Statistical stopping criteria for automated screening in systematic reviews

Max Callaghan, Finn Müller-Hansen

Mercator Research Institute on Global Commins (MCC), Berlin
Priestley International Center for Climate, University of Leeds
Potsdam Institute for Climate Impact Research (PIK)

January 21, 2021

Introduction

- Identifying studies for an evidence synthesis project can be very time-consuming
- Machine learning (ML) can be trained using human decisions to predict the relevance of unseen documents
- Active learning means iterations of
  - Humans screen documents
  - ML is trained using human decisions
  - ML predicts relevance probability for all remaining documents
  - Documents are ordered in descending order of predicted relevance
- With active learning we can identify all relevant documents without having to view every single document.

Demonstrable work savings

Numerous studies have used datasets from previous reviews to show that machine learning has the potential to save work in systematic review (O'Mara-Eves et al. 2015; Cohen et al. 2006; Przybyła et al. 2018).

However, most of the work savings are just *potential* work savings predicated on *a priori* knowledge of how many relevant studies we are looking for.

To achieve these savings, we need to know when to stop screening: and existing approaches to deciding this are insufficient (Shemilt et al. 2014; Howard et al. 2020; Jonnalagadda and Petitti 2013). We need to be confident that we don't miss studies, and we need to be able to communicate this confidence.

Screening documents

We can set up a toy dataset with 2,000 documents (N_{tot}), of which 100 are relevant (ρ_{tot})

```
N_tot <- 2000 # total documents
r_tot <- 100 # total relevant documents

docs <- rep(0,N_tot)
docs[1:r_tot] <- 1
docs <- sample(docs, replace=F)
docs
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
##  [ reached getOption("max.print") -- omitted 1965 entries ]
```

```
sum(docs)
```

```
## [1] 100
```

Screening documents

At any point during screening, we can calculate recall (τ) by dividing the number of relevant documents we have seen (ρ_{seen}) by the total number of relevant documents

$$\tau = \frac{\rho_{seen}}{\rho_{tot}}$$

We usually accept that we won't identify every last document and aim for a recall target (τ_{tar}) like 95%

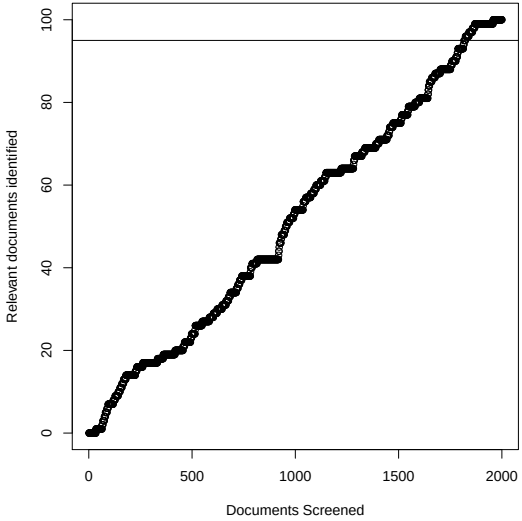
Screening documents

If we screen these at random, we will see 95% recall after seeing ~95% of all documents

```
par(pty="s")
plot(
  cumsum(docs),
  xlab="Documents Screened",
  ylab="Relevant documents identified",
  xlim=c(0,N_tot),
  ylim=c()
)
```

```
tau_target=0.95
tau <- cumsum(docs)/r_tot
which(tau>tau_target)[1]/N_tot
```

```
## [1] 0.912
```



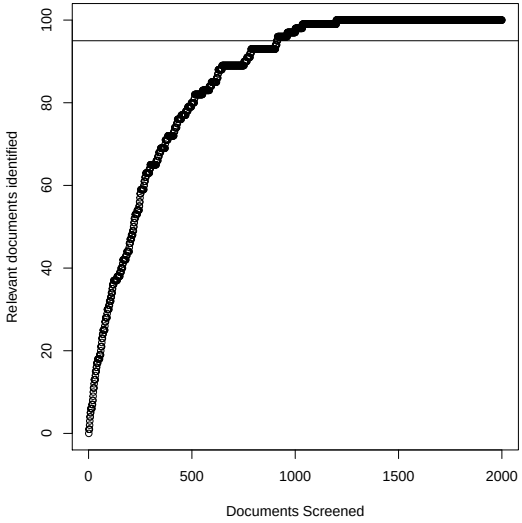
Screening documents with machine learning

