From Scripts to Projects: Learning a Modular, Auditable, and Reproducible Workflow by Example

Jessica Randall

12/29/2019

Contents

Introduction

Having been a big fan of HRDAG's work and fellow alumna of Emory's Rollin's School of Public Health, I have been delighted to have had opportunities to chat with both Megan and Patrick over the last year. I was equally thrilled when Patrick asked if I might want to work on an epidemiological problem the team had been presented with. While I had read about the organization's modular workflow (link) and watched Patrick's talk on how to do principled data processing (link), a lot of it was initially beyond my scope as someone who had only formally learned SAS and dabbled in self-taught Python and R. I fully admit to my R code looking suspiciously similar to all of the examples of amateur coding that Tarak uses in his post on why .Rproj is harmful to reproducible and auditable data science (link).

Excited to help out where I could and learn from a master, I was undettered by the prospect of a crash course in the HRDAG process necessitated by the inherent immediacy of the need for an answer; were the data suggestive of an epidemic or not? To that end and with a great deal of Patrick's guidance I began to shift towards working mainly in the command line as opposed to R Studio and to start thinking of scripts in terms of a larger project architecture as opposed to one-off bits of code that I'd write per project.

As I worked, Patrick also encouraged me to keep notes on what I liked and disliked about the process, where I felt it made things easier and where I noticed friction. The following post summarizes those notes and the work that went into determining the answers to the client's question.

To begin the project, we start with a task. Each task is a microchasm of the entire project and at HGRAG this facilitates the team's collaboration across time zones and programming languages. It also results in projects which are auditable and have reproducible results; anyone looking at the code can determine exactly how any output from it was produced and anyone working on the task at anytime can pick it up and recreate the same output you wrote. As a scientist, this need for reproducibility resonated with me and even if you are a team of one in your workplace (as many statisticians reading this might be) your closest collaborator

is you six months from now. Break up your project into discrete tasks and thank yourself later. I certainly plan to implement this style of project organization at my work outside of HRDAG.

To make the goals of each task even clearer, each task has a Makefile. The Makefile serves as an outline of each task not only for our benefit, but mainly for the computer to read it. Imagine being able to work on a group project with team members around the world, and still have an auditable trail of each person's work. The Makefile makes that scaffolding clear and traceable. For more information on Make files check out Mike Bostock's post here (link) and for the inspiration for HRDAG's project oriented workflow check out Jenny Bryan's post here (link).

For this project I have a clean task, a test task, and a write task each with its own Makefile, input, output, and src folder for the source code.

Clean

I start with the clean task. In the input folder of this task I include the two datasets of with reported deaths, the src folder contains the R code to clean up the code. I used RStudio to test the code for bugs (with frequent restarts and fresh environments) and use Microsoft Visual Studio Code to update the code that gets pushed to the project Github repository. I had a small bug early on with an error message I didn't want to immediately address so I created an issue on the GitHub to pick it back up later. I knew where the code had issues but I could count on the rest of it working as expected if (when) I'd need to spend time on other projects. This also had the added benefit (trap?) of making me feel like I'd actually made progress instead of staring down the endless empty field of .R file with however many tasks left to code. Win!

The first dataset includes initial deaths reported from 12 June 2019 – 10 September 2019. The second file contains additional deaths reported from 1 January 2019 – 14 December 2019. Our client would like us to determine if the number of deaths reported during a specific time period appeared to be unusually high, and the relationship of these reported detahs to dates of external events.

To clean the data, we check for any out of bounds dates, any values where one level of a variable may be coded different ways (ex: "seventeen" instead of "17"), and make sure that we don't have strings of text where we expect to see numbers and vicer versa. In this case we use informal (instead of manual) unit tests. This means that instead of checking the dimensions or the top 5 rows of a dataset visually, we specify some condition the data has to meet and tell it to stop running if that condition is not met. #FIXME: need to put in the tests

We also canonicalize the data. This process refers to making sure that the names of the variables we are comparing are standard across the datasets if we were to combine them. We are working with two datasets here and since they overlkap in terms of when they were collected, we'd like to keep them seperate while still having variables named consistently.

Since we are interested in examining only those cases in which people have been reported as dead, we remove all rows where the person's status is "Not_death".

We are also interested in a couple of dates of interest so we created a variable indicating whether or not a death occured prior to or following that date.

