Working with R

A University of Queensland Advanced Workshop

Session 10: Some Programming Techniques

Bill Venables, CSIRO/Data61, Dutton Park

Rhetta Chappell, Griffith University

2–5 February, 2021

Contents

1 A simulation example

Find by simulation the relative volume of a sphere of diameter 2 to a square box of side 2?

For dimensions $d = 1, 2, 3, \ldots$.

The answer is known mathematically. The volume of the sphere of radius r is

$$V_d^{(S)} = \frac{\pi^{d/2} r^d}{\Gamma\left(d/2+1\right)}$$

and the volume of the square box of side $2r$ is clearly $V_c^{(B)} = 2^d r^d$. Hence the relative size is

$$R =^{\text{def.}}= \frac{V_d^{(S)}}{V_d^{(B)}} = \frac{\pi^{d/2}}{\Gamma\left(d/2+1\right) 2^d}$$

```r
rvball <- local({
  pi <- base::pi  ## make sure it is the right one

  function(d)
  structure(exp(d/2 * log(pi) - d * log(2) - lgamma(d/2 + 1)),
            names = as.character(d))
})
floor(1/rvball(1:10))
```

```
  1   2   3   4   5   6   7   8   9  10 
  1   1   1   3   6  12  27  63 155 401 
```

1.1 A simulation strategy

- Generate samples uniformly within the box centred at $(0, 0, \ldots, 0)$
- Count the number at a Euclidean distance no further than 1 from the origin.
- Take ratio.

```
mcrvball <- function(d, N = 100000, blocksize = 10000) {
  n2 <- inside <- 0
  while(n2 < N) {
    n1 <- n2
    n2 <- min(n2 + blocksize, N)
    No <- n2 - n1
    samp <- matrix(runif(No * d, -1, 1), No, d)
    inside <- inside + sum(rowSums(samp^2) < 1)
  }
  res <- list(dimensions = d, inside = inside,
              total = N, call = match.call())
  class(res) <- "mcrvball"
  res
}
```

An alternative implementation using the doParallel family of packages.

```r
mcrvball2 <- function(d, N = 100000, blocksize = 10000) {
  chunks <- idiv(N, chunkSize = blocksize) ## division iterator
  inside <- foreach(No = chunks, .combine = sum) %dopar% {
    samp <- matrix(runif(No*d, -1, 1), No, d)
    sum(rowSums(samp^2) < 1)
  }
  structure(list(dimensions = d, inside = inside,
                 total = N, call = match.call()),
            class = "mcrvball")
}
```

To use it:

```r
suppressPackageStartupMessages(library(doParallel))
cl <- makePSOCKcluster(detectCores() - 1)
registerDoParallel(cl)              ## not needed in "parallel" package

tst <- mcrvball2(10, N = 1e7, blocksize = 1e5)

stopCluster(cl)
rm(cl)
```

1.2 S3 methods

```r
print.mcrvball <- function(x, ...) {
  with(x, cat("Dim.:", dimensions,
              "Estimated:", signif(inside/total, 4),
              "Actual:", signif(rvball(dimensions), 4), "\n"))
  invisible(x)
}

Ops.mcrvball <- function(e1, e2) {  ## group generic
  if(inherits(e1, "mcrvball"))
  e1 <- with(e1, inside/total)

  if(!missing(e2) && inherits(e2, "mcrvball"))
  e2 <- with(e2, inside/total)
  NextMethod()
}
```

```r
for(i in 4:10) print(mcrvball(i, 1000000))
```

```
Dim.: 4 Estimated: 0.3087 Actual: 0.3084
Dim.: 5 Estimated: 0.1645 Actual: 0.1645
Dim.: 6 Estimated: 0.08091 Actual: 0.08075
Dim.: 7 Estimated: 0.03692 Actual: 0.03691
Dim.: 8 Estimated: 0.01587 Actual: 0.01585
Dim.: 9 Estimated: 0.006474 Actual: 0.006442
Dim.: 10 Estimated: 0.002508 Actual: 0.00249
```

```r
r <- numeric(7)
for(d in 4:10) r[d-3] <- floor(1/mcrvball(d)) ; r
```

```
[1]   3   6  12  27  62 160 390
```

Having a `call` component allows `update` to be used:

```
p10 <- mcrvball(10)
p10a <- update(p10, N = 1000000, blocksize = 100000)
c(1/p10, 1/p10a)
```

```
[1] 411.5226 402.7386
```

```
"%+%" <- function(e1, e2)
    UseMethod("%+%")
"%+%.mcrvball" <- function(e1, e2) {
    if(e1$dimensions != e2$dimensions) stop("ball dimensions differ!")
    res <- list(dimensions = e1$dimensions, inside = e1$inside + e2$inside,
                total = e1$total + e2$total, call = e1$call)
    class(res) <- "mcrvball"
    res
}
# p10 %+% p10a
floor(1/(p10 %+% p10a))
```

```
[1] 403
```

Automatic (lazy) vectorization

```r
Mcrvball <- Vectorize(mcrvball, vectorize.args = c("d", "N"), SIMPLIFY = FALSE)
Mcrvball(4:8, N = 1e6, blocksize = 1e5)
```

```
[[1]]
Dim.: 4 Estimated: 0.3082 Actual: 0.3084


[[2]]
Dim.: 5 Estimated: 0.1645 Actual: 0.1645


[[3]]
Dim.: 6 Estimated: 0.08064 Actual: 0.08075


[[4]]
Dim.: 7 Estimated: 0.03692 Actual: 0.03691


[[5]]
Dim.: 8 Estimated: 0.01581 Actual: 0.01585
```

1.3 Some lessons

- Vectorization. (`rvball`) and vectorization tools: (`Vectorize`)
- Taking the "whole object view" of the problem. (`mcrvball`)
- Object orientation: put all the information you are likely to need into the object and give it a class. (`mcrvball`)
- Methods for existing generics. (`print.mcrvball`)
- Group generic functions. (`Ops.mcrvball`)
- Binary operators and new generic functions. (%+%)

2 Intermission: On text processing

Data in the form of text is becoming more common and important.