If machine learning means we are more likely to view a relevant document in each draw, we will achieve 95% recall much earlier

```
weights = rep(1,N_tot)
weights[which(docs==1)] <- seq(
  10,2,length.out=r_tot
)
ordered_docs <- sample(
  docs, prob=weights, replace=F
)
plot(cumsum(ordered_docs))
abline(h=r_tot*.95)
```

```
tau <- cumsum(ordered_docs)/r_tot
which(tau>=tau_target)[1]/N_tot
```

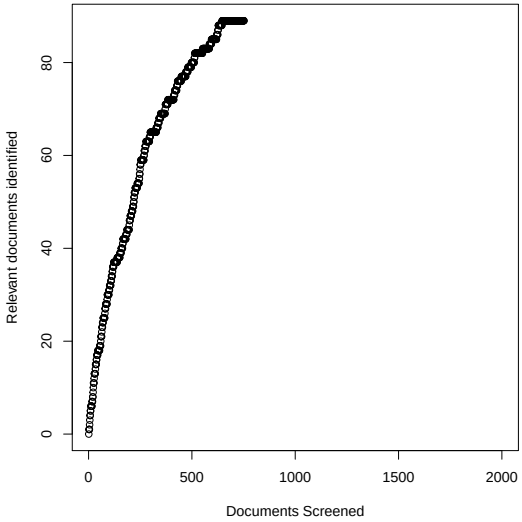
```
## [1] 0.4555
```



Screening documents with machine learning

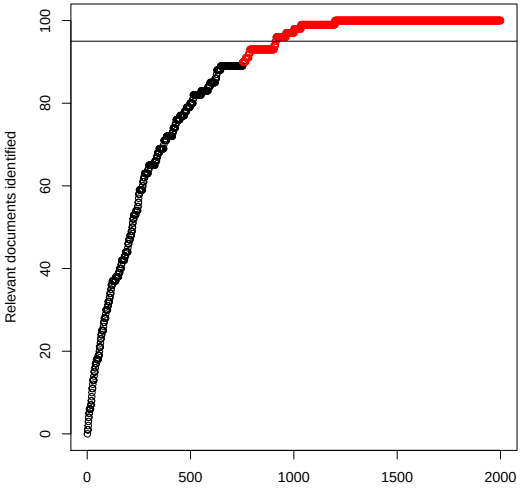
But if we descale the y axis, it's not so easy to tell when we have achieved 95% recall

```
r <- rle(ordered_docs) # calculate run lengths
cutoff <- cumsum(r$lengths)[
  r$lengths>40 & r$values==0
][1] # stop after 30 consecutive 0s
plot(
  cumsum(ordered_docs[1:cutoff]),
  xlim=c(1,N_tot),
  xlab="Documents Screened",
  ylab="Relevant documents identified"
)
```



Screening documents with machine learning

With a bit of maths, we can calculate when it is safe to stop, if we want to achieve 95% recall (or any other target) with a given level of confidence



In a nutshell, we use the distribution of relevant documents in those we previously screened, to infer likely distributions of relevant documents in documents not yet seen.

A statistical stopping criterion

Let's imagine we stop machine-learning prioritised screening at an arbitrary point and start drawing *at random* **without replacement** from the remaining documents.

In probability theory, we use the analogy of an urn with green marbles (relevant documents / successes) and red marbles (irrelevant documents / failures).