Since this is a sample and not a complete record of all people who have died in this time period, we expect that this data is incomplete but we cannot say to what extent our results will be generalizable to the population from which these data were obtained.

The data are cleaned as part of the clean task and all of that code is stored in the clean.R file in the clean/src folder.

To use the clean data, we pull it from the clean task's output folder.

```
files <- list(input1=here::here("clean/output/death1_clean.txt"),
              input2=here::here("clean/output/death2_clean.txt"))
```

```
death1 <- readr::read_delim(files$input1, delim="|")

death2 <- readr::read_delim(files$input2, delim="|")
```

In order to see if there appear to be an abnormally high number of deaths, we will use an epidemiological (or epi) curve. Epi curves are used in investigations of potential outbreaks and their shape can even suggest a means of propogation; curves for infections transmitted from person to person have different shapes than curves of diseases transmitted from a water source, for example.

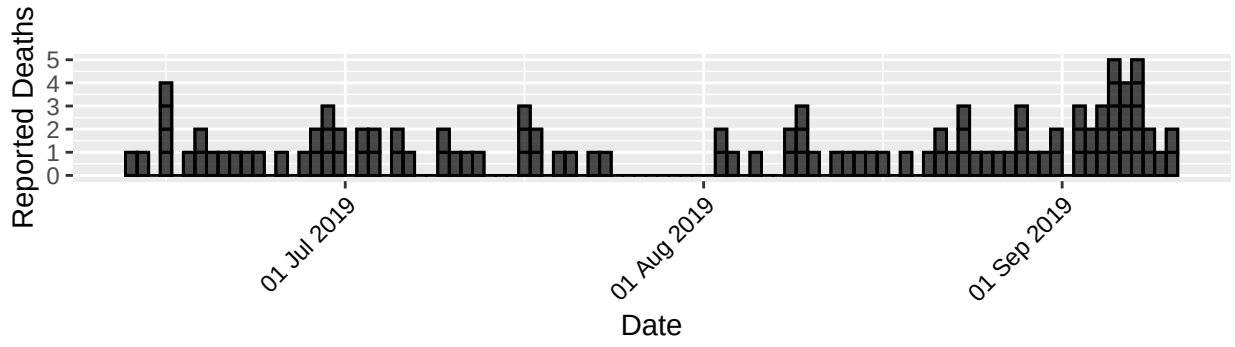
In our data, if an epidemic of deaths were occuring, we would expect to see a spike in the number of deaths reported per day.

Epi Curve 1

The first set includes deaths reported from 12 June 2019 – 10 September 2019, 91 days or approximately 3 months. The second set contains deaths reported from 1 January 2019 – 14 December 2019, 348 days or approximately 11 months.

In order to examine the frequency of reported deaths per day across the time period provided, we use the incidence package to obtain the number of deaths reported each day and plot this against all of the days for which we have data.

Since these graphs require relatively short blocks of code and don't require major changes to the data, they are not a seperate task.

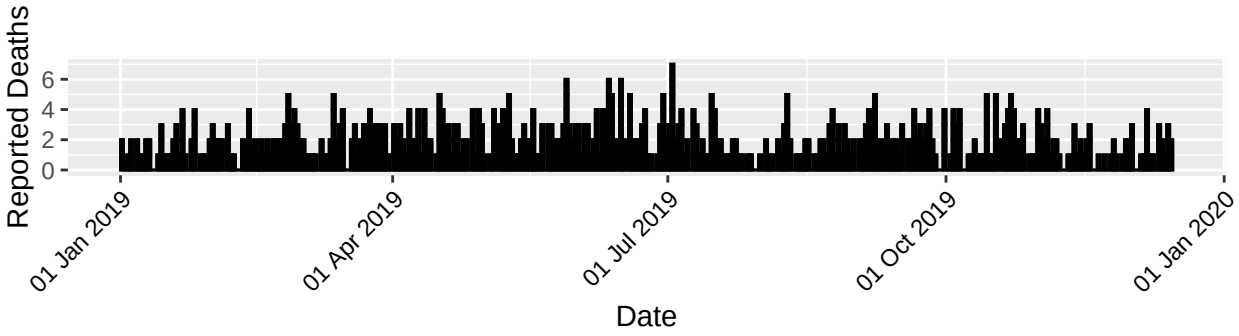


This pattern appears to suggest that deaths spiked in the beginning of June and end of August into

September. However, with such a small group of people in such a small amount of time, it is not clear that these numbers are abnormally high since we don't know what normal looks like for this population.

