Reproducible Contributions in Development Economics

Bastiaan Quast

16 June 2016

Contents

Introduction

As mentioned in the title, methodologically I focus on making the research in this thesis reproducible. Reproducibility is different from replicability in that is refers to regenerating the research results based on the same data, as opposed to replicablity, which uses newly gathered data.

A large part this methodology comes from the field of biostatistics and it is best explained by one of the key figures in this field.

> The replication of scientific findings using independent investigators, methods, data, equipment, and protocols has long been, and will continue to be, the standard by which scientific claims are evaluated. However, in many fields of study there are examples of scientific investigations that cannot be fully replicated because of a lack of time or resources. In such a situation, there is a need for a minimum standard that can fill the void between full replication and nothing. One candidate for this minimum standard is "reproducible research", which requires that data sets and computer code be made available to others for verifying published results and conducting alternative analyses.

I believe that this applies to at least the same extend an probably more in economics. Although there are situations in which replicable research may be conducted, specifically in experimental settings such as those often used in behavioural economicss or in field research using Randomised Control Trials (RCTs), there are also many cases in which this is not possible.

In this thesis I include two chapters where the identification strategy is based on a natural experiment. Firslty, a change in government policy, as a results of the unconstitutional nature of the sex-based discrimination in pension eligibility in South Africa. Secondly, a the introduction of an interface language on the South African Google Search website, as a spillover of that translation work being done for Botswana. In both these cases I use the National Income Dynamics Study, the most comprehensive panel data set in South Africa. Since it will be hard to find more relevant data, a full replication - using new data - will prove difficult. As such, it is all the more important for research to be as transparent as possible, making it at least as reproducible as can be.

There are a number of way in which research can be made reproducible, many of which are already becoming increasingly common. In addition to this, there are several methods which are less widespread, but nevertheless very useful.

First and moremost it is essential to make clear which data is being used and in case it is primary data, how it was produced (e.g. research instruments). Where possible, the data itself should be included. If this is impossible to due e.g. privacy concerns of licencing issues, a clear procedure for obtaining the data should be documented. The data used in both South African studies is available upon request through an online portal. In the publicly available Git project (more details below), I include a description how to obtain the data used through the South African Labour & Development Research Unit .

Secondly, the computer code for producing the research results should be made available, in case this code produces intermediate data sets, where possible also make available.

Comment the code, use e.g. markdown type conventions, # for section, use ## for subsection. Add a header to the file with:

1. file name
2. file purpose
3. author name
4. author email

In addition to describing the file purpose, the file name itself should also be meaningful. Typically the code in a file performs a certain action, as a rule, it can therefore best be described using a transitive verb, for instance. `import.R` the resulting output (here the imported data) can then be saved using an intransitive verb as a file name, e.g. `imported.RData`.

Development versioning using Git, also useful for registering research designs (since logged), retracing steps. In addition to this, changes in all of the project files are logged using a version control software package. I use the open-source version control software called Git. Projects using version control such as this are called repositories. These repositories are published on the internet on websites such as GitHub or BitBucket.

Good reasons to use open-source R. Using open-source software in order to make the computational procedures of the statistical method verifiable. In my case this means that all statistical analysis has been done using the statistical programming environment R.

Additionally, when implementing new algorithms, packaging these as R libraries (extensions) and releasing under an open-source licence such as General Public Licence version 3 or later on the Comprensive R Archive Network (CRAN) website is a step towards replicablity. For instance, I packages the procedures implementing the Wang-Wei-Zhu algorithm used for the analysis as the `decompr` package. I uploaded this package to CRAN[1] and it has since been downloaded over 6,000 times[2]. This makes code useful for replicable research, since the original procedures, which may have contained hardcoded procedures (e.g. doing a loop 34 times, one for each country) now have to generalised and transformed into a function, since R package can only contain functions. As a result of this, the code which we developed for the analysis of the TiVa data set, is now also being applied by other used to the wiod data set.

Adding weaving code . This also ties into `packrat`, since it makes clear which version of which libraries are used for the analysis.

[1] Available at: https://cran.r-project.org/web/packages/decompr/index.html

[2] Data from the RStudio servers only, other CRAN mirrors to not release statistics so the actual number is presumably higher. Note that this does not correspond directly to users, since updates require a new download.

