Journal of
open research software

SOFTWARE
METAPAPER

]u[ubiquity press

mRpostman: An IMAP Client for R

ALLAN V. C. QUADROS

ABSTRACT

Internet Message Access Protocol (IMAP) clients are a common feature in several programming languages. Despite having some packages for electronic message retrieval, the R language, until recently, lacked a broader solution, capable of coping with different mail providers along with a wide spectrum of features. `mRpostman` covers most of the IMAP 4rev1 functionalities by implementing tools for message searching, selective fetching of message attributes, mailbox management, attachment extraction, and several other IMAP features that can be executed in virtually any mail provider. By doing so, it enables users to perform data analysis based on email content. The goal of this article is to showcase the toolkit provided with the mRpostman package, describe its key features, and provide some application examples.

CORRESPONDING AUTHOR:

Allan V. C. Quadros

Kansas State University, US

quadros@ksu.edu

KEYWORDS:

IMAP; email; R

TO CITE THIS ARTICLE:

Quadros AVC 2024 mRpostman: An IMAP Client for R. *Journal of Open Research Software,* 12: 4. DOI: https://doi.org/10.5334/jors.480

(1) OVERVIEW

INTRODUCTION

The acknowledgment of the R programming language [17] as having remarkable statistical capabilities is much due to the excellence brought by its statistical and data analysis packages. This reputation also stands on the capabilities of a myriad of utility packages, which extends the use of the language by facilitating the integration of the steps involved in data collection, analysis, and communication. With that in mind, and considering the amount of data transmitted daily through email, `mRpostman` was conceived to fill the gap of an Internet Message Access Protocol (IMAP) client in the R statistical environment; therefore, providing an appropriate toolkit for electronic messages retrieval, and paving the way for email data analysis in R.

The Comprehensive R Archive Network (CRAN) has at least seven packages for sending emails (Table 1). Whereas some of these packages aim to provide a plain Simple Mail Transport Protocol (SMTP) client for R (e.g. `sendmailR` and `emayili`), others focus on more sophisticated implementations, using Application Program Interfaces (API), or providing seamless integration between SMTP and other R features such as integration with `rmarkdown` [1]. However, despite the surplus of available clients in R, the SMTP protocol is not suitable for receiving emails. It only allows clients to communicate with servers to deliver their messages.

For the purpose of message retrieval, there exists the Post Office Protocol 3 (POP3) and the Internet Message Access Protocol (IMAP). In comparison with IMAP, POP3 is a very limited protocol, working as a simple interface for clients to download emails from servers. IMAP, on the other hand, is a much more complex protocol and can be considered as the evolution of POP3, with a very different and broader set of functionalities. In contrast to POP3, all the messages are kept on the IMAP server and not locally. This means that a user can access the same mail account using parallel connections from different clients [7]. Besides the mail folder management, the capability to issue complex search queries also contributes to the higher level of sophistication of the IMAP protocol.

Amid CRAN packages for email communication, only `gmailr` has IMAP capabilities (Table 1). However, those capabilities are restricted to Gmail accounts and a few IMAP functionalities. Although `gmailr` supports both protocols, the package is more SMTP-focused, which explains its low count of IMAP features. Therefore, R was clearly lacking a broader IMAP client solution. It was in that mainstay that `mRpostman` was implemented.

In this article, we present a brief view of the main functionalities of the package and its applications.

IMPLEMENTATION AND ARCHITECTURE

`mRpostman` was conceived to be an easy-to-use session-based IMAP client for R. The package implements intuitive methods for executing the majority of the IMAP commands described in the Request for Comments 3501,[1] such as mailbox management, and selective search and fetch of message attributes. The package also implements complementary functions for decoding quoted-printable and Base64 content,[2] following the MIME[3] specification.