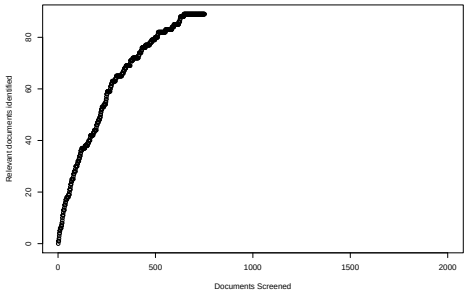
Using the hypergeometric distribution. we can calculate the probability of drawing

- k relevant documents
- in a sample of n documents
- from an urn that had N total documents
- of which K were relevant

A statistical stopping criterion

Let's return to a point that looked like a nice place to stop

```
plot(
  cumsum(ordered_docs[1:cutoff]),
  xlim=c(1,N_tot), xlab=xlab, ylab=ylab
)
```



N_{AL} is the number of documents screened after active learning, and ρ_{AL} is the number of relevant documents seen after active learning. So N (K) is the number of (relevant) documents remaining "in the urn".

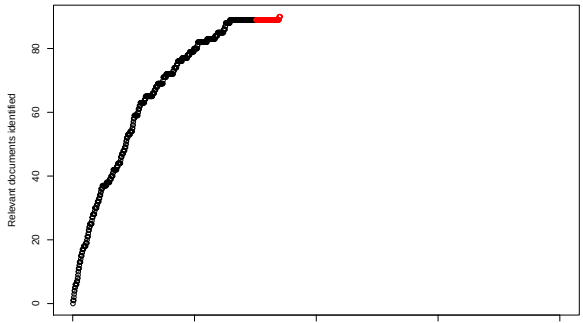
```
params <- list(
  N_al = cutoff,
  N = N_tot - N_al,
  r_al = sum(ordered_docs[1:cutoff]),
  K = r_tot-r_al
)
print(params)
```

```
## $N_al
## [1] 752
##
## $N
## [1] 1248
##
## $r_al
## [1] 89
##
## $K
## [1] 11
```

A statistical stopping criterion

We begin with a sample of 100 documents

```
n <- 100
urndocs <- sample(ordered_docs[cutoff:N_tot], replace=F)
s_docs <- urndocs[1:n]
k <- sum(s_docs)
plot(
  c(cumsum(ordered_docs[1:cutoff]),cumsum(s_docs)+r_al),
  xlim=c(1,N_tot), xlab=xlab, ylab=ylab,
  col=c(rep("black",cutoff),rep("red",n))
)
```



n is the number of documents in the sample (100) and k is the number of relevant documents drawn (1)

Using the hypergeometric distribution, we can calculate the probability of observing k relevant documents in a sample of n documents given an urn of N documents of which K are relevant.

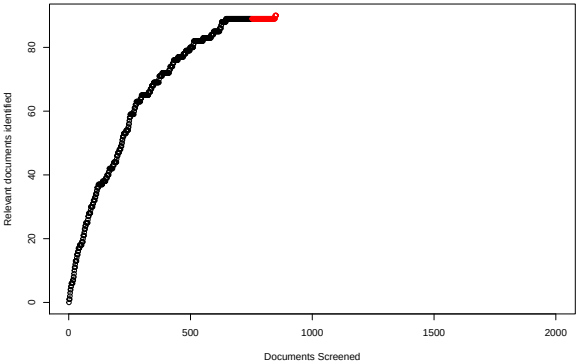
```
phyper(k, K, N, n)
```

```
## [1] 0.7846169
```

However, although we do know N, we don't know how many documents are in the urn (K).

A statistical stopping criterion - Hypothesis testing

```
plot(
  c(cumsum(ordered_docs[1:cutoff]),cumsum(s_docs)+r_al),
  xlim=c(1,N_tot), xlab=xlab, ylab=ylab,
  col=c(rep("black",cutoff),rep("red",n))
)
```



We form a null hypothesis that the target level of recall has not been achieved

$$H_0 : \tau < \tau_{tar} \tag{1}$$

Accordingly, our alternative hypothesis is that recall is at least as large as our target:

$$H_1 : \tau \geq \tau_{tar} \tag{2}$$

To operationalise this, we come up with a hypothetical value of K which is the lowest value compatible with our null hypothesis

