Exploring Models of Memory

Greg Cox

2021-09-08

Contents

This collection was written using the **bookdown** package (Xie, 2021), which was built on top of R Markdown and **knitr** (Xie, 2015).

Chapter 1

Strength Theory and Signal Detection

As applied to memory, signal detection theory aims to explain memory judgments as the result of comparing the result of a retrieval process against a set of criteria. The result of the retrieval process is a value on a unidimensional scale. Depending on where the value falls on that scale, a different response will be produced.

In this expedition, we will build, piece by piece, a model that instantiates recognition by signal detection. This will emphasize two broad perspectives on this model and modeling in general:

1. *Separability*: It is often possible to "swap out" one piece of a model for another, where the two pieces appear to perform the same basic function.
2. *Integrality*: The behavior of the model arises from all of its components and how they are put together; even if one piece seems functionally equivalent to another, they may lead to very different behavior from the model.

1.1 Follow the blazes to your own strength model

There are three basic ingredients to a strength model of recognition, instantiated in the following chunk of R code:

```r
# 1. Define one or more decision criteria
criterion <- 0.5

# 2. Define the strength of the target (studied) and foil (unstudied) items in memory
foil_strength <- 0
```

```
target_strength <- 1

# 3. Compare the strengths of each test item to the criteria to arrive at a decision
foil_response <- foil_strength > criterion
target_response <- target_strength > criterion
```

If you ran the preceding chunk, you can check what the model decided by running the following at the console

```
foil_response
```

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

```
target_response
```

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

Great! The model gave a “yes” response (coded as `TRUE`) to the target and a “no” response (coded as `FALSE`) to the foil.

1.1.1 Multiple targets and foils

If we wanted to explore how the model would behave with foils or targets of different strengths or with different criteria, we could change the above chunk. For example, let’s imagine that instead of a single target and a single foil, we present the model with multiple trials consisting of different targets and different foils. Each of these would have different strengths, potentially leading to different behavior.

```
# 1. Define one or more decision criteria
criterion <- 0.5

# 2. Define the strength of the target (studied) and foil (unstudied) items in memory
foil_strength <- c(0, -0.8, 0.4, 0.3, 1.3)
target_strength <- c(1, 0.9, 1.1, 0.2, -1.0)

# 3. Compare the strengths of each test item to the criteria to arrive at a decision
foil_response <- foil_strength > criterion
target_response <- target_strength > criterion
```

Now we can see the response the model made to *each* foil and target:

```
foil_response
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE
```

```
target_response
```

```
## [1]  TRUE  TRUE  TRUE FALSE FALSE
```

1.1.2 Multiple criteria

So far, we've assumed a single criterion leading to either a "yes" or "no" response. If we want to model situations with multiple responses like confidence ratings or different experimental conditions with different criteria (e.g., due to payoff structure), we can introduce multiple criteria.

This requires us to make two changes to the chunk of code that constitutes our model. First, we need to define the multiple criteria, like we did we multiple target/foil strengths. Note that we will now need to explicitly say that negative and positive infinity are the smallest and largest "response" criteria. Second, we need to swap out the way the model makes responses. Specifically, we will swap out the response lines with the `cut` function, which "cuts" the real line into pieces and tells us which piece each value falls into.

```r
# 1. Define one or more decision criteria
criteria <- c(-Inf, -0.5, 0, 0.5, 1, 1.5, Inf)

# 2. Define the strength of the target (studied) and foil (unstudied) items in memory
foil_strength <- c(0, -0.8, 0.4, 0.3, 1.3)
target_strength <- c(1, 0.9, 1.1, 0.2, -1.0)

# 3. Compare the strengths of each test item to the criteria to arrive at a decision
foil_response <- cut(foil_strength, breaks=criteria, labels=FALSE)
target_response <- cut(target_strength, breaks=criteria, labels=FALSE)
```

```r
foil_response
```

```
## [1] 2 1 3 3 5
```

```r
target_response
```

```
## [1] 4 4 5 3 1
```

1.1.3 Making an ROC

Great, now our model can make different responses, like we need to construct an ROC curve! Recall that to create such a curve, each point needs to represent the *cumulative* probability of making a response at *or above* a particular confidence level.