R has many sophisticated tools and packages for text manipulation—here we just glimpse two elementary ones: *grep* and *(g)sub*: the first for finding, the second twin pair for fixing.

2.1 The Authorised Version of King James

The extdata subdirectory of the WWRCourse package folder contains two compressed text files, of the first and last books of AV:

```
subdir <- system.file("extdata", package = "WWRCourse")
dir(subdir, pattern = "\\.txt.gz$")     ## NB regular expression

[1] "Genesis.txt.gz"     "Revelations.txt.gz"
```

Our task is to compare *chapter length phrase length* and *word length* in these two document, as an aspect of literary style

2.2 Reading in

```
gen <- file.path(subdir, "Genesis.txt.gz")
rev <- file.path(subdir, "Revelations.txt.gz")
Genesis <- scan(gzfile(gen), what = "")  ## no need to gunzip()
Revelations <- scan(gzfile(rev), what = "")
rbind(head(Genesis), head(Revelations)) %>% noquote
```

```
     [,1]  [,2] [,3]       [,4]      [,5]  [,6]   
[1,] Gen.1 In   the        beginning God   created
[2,] Rev.1 The  Revelation of        Jesus Christ,
```

Both have chapter markers in them of the form `Gen.1`, `Rev.23`, which are not part of the text. We can get the chapter lengths from them, though.

```
grep("^Gen\\.[[:digit:]]+$", Genesis)[1:5] ## the idea
```

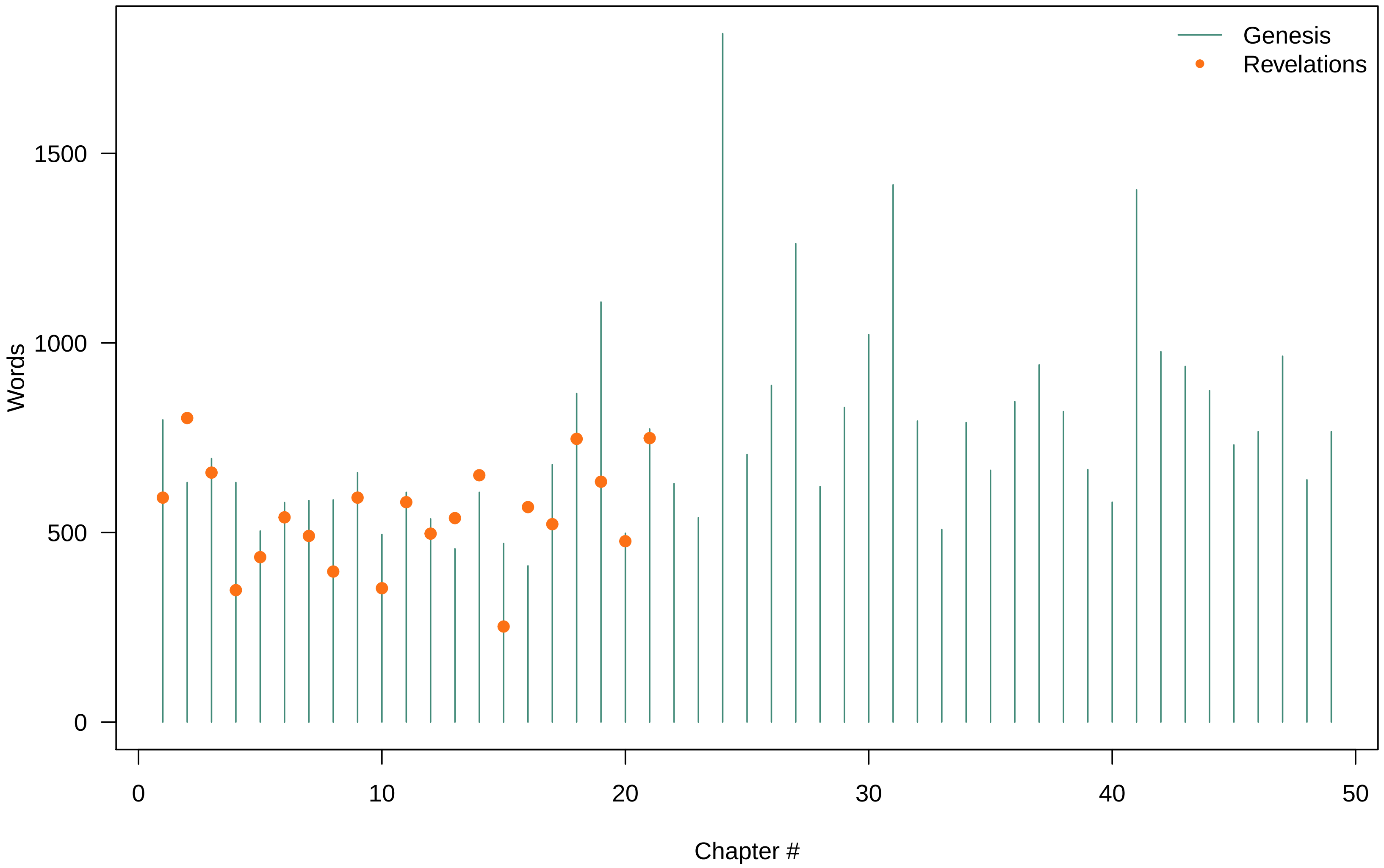
```
[1]    1  799 1432 2128 2761
```

```
chapterLengths <- function(txt,   ## cement it in a function & check
            book = substring(deparse(substitute(txt)), 1, 3)) {
  regex <- paste0("^", book, "\\.[[:digit:]]+$")
  where <- grep(regex, txt)
  diff(where - seq_along(where))
}
GenC <- chapterLengths(Genesis)
RevC <- chapterLengths(Revelations)
rbind(Gen = c(n = length(GenC), median = median(GenC)),
      Rev = c(n = length(RevC), median = median(RevC)))

     n median
Gen 49    695
Rev 21    540
```