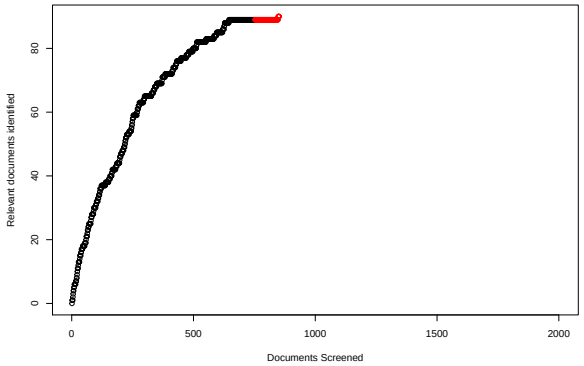
$$K_{tar} = \lfloor \frac{\rho_{seen}}{\tau_{tar}} - \rho_{AL} + 1 \rfloor \tag{3}$$

```
get_ktar <- function(r_al, r_seen, recall_target){
  return(floor(r_seen/recall_target-r_al+1))
}
r_seen <- r_al+k
K_tar <- get_ktar(r_al, r_seen, .95)
```

In other words, if there were 6 or more relevant documents in the urn when sampling began, the 89 relevant we identified before sampling, and the 1 we drew from the urn would not be enough to meet our target recall level

A statistical stopping criterion - Hypothesis testing

```
plot(
  c(cumsum(ordered_docs[1:cutoff]),cumsum(s_docs)+r_al),
  xlim=c(1,N_tot), xlab=xlab, ylab=ylab,
  col=c(rep("black",cutoff),rep("red",n))
)
```



The cumulative distribution function gives us the probability of observing what we observed, if our null hypothesis were true

$$p = P(X \leq k), \text{ where } X \sim \mathit{Hypergeometric}(N, K_{tar}, n) \tag{4}$$

When $p < 1 - \alpha$, we can stop screening, and report, for example, that we reject the null hypothesis that we achieve a recall below 95% at the 5% significance level

```
p <- phyper(k, K_tar, N-K_tar, n)
print(k)
```

```
## [1] 1
```

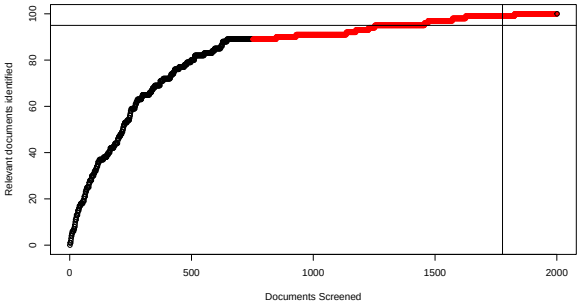
```
p
```

```
## [1] 0.922909
```

Based on what we observed in this sample, it is not unlikely at all that we have missed our target, so we keep screening

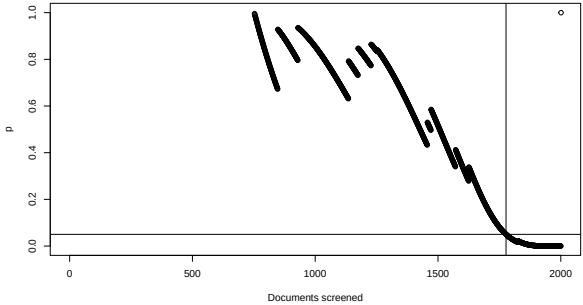
A statistical stopping criterion - Hypothesis testing

```
vec_K_tar <- vapply(
  cumsum(urndocs)+r_al, get_ktar, numeric(1),
  r_al=r_al, recall_target=.95
)
p <- phyper(cumsum(urndocs), vec_K_tar, N-vec_K_tar, seq(1,N))
plot(
  c(cumsum(ordered_docs[1:cutoff]),cumsum(urndocs)+r_al),
  xlim=c(1,N_tot), xlab=xlab, ylab=ylab,
  col=c(rep("black",cutoff),rep("red",N))
)
abline(h=r_tot*.95)
abline(v=N_al+which(p<0.05)[1])
```



