

Managing large codebases in R

Presented by
Alex Bertram
Ryo Nakagawara



Presentation outline

- Intro
- Principles & Practice
 - Adopting a coding style for your team
 - Organizing code into functions
 - Organizing functions into packages
 - Documenting code
 - Using version control

Introduction

ActivityInfo & R

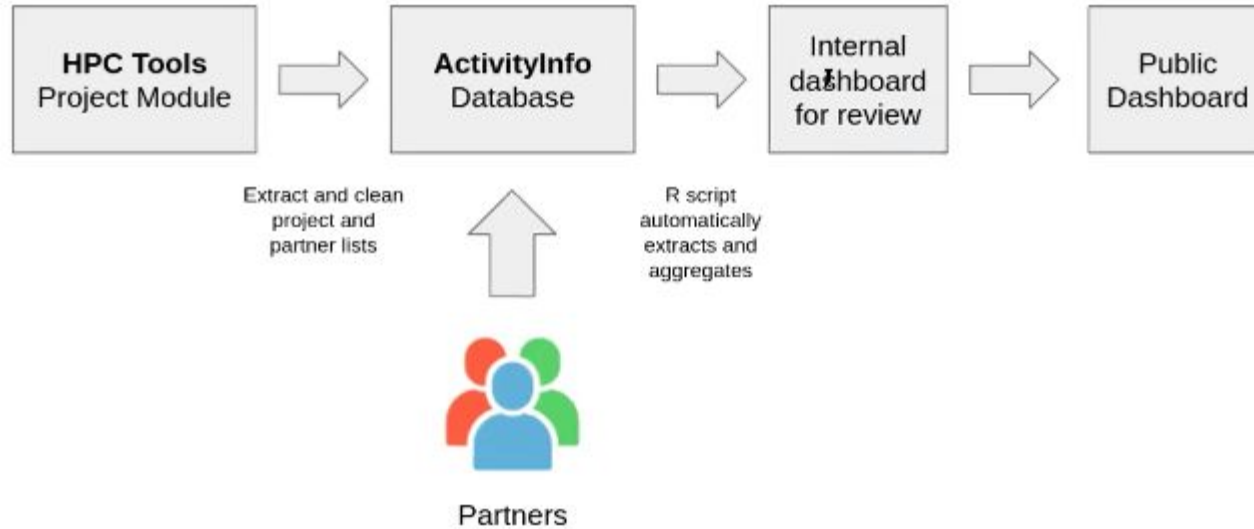
ActivityInfo is a user-friendly relational database for M&E, Case Management, and Humanitarian Coordination that seamlessly integrates with R.

<https://www.activityinfo.org/signUp>

What is a “large code base” ?

- More than one person working on the code
- More than a few files...

Examples - OCHA Libya



Examples - R4V

The screenshot shows a presentation slide within an Adobe Acrobat window. The slide title is "Activity Info Environment". It features a diagram with three "National SW" database icons at the top, each connected by a dashed arrow with an "R" in a circle to a single "Regional SW" database icon at the bottom. The "Regional SW" icon is labeled "Single form". To the left of the diagram is a "Strengths:" section with a bulleted list. To the right is a "Challenges:" section with a bulleted list. The slide footer contains "R4V.INFO" on the left and the "R4V" logo on the right. The Acrobat interface includes a top menu bar with options like "Transitions", "Animations", "Slide Show", "Review", "View", "Help", and "Acrobat". A toolbar below the menu bar contains various editing tools. In the top right corner of the Acrobat window, there is a video call window showing a man with a beard and headset, identified as "James Leon Dufour".

Activity Info Environment

National SW **National SW** **National SW**

Regional SW
Single form

Strengths:

- National database can have additional information adapted to their context
- Common reference tables
- Comparability through regional database

Challenges:

- 3 different languages throughout the region
- Quality of the data reported by partners
- Deadlines and double reporting between national and regional databases

R4V.INFO R4V

Examples - R4V Shiny App

The screenshot displays the R4V Shiny App interface. At the top, there are navigation tabs: "Data input", "Error and cleaning script", and "Download report". The main content area is titled "FOR COVID-19 ONLY (updated 20/04/2021, please read any comments in the Regional platform 40 team)". It features a "Final Report and cleaning scripts" panel with a "Country Name" dropdown menu set to "All". Below this, there are several sections of configuration options, each with a "TRUE" or "FALSE" radio button:

