

Estimating intrinsic survival rates for cardiac surgeons

James Geddes

Startup

```
library(methods)    # because otherwise RScript fails (I don't know why)
library(XML)        # needed by rvest, I think
library(httr)       # http requests
library(rvest)      # web scraping
library(xml2)
library(grid)
library(gridExtra)
library(tidyverse)  # tidy data and family
library(stringr)    # String manipulation
```

Overview

There exists published data on the outcomes of cardiac surgery, by surgeon, covering the period April 2012 to March 2015. One might be interested in seeing what one can say about one's own surgeon from this data. Unfortunately the data is in a form that is hard to parse into a useable dataset; that is the main task of this challenge.

Data on cardiac surgery outcomes is collected by the National Adult Cardiac Surgery Audit (NACSA), a survey managed by the National Institute for Cardiovascular Outcomes Research (NICOR) with direction from the Society for Cardiothoracic Surgeons (SCTS). NACSA itself is part of the National Clinical Audit Patient Outcomes Programme (NCAPOP) commissioned by the Healthcare Quality Improvement Partnership (HQIP).

The data has been heavily preprocessed, notably to adjust the outcome variable for case mix by surgeon but also to impute missing variables; the raw data is not made public.

SCTS publishes this data but in a form that it almost entirely unusable for aggregate calculations as it consists of a chart for each surgeon, available only as an image file.

However, the NHS republishes the data under the general rubric of “My NHS”, although they do not publish case mix. Nonetheless, the dataset is still awkward to obtain as a single entity and doing so provides an example of the “parsing” stage of data wrangling.

Obtaining the data

The following code either downloads the datasets from the original source and saves a copy to disk or reads in the saved files, depending on which part of the code is commented out. Re-reading the files is not quite equivalent to downloading but I don't think it makes a difference to the analysis.

```
gen_filename <- function(n) paste0("page", n, ".html")

### Download each of three pages of at most 100 records and write to a file

## url_NHS <- "https://www.nhs.uk"
```

```
## path_performance = "service-search/consultants/performanceindicators/1018"
##
## raw_data <- 1:3 %>%
##   map( ~ GET(modify_url(url = url_NHS,
##                         path = path_performance,
##                         query = list(CurrentPage = ., PageSize = 100))))
##
## walk(1:3, ~ cat(content(raw_data[[.]], "text"), file = gen_filename()))

### Read saved webpages from files
## Assuming this code is in surgeons/code and the data is in surgeons/data/raw

url_NHS <- "file://"
path_performance <- paste0(getwd())

raw_data <- 1:3 %>%
  map( ~ GET(modify_url(url = url_NHS,
                        path = paste0(path_performance, "../data/raw/", gen_filename()))))
```

A set of records, one for each surgeon, is published at: <https://www.nhs.uk/service-search/consultants/performanceindicators/1018>. However, it is not possible to obtain all records in a single view of a web page, as a maximum of 100 results may be shown at one time. Therefore one has to obtain at least three web pages. The above code downloads the three pages (chosen by the query string `?CurrentPage=n`) but does not check to ensure that three are either necessary or sufficient.

(I tried manually editing the `PageSize` parameter in the query string but the website appears to ignore requests for more than 100 results.)

Parsing the data

Description

Within the HTML file, the data is held as a table. The structure is roughly as follows:

```
...
<table class="indicators items-2">
  <tr class="fctitles"> HEADER_ROW </tr>
  <tr class="fcinfo"> INFO_ROW </tr>
  <tr> CONSULTANT_ROW </tr>
  <tr> CONSULTANT_ROW </tr>
  ...
</table>
```

There is one `CONSULTANT_ROW` per consultant. Its structure is as follows:

```
<td class="fcdetails consultantname fc-first">
  CONSULTANT
</td>
<td>
  VOLUME
</td>
<td>
  SURVIVAL_RATE
</td>
```

There are three cells in each row: one holds details of the consultant, one the volume of operations in the period under study, and one the (risk-adjusted) survival rate.

So far, fairly clear. Things are slightly hairier within each cell. A `CONSULTANT` looks like this:

```
<dl>
  <dt class="hidden">Name</dt>

  <dd class="consultantname">
    <a href="www.nhs.co.uk/profiles/consultant/4383798">
      Qamar Abid
    </a>
  </dd>

  <dt class="hidden">GMC number</dt>
  <dd>GMC membership number: 4383789</dd>
</dl>

<div class="consultant-trusts" >
  <p>Provides services for:</p>
  <dl>
    <dd>University Hospitals of North Midlands</dd>
    <dd>Another Hospital</dd>
    <dd>Another Hospital</dd>
  </dl>
</div>
```

The tags `<dl>`, `<dt>`, and `<dd>` were new to me. `<dl>` introduces a “description list”, a set of name/value pairs. The name (or term) is enclosed in `<dt></dt>` and the description is enclosed in `<dd></dd>`.