The p score declines as we see more consecutive irrelevant documents, and as there are fewer and fewer documents left in the urn.

```
plot(c(rep(NA,N_al),p), xlab="Documents screened", ylab="p")
abline(h=.05)
abline(v=N_al+which(p<0.05)[1])
```



Ranked quasi sampling

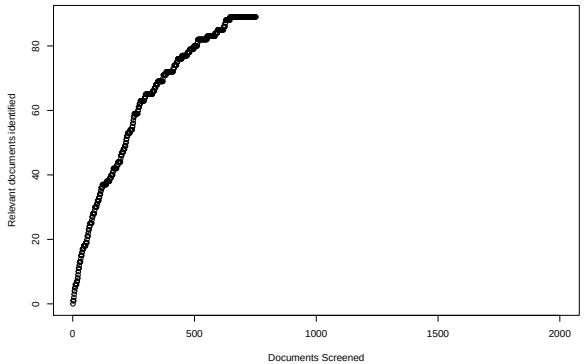
This is useful but not practical, because we don't know when to start a random sample, and doing so slows the whole process down.

-> We treat previously screened documents *as if* they were drawn from a random sample, which is conservative as long as the machine learning hasn't completely backfired.

```
last_docs <- c(0,0,0,1,0,0,1,0,0,1,0,1,0,1,0,0,1,0,0,0)
cols <- ifelse(last_docs>0,"green","red")
plot(
  rep(1,20), col = cols, pch = 19,
  ylim=c(0.5,4.5), axes=FALSE, xlab="",ylab=""
)
for (i in seq(1,6)) {
  lines(seq(20.5-i,20.5),rep(0.75+i*0.4,i+1))
  i_k <- sum(tail(last_docs,i))
  text(
    20.5-i*0.5,0.75+i*0.4,
    paste0("n=",i,", k=",i_k), adj=c(0.5,-0.25)
  )
}
```

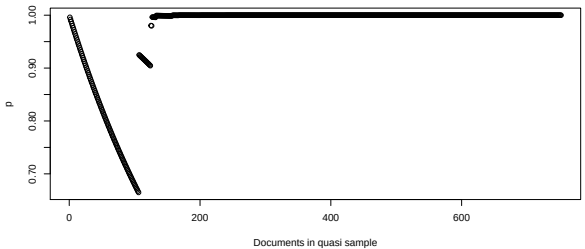
Ranked quasi sampling

```
plot(
  cumsum(ordered_docs[1:cutoff]),
  xlim=c(1,N_tot), xlab=xlab, ylab=ylab
)
```



We can then calculate a p value for quasi-samples consisting of the last 1 documents to all screened documents

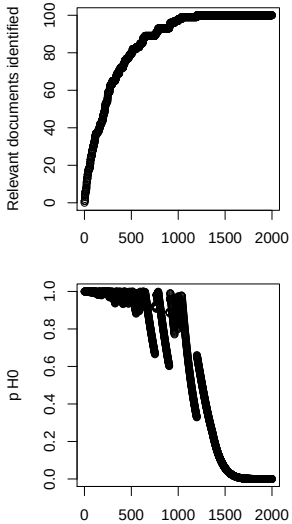
```
h0_p <- function(docs, N_tot, recall_target) {
  r_seen <- sum(docs)
  n_vec <- seq(1:length(docs))
  k_vec <- cumsum(rev(docs))
  r_al_vec <- r_seen - k_vec
  k_hat_vec <- vapply(r_al_vec, get_ktar, numeric(1),
                      recall_target=recall_target, r_seen=r_seen)
  red_ball_vec <- N_tot-(length(docs)-n_vec)-k_hat_vec
  p_vec <- phyper(k_vec, k_hat_vec, red_ball_vec, n_vec)
  n <- which.min(p_vec)
  return(list("min_p" = p_vec[n], "n" = n, "p_vec" = p_vec))
}
plot(h0_p(ordered_docs[1:cutoff],N_tot,.95)$p_vec,
     xlab="Documents in quasi sample",ylab="p")
```