Table 1: R `sessionInfo()`

```
sessionInfo()

## R version 3.3.0 (2016-05-03)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.11.4 (El Capitan)
##
## locale:
## [1] C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  base
##
## other attached packages:
## [1] dplyr_0.4.3  knitr_1.12.3
##
## loaded via a namespace (and not attached):
##  [1] R6_2.1.2       assertthat_0.1 magrittr_1.5   formatR_1.3
##  [5] parallel_3.3.0 DBI_0.4        tools_3.3.0    Rcpp_0.12.4
##  [9] stringi_1.0-1  methods_3.3.0  stringr_1.0.0  evaluate_0.9
```

Piping functions to make to more intuitive. As will most lines will have a structure beginning with the object to which the output is assigned followed by the assignment operator, followed by the function, followed by the input data, followed by the argument. After which the following line again will begin with the output object, etc. We can see this for instance, if we use the built-in data set ‘swiss‘, which examines fertility levels in the French speaking regions of Switzerland in 1888. In this example I reproduce a simplified version of an original study this data, looking at the difference in fertiliity levels between predominantly Catholic and predominantly protestant agricultural regions.

Table 2: swiss

```
# load data
data(swiss)

# inspect data
summary(swiss)

##    Fertility      Agriculture     Examination      Education    
##  Min.   :35.00   Min.   : 1.20   Min.   : 3.00   Min.   : 1.00  
##  1st Qu.:64.70   1st Qu.:35.90   1st Qu.:12.00   1st Qu.: 6.00  
##  Median :70.40   Median :54.10   Median :16.00   Median : 8.00  
##  Mean   :70.14   Mean   :50.66   Mean   :16.49   Mean   :10.98  
##  3rd Qu.:78.45   3rd Qu.:67.65   3rd Qu.:22.00   3rd Qu.:12.00  
##  Max.   :92.50   Max.   :89.70   Max.   :37.00   Max.   :53.00  
##     Catholic       Infant.Mortality
##  Min.   :  2.150   Min.   :10.80   
##  1st Qu.:  5.195   1st Qu.:18.15   
##  Median : 15.140   Median :20.00   
##  Mean   : 41.144   Mean   :19.94   
##  3rd Qu.: 93.125   3rd Qu.:21.70   
##  Max.   :100.000   Max.   :26.60   
```

The typical workflow is demonstrated below. We first define an *output* object for the filtered Agricultural regions, then the assignment operator, followed by the function, followed by the *input* data, followed by the argument. The second line then starts with a *new* intermediate object, etc. The final line starts with the function, then in last intermediate object, then the argument.

Table 3: R without pipe

```
swiss_agr     <- filter(swiss, Agriculture > 50)
swiss_agr_grp <- group_by(swiss_agr, Catholic > 50)
summarise(swiss_agr_grp, mean(Fertility, na.rm=TRUE) )

## Source: local data frame [2 x 2]
## 
##   Catholic > 50 mean(Fertility, na.rm = TRUE)
##           (lgl)                         (dbl)
## 1         FALSE                      65.62143
## 2          TRUE                      79.51667
```

In this example I now use the `magrittr` package, which has the following pipe operator `%>%`. The first thing is the input data, followed by the transforming function, followed by the argument (criterium), this is passed to the next line where the second transforming function is the first item, follow by its respective argument, this is passed on to last line, where the data is summarised, in this case by computing the mean.

Table 4: R with pipe

```
swiss %>%
        filter(Agriculture > 50) %>%
        group_by(Catholic > 50) %>%
        summarise( mean(Fertility, na.rm=TRUE) )

## Source: local data frame [2 x 2]
##
##   Catholic > 50 mean(Fertility, na.rm = TRUE)
##           (lgl)                         (dbl)
## 1         FALSE                      65.62143
## 2          TRUE                      79.51667
```

Chapter 1

Making the ‘Next Billion’ Demand Access

Chapter 2

decompr: Global Value Chain decomposition in R

Chapter 3

Global Value Chains in LICs

Nothing here het.

Chapter 4

rnn: Recurrent Neural Networks in R

Chapter 5

Male/Female Income and Child Growth

Final Remarks