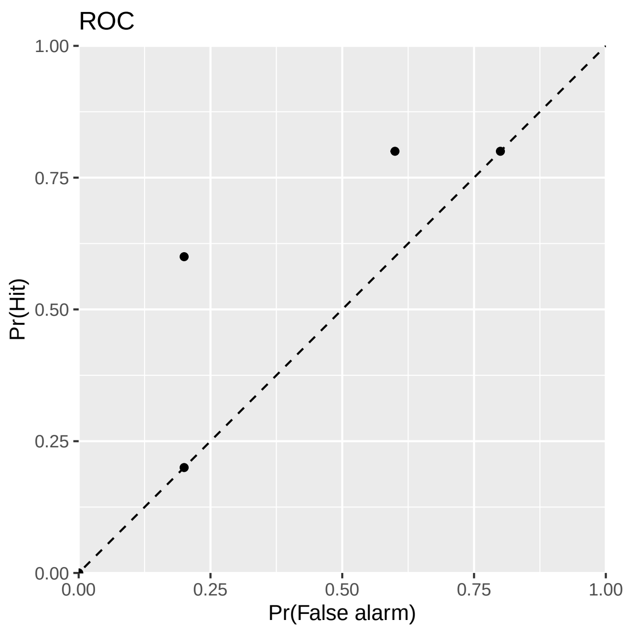
We can therefore make use of R's `ecdf` function—the Empirical Cumulative Distribution Function. The line `ecdf(*_response)` will calculate the ECDF using the responses made; we then *evaluate* this ECDF for all possible confidence levels, which in this case is 1 through 6, represented in R with `1:6`. The ECDF represent the probability of making a response at *or below* a particular confidence level, so we need to subtract the ECDF from one.

To plot the ROC, it will be handy to use the `ggplot2` library which makes prettier plots than the standard R functions, so let's load that library now.

```r
library(ggplot2)
```

Putting it all together, we have:

```r
# 1. Define one or more decision criteria
criteria <- c(-Inf, -0.5, 0, 0.5, 1, 1.5, Inf)

# 2. Define the strength of the target (studied) and foil (unstudied) items in memory
foil_strength <- c(0, -0.8, 0.4, 0.3, 1.3)
target_strength <- c(1, 0.9, 1.1, 0.2, -1.0)

# 3. Compare the strengths of each test item to the criteria to arrive at a decision
foil_response <- cut(foil_strength, breaks=criteria, labels=FALSE)
target_response <- cut(target_strength, breaks=criteria, labels=FALSE)

# 4. Calculate points on ROC curve
roc_foil <- 1 - ecdf(foil_response)(1:6)
roc_target <- 1 - ecdf(target_response)(1:6)

# 5. Plot ROC
ggplot(mapping=aes(x = roc_foil, y = roc_target)) +
    geom_point() +
    geom_abline(intercept = 0, slope = 1, color='black', linetype = 'dashed') +
    labs(x = 'Pr(False alarm)', y = 'Pr(Hit)', title = 'ROC') +
    coord_fixed(xlim=c(0, 1), ylim=c(0, 1), expand=FALSE)
```



1.1.4 Modeling strengths

We now have a model that instantiates the processes by which strength theory says recognition decisions are made. Pretty neat!

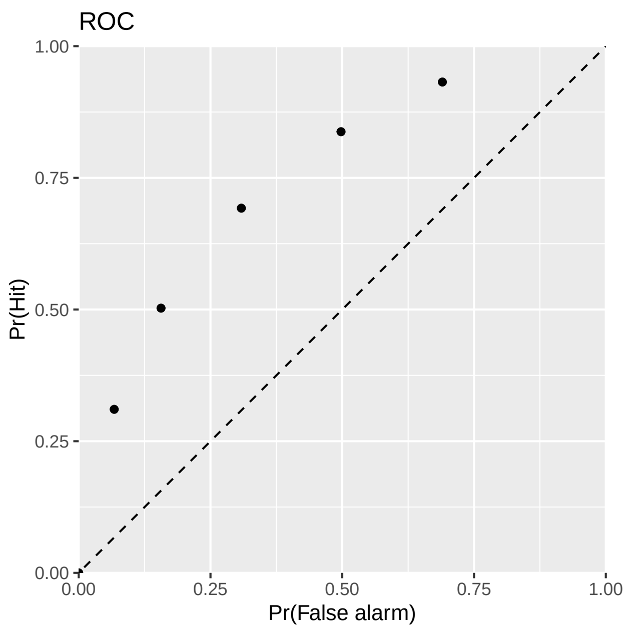
We have *not*, however, modeled the distributions of target and foil strength. Instead, we have just plugged in some numbers. Most SDT theories model these strengths as random variables that are sampled from a distribution.