Let's examine the 2nd set of data with around 900 people over a period of approximately 11 months.

Epi Curve 2



This graph suggests that spikes occured in June or July but with more data it appears even less clear if the days which have higher numbers of deaths are outside of the normal variation in this population.

As is common with data in the human rights context, it can often be difficult to obtain reliable statistics on incidence of a particular outcome of interest so we do not have a reference for the underlying rate of deaths in this population. Some group may have an interest in that data not existing in one central location or it may simply never have been collected. Either way, this suggests the need to try a classical statistical approach.

These deaths are counts collected in a specific time period. This means that they can be assessed in the context of whether or not they represent a normally distributed discrete random variable in a Poisson distribution. Data which follow a normal Posisson distribution exhibit expected amounts of variation around the mean number of events in a given time period.

In order to establish whether or not these data belong within that normal amount of variation we examine both the means and maximum deaths per day. Given the mean number of deaths reported per day prior to the date of interest, is the mean number of deaths per day significantly different after this date? Secondly, given the mean number of deaths per day, is the highest recorded number of deaths per day significantly different than the means for each time period?

Test

To determine this, we call in data from the test task. In this task we prepared our data for hypothesis testing by obtaining the counts for each date, i.e., the number of times each date is reported. Since a death wasn't reported at least once per day we also added in dates where we had no reported deaths with counts of zero.

We use the poisson.test function in R's stats package to determine if the differences in the means prior to and following 21 August 2019 are significantly different from one another at the 95% confidence interval.

Our null hypothesis is that the means are not statistically significantly different from one another (mupost-mupre = 0) and our alternative hypothesis is that the means are significantly different from one another (mupost-mupre =! 0). If we observe a p-value < 0.05, we would reject the null, indicating that the difference between the means is not zero.

```
files <- list(cts1 = here::here("test/output/counts1_pre.txt"),
              cts2 = here::here("test/output/counts1_post.txt"),
              cts3 = here::here ("test/output/counts2_pre.txt"),
              cts4 = here::here ("test/output/counts2_post.txt"))

cts1pre <- readr::read_delim(files$cts1, delim="|")
cts1post <- readr::read_delim(files$cts2, delim="|")
cts2pre <- readr::read_delim(files$cts3, delim="|")
cts2post <- readr::read_delim(files$cts4, delim="|")

# set 1
# mean deaths per day prior to 21 August 2019
mu1pre <- round(mean(cts1pre$n))

# mean deaths per day following 21 August 2019
mu1post <- round(mean(cts1post$n))

#test
res1 <-poisson.test(mu1pre, 1, mu1post, alternative = "g", conf.level = 0.95)
res1
```

```
##
##  Exact Poisson test
##
## data:  mu1pre time base: 1
## number of events = 1, time base = 1, p-value = 0.8647
## alternative hypothesis: true event rate is greater than 2
## 95 percent confidence interval:
##  0.05129329        Inf
## sample estimates:
## event rate
##          1
```

```
# set 2
# mean deaths per day prior to 21 August 2019
mu2pre <- round(mean(cts2pre$n))

# mean deaths per day following 21 August 2019
mu2post <- round(mean(cts2post$n))
```

```r
#test
res2 <- poisson.test(mu2pre, 1, mu2post, alternative = "g", conf.level = 0.95)
res2
```

```
##
##  Exact Poisson test
##
## data:  mu2pre time base: 1
## number of events = 2, time base = 1, p-value = 0.594
## alternative hypothesis: true event rate is greater than 2
## 95 percent confidence interval:
##  0.3553615       Inf
## sample estimates:
## event rate
##          2
```

In the first dataset, prior to 21 August 21 2019, there was an average of 1 death reported per day and after 21 August 2019 there was an average of 2 deaths reported per day. We cannot reject the hypothesis that the mean number of deaths reported after 21 August 2019 is not significantly greater than the mean number of deaths reported prior to 21 August 2019 at the 95% confidence interval ($p = 0.86$).