The next two fields are simpler but still not straightforward. A `VOLUME` is:

```
<div class="rawtext">450 operations</div>
<a href="http://scts.org/modules/surgeons/surgeon.aspx?id=358">View Source Information</a>
```

and a `SURVIVAL_RATE` is:

```
<br />
Within expected limits with a value of 98.95%
<a href="http://scts.org/modules/surgeons/surgeon.aspx?id=358">View Source Information</a>
```

Challenges

Apart from the problem of parsing HTML, there are five main challenges in extracting a usable dataset from this data:

1. The GMC number and volume and mortality rate figures are embedded in a text string.
2. Data is sometimes held in `<dd>` nodes, sometimes in `<div>` nodes, and sometimes neither.
3. The table is not in first normal form (and, hence, not a relation).
4. There may be any number of hospitals per consultant (possibly including zero).
5. The full dataset is broken over three tables.

As an aside, the use of HTML description lists appears to be inconsistent. One could imagine that their meaning is “a set of values, of the same type, but whose number is variable” (such as, for example, the set of hospitals for a given consultant). But this seems to clash with their use as “a fixed set of properties of

a consultant” (ie, name and GMC number). Furthermore, the HTML standard requires at least one <dt> associated with a set of <dd>s, and this requirement is violated in the hospital list.

Extracting data into a table

Before extracting the data, it’s worthwhile considering how we are going to represent the list of hospitals. Representing lists of values of unknown length is always tricky, because they don’t fit naturally into rigid tables. (The other problematic data structures are hierarchical relationships.)

Assuming that our main table will be a set of consultants, one row per consultant, there are three reasonably obvious choices for the set of hospitals associated with each consultant:

1. As a list (of varying length) within a single field;
2. As a boolean vector (whose length, the total number of hospitals, is fixed);
3. As a set of rows in a separate table of consultant/hospital pairs.

Option (3) is the “database solution”; option (2) would perhaps be chosen for modelling. I shall go with option (1) for now.

The R package `rvest` (“harvest”) is designed to extract particular components of a web page. First we parse the HTML:

```
## Parse the html returned by each web page request
parsed_data <- raw_data %>% map(~ content(., "parsed"))
```

```
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
```

Then extract consultant names and numbers:

```
## Extract consultant names
surgeons <- data_frame(name =
  parsed_data %>%
  map( ~ html_nodes(., ".consultantname a") %>% html_text()) %>%
  unlist())

## Extract GMC numbers
## The ~ selector finds one element preceded by another
## A GMC reference number is seven digits
surgeons$gmc_number <-
  parsed_data %>%
  map( ~ html_nodes(., ".consultantname~ dd") %>% html_text()) %>%
  unlist() %>%
  str_extract("[0-9]{7}")
```

Extracting hospitals is tricky because there can be more than one for each consultant.

```
surgeons$hospital <-
  parsed_data %>%
  map( ~ html_nodes(., ".consultant-trusts")) %>%
  flatten() %>% # Make one long list at this point
  map( ~ html_nodes(., "dd") %>% html_text()) # Extract one char vector per consultant
```

Extract patient volumes:

```
## A typical entry is "311 operations"; the regexp [0-9]* extracts the numeric part
surgeons$volume <-
  parsed_data %>%
  map( ~ html_nodes(., ".rawtext") %>% html_text()) %>%
  unlist() %>%
  str_extract("[0-9]*") %>%
  as.numeric()
```

Extracting the survival rates is more difficult, because we have to parse things like “99.53%” and “100%” out of the string.

```
## The "td:nth-child(3)" finds a td that is the third child element of its parent.
## It also picks up the header row, so we remove the first element:
## The function `(x, -1)` is equivalent to `x[-1]`.
surgeons$survival_rate <-
  parsed_data %>%
  map( ~ html_nodes(., "td:nth-child(3)") %>% html_text() %>% `[-1]` %>%
  unlist() %>%
  str_extract("[0-9]+(?:\\. [0-9]+)?(?:%)" ) %>%
  as.numeric() %>% `*`(0.01)
```

Finally, we should have a nice table:

```
surgeons

## # A tibble: 205 × 5
##       name gmc_number hospital volume survival_rate
##       <chr>      <chr>      <list> <dbl>      <dbl>
## 1 Qamar Abid    4383789 <chr [1]>    450      0.9895
## 2 Haitham Abunasra 4106663 <chr [3]>    469      0.9654
## 3 Yasir Abu-Omar  4409689 <chr [1]>    529      0.9907
## 4 Alsir Ahmed    4760313 <chr [2]>    148      0.9660
## 5 Ishtiaq Ahmed   4194671 <chr [2]>    278      0.9737
## 6 James Aitchison 4019811 <chr [1]>    320      0.9964
## 7 Enoch Akowuah   4399885 <chr [1]>    594      0.9873
## 8 Ayyaz Ali       4511416 <chr [1]>    391      0.9903
## 9 Simon Allen     2711197 <chr [1]>    540      0.9747
## 10 Shirish Ambekar 4687988 <chr [1]>    332      0.9959
## # ... with 195 more rows
```

Analysis

The dataset is summarised in the figure.¹ We use the following model: there is a parameter, $\pi_i \in [0, 1]$, for each surgeon i , giving that surgeon’s intrinsic patient survival rate. The parameters π_i are imagined to arise as independent draws from a beta distribution. The outcome of any operation by surgeon i is a random draw from a Bernoulli distribution with parameter π_i . For simplicity, the prior for the π_i s is taken to be an empirical fit of a beta distribution to the observed distribution of survival rates, as shown in the figure.

