Connecting teams and systems with R

Alexandra Mangold

4/27/2019

Slides here: bit.ly/2XLtJPn

Measure is a services company that uses drones to help customers with a wide range of use cases.

Measure is a services company that uses drones to help customers with a wide range of use cases.

Measure is a services company that uses drones to help customers with a wide range of use cases.

Measure is a services company that uses drones to help customers with a wide range of use cases.

Many people across the company need to come together to make these missions a **success**

Defining success

Reliable and low-lift: data should be able to flow across systems with no manual adjustment and minimal errors.

Organized and well-documented: solution should be able to be transferred to other team members with minimal pain.

Automated: solution should not be dependent on R-user's presence at desk.

API: a software intermediary that allows two applications to talk to each other

Mulesoft: https://bit.ly/2KzwGg4

Identifying early roadblocks

Does an API **exist?**

Does an API wrapper **exist?**

Is the **data compatible?**

Building an API wrapper

Getting authenticated with httr

This can vary a lot, depending on the system.

As someone new to HTTP and APIs, figuring out the correct syntax required a lot of trial and error!

What worked: using the Postman program, which helped reduce the number of ways things could be going wrong.

Building an API wrapper

Breaking down a sample API function

REST URL

Request + authentication

Extract request content

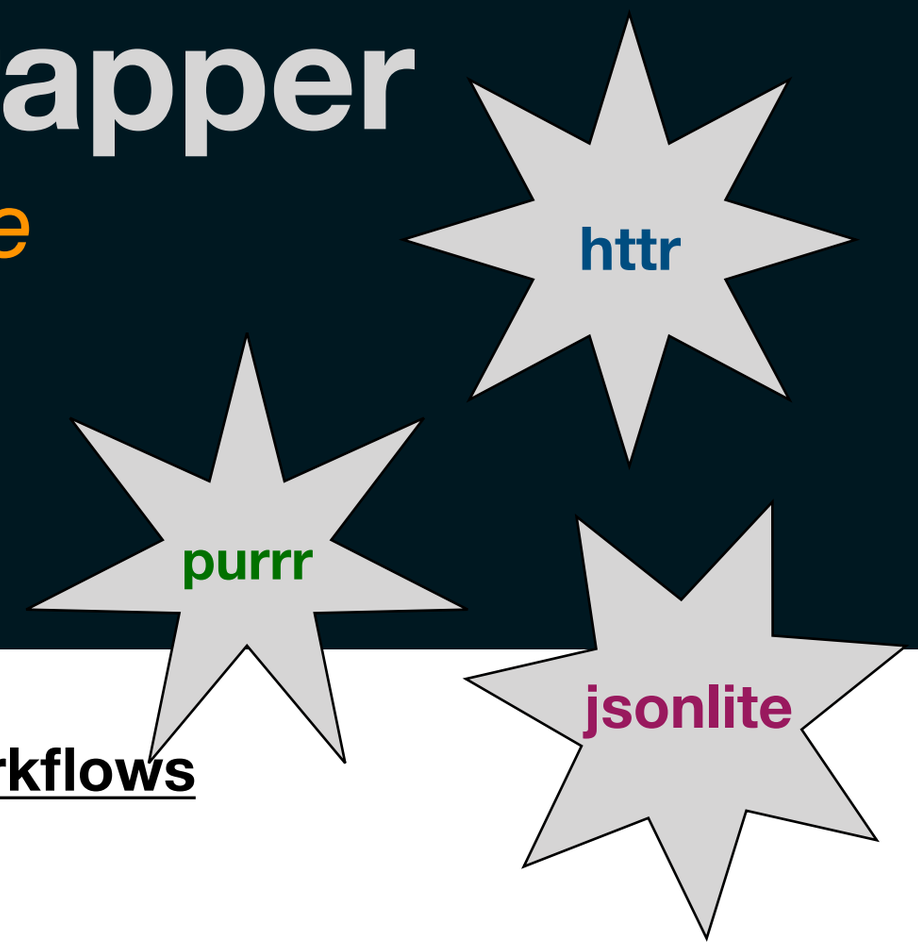
Extract fields into DF

```
wrike_folders <- function() {

  url <- paste0("https://www.wrike.com/api/v3/accounts/", account_id, "/folders")
  GETfolders <- httr::GET(url, httr::add_headers(Authorization = paste("Bearer",
                                                                       v3_key,
                                                                       sep = " ")))

  fold_content <- httr::content(GETfolders)[[2]]
  folders <- purrr::map_df(fold_content, magrittr::extract, c("id", "title"))
  return(folders)

}
```

Building an API wrapper

Key Packages: httr, purrr, jsonlite

httr: access API data + decode content
purrr: parse resulting JSON
jsonlite: convert data frame back to JSON



REST URL:
https://www.wrike.com/api/v3/accounts/*/workflows**


```
{
  "kind": "workflows",
  "data": [
    {
      "id": "IEABOPF6K4AFY73K",
      "name": "Data Solution Workflow",
      "standard": false,
      "hidden": true,
      "customStatuses": [
        {
          "id": "IEABOPF6JMAFY73K",
          "name": "Backlog",
          "standardName": false,
          "color": "Gray",
          "standard": false,
          "group": "Active",
          "hidden": false
        },
```


wrike_workflows() function

```
# A tibble: 77 x 5
   customStatusId   missionStatus    group     stage progress
   <chr>            <chr>            <chr>     <int> <chr>
 1 IEABOPF6JMAFY73K Backlog          Active        1 Stage 1 of 5
 2 IEABOPF6JMAFY73U To Do            Active        2 Stage 2 of 5
 3 IEABOPF6JMAFY736 In Progress      Active        3 Stage 3 of 5
 4 IEABOPF6JMAFY74I Ready for Review Active        4 Stage 4 of 5
 5 IEABOPF6JMAFY73L Completed        Completed     5 Completed
 6 IEABOPF6JMAFN3EA Waiting on Data  Active        1 Stage 1 of 6
 7 IEABOPF6JMAGTEIK Data Recieved    Active        2 Stage 2 of 6
 8 IEABOPF6JMAFN3GG Processing       Active        3 Stage 3 of 6
 9 IEABOPF6JMAFN3GQ Analysis         Active        4 Stage 4 of 6
10 IEABOPF6JMAFN3G2 Review           Active        5 Stage 5 of 6
# ... with 67 more rows
```

Connecting the dots

- Once the API wrapper was complete, I combined the datasets and **mapped the data flow** from one system to another.