All these methods and functions play an important role in facilitating email data analysis. We shall not overlook the amount of data analyses daily performed on email content. The package has proved to be very

	PROTOCOL	MAIL PROVIDERS	FEATURES				ACTIVE DEVELOPMENT
			SEARCH QUERIES	MESSAGE FETCH	ATTACHMENT EXTRACTION	MAILBOX MANAGEMENT	
sendmailR [12]	SMTP	all	–	–	–	–	yes
mailR [15]	SMTP	all	–	–	–	–	yes
mail [9]	SMTP	all	–	–	–	–	no
blatr [2]	SMTP	all	–	–	–	–	no
blastula [10]	SMTP	all	–	–	–	–	yes
emayili [5]	SMTP	all	–	–	–	–	yes
gmailr [8]	SMTP/IMAP	Gmail	no	limited	limited	no	yes
mRpostman	IMAP	all	yes	yes	yes	yes	yes

Table 1 Comparison of the currently available CRAN packages for email communication. The following attributes are evaluated: protocol – the supported protocol (SMTP or IMAP); mail providers – if the IMAP protocol is supported, which mail providers are supported by the package; Features – which type of IMAP features are available in the package; active development – if the package is currently under active development. If the package does not provide IMAP support, the remaining fields do not apply.

useful as an additional feature in this workflow by, for instance, enabling the possibility of automating the attachment retrieval step. Additionally, by fetching other message contents, users can, for example, apply statistical techniques to analyze the frequency of emails by sender or subject, run sentiment analysis on email content, etc.

Because `mRpostman` works as a session-based IMAP client, one can think of the provided methods following a natural order in which the steps shall be organized in the event of an IMAP session (Figure 1). For instance, if the goal is to search messages within a specific period and/or containing a specific word, first we need to configure the connection to the IMAP server; then, choose a mail folder where the search is to be performed; and execute the single criteria (left) or the custom multi-criteria search (right). If the user intends to fetch the matched message(s) or its parts, additional fetch steps can be chained together to the described schema.

`mRpostman` is flexible in the sense that the aforementioned steps can be used either under the tidy framework, with pipes [3], or via the conventional base R approach.

CONSTRAINTS AND FUTURE DEVELOPMENT

Because IMAP is such a complex protocol, this package is in constant development, meaning that new features described in the RFC 3501 are to be implemented in future versions. As of version 1.1.0, `mRpostman` does not have support for the following IMAP4rev1 client commands: `NOOP` – used to reset any inactivity auto-logout timer on the server; `LSUB` – returns a subset of names from the set of names that the user has declared as being "active" or "subscribed"; `SUBSCRIBE/UNSUBSCRIBE` – adds/remove the specified mailbox name to the server's set of "active" or "subscribed" mailboxes returned by the `LSUB` command; `STATUS` – requests the status of the indicated mailbox; `APPEND` – appends the literal argument as a new message to the end of the specified destination mailbox; `CHECK` – requests a checkpoint of the currently selected mailbox; `CLOSE` – permanently removes all messages that have the "\Deleted" flag set from the currently selected mailbox, and returns to the authenticated state from the selected state.

DEMONSTRATION OF FUNCTIONALITY

Configuring an IMAP connection

As we demonstrated in Figure 1, the first step when using `mRpostman` is to configure an IMAP connection. It consists of creating a connection-token object of class `ImapCon` that will retain all the relevant information to issue requests to the server. `configure_imap` is the function used to configure and create a new IMAP connection. There are three mandatory string arguments: `url`, `username`, and `password` for plain authentication; or `url`, `username`, and `xoauth2_bearer` for OAuth2.0 authentication.[4] The following example illustrates how to configure a connection to a Microsoft Exchange IMAP 4 server; more specifically, to an Office 365 Outlook account using plain authentication.

```
library("mRpostman")
con <- configure_imap(url = "imaps://outlook.office365.com",
                      username = "user@agency.gov",
                      password = rstudioapi::askForPassword())
```

We opted for using an Outlook Office 365 account as an example to highlight the difference between `mRpostman`

