Brian Law is a customer success representative at RStudio and a new R Views contributor.

Jason Rich manages all US Data Engineering for PRA Group and studies computer science at Old Dominion University

There are times when it would be really nice to get an email from R. Maybe you have a long running job that you would just like to leave alone while you go off and do other things. When this happens, it would be nice if R notified you when it was done; something like calling o monitoring a process for anomalies. Usually, everything's fine, except when it's not, and then we want a "fire alarm" to go off in R. Below we'll walk through how to automate having R send you an: email, text message, `Slack` message, or `Microsoft Teams` message.

Email

```
library(blastula)
# First let's build a rich HTML email using library(blastula)
owl <- compose_email(body = md(c("Hello from Hogwarts.

 The polyjuice potion is complete!")))
# Second, store credentials to later send our message via smtp (using gmail here but could be
others)
create_smtp_creds_file(
  file = "gmail_creds",
  user = "name@gmail.com",
  provider = "gmail"
) # Note, a pop up will ask for the pwd for the user you provided
# Third, send the email (using gmail here but could be others)
owl %>%
  smtp_send(
    to = "someone@email.com",
    from = "name@gmail.com",
    subject = "Mischief Managed",
    credentials = creds_file("gmail_creds")
  )
```

Text

If you prefer to get notifications via text messages then you can use the old trick to send an email that will convert into a text message to a phone number. Each mobile phone plan provider has a slightly different format for how to do this and so the first step is to google "how t digit-phone-number@mms.att.net. Let's run through an example.

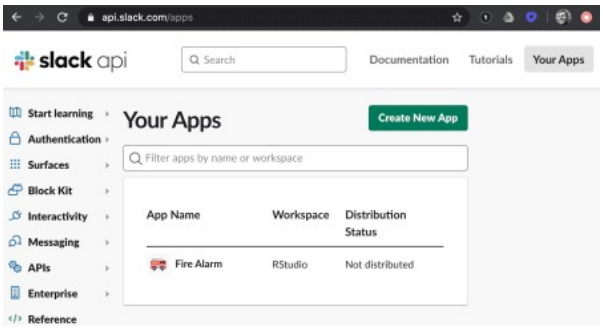
```
owl2 <- ""
owl2 %>%
  smtp_send(
    to = "xxxxxxxxx@mms.att.net",
    from = "name@gmail.com",
    subject = "Mischief Managed",
    credentials = creds_file("gmail_creds")
  )
```

Note that the text message above will only render the subject line currently but you can tinker further.

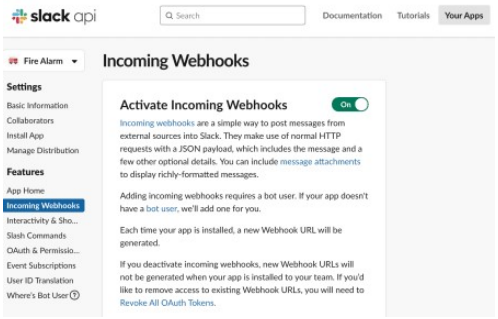
Slack

Who knew that chat rooms would make such a comeback? If you use `Slack` and want to send a message there, you can do so using R. Here we'll walk through a stripped down method that posts messages directly to the `Slack` API using an RStudio library `httr`. If you war more control and features.

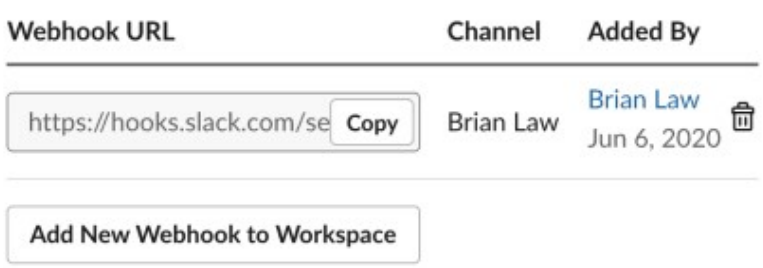
There are a few steps to set things up. First, you will need a Slack account. Second, go to api.slack.com/apps and login.