That is to say, for a given surgeon i , the probability of a successful outcome in some operation is

$$p(\text{success}) = \pi_i,$$

¹It’s possible to imagine building a more sophisticated model which includes “hospital-level” effects (using data published on the STCS website) but I have not done so here.

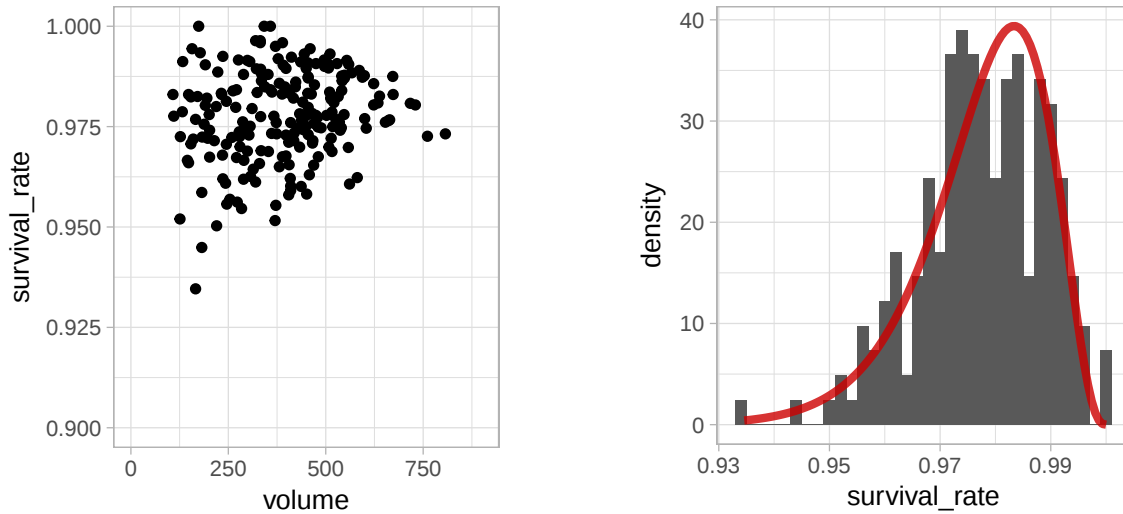


Figure 1: Summary of survival rates by surgeon. The left-hand plot shows the survival rate of each surgeon, in what is sometimes called a “funnel plot”. The right-hand plot shows the spread of survival rates for each surgeon together with an empirical fit to a beta distribution.

where

$$\pi_i \sim \text{beta}(\alpha_0, \beta_0),$$

for some parameters α_0 and β_0 .

```
## Calculation of the parameters of the empirical prior
## by means of the "method of moments"
pi_mean <- mean(surgeons$survival_rate)
pi_var <- var(surgeons$survival_rate)

temp <- pi_mean * (1 - pi_mean) / pi_var - 1

alpha0 <- pi_mean * temp
beta0 <- (1 - pi_mean) * temp
```

The estimated parameters are: $\alpha_0 = 165.5$ and $\beta_0 = 3.78$.

For a particular surgeon, i , we observe N_i operations (called `volume` in the R code), of which a proportion $r_i N_i$ are successful (r_i is called `survival_rate` in the R code). The posterior distribution for s_i is then:

$$\pi_i | N_i, r_i \sim \text{beta}(\alpha_0 + r_i N_i, \beta_0 + N_i(1 - r_i)).$$

So one answer to the challenge is to compute the posterior distribution of π_i for each surgeon. This is done below; the figure shows, for each surgeon, upper and lower quartiles as well as the posterior median.

```
### Compute posterior
surgeons <- surgeons %>%
  mutate(alpha = alpha0 + volume * survival_rate,
         beta = beta0 + volume * (1 - survival_rate)) %>%
  mutate(posterior_median = qbeta(0.5, alpha, beta),
         lower_q = qbeta(0.25, alpha, beta),
         upper_q = qbeta(0.75, alpha, beta))

### Plot posterior median, and upper and lower quartiles
ggplot(data = surgeons, aes(x = volume, y = posterior_median, ymin = lower_q, ymax = upper_q)) +
```

```
geom_point() + geom_linerange() +  
xlim(c(0, 900)) + ylim(c(0.9, 1.0)) +  
theme_light()
```

