

Statistical tools and Data Management

Christopher Maronga



KEMRI | Wellcome Trust

Session assumption

- Already installed R and Rstudio
- Know how to use R at it's basics (*NOT required*)
- Have atleast handled a messy dataset once before
- Basic understanding of dataset components(variables/rows/cells)

The tool of trade

This session will focus on learning the basics of data wrangling using R statistical software

Why R?

There are other commercial statistical tools such as STATA, SAS and SPSS that can be employed in data management. We chose R over the rest because:-

- Open source (free to use)
- Relates to other languages and systems
- It's flexible, fun and easy to learn
- Outstanding visualization
- Advanced statistical language (a tool for Machine Learning)

Supports user defined extensions (packages) and its cross-platform

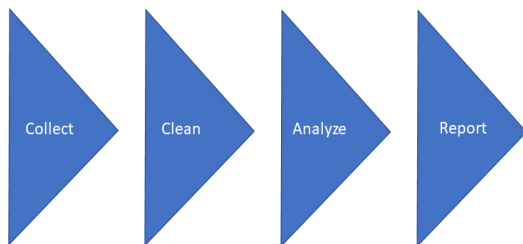
What is R and Rstudio



Why should I care about cleaning my data??

Why should I care about cleaning my data?

- Messy data is everywhere and most real world datasets start off messy in nature
- As data grows bigger, number of things that can go wrong grows in the same magnitude

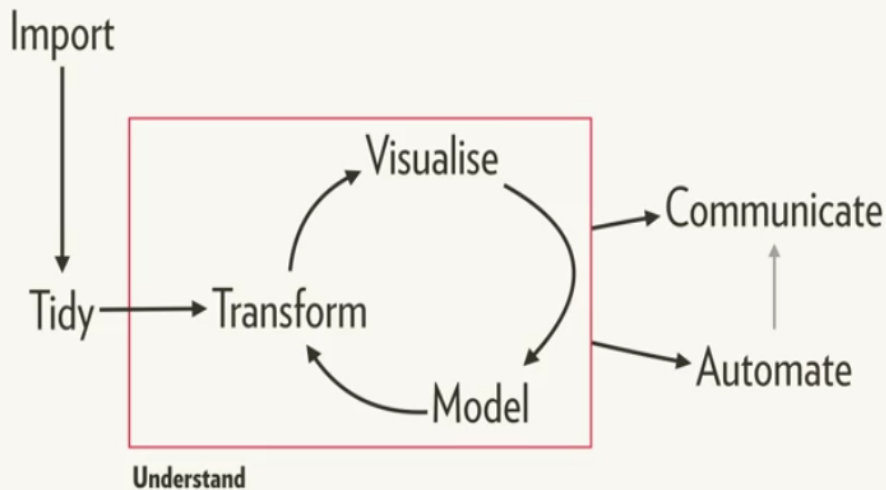


What are some of the issues you get to deal with when it comes to data management??

What are some of the domains for inclusion into your error-checking plan??

-
-
-

The almost familiar data cycle?



Session Objectives

By the end of the presentation:-

- Get introduced to R for data wrangling
- Be able to read data into R from whichever source it's stored
- Explore your raw data with the aim of deeply understanding it
- Tidy/reshape your data to required format
- Explore toolkit for general data housekeeping in R
- Prepare a dataset for analysis
- Fundamentals of reproducible research

Follow up plan

In the next couple of minutes, we will explore data munging tasks like:-

- General data validation checks (missing values, outlier values etc.)
- Identifying and removing duplicates
- Manipulating variables
- Creating new variables
- subsetting/filtering data
- Reshaping and Merging datasets

Data cleaning and reshaping is one of the task that we spend much time on while undertaking data management activities

We will introduce a complete toolkit of R packages for cleaning, manipulating, reshaping and visualizing your data

Getting data into R

Research data can come in different flavours:-

- Flat files (.xlsx, .csv, .txt)
- Data on web-based databases (REDCap, MySQL databases and OpenClinica)
- Data from other statistical softwares (STATA, SAS and SPSS)

We will briefly cover how to read your data into R from the first 2 sources above and begin managing it

```
library(readxl) # for reading .xlsx files  
read_excel()
```

```
library(readr) # for reading .csv and .txt files  
read_csv() and read_tsv()
```

