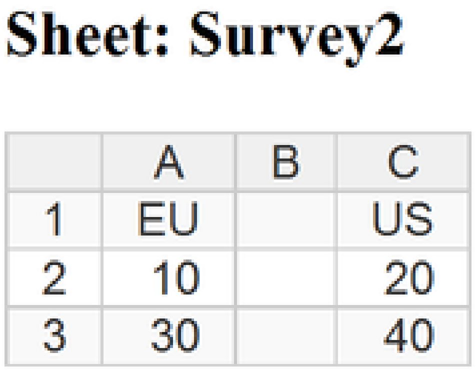
Use it in the simplest form

Assume you have the spreadsheet `survey.xlsx`, with consisting of an empty sheet and two sheets `Survey1` and `Survey2` with plain formatted data like the following (anyway a `survey.xlsx` spreadsheet should accompany this manual):



Sheet: Survey1

	A	B
1	Boys	Girls
2	10	20
3	30	40
4		
5		
6	Young	Old
7		20
8	30	40



Sheet: Survey2

	A	B	C
1	EU		US
2	10		20
3	30		40

A file should come together with this manual, `survey.xlsx` which you can use to run the same code as here.

To import the file with all sheets simply run:

```
surv=read.xlx('survey.xlsx')
```

The result of your import is:

```
class(surv)
```

```
## [1] "list"
```

```
surv
```

```
## $Survey1
##       1     2
## 1  Boys Girls
## 2    10    20
## 3    30    40
## 6 Young   Old
## 7  <NA>    20
## 8    30    40
##
## $Survey2
##    1  3
## 1 EU US
## 2 10 20
## 3 30 40
```

```
lapply(surv, class)
```

```
## $Survey1
## [1] "data.frame"
##
## $Survey2
## [1] "data.frame"
```

```
names(surv$Survey1)
```

```
## [1] "1" "2"
```

Following the general convention for the `read.*` family of functions, the sheets are converted into data frames, plus data frames are wrapped into a list. The data frame comprising the list are named like the equivalent Excel sheet, here `Survey1` and `$Survey2`. As you guess from the output:

1. Blank sheets are by default not imported.
2. By default Excel column letters are converted to digits and set as the name of the data frame columns.
3. Blank lines, that is whole blank rows or columns, are removed, instead single blank cells inside data tables are reported as NA.
4. By default it is possible to identify skipped blank lines by reading the sequence of row and column numbers.

In case cells are not plain formatted and formatting inside columns is inconsistent R will coerce incoherent cell to the column prevailing format and issue a warning.

Import individual workbook items

We can customise the default behaviour. For example let us assume we want to import only the sheet "Survey2"

```
surv= read.xlx('survey.xlsx', sheets=c('survey2'))
```

As usual for a single item `sheets=c("survey2")` can be shortened as `sheets='survey2'`

Note that, respecting Excel convention the name is not case sensitive, so `survey2` works even if the actual sheet name is `Survey2`. The name used in the importing command will be the one stored in R, in case you later need to address it.

Another thing is that, since you asked for a single sheet, there is no need to wrap it in a now worthless list:

```
class(surv)
```

```
## [1] "data.frame"
```

```
surv
```

```
##    1  3
## 1 EU US
## 2 10 20
## 3 30 40
```

What about using the first sheet row as the data table names?

```
surv=read.xlx('survey.xlsx', sheets=c("survey1", "survey2"), header.sheets=c(FALSE, TRUE))
```

With `header.sheets` we say for which imported sheets first row should be used for labelling data frame. In our case this is resp. `FALSE`, `TRUE` for "survey1", "survey2". In fact:

```
surv
```

```
## $survey1
##       1     2
## 1  Boys Girls
## 2    10    20
## 3    30    40
## 6 Young   Old
## 7  <NA>    20
## 8    30    40
##
## $survey2
##   EU US
## 2 10 20
## 3 30 40
```

```
names(surv$survey2)
```

```
## [1] "EU" "US"
```

There are no more digits for data frame names, but the names, taken from the first sheet row, are `"EU" "US"`.

`header.sheets` are recycled. So `sheets=c("survey1", "survey2"), header.sheets=TRUE` means that for both “survey1” and “survey2” the first row will be used for labeling.

Note that the letter case is the same as the issued command, therefore now it is lowercase, while before it was the original case used in the file.

It is possible to query for named ranges too. The file `survey.xlsx` contains the named ranges `education` and `students`.

```
surv= read.xlx('survey.xlsx', ranges=c('education', 'students'))
```

```
surv
```

```
## $education
##       1   2
## 6 Young Old
## 7  <NA>  20
## 8    30  40
##
## $students
##      1     2
## 1 Boys Girls
## 2   10    20
## 3   30    40
```

Again we can use the first line as header:

```
surv= read.xlx('survey.xlsx', ranges='education', header.ranges=TRUE)
```

```
surv
```

```
##   Young Old
## 7  <NA>  20
## 8    30  40
```

Details for the non causal user

Sheets are converted into data frame following other `read.*` function behaviour, which means that the values of a column share a common type. Anyway in the same Excel column different cells can have different formats. Why loosing this information? It would have been possible to use a list object to model a sheet and so retain the differences, but most of the R statistic functions can effectively operate when at least at column level the formats are the same. That being said, when in a column there are different cell formats the prevailing compatible styles, will be applied to all. This willoften involve a the “character”, because it is always compatible with numeric formats too.

Given this recognised Excel style as R equivalent are:

- number, accounting, currency, fraction, scientific: converted to R numeric format
- percent: converted to R numeric format, with “percent” attribute
- date: converted to R date or datetime format
- time: converted to R time format

- text: converted to R character format
- general: converted to R character format

To see how this works in practice:

```
surv=read.xlx('survey.xlsx', "survey2")
```

```
surv
```

```
##    1  3
## 1 EU US
## 2 10 20
## 3 30 40
```

If the Excel user has not set a specific cell style. All cells have the Excel "general" format and the general format is mapped to R "character".

Let us assume that the user has explicitly set the values in row 2 and 3 to the Excel number style. Given the previous command

```
 surv[[1]]
```

```
## [1] "EU" "10" "30"
```

```
class(surv[[1]])
```

```
## [1] "character"
```

Numeric values 10 and 30 are converted to character to be compatible with the string "EU".

If you want to reduce the progress messages printed (perhaps because you are using a number of batch jobs), use:

```
suppressMessages( x=read.xlx('survey.xlsx') )
```

You will only get one line of +'s. I am thinking if it is convenient to totally abolish it.