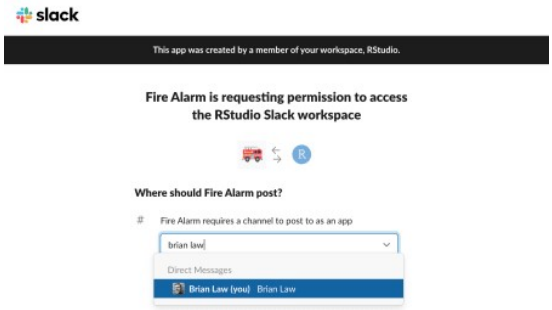
Third, click on "Create New App". Fourth, on the left sidebar, click on "Incoming Webhooks" and make sure "Activate Incoming Webhooks" is "On".



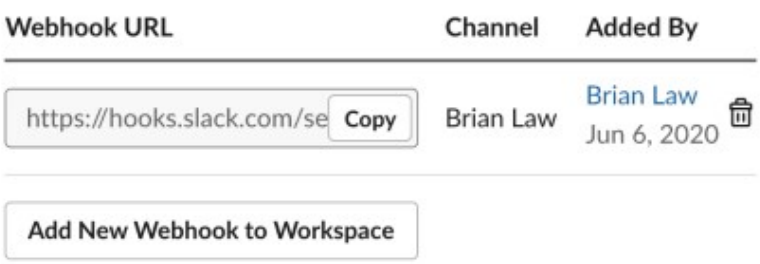
Then, scroll to the bottom of the page and click "Add New Webhook to WorkSpace".



Next, choose what `Slack` channel you want to generate a webhook to connect with, for example, your own `Slack` username, or a more general `Slack` channel at your company, like "cat_photos".



Lastly, click on the "Copy" button to copy the webhook, which is what we'll use in our R code below to actually send the message.

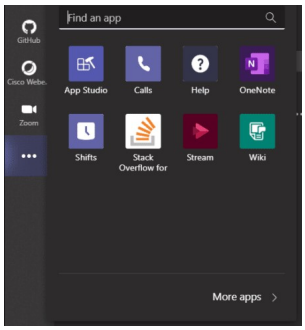



```
library(httr)
test_msg <- list(text="hello world!")
hook_to_me <- "https://hooks.slack.com/services/some_long_hash"
POST(hook_to_me, encode = "json", body = test_msg)
if (2 < 3) { # make this conditional aka a "fire alarm"
  hook_to_cats_channel <- "https://hooks.slack.com/services/some_long_hash"
  POST(hook_to_cats_channel, encode = "json", body = test_msg)
}
```

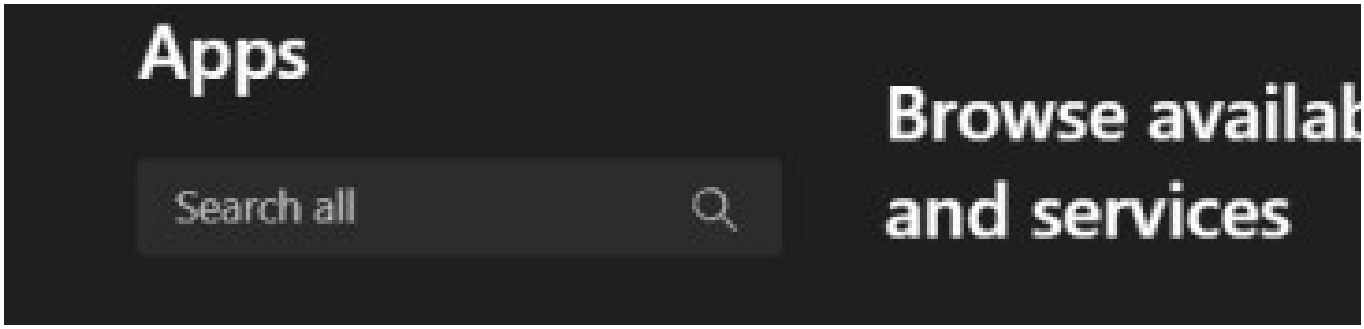
Microsoft Teams

The process of connecting to `Microsoft Teams` in similar to connecting to `Slack` under the covers, using a webhook and leveraging the incoming webhook app, installed from the `Teams` app store. To get started, let's install the `teamr` package from `CRAN`, call the library ar

