My Organization's First R Package

2 — Package Functions

Planning out the functions you'll want to include

So you’re making a work package… What’s going in it? (We need a plan.)

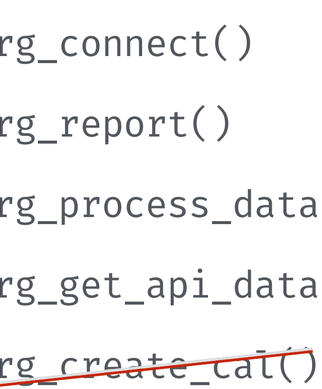
💯 `org_connect()`

💯 `org_report()`

❤️❤️❤️❤️❤️❤️❤️❤️❤️❤️ `org_process_data()`

`org_get_api_data()` 😎

FOR NOW 👎 ~~`org_create_cal()`~~



Think of the most pressing needs and make functions for them first.

Try to make ‘families’ of functions separated by what they do.

Think about what ‘helper functions’ might be useful to simplify things.

The amount of functions planned doesn’t have to be big; you won’t implement everything at once.

Let's make some functions that get a few KPIs

Measurement and reporting of key performance indicators (KPIs) can be helpful in decision-making.

Let’s use package functions that work on data in the `intendo` database, do all the calculations in-database, and return numeric values for the KPIs.

We’ll have a separate function per KPI. That’ll reduce confusion and, also, each KPI function can have its own interface.

PACKAGE FUNCTIONS

We need to define the KPI definitions and calculations first

Intendo typically cares about these four key performance indicators (KPIs):

KPI	Description
DAU	Daily active users, the number of users active in a day.
MAU	Monthly active users, the number of users active in a month.
DAC	Daily active customers, the number of customers active in a day.
ARPU	Average revenue per user, over some defined span of time.

Including functions that calculate KPIs from a DB table

Before we use the functions available in `scripts`, we should understand and test the code for correctness.

For the DAU calculation on a specific date, we need to:

– have a date stored in a variable (call it, `date`)
– create a connection to the `intendo` database
– access our `daily_users` table through `dplyr::tbl()`
– filter that table by the `date`
– select just the user names, make the table distinct
– count the rows
– `dplyr::collect()` the table
– pull the single value out, giving us an **R** vector

CODE DEMO: `02_01_pkg_functions.Rmd`

We are going to move over to the `.Rmd` file named:

`exercises/02_01_pkg_functions.Rmd`

In that, we'll work through code that calculates the DAU, MAU, DAC, and ARPU KPI values. We'll do this together.

These bits of code won't actually be functions but just some dplyr exercises to better understand how these KPIs are calculated.

YOUR TURN (WORKING ON THE intendo PACKAGE)

30 minutes

Open up the package-in-progress at:

`p_02_intendo/02_intendo.Rproj`

Looks at the state of the package. Try the functions that are already included.

The `scripts` folder contains the `main_kpis.R` file that has the functions we need to test and add to the package.

Here are two **usethis** functions we could take advantage of:

```
use_r() use_package()
```

Work together with your neighbors. Discuss things. Ask us anything.

PACKAGE FUNCTIONS

Using segmentation: it's really just filtering data

Segmentation of the relevant data typically happens before calculating any metric. This can give us a segmented KPI.

EXAMPLES OF SEGMENTED KPIS

ARPU for users in Germany

DAU for Android users

DAC for users that spent over $50

Segmenting can be as simple as filtering a table, or, it could involve joining tables and even doing mutations.

The next addition to the package is making a segmentation function that does the heavy lifting and only requires simple inputs.

YOUR TURN (WORKING ON THE intendo PACKAGE)

30 minutes

Still using the package-in-progress at:

`p_02_intendo/02_intendo.Rproj`

The `scripts` folder contains the `segmented.R` file that has some functions and manual tests.

The main part is the augmentation of the `get_dau()` function. Test that function and eventually add it to the package.

Is the design of that function good in your eyes? Is it merely good enough?

Work together with your neighbors. Discuss things. Ask us anything.

Let's discuss what we worked on

What kind of interface did the segmentation function use? What does it return?

What approach would you take to incorporate some basic segmentation in `get_dau()`?

Do you see any potential issues for user experience? (That is, for users of the functions)