NOTE: A lot of functions are available for reading flat files into R, just Google them

Reading flat files

Lets see how this works with a practical example

Connecting R to Web-based databases(1/2)

Special packages and functions are available to help you connect and fetch data stored in your web-based databases from within R software

Connecting to REDCap

```
library(redcapAPI) # load required package
# create a connection
con<-redcapConnection(url='https://redcaplink/api/',
                      token = 'your account token here')

# export data
my_data<-exportRecords(con, fields = NULL, forms = null ,
                      records = NULL, events = NULL ,
                      labels = TRUE, dates = TRUE,
                      survey = FALSE,factors=F, dag = T,
                      checkboxLabels = TRUE)

?redcapAPI # get more help
```

Connecting to REDCap

Lets see how this works with a practical example

Connecting R to Web-based databases(2/2)

Connecting to MySQL databases

```
library(DBI) # provides the interface
library(RMySQL) # implements the process
# create a connection
con <- dbConnect(MySQL(),
                  host= "hostname",
                  dbname = "databasename",
                  user = "username",
                  password = "password")

# list database tables
dbase_tables <- dbListTables(con)

# export data
my_data <- dbGetQuery(con, "SQL query here")

# close database connection
dbDisconnect(con) # remember to close connection
```

Connecting to MySQL databases

Lets see how this works with a practical example

Cleaning data in R

Steps involved in cleaning data

- Explore the raw data
- Tidy/reshape your data
- Perform general housekeeping
- Preparing for final analysis

Cleaning data in R – Sources of errors

Possible sources of error

- Experimental error
- Data entry error
- Valid measurement (might be) etc.

Identifying errors:-

- **Focus on context** (*“tidy datasets are all alike but every messy dataset is messy in its own way”* **Hadley Wickham**)
- Possible ranges

Cleaning data in R – Exploring raw data

This step is for understanding the structure of your data. Looking through the data components (what variables, types and scope)

- `class()`
- `dim()`
- `names()`
- `str()` or `glimpse()`
- `summary()`
- `head()`
- `tail()`

You can also visualize data to quickly identify extreme or suspicious values in your data

- histogram
- scatterplots
- boxplots

Cleaning data in R – Tidying data

Violations of principles of tidy data

- Column headers are values, not variables
- Variables stored in both rows and columns
- Multiple variables are stored in one column

`tidyr()` functions to reshape and restored tidy data

- `gather()`
- `spread()`
- `separate()`
- `unite()`

`gather()` and `spread()` outputs gives rise to most commonly referred types of datasets

- wide datasets *more columns than rows*
- long datasets *more rows than columns*

Data Cleaning in R – General Housekeeping

We will explore a sample dataset using `dplyr` functions among others

dplyr functions

- `select()` : Subset columns
- `filter()` : Subset rows
- `arrange()` : Reorders rows
- `mutate()` : Add columns to existing data
- `summarise()`: Summarizing data set

additional functions

- `summary()` : printing general summary
- `is.na()` : checking for missing values
- `merge()` : to merge 2 datasets

Lets see how this works with a practical example

Cleaning data in R – Prepare for data analysis

Properly licensing your data for an analysis

Type conversion at its basic (*putting variables into their required formats*)

- **character**
- integer
- logical
- **factor**
- string manipulation

Its extremely important to know how to convert your variables from one type to another just incase you require it

- as.* family of functions for type conversion
- dealing with missing values `na.omit()` and `complete.cases()`
- outliers/obvious errors – histograms and boxplots

Exporting datasets/saving files (Output)

R allows you to save or export datasets from the workspace into .csv or tab delimited

```
write.csv(dataset_name, "dest_folder/preferred_name.csv")
```

You can also save an entire workspace and load it later to an R session like this

```
save(list = ls(),file = "pref_name.RDA") # save datasets in workspace  
load("pref_name.RDA") # load back items into workspace
```