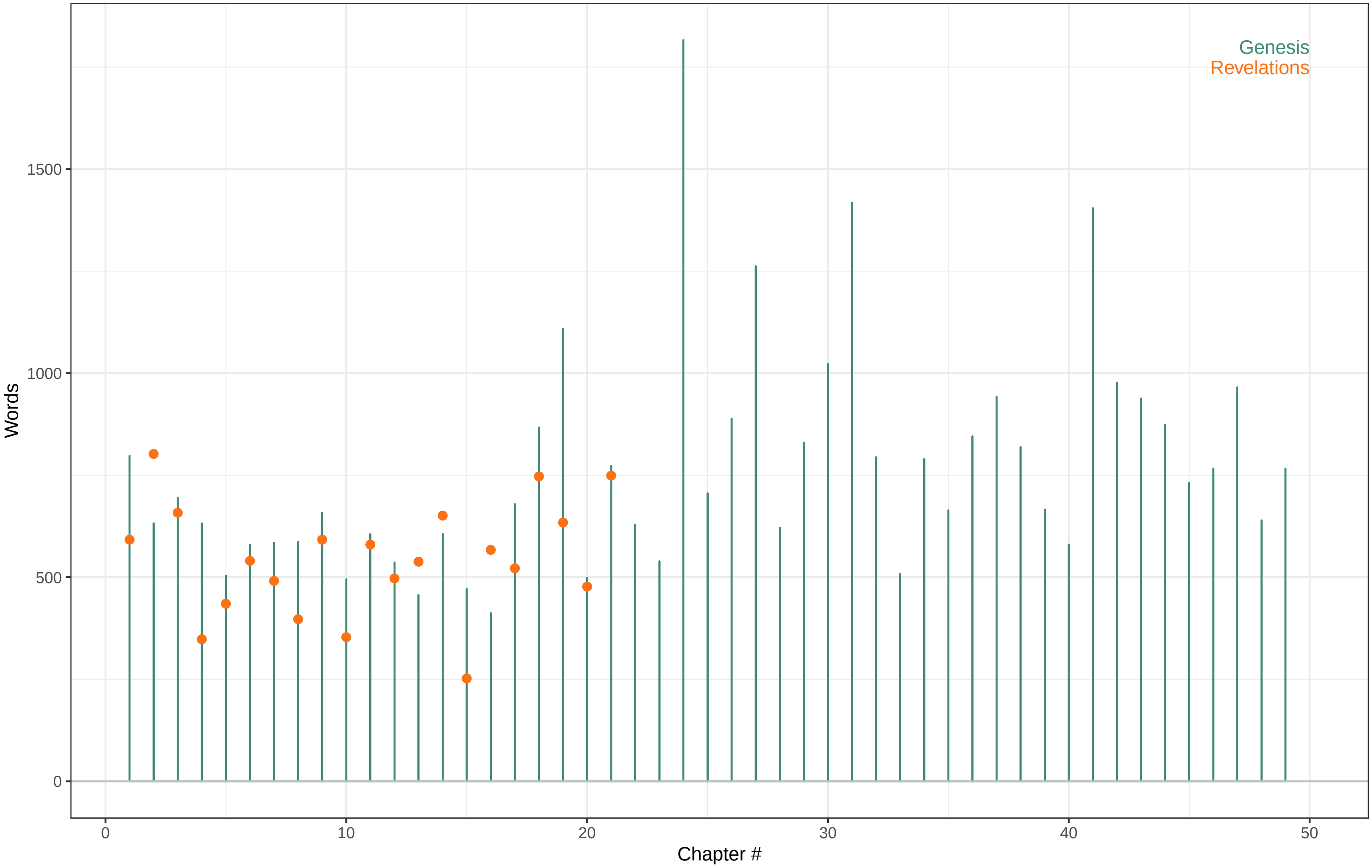
Revelations has fewer, and shorter chapters, but the chapter lengths are quite similar to the opening chapter lengths of Genesis:

```
plot(GenC, type="h", col = "#418A78", xlab = "Chapter #",
     ylab="Words", ylim = range(0, GenC, RevC))
points(RevC, pch=20, col="#FC7115", cex=1.5)
legend("topright", c("Genesis", "Revelations"), lty = c("solid", NA),
       pch=c(NA, 20), col = c("#418A78", "#FC7115"), bty="n")
```


