…it seems reasonable to assign a baseline success rate of about 0.1 (i.e., about one application in 10 yields an interview). Suppose the one-page version is twice as effective and we apply to 100 positions with each. Then the statistical power, i.e. the probability of detecting a statistically significant effect, is given by:

```
library(Exact)
power.exact.test(p1 = 0.2, p2 = 0.1, n1 = 100, n2 = 100)
```

```
##
##      Z-pooled Exact Test
##
##          n1, n2 = 100, 100
##          p1, p2 = 0.2, 0.1
##           alpha = 0.05
##           power = 0.501577
##     alternative = two.sided
##           delta = 0
```

That is, we have only about 50% chances of detecting the effect with 0.05 confidence. This is not going to work; at a rate of about 10 applications per month, this would require 20 months.

Instead I'm going to frame this as a multi-armed bandit problem: I have two resumes and I don't know which one is the most effective, so I'd like to test them both *but* give preference to the one that seems to have the highest rate of success—also known as trading off exploration vs exploitation.

We'll begin by assuming again that we think each has about 10% chance of success, but since this is based on a limited experience it makes sense to treat this probability as the expected value of a beta distribution parameterized by, say, 1 success and 9 failures.

So whenever we apply for a new job, we:

- draw a new $p1$ and $p2$ from each beta distribution
- apply to the one with the highest drawn probability
- update the selected resume's beta distribution according to its success or failure.

Let's simulate this, assuming that we know immediately if the application was successful or not. Let's take the "true" probabilities to be 0.14 and 0.11 for the one-page and two-page resumes respectively. We'll keep track of the simulation state in a simple list:

```
new_stepper <- function() {
  state <- list(k1 = 1, n1 = 10, p1 = 0.14, k2 = 1, n2 = 10, p2 = 0.11)
  step <- function() {
    old_state <- state
    state <<- next_state(state)
    old_state
  }
  step
}
```

`new_stepper()` returns a closure that keeps a reference to the simulation state. Each call to

that closure updates the state using the `next_state` function:

```
next_state <- function(state) {
  p1 <- rbeta(1, state$k1, state$n1 - state$k1)
  p2 <- rbeta(1, state$k2, state$n2 - state$k2)
  pull1 <- p1 > p2
  result <- rbinom(1, 1, ifelse(pull1, state$p1, state$p2))
  if (pull1) {
    state$n1 <- state$n1 + 1
    state$k1 <- state$k1 + result
  } else {
    state$n2 <- state$n2 + 1
    state$k2 <- state$k2 + result
  }
  state
}
```

So let's now simulate 1000 steps:

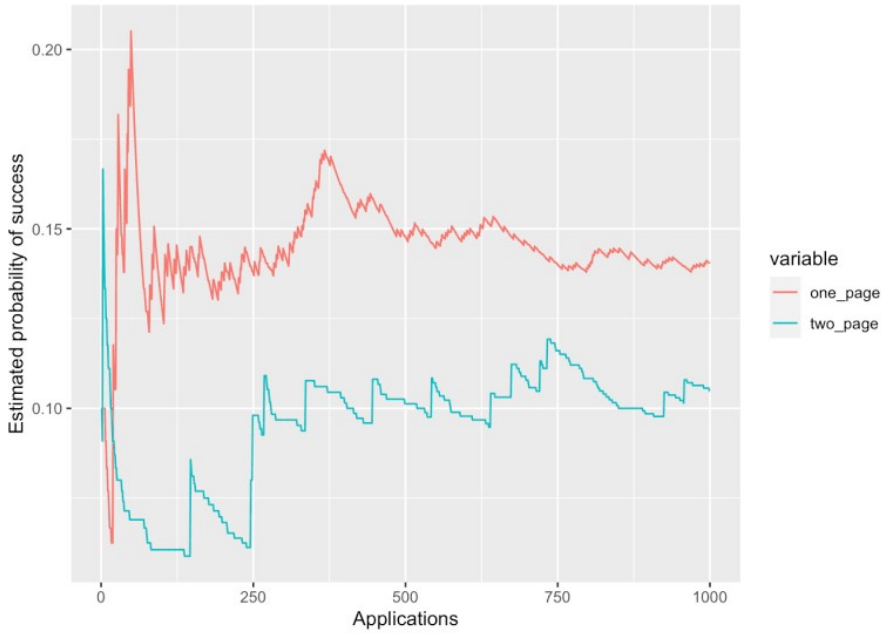
```
step <- new_stepper()
sim <- data.frame(t(replicate(1000, unlist(step()))))
```

The estimated effectiveness of each resume is given by the number of successes divided by the number of applications made with that resume:

```
sim$one_page <- sim$k1 / sim$n1
sim$two_page <- sim$k2 / sim$n2
sim$id <- 1:nrow(sim)
```

The follow plot shows how that estimated probability evolves over time:

```
library(reshape2)
library(ggplot2)
sim_long <- melt(sim, measure.vars = c('one_page', 'two_page'))
ggplot(sim_long, aes(x = id, y = value, col = variable)) +
  geom_line() +
  xlab('Applications') +
  ylab('Estimated probability of success')
```



Wouldn't that be nice

As you can see, the algorithm decides pretty rapidly (after about 70 applications) that the one-page resume is more effective.

So here's the protocol I've begun to follow since about mid-November:

- Apply only to jobs that I would normally have applied to
- Go through the entire application procedure, including writing cover letter etc, until uploading the resume becomes unavoidable (I do this mainly to avoid any personal bias when writing cover letters)
- Draw $p1$ and $p2$ as described above; select resume type with highest p
- Adjust the resume according to the job requirements, but keep the changes to a minimum and don't change the overall format
- Finish the application, and record a failure until a call for an interview comes in.