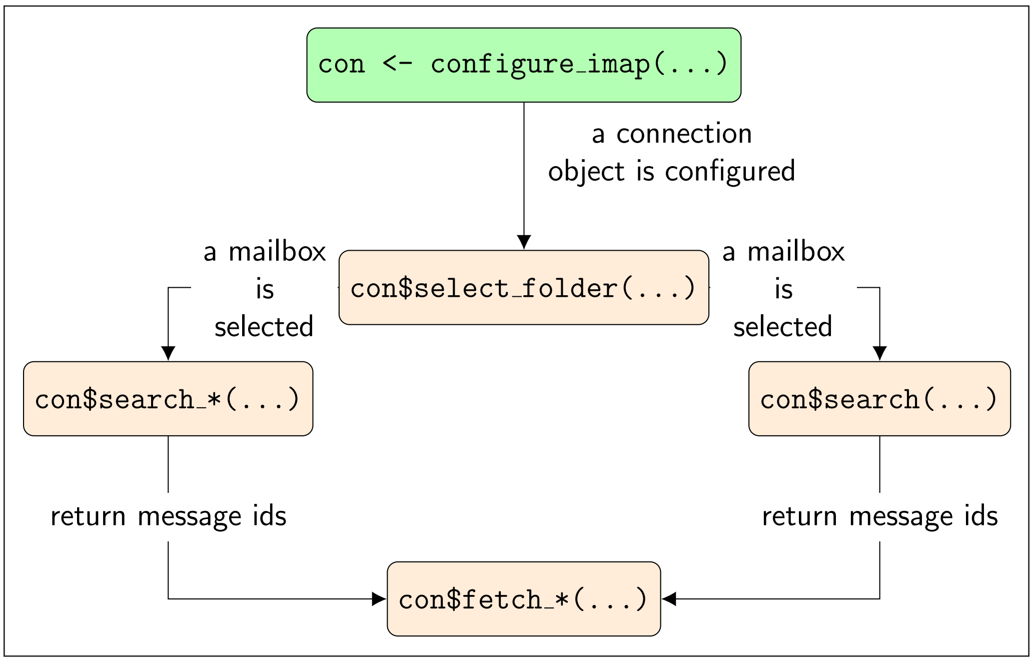

Figure 1 Basic schema for fetching the full content of a message or its parts after a search query.

and the other two CRAN packages which, although also capable of receiving emails, are restricted to Gmail accounts and have fewer IMAP functionalities. Although `mRpostman` can theoretically connect to any mail provider,[5] the Outlook Office 365 service is broadly used by universities and companies. This enriches the range of data analysis applications of this package, thus justifying our choice.

In a hypothetical situation where the user needs to simultaneously connect to more than one email account (in different providers or not) in the same R session, this can be easily attained by creating and configuring multiple connection tokens, such as `con1`, `con2`, and so on.

SELECTING A MAIL FOLDER

Mailboxes are structured as folders in the IMAP protocol. This allows us to mimic many of the operations executed in a local directory such as creating, renaming, or deleting folders. As messages are kept inside the mail folders, users need to select one of the folders whenever they intend to execute a search, fetch, or other message-related operation, as presented in Figure 1.

In this sense, the `select_folder` method is one of the key features of this package. It selects a mail folder for the current IMAP section. The mandatory argument is a character string containing the `name` of the folder to be selected.

Assuming we want to select the "`INBOX`" folder and considering that we are going to use the same connection object (`con`) that has been previously created, the command would be:

```
con$select_folder(name = "INBOX")
```

Further details on mailbox management features are provided with the package's official documentation (16).

MESSAGE SEARCH

The IMAP protocol is designed to allow single or multi-criteria queries on the mailboxes. This package implements a vast range of IMAP search commands, which consists of a critical feature for performing data analysis on email content.

As of its version 1.1.0, `mRpostman` has five types of single-criterion search methods implemented: by date; string; flag, size; and time span (`WITHIN` extension).[6] The custom search, on the other hand, enables the execution of multi-criteria queries by allowing the combination of two or more types of search. However, in this article, we will focus on the single-criterion search-by-string type.

The `search_string` method searches for messages containing a specific string or expression. One or more specific sections of a message, such as the `TEXT` section or the `TO` header field, for example, must be specified.

In the following code snippet, we search for messages from senders whose mail domain is "`@ksu.edu`".

```
ids <- con$search_string(expr = "@ksu.edu", where = "FROM")
```

The resulting object is a vector containing the matched unique IDs (UID) or the message sequence numbers[7] such as presented below:

```
 [1]  60 145 147 159 332 333 336 338 341 428
```

Further details on the other single-search methods and the custom-search method available in this package are provided in [16].

MESSAGE FETCH

After executing a search query, users may be interested in fetching the full content or some part of the messages indicated in the search result. In this regard, the package implements six types of fetching features:

- `fetch_body` – fetches the message body (message's full content) or a specified MIME level, which can refer to the text or the attachments if there are any.
- `fetch_header` – fetches the message header, which comprises all the components of the `HEADER` section of a message. Besides the traditional ones (from, to, cc, subject), it may include several more fields.
- `fetch_metadata` – fetches the message metadata, which consists of some message attributes such as the internal date, and the envelope (from, to, cc, and subject fields).
- `fetch_text` – fetches the message text section, which can comprise attachment MIME levels if applicable.