Genesis
Revelations
Words
0
500
1000
1500
Chapter #
0
10
20
30
40
50

A (contrived) *ggplot* version

```r
gendat <- data.frame(x = c(1, rep(seq_along(GenC), each = 3)),
                     y = c(0, rbind(0, GenC, 0)))
revdat <- data.frame(x = seq_along(RevC), y = RevC)
ggplot() + aes(x,y) + geom_path(data = gendat, colour = "#418A78") +
  geom_point(data = revdat, colour = "#FC7115", size = 2) +
  geom_hline(yintercept = 0, colour = "grey", size = 0.5) +
  labs(x = "Chapter #", y = "Words") +
  annotate("text", x = 50, y = 1800, label = "Genesis",
           colour = "#418A78", hjust = 1) +
  annotate("text", x = 50, y = 1750, label = "Revelations",
           colour = "#FC7115", hjust = 1) + theme_bw()
```


Genesis
Revelations
Words
1500
1000
500
0
0
10
20
30
40
50
Chapter #

2.3 Phrase length

The first step is to strip out the chapter markers and leave the words, and punctuation, only.

```
Gen <- grep("^Gen\\.[[:digit:]]+$", Genesis, invert = TRUE, value = TRUE)
Rev <- grep("^Rev\\.[[:digit:]]+$", Revelations, invert = TRUE, value = TRUE)
noquote(rbind(head(Gen), head(Rev)))

     [,1] [,2]       [,3]      [,4]  [,5]    [,6] 
[1,] In   the        beginning God   created the  
[2,] The  Revelation of        Jesus Christ, which
```

We define a phrase as the words between successive (terminal) punctuation marks. These indicate a pause in the reading.

```
GenP <- diff(c(0, grep("[[:punct:]]$", Gen)))
RevP <- diff(c(0, grep("[[:punct:]]$", Rev)))
rbind(Gen = head(GenP), Rev = head(RevP))

    [,1] [,2] [,3] [,4] [,5] [,6]
Gen   10    6    2    9   12    3
Rev    5    5   12   13    8    7
```

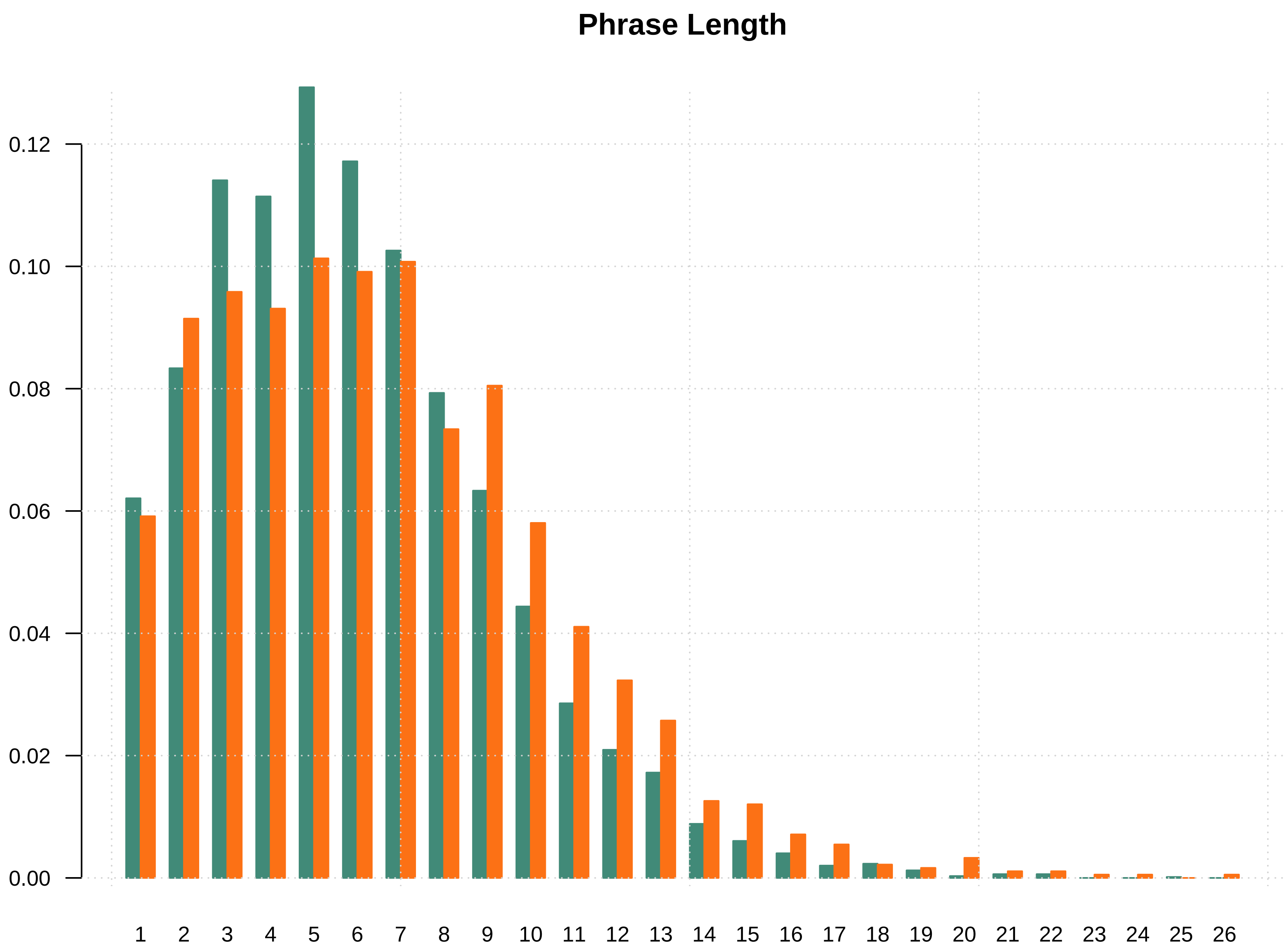
Put into a data frame and look at the relative frequencies.[a]

```
Phrase <- rbind(data.frame(Book = "Gen", Length = GenP),
                data.frame(Book = "Rev", Length = RevP))
tab <- with(Phrase, table(Book, Length)); t(tab)

      Book
Length Gen Rev
    1  400 108
    2  537 167
....
    23   0   1
    24   0   1
    25   1   0
    26   0   1

colours <- c(Genesis = "#418A78", Revelations = "#FC7115")
par(cex.axis = 0.8)
barplot(tab/rowSums(tab), beside = TRUE, main = "Phrase Length",
        fill = colours, colour = colours)
grid()
```

