Package 'muRL'

June 13, 2017

Type Package

Title Mailmerge using R, LaTeX, and the Web

Version 0.1-11

Date 2017-06-12

Author Ryan T. Moore <rtm@american.edu> and Andrew Reeves <reeves@wustl.edu>

Maintainer Ryan T. Moore <rtm@american.edu>

Depends utils, maps, stringr

Imports graphics

Description Provides mailmerge methods for reading spreadsheets of addresses and other relevant information to create standardized but customizable letters. Provides a method for mapping US ZIP codes, including those of letter recipients. Provides a method for parsing and processing html code from online job postings of the American Political Science Association.

License GPL-2 | file LICENSE

URL http://www.ryantmoore.org/software.murl.html

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2017-06-13 06:55:35 UTC

R topics documented:

`murl-package` *Mailmerge methods for reading addresses, creating letters, and mapping US ZIP codes of recipients*

Description

Provides mailmerge methods for reading spreadsheets of addresses and other relevant information to create standardized but customizable letters. Provides a method to map US ZIP codes, including those of letter recipients. Provides a method for parsing and processing html code from online job postings of the American Political Science Association.

Details

Package:	muRL
Type:	Package
Version:	0.1-11
Date:	2017-06-12
License:	GPL version 2.0 or newer
URL:	http://www.ryantmoore.org/software.murl.html
LazyLoad:	yes

Author(s)

Ryan T. Moore <rtm@american.edu> and Andrew Reeves <reeves@wustl.edu>

Maintainer: Ryan T. Moore <rtm@american.edu>

`apsahtml2csv` *Read, parse, and write to a .csv file APSA eJobs html files*

Description

Reads American Political Science Association (APSA) "eJobs" html files, parses the content of these files into a format for muRL to read, and writes that content to a .csv file.

Usage

```
apsahtml2csv(directory, file.name, file.ext = ".htm", verbose = TRUE)
```

Arguments

directory	a character string specifying the directory to which a set of APSA job announcement web pages have been downloaded.
file.name	a character string specifying the name of the file to which the data should be written.
file.ext	a character string specifying the extension of the files from which the data will be harvested.
verbose	a logical specifying whether the file name and current working directory should be printed.

Details

After logging in to eJobs, the job announcement site of the American Political Science Association (APSA), the user can search for and find the APSA web page announcing a single job listing. The user can download the html from several such pages (usually with a simple "Save As" command, depending on one's operating system). apsahtml2csv then parses the html code from these pages, and sorts and stores the relevant content. A .csv file is written containing this content.

If the user downloads the APSA webpages using a different (or no) file extension, that extension (or "") should be specified using the file.ext argument. Because apsahtml2csv uses the value of file.ext in a grep command, we strongly recommend that the directory specified by directory include only the downloaded webpages, and no other files or directories.

Institutions are inconsistent in how they enter the names of their jobs' contact representatives. Thus, some tweaking of the output of apsahtml2csv may be required in order to create a .csv file that can be seemlessly read by read.murl. Specifically, the user may have to take the single column of the output of apsahtml2csv called contact, and create columns called title, fname, and lname. Additionally, the user may have to adjust the position and subfield columns, and institutions may report these somewhat differently.

Value

An R dataframe is created and a .csv file is written. These include columns containing the APSA job listing ID number, the date the job advertisement was posted, the type of institution, the title and subfield of the position, the start date, salary, and region, the name of the institution and department, the name, address, city, state, ZIP code, and phone number of the individual to contact, the department or institution's web address, and a full paragraph description of the position.

The full paragraph description is stored in a column named desc. Due to the current parsing strategy, this field may include some excess characters from the APSA html page.

Author(s)

Ryan T. Moore <rtm@american.edu> and Andrew Reeves <reeves@wustl.edu>

See Also

read.murl

murljobs *A sample dataframe of recipient information and addresses*

Description

This sample dataframe of recipient information and addresses includes columns with information required by the muRL package, as well as auxiliary columns with information related to a hypothetical mailmerge, but not required by muRL.

Usage

```
data(murljobs)
```

Format

A data frame with 8 observations on the following 15 variables.

`institution` a factor containing sample institution names (with levels `Christopher College`, ..., `University of State University`).

`type` an auxiliary factor for sorting sample entries (with level `am`).

`deadline` an auxiliary factor containing sample deadlines (with levels `1/5/2010`, `12/1/2009`).

`title` a factor containing sample recipient titles (with levels `Dean`, ..., `Sargent`).