Each of these methods can be seamlessly integrated into a previous search operation so that the returned IDs are used as input for the fetch method.

Above all, these methods consist of a powerful tool of information retrieval for performing data analysis on email content. Here, we mimic the extraction of the `TEXT` portion of a message. Although there is a `fetch_text` method, the recommended approach is to use `fetch_body(…, mime_level = 1L)` because the former may collect attachment parts along with the message text in some situations.

```
out <- ids %>%
  fetch_body(mime_level = 1L)
```

Once the messages' content is fetched, the text can be cleaned and decoded with the `clean_msg_text` helper function. A subsequent call to the base R function `writeLines` produces a neat printing of the fetched text.

```
cleaned_text <- clean_msg_text(msg_list = out)
writeLines(cleaned_text[[1]])
```

```
Receipt Number: XXXXXXX
Customer: Vieira de Castro Quadros, Allan
Kansas State University
Current Date: 04/15/2020
Description                                                     Amount
----------------------------------------------------------------------------
HOUSING & DINING                                               $30.00
     User Number: XXXXXXXXX
                                                 Total          $30.00
Payments Received                                               Amount
----------------------------------------------------------------------------
07 CREDIT CARD PAYMENTS                                        $30.00
     Visa XXXXXXXXXXXX8437
     Authorization # XXXXXX
                                                 Total          $30.00
Thank you for the payment.
```

Besides other applications, the exported function `clean_msg_text` can be used to decode hexadecimal and Base64 characters in the text and other parts of the message. In some locales such as French, German, or Portuguese-speaking countries, message parts may contain non-ASCII characters. SMTP servers, then, encode it using the RFC 2047 specifications when sending the email. In these cases, `clean_msg_text` is capable of correctly decoding the non-ASCII characters.

ATTACHMENT EXTRACTION

In its pretension to be an IMAP client for R, `mRpostman` provides methods that enable users to list and download message payloads. This feature can be particularly critical for automating the analysis of attachment data files, for example.

Attachments can be downloaded using two different approaches in this package: extending the `fetch_text/fetch_body` operation by adding an attachment extraction step at the end of the workflow with `get_attachments`; or directly fetching attachment parts via the `fetch_attachments` method. In this article, we focus on the first type of attachment method, adding a step to our previous workflow.

The `get_attachments` method extracts attachment files from the fetched messages and saves these files to the disk. In the following code excerpt, we extract attachments in a unique pipeline that gathers fetching and search steps.

```
con$search_string(expr = "@ksu.edu", where = "FROM") %>%
  con$fetch_text() %>%
  con$get_attachments()
```

During the execution, the software locally saves the extracted attachments into sub-folders inside the user's working directory. These sub-folders are named following the messages' IDs. The attachments are placed into the respective message sub-folders as demonstrated in Figure 2. Note that the parent levels are

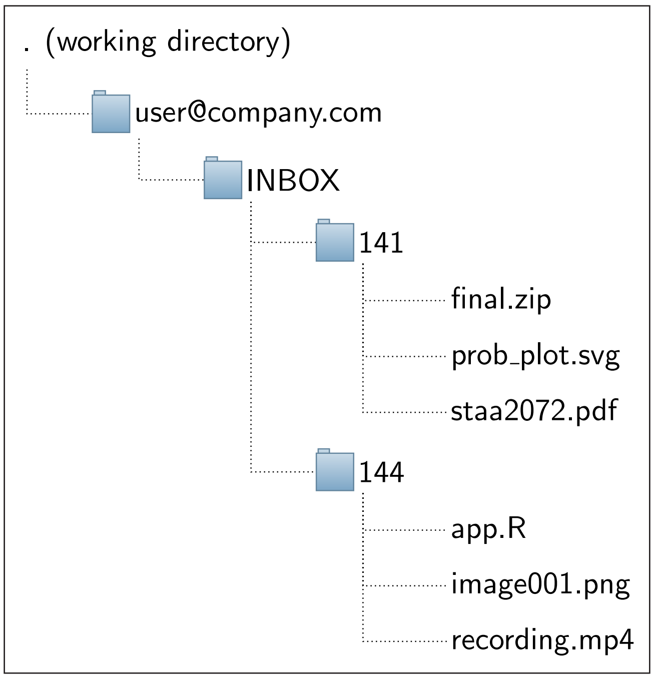

Figure 2 Local directory tree for the extracted attachment files.

named after the informed username and the selected mail folder.