[a] Exercise: Assuming Length-1 is Poisson, test for a difference in mean phrase length between the two books


Phrase Length
0.12
0.10
0.08
0.06
0.04
0.02
0.00
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

Finally we come to word length distributions. For this we strip out any non-letter character and count the string lengths of what is left.

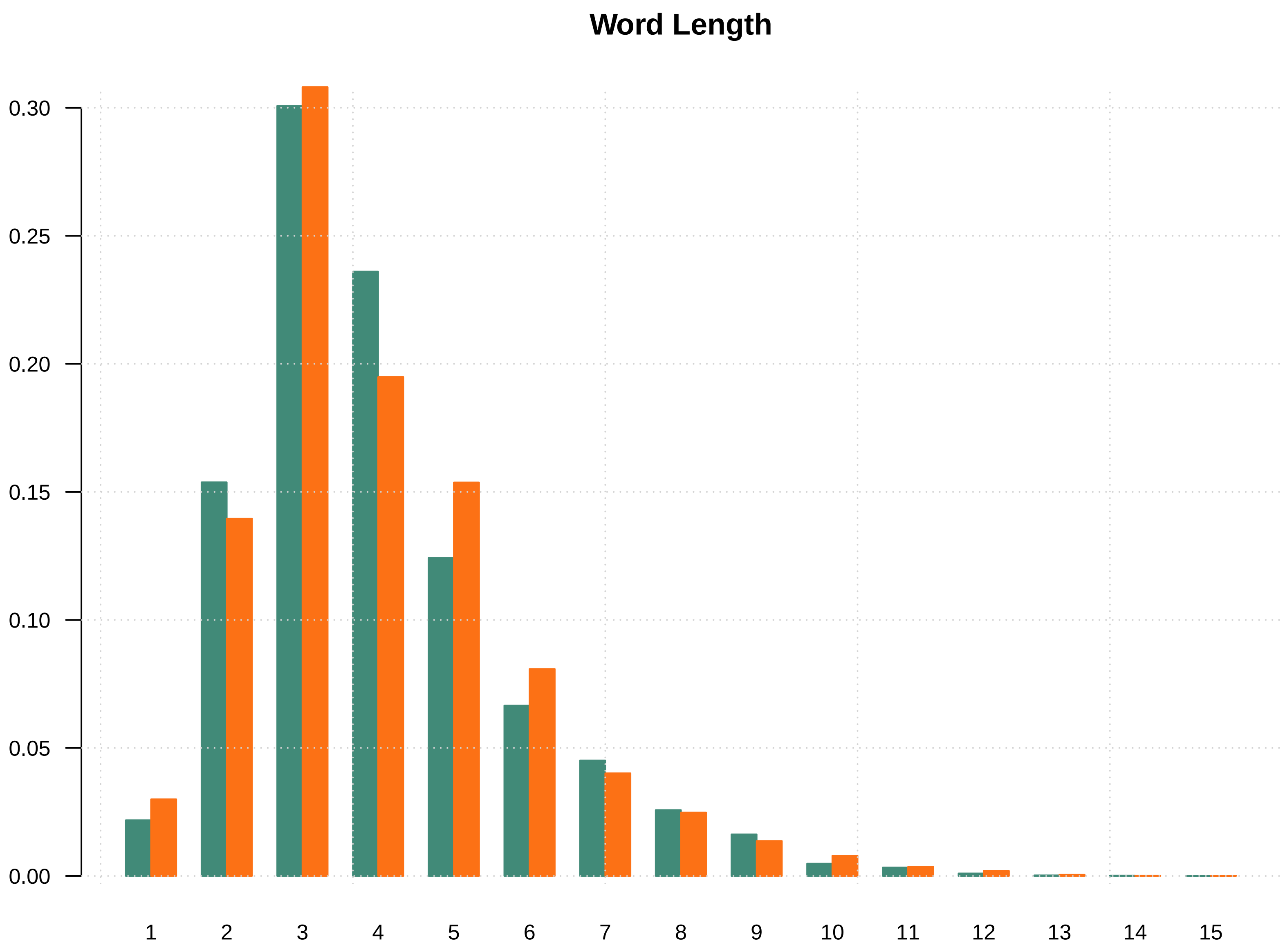
```
GenW <- nchar(gsub("[^[:alpha:]]", "", Gen))
RevW <- nchar(gsub("[^[:alpha:]]", "", Rev))
noquote(rbind(head(Rev), head(RevW)))  ## word 5 comma excluded

     [,1] [,2]       [,3] [,4]  [,5]    [,6]
[1,] The  Revelation of   Jesus Christ, which
[2,] 3    10         2    5     6       5

Words <- rbind(data.frame(Book = "Gen", Length = GenW),
               data.frame(Book = "Rev", Length = RevW))
tab <- with(Words, table(Book, Length)); t(tab)

      Book
Length  Gen  Rev
....
    14    7    2
    15    2    1

par(cex.axis = 0.8)
barplot(tab/rowSums(tab), beside = TRUE, main = "Word Length",
        fill = colours, colour = colours)
grid()
```