`fname` a factor containing sample recipient first names (with levels `Frank`, ..., `Tim`).

`lname` a factor containing sample recipient last names (with levels `Anderson`, ..., `Smithers`).

`dept` a factor containing sample recipient information (with levels `Department of Political Science`, `Department of Politics`).

`position` a factor containing sample position titles (with levels `assistant professor`, ..., `postdoctoral associate`).

`subfield` a factor containing sample recipient information (with levels `American politics`, ..., `Governance Studies`).

`address1` a factor containing sample recipient address first lines (with level `Graduate Admissions Committee`).

`address2` a factor containing sample recipient address second lines (with levels `11 Smith Rd.`, ..., `Dept of Political Science`).

`address3` a factor containing sample recipient address third lines (with levels `123 Main St`, ..., `Dept of Rock Music`).

`city` a factor containing sample recipient cities (with levels `Allentown`, ..., `Topeka`).

`state` a factor containing sample recipient states or provinces (with levels `CA`, ..., `WY`).

`zip` a numeric vector containing sample recipient ZIP codes.

Source

Created by package authors.

Examples

```
data(murljobs)
```

read.murl *Read a .csv file or R dataframe of letter recipients*

Description

Reads a .csv file or R dataframe of letter recipients, processes the column names for write.murl, checks whether United States ZIP codes conform to standard formats, and reports potential problems to the user.

Usage

```
read.murl(file = "murljobs.csv", header = TRUE, stringsAsFactors =
FALSE, field.title = "title", field.fname = "fname", field.lname =
"lname", fields.address = "address", field.city = "city",
field.state = "state", field.zip = "zipcode", field.position =
"position", field.subfield = "subfield", field.dept = "dept",
field.institution = "institution", colClasses = c("character"), ...)
```

Arguments

file	the name of a .csv file or R dataframe.
header	a logical for whether the first row of the input file or dataframe is a header row.
stringsAsFactors	a logical for whether character strings should be stored as factors with levels.
field.title	a character string giving the name of the column containing recipients' titles (such as "Doctor", "Mrs.", etc.).
field.fname	a character string giving the name of the column containing recipients' first names.
field.lname	a character string giving the name of the column containing recipients' last names.
fields.address	a character string common to the name(s) of the column(s) containing recipients' street mailing address information. Each column will be printed as its own row in the mailing address. See Details below for more.
field.city	a character string giving the name of the column containing recipients' cities.
field.state	a character string giving the name of the column containing recipients' states or provinces.
field.zip	a character string giving the name of the column containing recipients' United States ZIP or other postal codes.
field.position	a character string giving the name of the column containing recipient-specific information, such as the specific position for which one is applying.
field.subfield	a character string giving the name of the column containing recipient-specific information, such as the specific subfield for which one is applying.

`field.dept` a character string giving the name of the column containing additional information, such as the specific department offering the position for which one is applying.

`field.institution` a character string giving the name of the column containing additional information, such as the institution offering the position for which one is applying.

`colClasses` a vector of character strings indicating the class of each column. Using `c(``character'')` ensures leading zeros in, for example, ZIP codes will be preserved.

`...` other arguments to pass to `read.csv()` if the input file is a .csv file.

Details

Recipients' addresses are formatted for mailing as follows. The first row contains the contents of the fields defined by `field.title`, `field.fname`, and `field.lname`. Each of the fields defined by `fields.address` is formatted as a unique row. The last row contains the contents of the fields defined by `field.city`, `field.state`, and `field.zip`.

`fields.address` specifies the string common to the names of the columns containing the recipients' street addresses. For example, if the user's file has the street address in columns named `addr1`, `addr2`, `...`, then the user should set `fields.address = "addr"`.

If the input file is an R dataframe, then the argument `...` is ignored.

Value

An R dataframe containing the relevant information for creating a set of standardized but customizable letters to be mailed.

Author(s)

Ryan T. Moore <rtm@american.edu> and Andrew Reeves <reeves@wustl.edu>

See Also

`write.murl`, `zip.plot`

Examples

```
## Specify path to .csv database of sample addresses
fpath <- system.file("extdata", "murljobs.csv", package = "muRL")

murljobs <- read.murl(fpath)
```

write.murl *Write a .tex file of recipient information to be processed by LaTeX*

Description

Reads an R dataframe of letter recipient- and position-specific data, such as the output of `read.murl`. Creates a `.tex` file of the relevant data and LaTeX code, which can then be processed directly by pdflatex, for example.