Reports authoring, visualization and reproducible research

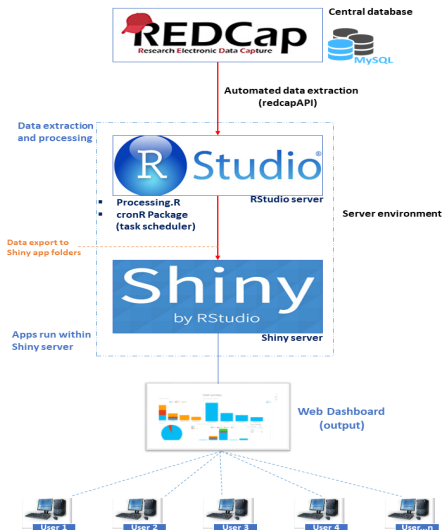
Two most powerful tools for report authoring, visualization and process automation

- **Rmarkdown** is file format for making dynamic documents with R
- **Shiny** is an open source R package that provides an elegant and powerful web framework for building web applications using R

NOTE: R Shiny and creating reports dashboards is beyond the scope of this workshop

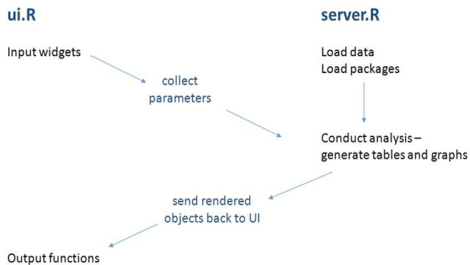
R shiny dashboard workflow

How does it work?

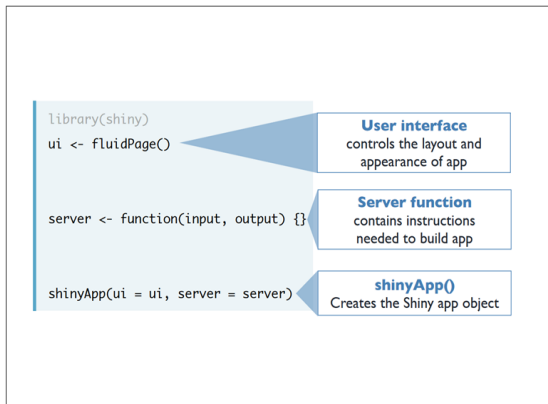


R Shiny framework(1/2)

Anatomy of R shiny framework



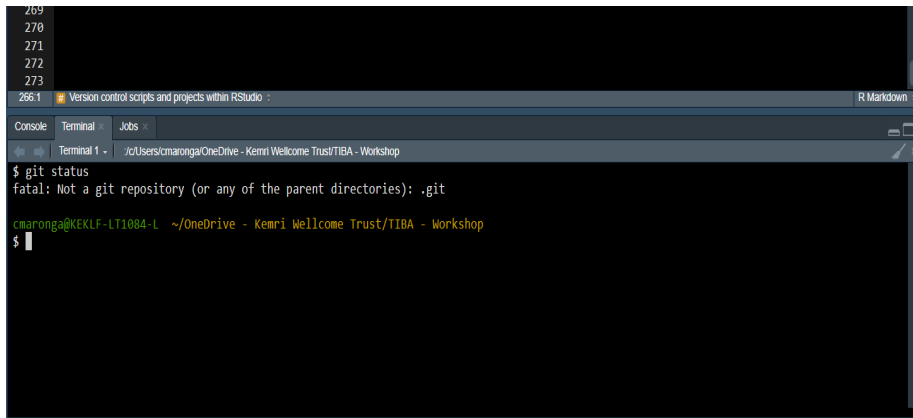
R Shiny framework(2/2)



A built example is **CHAIN Network Reports Dashboard**

Version control scripts and projects within RStudio

Rstudio IDE supports version control systems such as Git and SVN. Enables version control data analysis/management codes as well documents withing the R project.



The screenshot shows the RStudio IDE interface. At the top, a script editor displays line numbers 269 through 273. Below it, a pane shows a file named '266.1' with the title 'Version control scripts and projects within RStudio'. The bottom pane is divided into 'Console', 'Terminal', and 'Jobs' tabs. The 'Terminal' tab is active, showing a terminal window with the following text:

```
$ git status
fatal: Not a git repository (or any of the parent directories): .git

cmaronga@KEKLF-LT1084-L ~/OneDrive - Kemri Wellcome Trust/TIBA - Workshop
$
```

Acknowledgements



KEMRI | Wellcome Trust