Word Length
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

3 Convolutions

Given two sequences of numbers, $a_i, i = 0, 1, \ldots$ and $b_j, j = 0, 1, \ldots$ their *convolution* is defined by

$$(ab)_k = \sum_{i+j=k} a_i b_j$$

(this is the operation involved in polynomial multiplication).

Consider some methods for doing this in **R**.

```
convolve0 <- function(a, b) {
  ab <- rep(0, length(a) + length(b) - 1)
  for(i in 1:length(a))
    for(j in 1:length(b))
      ab[i+j-1] <- ab[i+j-1] + a[i]*b[j]
  ab
}
###
convolve1 <- function(a, b) {
  ab <- rep(0, length(a) + length(b) - 1)
  ind <- 1:length(a)
  for(j in 1:length(b)) {
    ab[ind] <- ab[ind] + a*b[j]
    ind <- ind + 1
  }
  ab
}
```

```
convolve1a <- function(a, b) {
  if(length(a) < length(b)) Recall(b,a) else {
    ab <- rep(0, length(a) + length(b) - 1)
    ind <- 1:length(a)
    for(j in 1:length(b)) {
      ab[ind] <- ab[ind] + a*b[j]
      ind <- ind + 1
    }
    ab
  }
}
###
convolve2 <- function(a, b) {
  p <- outer(a, b)
  as.vector(tapply(p, row(p) + col(p), sum))
}
### Uses much less memory than convolve2
convolve2a <- function(a, b)
  as.vector(tapply(outer(a, b),
            outer(seq(along = a), seq(along = b), "+"), sum))
```

The young geek's version.

```
convolve_hw <- function(a, b) {
  stopifnot(require(dplyr))
  data.frame(x = as.vector(outer(a, b)),
             g = as.vector(outer(seq_along(a),
                                 seq_along(b), "+"))) %>%
    group_by(g) %>%
    summarise(conv = sum(x), .groups = 'drop') %>%
    .[["conv"]]
}
```

A C code version

File `VR_convolve.c` has the code:

```
void VR_convolve(double *a, int *na,
                 double *b, int *nb,
                 double *ab)
{
  int i, j, nab = *na + *nb - 1;

  for(i = 0; i < nab; i++) ab[i] = 0.0;
  for(i = 0; i < *na; i++)
    for(j = 0; j < *nb; j++)
      ab[i + j] += a[i] * b[j];
}
```

To compile:

```
$ R CMD SHLIB VR_convolve.c
```

```r
convolve3 <- function(a, b) {
  if(!is.loaded("VR_convolve")) {
    path <- file.path("SharedObjects",
                      paste("VR_convolve",
                            .Platform$dynlib.ext, sep=""))
    dyn.load(path)
  }
  storage.mode(a) <- "double"
  storage.mode(b) <- "double"

  .C("VR_convolve",
     a,
     length(a),
     b, length(b),
     ab = double(length(a) + length(b) - 1))$ab
}
```

The Rcpp revolution

- Makes it easy to write compiled code *without* the need for an **R**-side interface function (such as the above).
- Can interact directly with **R**-objects, using **R** to make *most* of the mode conversions
- Good tools for creating packages using such code, (*Rcpp.package.skeleton*, in particular).
- Drawbacks:
  - The code has to be written specifically for Rcpp, using the extensive **C++** header files,
  - There *can be* a small performance overhead using “RcppSugar”, (but quicker programming and fewer bugs).

Use Rcpp to extend the **R** system; Use the *dyn.load* to use of existing code, or minimally modified.

