**Counting digits appearing in page numbers**

The other day in a training session, the facilitators warmed people up into intellectual work with this group exercise:

* Count the number of times the digit “1” appears in the page numbers of a 90 page book.
* Then, count the number of times it appears in the page numbers of a 2,000 page book.

(Actual numbers vary slightly from those used so I don’t just give away the answers).

The first task is easy enough; you can pretty much count them in your head (something like “1, 2, 3-4, plus 8 equals 12, plus 7 equals 19” – if you follow me). The second task is obviously harder and the aim was to illustrate the need to break down more complicated problems into a structure for a solution.

But the alternative message that occurred to me of course is that “this is why we invented computers isn’t it?”. Here’s the one-liner in R that answers the second question, about the 2,000 page book:

length(stringr::str\_extract\_all(paste(1:2000, collapse = ""), pattern = "1", simplify = TRUE))

Basically, starting from the middle, this tells the computer:

* make a vector of all the numbers from 1 to 2,000
* collapse them all into a single big character string
* extract all the digits “1” from that string into a new vector which each element is just the extracted “1”
* count the length of that vector

The answer is 1,600.

What about if we wanted to generalise this though – for more page numbers, and different digits? After the training session was over, I had a go at this. For the sake of efficiency and speed, I treated this as a slightly different problem and I solved it with the help of the count\_digits function defined below which extracts the number of times a digit appears in a given page number; and then we use cumulative sums to get the eventual result for books of different lengths with minimal duplication of calculations.

Here’s the code to do this, and its application to a book of 2,500 pages, for all digits.

# more considered approach

library(tidyverse)

library(scales)

library(testthat)

library(stringr)

library(gridExtra)

count\_digits <- function(data, page\_number){

length(stringr::str\_extract\_all(paste(data, collapse = ""),

pattern = as.character(page\_number), simplify = TRUE))

}

# test that it still gets the known result for 2000 pages:

expect\_equal(sum(Vectorize(count\_digits)(data = 1:2000, page\_number = 1)), 1600)

#------------2500-------------

d <- expand.grid(n = 1:2500, digit = 0:9)

d$freq <- apply(d, 1, function(x){count\_digits(x[1], x[2])})

d <- d %>%

group\_by(digit) %>%

mutate(cum\_freq = cumsum(freq))

d %>%

ggplot(aes(x = n, y = cum\_freq, colour = as.factor(digit))) +

geom\_line() +

geom\_text(data = filter(d, n == max(n)), aes(label = digit), hjust = 0) +

theme(legend.position = "none") +

scale\_x\_continuous("Pages in book", label = comma) +

scale\_y\_continuous("Number of occurrences of digit in page number", label = comma) +

scale\_colour\_brewer(palette = "Spectral") +

ggtitle("Occurrence of digits in numbers in a simple sequence",

"How many times does each digit appear in the page number of a book of different lengths")

It’s quite a pretty pattern:

When we get a book of 75,000 pages it gets even prettier:



(Code not shown as it’s basically identical to the 2,500 page case).

Funnily enough, after doing the thinking above, I found out that one of my work colleagues who had done the same training session a few months earlier had reacted almost identically, in using a computer to count the results. If you’re at all interested in counting the digits appearing in page numbers, I thoroughly recommend [Paul Geil’s DigitCounter Shiny app](https://paul-geil.shinyapps.io/DigitCounter/) that he built in response.

**Benford’s law**

All this digit counting reminded me that I’d wanted to look into [“Benford’s ‘law’”](https://en.wikipedia.org/wiki/Benford%27s_law) at some point, and prompted me to make today the day. Benford observed that the frequency distribution of the leading digit in many real-life sets of numerical data follows a pattern. 1 appears as the first digit about 30.1% of the time, 2 17.6%, and so on.

If Benford’s law applies to a series of numbers, the probability of a number beginning with the digit d is:

Benford wasn’t the first person to notice this, although he did go into it particularly systematically. The reasons why it applies are interesting but I won’t go into them, they’re nicely explained in the Wikipedia page. Obviously in fact it is not always going to apply; for instance, the leading digits of the page numbers of a book aren’t going to follow this distribution. It does apply often enough though to be useful in detecting fraud (although in practice, automated fraud detection should use much more than this).

Benford’s law comes closer to application in distributions where the data cover at least two orders of magnitude. Take a look at these examples:

The normal distribution isn’t a bad fit because it covers a couple of orders of magnitude (lots of data in 0-1, some in the 1-10 range), but the best fits are the distributions with fatter tails in at least one direction, like the log-normal and the Cauchy.