Usage

```
write.murl(object, file.name = "mailmerge.tex", salutation = "Dear",
sal.punct = ":", address.string = "123 Venus Flytrap Way\\\\Cincinnati,
OH 45201\\\\ \\texttt{jfever@wkrp.edu}\\\\
\\texttt{http://www.wkrp.edu/jfever}\\\\513-555-5664",
date = "\\today", letter.file = NULL, letter.text = NULL, valediction = "Sincerely,",
signature = "Johnny Fever", opening = "", include.opening = FALSE, verbose = TRUE)
```

Arguments

`object`	a dataframe of mailmerge data, such as an output from `read.murl()`.
`file.name`	a character string specifying the file name (and optionally, path) for the output `.tex` file. See Details below for more.
`salutation`	a character string specifying the salutation to be used in the letters.
`sal.punct`	a character string specifying the punctuation to be used at the end of the salutation.
`address.string`	a character string specifying the return address to be used in the letters. Note that two slashes (\\) should be used for every one usual LaTeX slash.
`date`	an optional character string specifying the date. Defaults to the current date.
`letter.file`	an optional character string specifying a file containing the body text of the letters. See Details below for more.
`letter.text`	an optional character string containing the body text of the letters. See Details below for more.
`valediction`	a character string specifying the valediction to be used in the letters.
`signature`	a character string specifying the signature to be used in the letters.
`opening`	a character string specifying the opening line to be used in the letters. See Details below for more.
`include.opening`	a logical indicating whether an opening, customized line is to be used in the letters. See Details below for more.
`verbose`	a logical indicating whether the `file.name` and the current working directory should be printed after the file has been created.

Details

The dataframe used by `write.murl` should include columns for recipients' titles, first names, last names, addresses, cities, states, and ZIP codes, as well as information specific to the position for which the letter is in application. `write.murl` is intended to operate on the output of `read.murl`, and thus requires that the column names for the fields above be "title", "fname", "lname", "address1" (and "address2", etc.), "city", "state", "zip", "position", "subfield", "dept", and "institution". These field names are automatically created by `read.murl`.

The user may define the main body text of the letter in at least three ways. First, `write.murl` includes some sample text by default. The user could simply edit this text in the `.tex` file created by `write.murl`. Second, the user could write the body text in a separate file (such as a `.txt` file) and specify that file's name using the `letter.file` argument. Third, the user could define the entire body text as a string passed to the `letter.text` argument.

If both `letter.file` and `letter.text` are specified, `write.murl` appends the value of `letter.string` below the contents of the file specified by `letter.file`.

The opening line specified by argument `opening` should be of a grammatical form consistent with "I write to apply for the position in". This phrase will then be followed by customized input, using the fields "position", "subfield", "dept", and "institution", as in the example in Value below. To omit such a customized opening line, set the `include.opening` toggle to `FALSE` (the default). The example in Value below, and thus each letter, will include only the content defined in the LaTeX-defined "body".

Value

A `.tex` file of LaTeX code and recipient-specific content, to be processed directly by LaTeX. Using the included `murljobs.csv` sample data, the `.tex` file created by `write.murl` includes for each position one code snippet that looks like the following:

```
\begin{letter}
{Dr. Richard Sanders\\Graduate Admissions Committee\\123 Hello Way\\Frederick MD 21701}
\opening{Dear Dr. Sanders:}
 \body
\closing{Sincerely,}
\end{letter}
```

Author(s)

Ryan T. Moore <rtm@american.edu> and Andrew Reeves <reeves@wustl.edu>

See Also

`read.murl`

Examples

```
data(murljobs)

## Create mailmerge.tex required for LaTeX import
write.murl(murljobs)
```

```
## Specify a file containing the letters' body text
## write.murl(murljobs, letter.file = "mybodytext.txt")

## Specify a string containing the letters' body text
write.murl(murljobs, letter.text = "This is the whole body of my letters.")

## Specify salutation, valediction options (overwrites previous mailmerge.tex)
write.murl(murljobs, file.name = "mailmerge.tex", salutation = "Greetings",
     sal.punct = ",", valediction = "Truly Yours,", include.opening = FALSE)

## Specify opening line also (overwrites previous mailmerge.tex)
write.murl(murljobs, file.name = "mailmerge.tex", salutation = "Greetings",
     sal.punct = ",", valediction = "Truly Yours,",
     opening = "I am applying for the job in", include.opening = TRUE)
```

`zip.plot` *Plot US ZIP codes, including locations of letter recipients.*

Description

Using United States ZIP codes, plots on a map the location of letter recipients. State or county boundaries may be displayed.