An Rcpp convolution function: file src/convolve3a.Cpp

```
#include <Rcpp.h>
using namespace Rcpp;

// [[Rcpp::export]]
NumericVector convolve3a(NumericVector x, NumericVector y)
{
    int nx = x.size(), ny = y.size(), nz = nx + ny - 1;
    NumericVector z(nz);  // set to 0 on creation. NB z() not z[] here!

    for(int i = 0; i < nx; ++i) {
        for(int j = 0; j < ny; ++j) {
            z[i+j] += x[i]*y[j];
        }
    }
    return z;
}
```

To make the code available as an **R** function, you need to have the necessary tools installed, but then:

```
library(Rcpp)
sourceCpp("src/convolve3a.cpp")
```

The **R** version of the function looks like:

```
convolve3a

function (x, y)
.Call(<pointer: 0x7f60987ca630>, x, y)
```

Alterntively, small functions can be compiled directly as a text string. The headers are assumed:

```
Rcpp::cppFunction('
NumericVector convolve3a(NumericVector x, NumericVector y)
{
    int nx = x.size(), ny = y.size(), nz = nx + ny - 1;
    NumericVector z(nz);  // set to 0 on creation. NB z() not z[] here!
    for(int i = 0; i < nx; ++i) {
        for(int j = 0; j < ny; ++j) {
          z[i+j] += x[i]*y[j];
        }
    }
    return z;
}
')
```

Such functions *must* be re-made the first time they are needed in an **R** session, in order to integrate them with the **R** program. They cannot be `save()`d in one session and `load()`ed in another, for example.

Permanent versions *can* be made in packages, however, with the system integration automatically happening when the package is loaded.

Some checks:

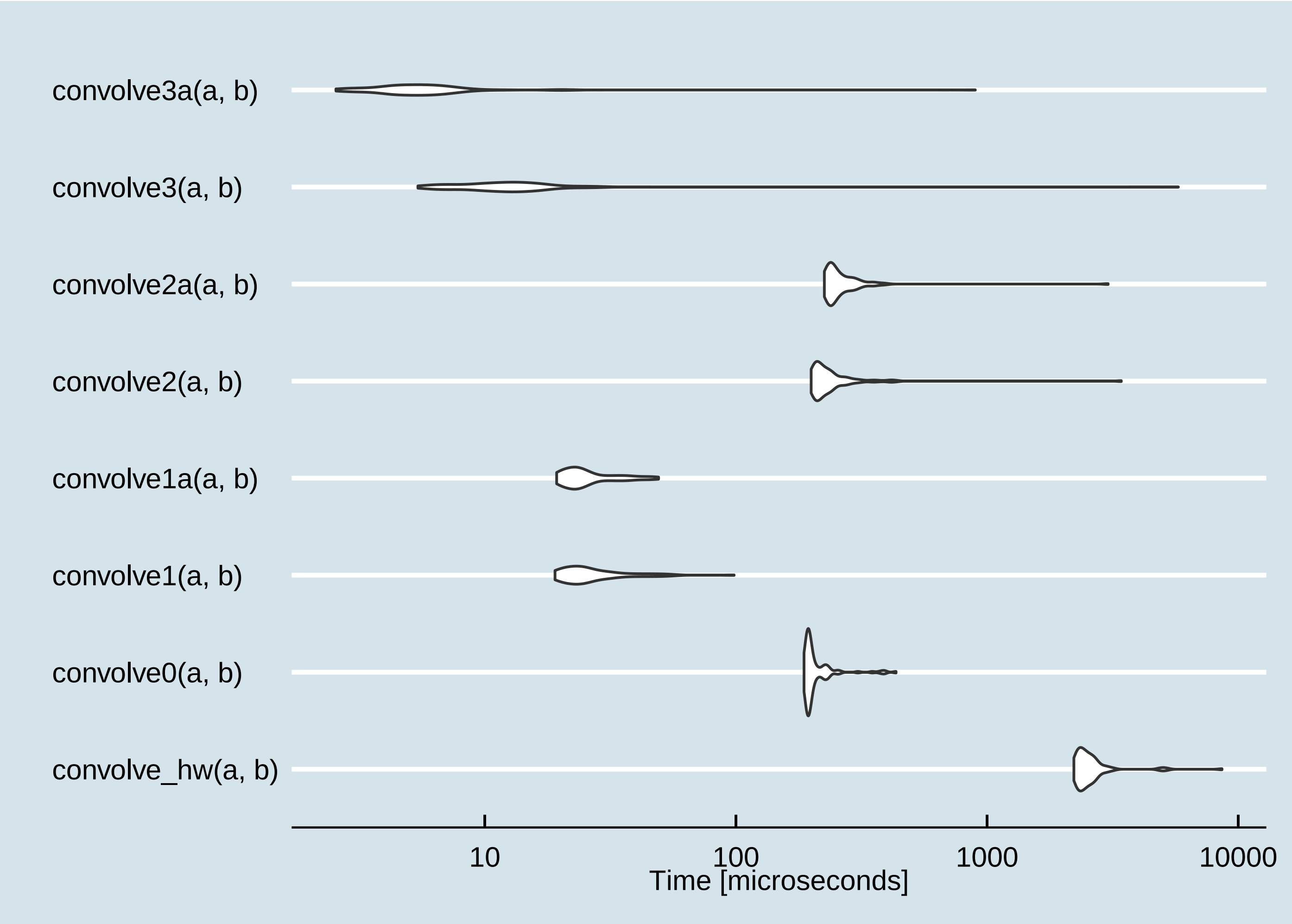
```r
a <- 1:3; b <- 4:7
rbind(convolve0(a,b), convolve1(a,b), convolve1a(a,b),
      convolve2(a,b), convolve2a(a,b), convolve3(a,b),
      convolve3a(a,b), convolve_hw(a, b))
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    4   13   28   34   32   21
[2,]    4   13   28   34   32   21
[3,]    4   13   28   34   32   21
[4,]    4   13   28   34   32   21
[5,]    4   13   28   34   32   21
[6,]    4   13   28   34   32   21
[7,]    4   13   28   34   32   21
[8,]    4   13   28   34   32   21
```

```
library(microbenchmark)
a <- 1:300; b <- 4:7
(b <- microbenchmark(convolve_hw(a, b),
                     convolve0(a,b),
                     convolve1(a,b),
                     convolve1a(a,b),
                     convolve2(a,b),
                     convolve2a(a,b),
                     convolve3(a,b),
                     convolve3a(a,b))) %>% summary() %>% arrange(median) %>%
                                .[, cs(expr, min, median, mean, max, cld)]
```

```
               expr      min    median       mean      max  cld
1  convolve3a(a, b)    2.555    5.3060   14.72706  897.728 a
2   convolve3(a, b)    5.417   12.1750   70.08196 5776.490 ab
3  convolve1a(a, b)   19.331   23.7105   26.52728   49.217 a
4   convolve1(a, b)   19.043   24.2050   28.04480   98.359 a
5   convolve0(a, b)  186.723  195.9835  212.63153  434.143  bc
6   convolve2(a, b)  199.330  221.9620  270.35601 3422.914   c
7  convolve2a(a, b)  224.943  246.1995  288.69014 3034.108   c
8 convolve_hw(a, b) 2216.659 2459.1560 2660.45093 8618.315    d
```

```
suppressMessages(autoplot(b)) + theme_economist()
```