For more information on other attachment-related methods, the reader should refer to the package's official documentation in [16].

QUALITY CONTROL

Tests for `mRpostman` should be performed on the user's real email data, where the existent messages on the mail provider webpage or app can be compared with the results returned by the package's functions and methods. Development-wise, new tests on real email data are performed with each software update and on different mail providers, such as GMail, Yandex, AOL, Hotmail, Outlook/Office 365, and Yahoo. These tests check for all the available package functionalities, when applicable.[8]

One of the critical aspects to make sure that the package works correctly is to establish a connection with the IMAP server successfully. For this reason, practical reproducible tests are not available, given that mimicking a connection to an IMAP server to simulate searching or fetching of fake message content would be innocuous for testing purposes. Nonetheless, the use of the GitHub Actions continuous integration (CI) platform allows automatic building and compilation testing after each change of the source code uploaded to `mRpostman's` main repository. The tests on GitHub Actions have been performed using different platforms, such as MacOS, Ubuntu, and Windows with different R versions (old, release, and development).

As for the package support, users may raise issues on GitHub or StackOverflow for additional assistance on the package functionalities.

LANGUAGE

English

(2) REUSE POTENTIAL

To demonstrate the capabilities of the proposed software, we explore two use cases of this package in support of data analysis tasks: a simple study of the frequency of emails grouped by senders and filtered by date; and a sentiment analysis that was run on a set of emails also filtered by a specific period. The R scripts needed for reproducing these examples are provided in the following subsections. Although the results cannot be exactly reproduced once they reflect the author's mailbox contents, they can be easily adapted to the reader's context.

FREQUENCY ANALYSIS OF EMAIL DATA

In the first example, we run a simple analysis of the email frequency by senders. This can be especially useful in areas such as marketing and customer service departments. A period of analysis was defined, and a search-by-date was performed using the `search_period` method. Then, senders' information for the returned IDs is fetched via `fetch_metadata`, using the `ENVELOPE` attribute. After some basic manipulation with regular expressions, the data is ready to be plotted as shown in Figure 3.

The same kind of analysis can be replicated for message subjects with only a few modifications in the regular expression code chunks. Considering that some companies/users deal with subject-standardized emails, this approach can be useful in analyzing the frequency of emails regarding different categories of subjects.