- Creation of **dedicated fields** to hold the other system's data allowed users to stay within a single location.

- I created a **package** to hold my integration functions, along with data validation functions created to monitor that everything I thought should be happening, was happening.

Knowledge Transfer

How can I make this live beyond me?

Packages:

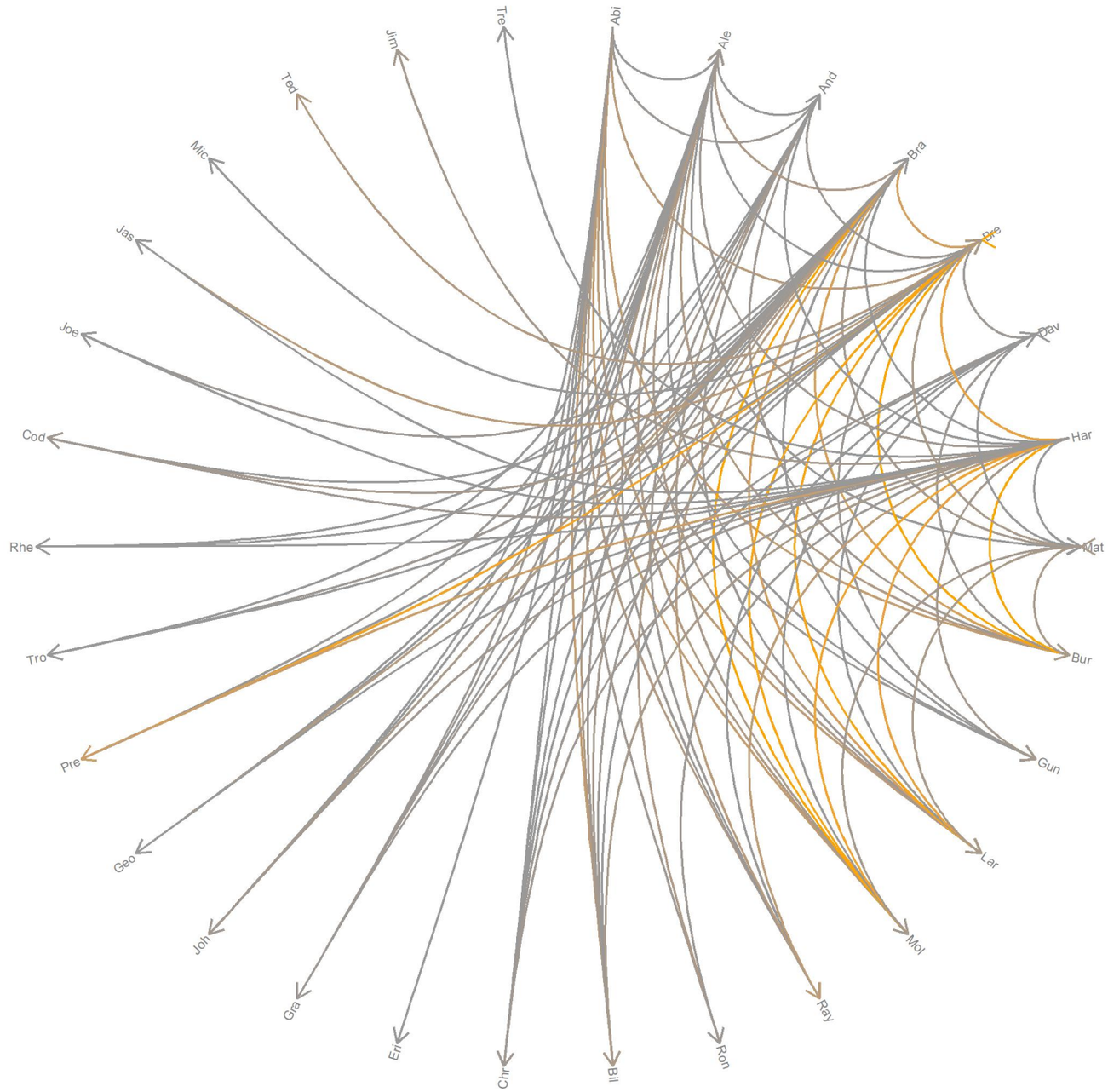
- **Easy installation** for future users
- Promote **good documentation** habits

Github:

- Make **package installation** even easier with devtools::install_github()
- Give confidence to users that they have the **latest version**
- Now offering **free private repos**!

It takes a village

2019 Mission Work



Code inspiration from @WireMonkey

Wrapping up

Key lessons learned

1. Make sure you have the right **foundations**

2. Stand on the shoulders of **giants**, if you can

3. If you need to write an API wrapper: **httr, purrr, and jsonlite** are your friends

4. Keep it **organized** + well-documented by making packages

Links

Github presentation link: **bit.ly/2XLtJPn**

APIs in English: **bit.ly/2AgbyJ6**

Purrr tutorials from Jenny Bryan: **bit.ly/2aR2c9O**

Getting started with httr: **bit.ly/2EYOHSy**

THANK YOU!