Usage

```
zip.plot(data, zip.file = system.file("extdata", "zips.tab", package =
"muRL"), map.type = "state", cex = 1, col = "black", pch = 20,
jitter.factor = NULL, ...)
```

Arguments

`data`	a dataframe with ZIP codes in a column named 'zip', such as the output of `read.murl`.
`zip.file`	a character string naming a `.tab` file with the columns for the latitude and longitude of ZIP codes, such as 'zips.tab' provided in the `murl` package (the default).
`map.type`	the type of map for `map()` from the `maps` library to create. See Details for more.
`cex`	a numerical value giving the amount by which plotting text and symbols should be magnified relative to the default. Accepts, for example, a vector of values which are recycled.
`col`	a specification for the plotting color.
`pch`	the plotting character for `map()` from the `maps` library to use.
`jitter.factor`	a numeric specifying by how much points should be jittered before plotting. See Details below for more.
`...`	other arguments to pass to `map()` from the `maps` library. See Details below for more.

Details

map.type can be any valid map from the maps package. For plotting the location of United States ZIP codes, usa, state, or county should be used.

See help(par) for more details on cex, col, and pch.

See help(jitter) for more details on jitter.factor. zip.plot jitters latitude and longitude separately using the same factor.

To plot only a region within the selected map.type, include the map argument region = . For example, zip.plot(..., region = ``Maryland'') would plot only the recipients with ZIP codes in the US state of Maryland.

Note

zip.plot calls the map function in the maps package. The map function places an object called stateMapEnv in the user's workspace.

Author(s)

Ryan T. Moore <rtm@american.edu> and Andrew Reeves <reeves@wustl.edu>

See Also

read.murl, zips

Examples

```
## Call murl object of sample addresses
data(murljobs)
zip.plot(murljobs)

## Read .csv to murl object
murljobs <- read.murl(system.file("extdata", "murljobs.csv", package = "muRL"))
## Specify US state to map
zip.plot(murljobs, map.type = "state", region = "maryland")
```

---

zips *A .tab file of US ZIP code data for mapping recipients*

Description

A .tab file of United States ZIP code data for mapping recipients. Called by zip.plot to match ZIP codes from letters to latitude and longitude coordinates, and then plot latitudes and longitudes on a user-selected map type.

Usage

```
data(zips)
```

Format

A data frame with 33309 observations on 4 variables.

`state` a factor containing state and territory abbreviations (with levels `AK`, `AL`, ..., `WY`).

`zip` a factor containing three-digit, four-digit, five-digit, and three-digit-plus-wildcard formatted ZIP codes (with 33188 levels).

`lat` a numeric vector of latitude coordinates.

`lon` a numeric vector of longitude coordinates.

Details

A few ZIP codes span more than one state, and thus appear more than once in `zips`. See the Examples below for hints on extracting latitude and longitude.

Note

Not all US ZIP codes are currently included in this file. If you have a ZIP code you would like included for plotting, please email the package maintainer with the following four pieces of information: the state in which the ZIP code is located, the ZIP code itself, the latitude of the ZIP code to six decimal places (such as 38.643248), and the longitude of the ZIP code to six decimal places (such as -75.611025). Please also provide the city and any other information required to verify the latitude and longitude for inclusion.

Source

The original file upon which `zips.tab` is based is available at `http://www.census.gov/ tiger/tms/gazetteer/zcta5.t`
which is linked from `http://www.census.gov/ geo/www/gazetteer/places2k.html`. The US Census Bureau's Geography Division produced these documents. A few additions to the originals have been made. See the `muRL` CHANGELOG for details.

References

Further information about ZIP Code Tabulation Areas (ZCTAs) is available at

`http://www.census.gov/geo/ZCTA/zcta.html`.

Examples

```
data(zips)

summary(zips$lat)
summary(zips$lon)

## Extracting latitude and longitude.
## Create a sample survey data frame with an ID variable,
##   respondent ZIP code, state, and survey response:
svy1 <- data.frame(id = c(1,2,3,4), zip = c("10001", "10001", "63130", "380HH"),
          state = c("NY", "NY", "MO", "AR"), resp = c(1,2,1,5))
svy1
## Since ZIP 380HH spans three states, all are included:
```

```
svy2 <- merge(svy1, zips, by = "zip", all.x = TRUE)
svy2
## Merging by ZIP and state omits the duplicate 380HH entries:
svy3 <- merge(svy1, zips, by = c("zip", "state"), all.x = TRUE)
svy3
```

Index