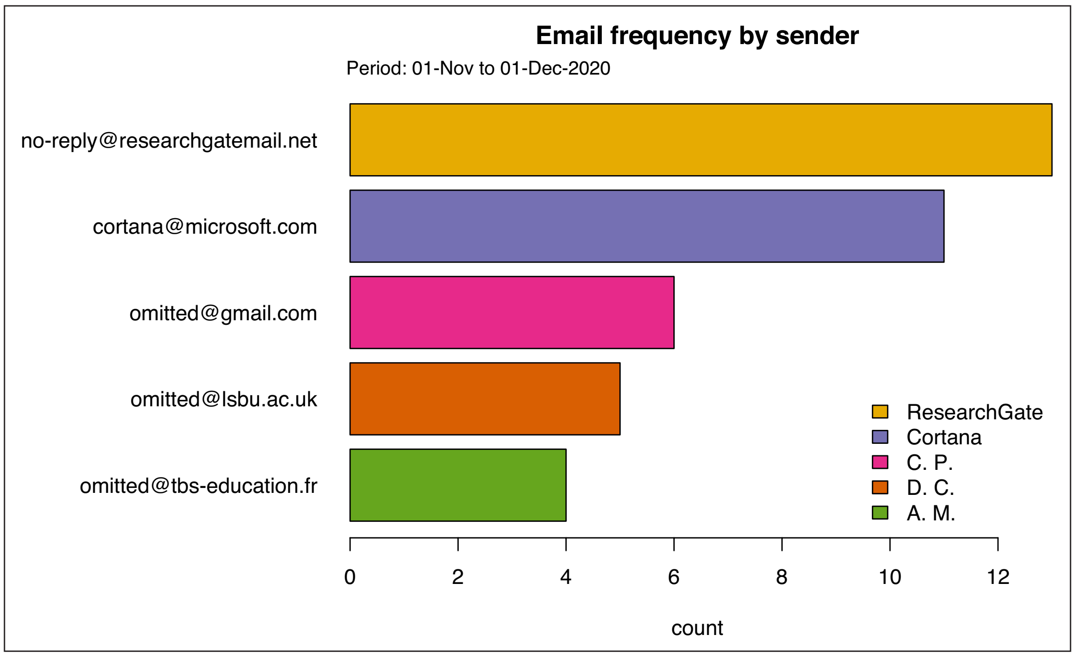

Figure 3 An example of email frequency analysis grouped by sender. Some of the names and emails were anonymized for privacy reasons.

```
library(mRpostman)
con <- configure_imap(
  url="imaps://outlook.office365.com",
  username="user@company.com",
  password=rstudioapi::askForPassword()
)
con$select_folder(name = "INBOX")
meta_res <- con$search_period(since_date_char = "01-Nov-2020",
                              before_date_char = "01-Dec-2020") %>%
  con$fetch_metadata(attribute = "ENVELOPE")
# cleaning
# step 1
clean_meta <- lapply(meta_res, function(x){
  regmatches(x, regexpr(pattern = "\\(\\(.*\"(.*?)\"\\)\\)", x, perl = TRUE))
})
# step 2
# cleaning Ccs
senders1 <- lapply(clean_meta, function(x){
  gsub(")) NIL .*$|)) .*$|))$", "", x)
})
# step 3
senders1 <- lapply(senders1, function(x){
  gsub('^\\(\\(|\"+', "", x)
})
# splitting
name <- c()
email <- c()
for (i in seq_along(senders1)){
  # i = 1
  out <- unlist(strsplit(senders1[[i]], " NIL "))
  name <- c(name, out[1])
  email <- c(email, gsub(" ", "@", out[2]))
}
df <- data.frame(name, email)
df$name <- decode_mime_header(string = as.character(df$name))
df2 <- as.data.frame(table(df$email))
colnames(df2) <- c("email", "count")
df2 <- df2[order(-df2[,2]), ][1:5,]
df2$name <- unique(df$name[df$email %in% df2$email])
par(mar=c(5,13,4,1)+.1)
pal_cols <- rev(c("#e6ab02", "#7570b3", "#e7298a", "#d95f02", "#66a61e"))
barplot(rev(df2$count),
        main = "Email frequency by sender",
        xlab = "count",
        names.arg = rev(df2$email),
        las = 1,
        col = pal_cols,
        horiz = TRUE)
mysubtitle <- "Period: 01-Nov to 01-Dec-2020"
legend(x = "bottomright", legend = df2$name, fill = rev(pal_cols), bty = "n",
       y.intersp = 1)
mtext(side=3, line=0.3, at=-0.07, adj=0, cex=0.9, mysubtitle)
```

SENTIMENT ANALYSIS ON EMAIL DATA

For the sentiment analysis example, we also define a period of analysis and run a `search_period` query. Then, we retrieve the text part of the messages by fetching the first MIME level with `fetch_body(…, mime_level = 1L)`. The texts go through a first cleaning step with a call to the `clean_msg_text` function. After a few additional cleaning procedures, we use a lexicon [13] via the `syuzhet` [11] package to evaluate the sentiment of each email. The output below is a subset of the resulting data frame. The last two columns indicate, respectively, the counts of negative and positive words for each message. The other columns provide counts related to more specific emotions, which are not necessarily positive or negative.

```
library(mRpostman)
con <- configure_imap(url="imaps://outlook.office365.com",
                      username="user@company.com",
                      password=rstudioapi::askForPassword(),
                      timeout_ms = 20000
)
con$select_folder("INBOX")
ids <- con$search_period(since_date_char = "10-Oct-2020",
                         before_date_char = "20-Dec-2020")
fetch_res2 <- ids %>%
  con$fetch_body(mime_level = 1L)
cleaned_text_list <- clean_msg_text(msg_list = fetch_res2)
cleaned_text_list[[4]]
for (i in seq_along(cleaned_text_list)) {
  clean_text <- gsub("\r\n", " ", cleaned_text_list[[i]])
  clean_text <- unlist(strsplit(clean_text, " "))
  words <- clean_text[!grepl("\\d|_|http|www|nbsp|@|(?<=[[:lower:]])(?=[[:upper:]])",
                                 clean_text, perl = TRUE)]
  words <- tolower(gsub("\\W+", "", words))
  words <- gsub('[^a-zA-Z|[:blank:]]', "", words)
  cleaned_text_list[[i]] <- paste(words, collapse = " ")
}
cleaned_text_df <- do.call("rbind", cleaned_text_list)
library(syuzhet)
email_sentiment_df <-get_nrc_sentiment(cleaned_text_df)
rownames(email_sentiment_df) <- rownames(cleaned_text_df)
head(email_sentiment_df,10)
```

```
        anger anticipation disgust fear joy sadness surprise trust negative positive
body91      1            5       1    1   2       2        0     9        1       13
body92      0            1       0    0   1       0        0     3        0        1
body93      0            3       0    2   0       1        2     2        1        3
body94      0            1       0    1   0       0        1     4        1        4
body95      0            5       0    0   3       0        2     8        0       13
body96      0            0       0    0   0       0        0     0        0        0
body97      4           20       4   11  13      11        4    25       16       51
body98      0            3       0    0   2       0        1     4        0        6
body99      3            9       1    6   1       5        2    16       14       24
body100     4           15       1   13   6       7        6    15       16       31
```