In the second dataset, prior to 21 August 21 2019, there was an average of 2 deaths reported per day and after 21 August 2019 there was an average of 2 deaths reported per day. We cannot reject the hypothesis that the mean number of deaths reported after 21 August 2019 is not significantly greater than the mean number of deaths reported prior to 21 August 2019 at the 95% confidence interval ($p = 0.60$).

Though the means are not significantly different from one another, there are some days within each dataset on which we see numbers of reported deaths which are higher than the average, sometimes much higher. Does this suggest a spike or is this part of the normal amount of variation we would expect to see within a randomly distributed Poisson variable?

Our null hypothesis is that the maximum number of reported deaths per day is not statistically significantly different from the mean number of deaths reported per day (maxpre - mupre = 0) and our alternative hypothesis is that the maximum is significantly different from the mean (maxpre-mupre =! 0). If we observe a p-value < 0.05, we would reject the null, indicating that the maximum value is significantly different from the mean reported deaths per day.

```r
# set 1
# maximum deaths per day prior to 21 August 2019
max1_pre <- max(cts1pre$n)

# maximum deaths per day following 21 August 2019
max1_post <- max(cts1post$n)

# set 2
# maximum deaths per day prior to 21 August 2019
max2_pre <- max(cts2pre$n)

# maximum deaths per day following 21 August 2019
max2_post <- max(cts2post$n)

# likelihood of observing the maximum deaths/day given the mean

# set 1
```

```
like1pre<-poisson.test(mu1pre, 1, max1_pre, alternative = "g", conf.level = 0.95)

like1post<-poisson.test(mu1post, 1, max1_post, alternative = "g", conf.level = 0.95)

# set 2
like2pre<-poisson.test(mu2pre, 1, max2_pre, alternative = "g", conf.level = 0.95)

like2post<-poisson.test(mu2post, 1, max2_post, alternative = "g", conf.level = 0.95)
```

In the first dataset, prior to 21 August 21 2019, there was an average of 1 death reported per day and a maximum of 4 deaths reported per day. We cannot reject the hypothesis that the maximum number of deaths reported prior to 21 August 2019 is not significantly greater than the mean number of deaths reported prior to 21 August 2019 at the 95% confidence interval (p = 0.98).

After 21 August 2019 there was an average of 2 deaths reported per day and a maximum of 5 deaths reported per day. We cannot reject the hypothesis that the maximum number of deaths reported following 21 August 2019 is not significantly greater than the mean number of deaths reported following 21 August 2019 at the 95% confidence interval (p = 0.96).

In the second dataset, prior to 21 August 21 2019, there was an average of 2 deaths reported per day and a maximum of 7 deaths reported per day. We cannot reject the hypothesis that the maximum number of deaths reported prior to 21 August 2019 is not significantly greater than the mean number of deaths reported prior to 21 August 2019 at the 95% confidence interval(p = 1.0).

After 21 August 2019 there was an average of 2 deaths reported per day and a maximum of 5 deaths reported per day. We cannot reject the hypothesis that the maximum number of deaths reported following 21 August 2019 is not significantly greater than the mean number of deaths reported following 21 August 2019 at the 95% confidence interval (p = 0.96).

Write

Finally, in what struck me as the most incredible part of this process, is the write task. This entire report was generated from an R markdown file stored in the write task's src folder. This still excites me and now I want to write as many reports as I can this way. An especially fun detail is the use of embedded variables. Throughout this report I've reported statistics about this dataset like the means . If we suddenly received more data and I had to regenerate this report, instead of having to manually check that each number I cited was updated, I could drop the files into the clean task's import folder and update the entire report.

Acknowledgements

Learning HRDAG's data analysis process has been an incredible learning opportunity for me. Looking at projects from the very macro ("How many tasks do I have?") to the very micro ("Can I pipe this line?") levels is a fundamental shift in the way I had been working and I know I will take this into my future projects.

The project architecture structure discussed in this post, with discrete tasks using Makefiles, was developed by Drs. Scott Weikart and Jeff Klingner and formalized around 2007. I like to thank Tarak Shah for his incredibly motivating post on the dangers of .Rproj style of project management, Dr. Amelia Hoover Green for her detailed post on what it means to clean and canonicalize data, Dr. Megan Price for her encouragement, and most of all Dr. Patrick Ball for inviting me to solve this puzzle and for his patience as learned from the best.