Ranked quasi sampling

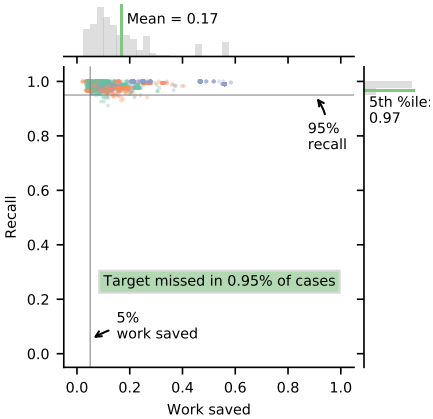
We can calculate this after each document is screened and figure out when it is safe to stop

```
df <- data.frame(
  i=seq(1,N_tot), r_seen=cumsum(ordered_docs),
  docs=ordered_docs, pmin=0
)
for (i in df$i) {
  p <- h0_p(df$docs[1:i], N_tot, .95)
  df$pmin[i] <- p$min_p
}
par(mfrow = c(2,1), mar = c(2, 2, 2, 2),pty="s")
plot(df$r_seen)
plot(df$pmin)
```



Evaluation

We tested this on 20 systematic review datasets, simulating 100 machine learning assisted reviews of each.



Our criterion performed reliably. Although work savings were small, this is partly due to small datasets with few relevant documents. Because of the way the hypergeometric distribution works, the extra effort required to prove that it is safe to stop gets proportionally smaller.

Conclusion

We provide a reliable way to estimate if it is safe to stop screening if you want to achieve a certain recall target. All you need are the R function in this presentation, a vector of 1s and 0s representing human screening decisions, and the number of documents in a database

Thanks!

Paper: Callaghan, M.W., Müller-Hansen, F.. Statistical Stopping Criteria for Automated Screening in Systematic Reviews. *Systematic Reviews*. **9** 273 (2020). https://doi.org/10.1186/s13643-020-01521-4.

Contact: callaghan@mcc-berlin.net, Twitter: @MaxCallaghan5

Code: https://github.com/mcallaghan/rapid-screening

This presentation:
https://github.com/mcallaghan/rapid-screening/blob/master/pres/rapid-review-esmar.Rmd

References

Cohen, A. M., W. R. Hersh, K. Peterson, and Po Yin Yen. 2006. "Reducing workload in systematic review preparation using automated citation classification." *Journal of the American Medical Informatics Association* 13 (2): 206–19. doi:10.1197/jamia.M1929.

Howard, Brian E, Jason Phillips, Arpit Tandon, Adyasha Maharana, Rebecca Elmore, Deepak Mav, Alex Sedykh, et al. 2020. "SWIFT-Active Screener : Accelerated document screening through active learning and integrated recall estimation." *Environment International* 138 (April 2019). Elsevier: 105623. doi:10.1016/j.envint.2020.105623.

Jonnalagadda, Siddhartha, and Diana Petitti. 2013. "A new iterative method to reduce workload in systematic review process." *International Journal of Computational Biology and Drug Design* 6 (1/2): 5. doi:10.1504/ijcbdd.2013.052198.

O'Mara-Eves, Alison, James Thomas, John McNaught, Makoto Miwa, and Sophia Ananiadou. 2015. "Using text mining for study identification in systematic reviews: A systematic review of current approaches." *Systematic Reviews* 4 (1): 1–22. doi:10.1186/2046-4053-4-5.

Przybyła, Piotr, Austin J. Brockmeier, Georgios Kontonatsios, Marie Annick Le Pogam, John McNaught, Erik von Elm, Kay Nolan, and Sophia Ananiadou. 2018. "Prioritising references for systematic reviews with RobotAnalyst: A user study." *Research Synthesis Methods* 9 (3): 470–88. doi:10.1002/jrsm.1311.

Shemilt, Ian, Antonia Simon, Gareth J. Hollands, Theresa M. Marteau, David Ogilvie, Alison O'Mara-Eves, Michael P. Kelly, and James Thomas. 2014. "Pinpointing needles in giant haystacks: Use of text mining to reduce impractical screening workload in extremely large scoping reviews." *Research Synthesis Methods* 5 (1): 31–49. doi:10.1002/jrsm.1093.