CONCLUSIONS

`mRpostman` aims to provide an easy-to-use IMAP client for R. Its object-oriented design [4] allows efficient, elegant, and intuitive execution of several IMAP commands on a wide range of mail providers. As a result, users cannot only manage their mailboxes but also conduct email data analysis from inside R seamlessly integrating all the steps involved.

(3) AVAILABILITY

OPERATING SYSTEM

MacOS (≥13.10), Ubunutu (≥18.04), Windows (≥8.0).

PROGRAMMING LANGUAGE

R (≥3.1.0)

ADDITIONAL SYSTEM REQUIREMENTS

None

DEPENDENCIES

libcurl (≥ 7.65) [19].

LIST OF CONTRIBUTORS

None

SOFTWARE LOCATION

Archive

Name: CRAN
Persistent identifier: https://cran.r-project.org/package=mRpostman
Licence: GPL-3
Publisher: Allan V. C. Quadros
Version published: 1.1.0
Date published: 27/10/2022

Code repository GitHub

Name: mRpostman
Persistent identifier: https://github.com/allanvc/mRpostman
Licence: GPL-3
Date published: 27/07/2023

NOTES

1 The RFC 3501 [6] is a formal document from the Internet Engineering Task Force (IETF) specifying standards for the IMAP, Version 4rev1 (IMAP4rev1).

2 The RFC 2047 [14] specifies rules for encoding and decoding non-ASCII characters in electronic messages.

3 The Multiple Internet Mail Extensions (MIME) is an internet standard defined by the IETF in a series of Requests for Comments manuals, the RFCs 2045, 2046, 2047, 2048, 4288, 4289, and 2049. The so-called MIME standard *"specifies a standard format for encapsulating multiple pieces of data into a single Internet message"* [20]. This standard comprises all parts of an email message, ranging from the header to attachment files.

4 Please refer to the *"IMAP OAuth2.0 authentication in mRpostman"* vignette in [16].

5 Besides Outlook Office 365, the package has been successfully tested with Gmail, Yahoo, Yandex, AOL, and Hotmail accounts.

6 The WITHIN extension is not supported by all IMAP servers. A call to the list_server_capabilities method will present all the IMAP extensions supported by the mail provider [16].

7 Further details on the message identification methodology deployed by the IMAP protocol are provided in [18, 6, 16].

8 IMAP capabilities may vary among mail providers. Users can list all the IMAP4rev1 capabilities supported by the server using the list_server_capabilities method.

ACKNOWLEDGEMENTS

The author would like to acknowledge the Department of Statistics at Kansas State University (K-State) for his doctoral studies scholarship. He wants to especially thank Dr. Michael Higgins, Dr. Christopher Vahl, Dr. Perla Reyes, and Dr. Brian Silverstein for their mentorship and academic support. The author also acknowledges the intellectual guidance of Dr. George von Borries and Dr. André Cançado at Universidade de Brasilia (UnB). The contents of this article are the responsibility of the author and do not reflect the views of K-State or UnB.