convolve3a(a, b)
convolve3(a, b)
convolve2a(a, b)
convolve2(a, b)
convolve1a(a, b)
convolve1(a, b)
convolve0(a, b)
convolve_hw(a, b)
10
100
1000
10000
Time [microseconds]

Some timings (the old fashioned way):

```
a <- 1:1000; b <- 1:10000
library(rbenchmark)
benchmark(convolve0 (a,b), convolve1 (a,b), convolve1 (b,a),
          convolve1a(a,b), convolve1a(b,a), convolve2 (a,b),
          convolve2a(a,b), convolve3 (a,b), convolve3a(a,b),
          columns = c("test", "replications", "elapsed", "relative"),
          order = "relative", relative = "elapsed", replications = 20)
```

```
              test replications elapsed relative
9 convolve3a(a, b)           20   0.096    1.000
8  convolve3(a, b)           20   0.099    1.031
3  convolve1(b, a)           20   2.804   29.208
4 convolve1a(a, b)           20   2.810   29.271
5 convolve1a(b, a)           20   2.948   30.708
2  convolve1(a, b)           20   3.068   31.958
6  convolve2(a, b)           20  16.996  177.042
7 convolve2a(a, b)           20  17.492  182.208
1  convolve0(a, b)           20  22.478  234.146
```

Session information

Date: 2021-01-29

- R version 4.0.3 (2020-10-10), `x86_64-pc-linux-gnu`
- Running under: `Ubuntu 20.04.1 LTS`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0`
- LAPACK: `/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, utils
- Other packages: doParallel 1.0.16, dplyr 1.0.3, english 1.2-5, forcats 0.5.1, foreach 1.5.1, ggplot2 3.3.3, ggthemes 4.2.4, gridExtra 2.3, iterators 1.0.13, knitr 1.31, lattice 0.20-41, microbenchmark 1.4-7, patchwork 1.1.1, purrr 0.3.4, Rcpp 1.0.6, readr 1.4.0, scales 1.1.1, stringr 1.4.0, tibble 3.0.5, tidyr 1.1.2, tidyverse 1.3.0, WWRCourse 0.2.3, WWRData 0.1.0, WWRGraphics 0.1.2, WWRUtilities 0.1.2, xtable 1.8-4
- Loaded via a namespace (and not attached): assertthat 0.2.1, backports 1.2.1, broom 0.7.3, cellranger 1.1.0, cli 2.2.0, codetools 0.2-18, colorspace 2.0-0, compiler 4.0.3, crayon 1.3.4, DBI 1.1.1, dbplyr 2.0.0, digest 0.6.27, ellipsis 0.3.1, evaluate 0.14, fansi 0.4.2, farver 2.0.3, fractional 0.1.3, fs 1.5.0, generics 0.1.0, glue 1.4.2, grid 4.0.3, gtable 0.3.0, haven 2.3.1, highr 0.8, hms 1.0.0, httr 1.4.2, jsonlite 1.7.2, labeling 0.4.2, lazyData 1.1.0, lifecycle 0.2.0, lubridate 1.7.9.2, magrittr 2.0.1, MASS 7.3-53, Matrix 1.3-2, modelr 0.1.8, multcomp 1.4-15, munsell 0.5.0, mvtnorm 1.1-1, PBSmapping 2.73.0, pillar 1.4.7, pkgconfig 2.0.3, R6 2.5.0, randomForest 4.6-14, readxl 1.3.1, reprex 1.0.0, rlang 0.4.10, rpart 4.1-15, rstudioapi 0.13, rvest 0.3.6, sandwich 3.0-0, SOAR 0.99-11, splines 4.0.3, stringi 1.5.3, survival 3.2-7, TH.data 1.0-10, tidyselect 1.1.0, tools 4.0.3, vctrs 0.3.6, withr 2.4.1, xfun 0.20, xml2 1.3.2, zoo 1.8-8