Those simulations were measured with the count\_first\_digit function below:

#================Benford's 'law'============

#-----------set up---------------

#' Function to draw a chart of frequency of first digits and compare to that predicted by Benford's 'law'

#'

#' @param x a univariate numeric vector

#' @param desc description of x to use as subtitle

#' Other parameters are self-explanatory

#'

#' See https://en.wikipedia.org/wiki/Benford%27s\_law

count\_first\_digit <- function(x, desc, main\_title = "Frequency of first digits",

seed = 123, legend.position = "none",

bottom\_lab = "", left\_lab = "Leading digit"){

set.seed(seed)

# Extract the first digit from 1 to 9

y <- as.character(x)

y <- str\_extract(y, "[1-9]")

refs <- data\_frame(

digit = 1:9,

`Benford's law` = log10(1 + (1/1:9))

)

z <- table(y)

p <- data\_frame(digit = as.numeric(names(z)),

freq = as.numeric(z)) %>%

mutate(freq = freq / sum(freq)) %>%

rename(`Observed` = freq) %>%

right\_join(refs, by = "digit") %>%

gather(variable, value, -digit) %>%

ggplot(aes(x = as.ordered(digit), y = value, fill = variable)) +

geom\_col(position = "dodge") +

coord\_flip() +

scale\_fill\_manual("", values = c("grey85", "darkblue"),

guide = guide\_legend(reverse = TRUE)) +

ggtitle(main\_title,

desc) +

labs(x = left\_lab, y = bottom\_lab) +

theme(legend.position = legend.position)

return(p)

}

#---------------with simulated distributions---------------------

n <- 1000000

p1 <- count\_first\_digit(rnorm(n), "Standard normal distribution", legend.position = c(0.7, 0.7),

main\_title = "Frequency of first digits, simulated")

p2 <- count\_first\_digit(exp(rnorm(n)), "Standard log-normal distribution", "", left\_lab = "")

p3 <- count\_first\_digit(runif(n, 0, 1), "Standard uniform distribution", "", left\_lab = "")

p4 <- count\_first\_digit(runif(n, -85, 85), "[-85, 85] uniform distribution", "")

p5 <- count\_first\_digit(rgamma(n, 2), "Gamma distribution with shape 1, rate 1", "",

bottom\_lab = "Frequency", left\_lab = "")

p6 <- count\_first\_digit(rcauchy(n), "Standard Cauchy distribution", "", left\_lab = "")

grid.arrange(p1, p2, p3, p4, p5, p6, ncol = 3)

Having built that function, I tried out Benford’s law on some famous datasets built into R. The results are illuminating I think:

Only the closing prices of stock indices comees close to following the predicted pattern. The failure of the law to apply is partly because of smallish sample sizes but mostly because these datasets all cover too small a range of results for the first digit law to apply. For example, the monthly road casualty dataset is mostly in the low 100s, with an occasional value in the 90s or 80s. So the first digit law doesn’t apply. But I’m pretty sure that if we compared the monthly road casualties of many different countries, it would be a much closer fit.

When the data is from multiple diverse units of observation and hence covers several orders of magnitude, the law is much more likely to apply. For example, consider the population data set in tidyr package, a subset of World Health Organization tuberculosis data:

We see quite a close match.

The take-home story is to not expect the first-digit law, or Benford’s law, to apply unless you’ve got the right sort of data for it – data that ranges across several orders of magnitude.

Here’s the code that produced those real life tests of Benford’s law.

#--------with some real datasets----------

p1 <- count\_first\_digit(AirPassengers, "Monthly Airline Passenger Numbers\n1949-1960",

main\_title = "Frequency of first digits, real data",

legend.position = c(0.7, 0.7))

p2 <- count\_first\_digit(BJsales, "\nBox & Jenkins Sales Data", "", left\_lab = "")

p3 <- count\_first\_digit(CO2$uptake, "\nCarbon dioxide uptake in Grass Plants", "", left\_lab = "")

p4 <- count\_first\_digit(EuStockMarkets, "Daily Closing Prices of Major European\nStock Indices, 1991-1998",

"")

p5 <- count\_first\_digit(Seatbelts[ , "DriversKilled"], "\nRoad Casualties in Great Britain 1969-84",

"", left\_lab = "")

p6 <- count\_first\_digit(morley[, "Speed"], "\nSpeed of light estimated by Michelson", "", left\_lab = "")

grid.arrange(p1, p2, p3, p4, p5, p6, ncol = 3)

count\_first\_digit(population$population, "National population figures")