- COVID-19:** If the COVID-19 field is empty, automatically registers as "No".
- SUPPORTSPACE:** If the Support Space field is empty, automatically registers as "No".
- EMER:** If the EMER field is empty, automatically registers as "No".
- CHANGING:** If CIA is registered as No, but there is a value of transfer that has been registered, automatically changes the CIA to No.
- FIN BENEF FILL:** If new Beneficiaries and Total are equal 0, copy values from breakdown (only if they are equal or from equality values, if breakdowns are not equal).
- TOTAL LOWER NEW:** If the new Beneficiaries value is superior to the Total of Beneficiaries, automatically change the value of Total to equal the value of new Beneficiaries.
- ASIA IN DESTINATION:** For Pin indicators, in the Population Type breakdown is empty, assign the new Beneficiaries values to the Refugee and Migrants category, as recommended in the address.
- ASIA IN ALL DESTINATION:** If new Beneficiaries value is not equal to the population type breakdown, assign the difference to the NA2 category of the breakdown.
- FINACTY AFFORTIONING:** For Pin indicators, if the Age and Gender breakdown is empty, applies an automatic breakdown extracted from the Pin-2021 Country and Sector Information based on new beneficiaries values.
- COMPLETE PIN AFFORTIONING:** For Pin indicators, if the Age and Gender breakdown is empty, applies an automatic breakdown extracted from the Country and Sector Information based on new beneficiaries values. USE WITH CARE AS IT OVERWRITES DATA !
- ROUNDING OFF:** Rounding differences due to roundings, differences in the breakdowns can be +1 or -1, aim to reset this difference to the max age and gender category.
- PIN ERASE QUANTITY:** For Pin indicators as Pin indicators should not contain data in Quantity, change all values for this field to 0. USE WITH CARE AS IT ERASES DATA !
- PEOPLE FILL:** People indicators: If Quantity of with equal 0, copy values from Total of beneficiaries or if total empty, from age and gender breakdown sum.
- PEOPLE ERASE EXTRA DATA:** For People indicators: as People indicators should not contain data in population type breakdown, change all values for this field to 0. USE WITH CARE AS IT ERASES DATA !
- OTHER FILL:** People indicators: If Quantity of not equal 0, copy values from Total of beneficiaries.
- OTHER ERASE EXTRA DATA:** For Other indicators: as Other indicators should not contain data in age gender and population type breakdown, change all values for this field to 0. USE WITH CARE AS IT ERASES DATA !

At the bottom of the configuration panel, there are buttons for "Run Script", "Download Predefined data", and "Download Cleaned data". To the right of the configuration panel, there is a "Statistics" panel with a list of metrics: "Number of activities", "Number of errors Pre cleaning", "Number of errors Post cleaning", and "Percentage of errors".

Principles

What is a coding style?

Adopting a common code style

- Rules that the whole team agrees on
- How to name functions, variables, datasets
- When to use spacing
- When and how to document functions

Why is a common code style important?

Adopting a common code style

- Code is written once, read a hundred times
- Is it ...?
 - `check_duplicates()`
 - `checkDuplicates()`
 - `check.duplicates()`
 - `CheckDuplicates()`
 - ???



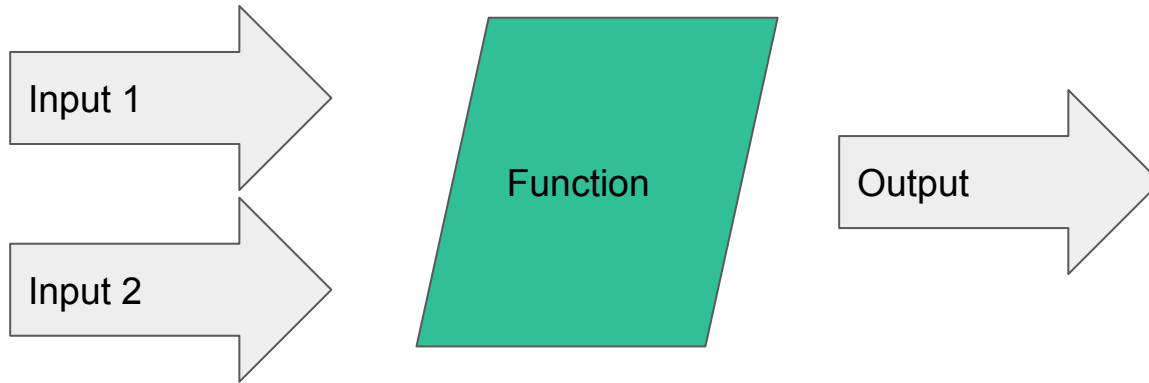
How to get started

Adopting a common code style

- **Recommendation:** <http://adv-r.had.co.nz/Style.html>
- Automated with formatR: <https://yihui.org/formatr/>

What is a function?