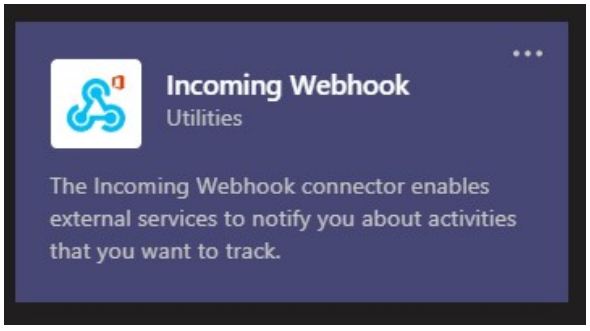
Just like with `Slack`, you will first need an organizational `Teams` account. Secondly, navigate to the `add more apps` button from within the desktop client, which is the ellipsis below the pinned apps on the left task bar.



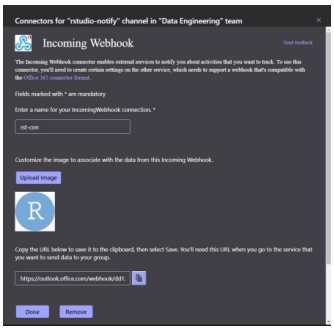
Third, use the search bar, located in the upper left corner to search for the `Incoming Webhook` app



click on the icon to bring up the install screen, where you can name the webhook, and assign it a specific channel within `Teams`



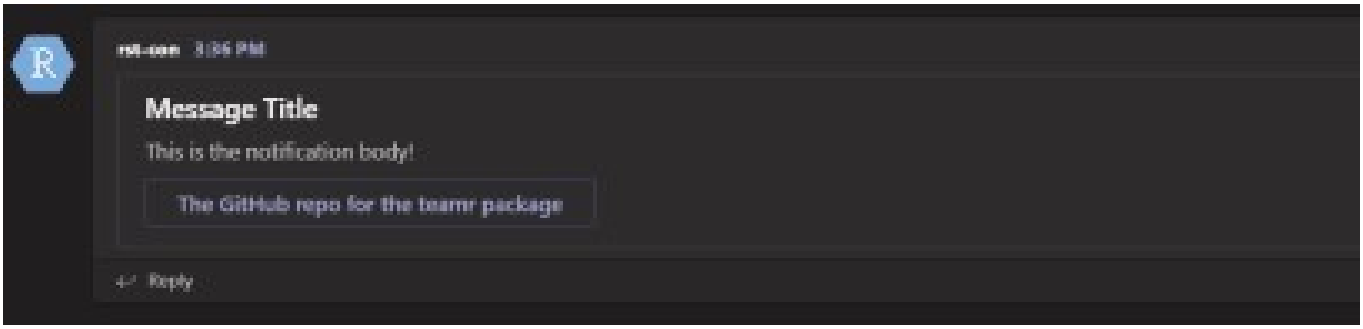
After following the prompts to name the connection and assign it to a channel, you are given the option of uploading an image so the webhook is easily recognizable when notifications are posted to your channel. Here, I have chosen to use an `RStudio` logo, which we will se post originates. After you are satisfied, click the done button in the bottom left corner.



The code to test your webhook in reasonably simple, and requires only six lines of code. Although, in practice, we leverage many of, if not all, the customizable features provided within the `teamr` package. You can read more about these feature either in the official document teamr

```
library(teamr)
con <- connector_card$new(hookurl = "https://outlook.office.com/webhook/...")
con$text("This is the notification body!")
con$title("Message Title")
con$add_link_button("The GitHub repo for the teamr package", "https://github.com/wwwjk366/teamr")
con$send()
#[1] TRUE
```

The final step, after running this code, is to verify everything is working as it should. The above code snippet sends a notification to `Teams`, with a link button pointing the package's GitHub repo, from a webhook bot containing the `RStudio` logo.



We hope you found this discussion helpful. The next time you wish there was a way to have `R` notify you, you can incorporate one of these into your workflow.