Another benefit to modeling the strengths is that we can model a situation with a huge number of target/foil trials, far more than would be feasible in any experiment with living creatures. As a result, we can essentially obtain perfect estimates of things like ROC curves by simulating many trials.

1.1.4.1 Variability in initial strength

Let's begin with a simple model in which we assume that all items initially have a strength that is sampled from a normal distribution with mean 0 and standard deviation 1. When an item is studied, its strength is incremented by *exactly* 1.

```r
# 1. Define one or more decision criteria
criteria <- c(-Inf, -0.5, 0, 0.5, 1, 1.5, Inf)

# 2. Define the strength of the target (studied) and foil (unstudied) items in memory
foil_strength <- rnorm(n = 10000, mean = 0, sd = 1)
target_initial_strength <- rnorm(n = 10000, mean = 0, sd = 1)
study_strength_increment <- 1
target_strength <- target_initial_strength + study_strength_increment

# 3. Compare the strengths of each test item to the criteria to arrive at a decision
foil_response <- cut(foil_strength, breaks=criteria, labels=FALSE)
target_response <- cut(target_strength, breaks=criteria, labels=FALSE)

# 4. Calculate points on ROC curve
roc_foil <- 1 - ecdf(foil_response)(1:6)
roc_target <- 1 - ecdf(target_response)(1:6)

# 5. Plot ROC
ggplot(mapping=aes(x = roc_foil, y = roc_target)) +
    geom_point() +
    geom_abline(intercept = 0, slope = 1, color='black', linetype = 'dashed') +
    labs(x = 'Pr(False alarm)', y = 'Pr(Hit)', title = 'ROC') +
    coord_fixed(xlim=c(0, 1), ylim=c(0, 1), expand=FALSE)
```

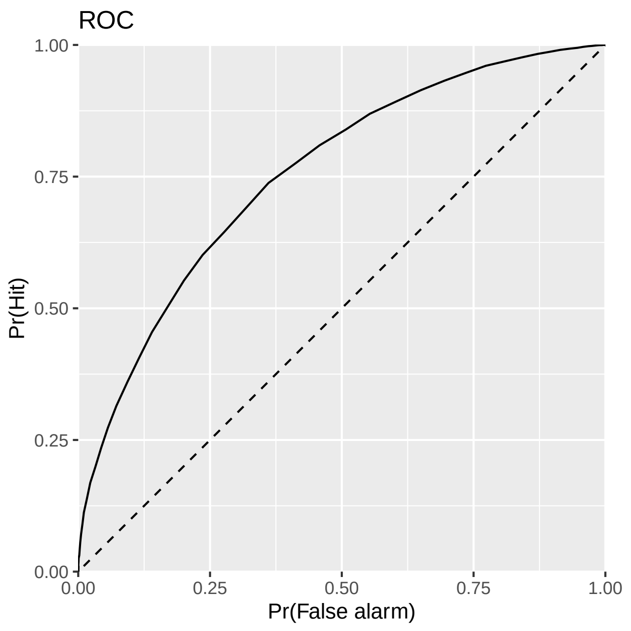


1.1.4.2 Fine-grained ROC's

Cool! Again, since we're not using a living creature, we can also ask the model to use an absurdly large number of criteria. The value of this is that it gives us a much more fine-grained picture of the ROC. Even though we couldn't get such a picture in real life, if the model is a reasonably good approximation to reality, knowing the detailed model ROC let's us know the kind of shapes we would *expect* to see in an experiment.

```
# 1. Define one or more decision criteria
num_criteria <- 100
criteria <- c(-Inf, seq(-6, 6, length.out = num_criteria - 1), Inf)

# 2. Define the strength of the target (studied) and foil (unstudied) items in memory
foil_strength <- rnorm(n = 10000, mean = 0, sd = 1)
target_initial_strength <- rnorm(n = 10000, mean = 0, sd = 1)
study_strength_increment <- 1
target_strength <- target_initial_strength + study_strength_increment

# 3. Compare the strengths of each test item to the criteria to arrive at a decision
foil_response <- cut(foil_strength, breaks=criteria, labels=FALSE)
target_response <- cut(target_strength, breaks=criteria, labels=FALSE)

# 4. Calculate points on ROC curve
roc_foil <- 1 - ecdf(foil_response)(1:num_criteria)
```

```
roc_target <- 1 - ecdf(target_response)(1:num_criteria)

# 5. Plot ROC
ggplot(mapping=aes(x = roc_foil, y = roc_target)) +
    geom_line() +
    geom_abline(intercept = 0, slope = 1, color='black', linetype = 'dashed') +
    labs(x = 'Pr(False alarm)', y = 'Pr(Hit)', title = 'ROC') +
    coord_fixed(xlim=c(0, 1), ylim=c(0, 1), expand=FALSE)
```