Organizing code into functions



Pure vs Impure functions

Pure functions

- Same inputs, same outputs
- No side effects (no reading, writing)
- Can be tested

Examples:

- `flag_duplicates(hh_list)`
- `score_eligibility(hh_list)`

Impure (imperative) functions

- Outputs depend on the outside world (reading from a file, from a server)
- Same inputs, (maybe) different outputs

Examples

- `launch_missiles()`
- `read_hh_from_ai_form()`
- `write_updates_to_db(df)`

Why functions?

Organizing code into functions

- Breaking code into smaller functions makes the code easier to read and understand
- Easier to compose functions together
- Individual (pure) functions can be tested

Function length

Organizing code into functions

- Strive for functions with max **twenty** lines.
- A function should “fit in your head”.



```
##### Make Arabic names consistence
{
  arabic_letter <- read.xlsx("../pcodes/arabic_letters.xlsx", sheet = "consis
  raw_data$hoh_arabic_name <- mgsub("\\s+", " ", raw_data$hoh_arabic_name)
  raw_data$hoh_arabic_name <- mgsub(arabic_letter$old_char, arabic_letters$new_cf

  raw_data$hoh_spouse_name <- mgsub("\\s+", " ", raw_data$hoh_spouse_name)
  raw_data$hoh_spouse_name <- mgsub(arabic_letter$old_char, arabic_letters$new_cf
}

##### HH names duplicates - Similarities check
{
  HH_duplicates_checks <- raw_data %>% dplyr::select(c("X.id", "QA_Code", "intervi
      "district", "sub_district"
      #second row should be spe
      "hoh_arabic_name"))%>%

  dplyr::mutate(HH_duplicates_checks=0)

  if (nrow(HH_duplicates_checks) > 0) {
    for(i in 1:nrow(HH_duplicates_checks)){
      if (HH_duplicates_checks$interviewer[i] == "Master data") {
        break
      }else{
        duplicate_vector_simi<- 1 - stringdist(HH_duplicates_checks$hoh_arabic_r
          HH_duplicates_checks$hoh_arabic_r
          nchar(as.character(HH_duplicates_checks$hoh_arabic_name[i])))

        HH_duplicates_checks$HH_duplicates_checks[!is.na(duplicate_vector_simi)
```

Splitting pure and imperative parts

Organizing code into functions

- The “Functional Core, Imperative Shell” Pattern

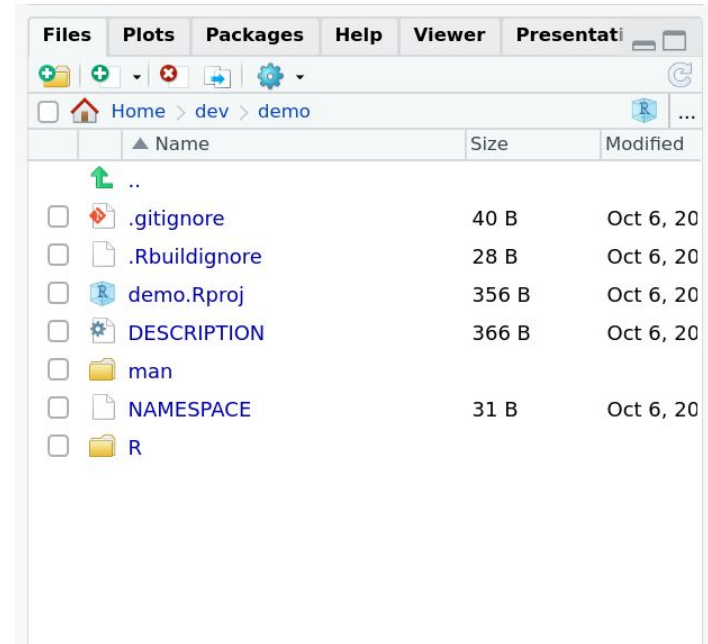
```
upload_from_csv()
```

```
read_hh_from_csv() |>  
  remove_duplicates() |>  
  score_eligibility() |>  
  import_hh_to_ai()
```

R Packages

Organizing functions into packages

- Combination of functions, dependencies, and documentation
- Standard structure



Why packages?

Organizing functions into packages

- Re-use common code
- Easier to work with many files than endlessly `source()`'ing.

Version control system

Using version control

- Most commonly used VCS today is Git
- Free hosting at GitHub.com, GitLab.com, BitBucket.com
- Supports collaboration