COMPETING INTERESTS

The author has no competing interests to declare.

AUTHOR AFFILIATIONS

Allan V. C. Quadros orcid.org/0000-0003-3250-5380
Kansas State University, US

REFERENCES

1. **Allaire J, Xie Y, McPherson J, Luraschi J, Ushey K, Atkins A, Wickham H, Cheng J, Chang W, Iannone R.** *rmarkdown: Dynamic Documents for R*. R package version 2.5; 2020. URL: https://CRAN.R-project.org/package=rmarkdown.
2. **Bache SM.** *blatr: Send Emails Using 'Blat' for Windows*. R package version 1.0.1; 2015. URL: https://CRAN.R-project.org/package=blatr.
3. **Bache SM, Wickham H.** *magrittr: A Forward-Pipe Operator for R*. R package version 1.5; 2014. URL: https://CRAN.R-project.org/package=magrittr.
4. **Chang W.** 2020. *R6: Encapsulated Classes with Reference Semantics*. R package version 2.5.0; 2020. URL: https://CRAN.R-project.org/package=R6.
5. **Collier AB.** *emayili: Send Email Messages*. R package version 0.4.4; 2020. URL: https://CRAN.R-project.org/package=emayili.
6. **Crispin M.** 'Internet message access protocol – version 4rev1'. Request for Comments 3501 (RFC 3501), Internet Engineering Task Force (IETF); 2003. URL: https://tools.ietf.org/html/rfc3501.
7. **Heinlein P, Hartleben P.** *The Book of IMAP: Building a Mail Server with Courier and Cyrus*, No Starch Press; 2008.
8. **Hester J.** *gmailr: Access the 'Gmail' 'RESTful' API*. R package version 1.0.0; 2019. URL: https://CRAN.R-project.org/package=gmailr.
9. **Himmelmann L.** *mail: Sending Email Notifications from R*. R package version 1.0; 2011. URL: https://CRAN.R-project.org/package=mail.
10. **Iannone R, Cheng J.** *blastula: Easily Send HTML Email Messages*. R package version 0.3.2; 2020. URL: https://CRAN.R-project.org/package=blastula.
11. **Jockers ML.** *Syuzhet: Extract Sentiment and Plot Arcs from Text*. R package version 1.0.4; 2015. URL: https://CRAN.R-project.org/package=syuzhet.
12. **Mersmann O.** *sendmailR: send email using R*. R package version 1.2–1; 2014. URL: https://CRAN.R-project.org/package=sendmailR.
13. **Mohammad S, Turney P.** Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In: *'CAAGET '10: Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text'*, Los Angeles, California; 2010: 26–34. June, 2010. URL: http://saifmohammad.com/WebPages/lexicons.html.
14. **Moore K.** 'Multipurpose Internet Mail Extensions (MIME), part three: Message header extensions for non-ascii text'. *Request for Comments 2047 (RFC 2047), Internet Engineering Task Force (IETF)*; 1996. URL: https://tools.ietf.org/html/rfc2047.
15. **Premraj R.** *mailR: A Utility to Send Emails from R*. R package version 0.4.1; 2015. URL: https://CRAN.R-project.org/package=mailR.
16. **Quadros A.** *mRpostman: An IMAP Client for R*. R package version 1.1.0; 2022. URL: https://CRAN.R-project.org/package=mrpostman.
17. **R Core Team.** *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing; 2020. URL: https://www.R-project.org/.
18. **Resnick P.** 'Internet message format'. Request for Comments 5322 (RFC 5322). *Internet Engineering Task Force (IETF)*; 2008. URL: https://tools.ietf.org/html/rfc5322.
19. **Stenberg D.** 'libcurl – the multiprotocol file transfer library'. version 7.69.1; 2020. URL: https://curl.haxx.se/.
20. **Troost R, Dooner S, Moore K.** 'Internet message format'. Request for Comments 2183 (RFC 2183). *Internet Engineering Task Force (IETF)*; 1997. URL: https://tools.ietf.org/html/rfc2183.

TO CITE THIS ARTICLE:
Quadros AVC 2024 mRpostman: An IMAP Client for R. *Journal of Open Research Software,* 12: 4. DOI: https://doi.org/10.5334/jors.480

Submitted: 28 July 2023 **Accepted:** 21 February 2024 **Published:** 07 March 2024