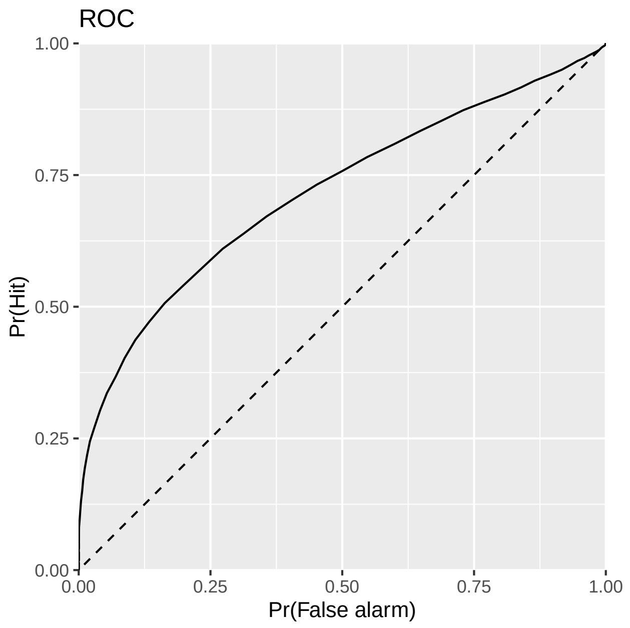


1.1.4.3 Variability in study strength increments

Finally, let's introduce variability in the increment in strength that results from studying an item. Again, maybe we can imagine this strength increment comes from a normal distribution (this is the assumption of the unequal-variance SDT model). In that case, we can define its mean and standard deviation.

```
# 1. Define one or more decision criteria
num_criteria <- 100
criteria <- c(-Inf, seq(-6, 6, length.out = num_criteria - 1), Inf)

# 2. Define the strength of the target (studied) and foil (unstudied) items in memory
foil_strength <- rnorm(n = 10000, mean = 0, sd = 1)
target_initial_strength <- rnorm(n = 10000, mean = 0, sd = 1)
study_strength_increment <- rnorm(n = 10000, mean = 1, sd = 1)
target_strength <- target_initial_strength + study_strength_increment
```

```r
# 3. Compare the strengths of each test item to the criteria to arrive at a decision
foil_response <- cut(foil_strength, breaks=criteria, labels=FALSE)
target_response <- cut(target_strength, breaks=criteria, labels=FALSE)

# 4. Calculate points on ROC curve
roc_foil <- 1 - ecdf(foil_response)(1:num_criteria)
roc_target <- 1 - ecdf(target_response)(1:num_criteria)

# 5. Plot ROC
ggplot(mapping=aes(x = roc_foil, y = roc_target)) +
    geom_line() +
    geom_abline(intercept = 0, slope = 1, color='black', linetype = 'dashed') +
    labs(x = 'Pr(False alarm)', y = 'Pr(Hit)', title = 'ROC') +
    coord_fixed(xlim=c(0, 1), ylim=c(0, 1), expand=FALSE)
```



1.2 Going off-trail

Following the trail this far has gotten us to a reasonably complete model of recognition by signal detection, according to strength theory. Now it's your turn to go off-trail and explore on your own!

1. If study strength increments are normally distributed, this means it is possible for them to be *negative*, that is, for studying an item to actually decrease its strength from its initial level. Could this be plausible under

some circumstances? If it is plausible, what does this say about how we should interpret what "strength" means in this context?

2. Try swapping out the normally-distributed strength increment for one that is *exponentially* distributed. In R, you can draw random samples from an exponential distribution with the `rexp` function (check out the help pages to see what parameters the function takes). How close can you get the resulting ROC to looking linear across most of its range? What does this say about the kinds of conclusions we can draw based on the shape of ROC's? Are there reasons why the exponential distribution might be a better or worse model for study strength increments?
3. Pick an alternative recognition model (e.g., a threshold model or a hybrid model or one of your own ideas) and implement it by modifying the code we built on trail. What choices did you make about how to implement the model you chose and why? Did you stick with the same basic structure as the model we built above, or did you modify it at all? Were you able to find any unique or striking—but still plausible—predictions from your chosen model?

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2021). *bookdown: Authoring Books and Technical Documents with R Markdown.* R package version 0.22.