CatContModel_PlatSpline

Kyle O Hardman

10/25/2019

The psychological process model implemented by the CatContModel package has as one of its elements a theoretical method by which people categorize observed continuous stimuli. For example, an individual might encode any color in a certain range of greenish colors as belonging to the "green" category or encode an orientation somewhat above or below canonical left as belonging to the left category. This categorization may be useful if there is a limit to the number of memory items that can be encoded with continuous information, but items that are encoded categorically may be easier to remember, or at least use different mental resources than continuous items.

This document focuses on the method by which participants convert a continuous stimulus representation into a categorical representation. I don't intend to make any theoretical claims here, but rather to explain assumptions about how the model does the categorization and choices that users of the model can make about categorization.

Encoding Process

The structure of the model shown in Figure 2 of Hardman, Vergauwe, and Ricker does not map nicely onto thoeries of how people might go about encoding items into WM. Instead, the model is a simplification that works well mathematically but which is not intended to make strong theoretical statements about encoding and storage in WM. For the kinds of simplistic psychological process models for which I know how to estimate parameters, it is difficult or impossible to embed assumptions about the time course of memory.

An important question is whether items are more likely to be chosen for encoding if those items are determined to be easily categorizable. If it is easier for people to remember categories than continuous values (which seems to be the case), it could be that stimulus values that are similar to categories might be more likely to be chosen for encoding.

The model tree in Figure 2 of Hardman et al. (2017), if read literally, makes the claims that: 1. People either do or do not remember the item (based on P^M). 2. Then if the item is remembered, it is remembered either categorically or continuously. This is of course totally bizarre: Whether or not people remember the item depends on the choice of how the item was encoded (cont or cat), not the other way around.

Statistical Properties of Categorization

Regardless of how categorization actually happens in the mind/brain, it seems likely to have some statistical properties that a model of the psychological process would be advised to account for.

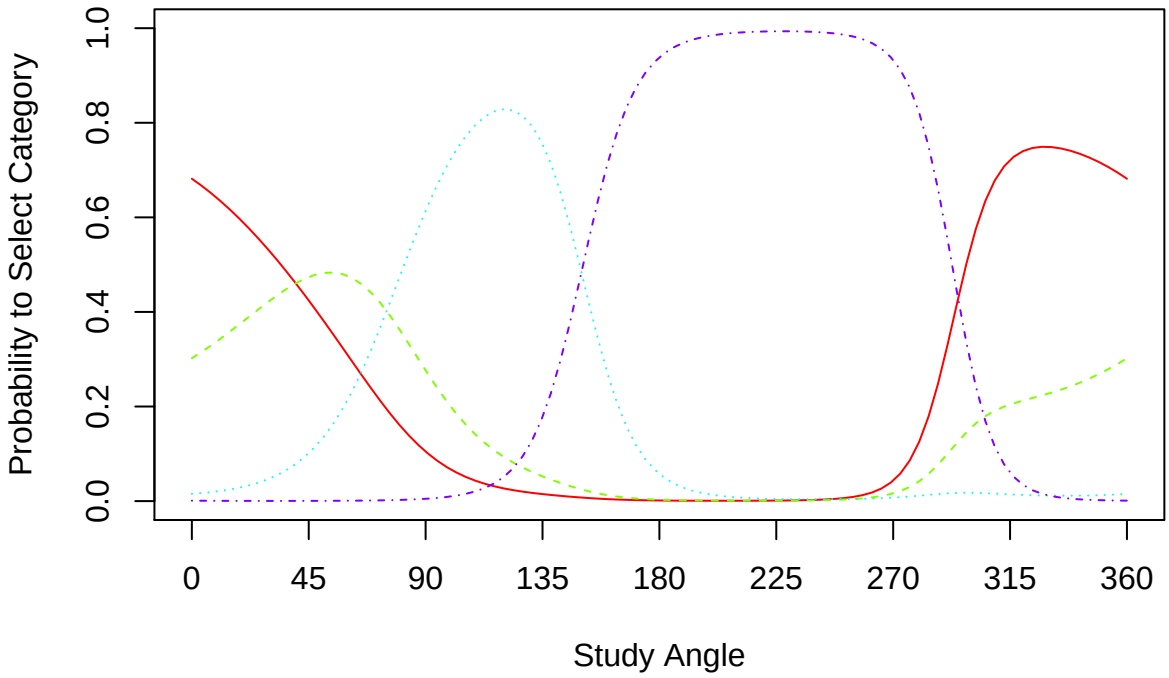
1. Categorization is likely somewhat probabilistic. A yellow-green color might be categorized as either yellow or green on a probabilistic basis. As such, the model should allow for stimuli between categories to be categorized into either surrounding category.
2. People are probably unlikely to categorize something that is red as green, or if they did so it would be indistinguishable from memory failure (or color blindness, which would produce a special pattern). So the categorization method should probably assign very low or zero probability that a color in the green range would be categorized as any color other than green or an adjacent color.

Mixture Model Structure

The mixture model shown in Figure 2 of Hardman, Vergauwe, and Ricker (2017) starts

The Hardman, Vergauwe, and Ricker (2017) Weights Distribution

```
CatContModel:::plotWeightsFunction(c(30, 50, 100, 200), 30)
```



Limitations of the Weights Distribution

One limitation of the weights distribution used by Hardman et al. (2017) is that the sum of the weights for any given studied value is always 1. In other words, there are no studied values for which categorization happens any less or more than for any other studied value. This was by design: The nature of the mixure model meant that if a studied item is to be categorized (in a conceptual sense that the single-item `pContBetween` is 0) it must be put into one of the available categories. No to-be-categorized items can end up without a category.

It is possible to imagine, however, that some studied values are less likely to be placed into a category than other values. For example, for orientation stimuli, it could be that values near the four primary directions (up, down, left, right) are more likely to be categorized than values on the off-diagonals. There are various ways to account for this possibility, but many of them are suprisingly complex to implement and have strange problems.

I'm going to develop one possibility, which I'm calling the PlatSpline distribution, and try to work it into the mixure model structure of these models.

The PlatSpline Distribution

The Spline

The transition between the top of the plateau and the "ground" (zero) needs a special curve for the PlatSpline distribution to be continuous. In particular, what is needed is a curve with a first derivative of 0 at either end.

I found a method to satisfy that requirements with a cubic spline, which is described by Carstensen (https://www.tf.uni-kiel.de/matwis/amat/comp_math/kap_1/backbone/r_se16.html). The method can be generalized to higher order splines, but a cubic is good enough.

The spline section of the PlatSpline distribution is a cublic spline that fits the points (0, 1), (1, 0.5), and (2, 0) and has a first derivative of 0 at either end. This basic spline is easily scaled in the x or y dimension.

I came up with cubic spline coefficients as follows.

```
# Points to fit
x = c(0, 1,   2)
y = c(1, 0.5, 0)

n = length(x) - 1


# Helper functions:

# 0th deriv
d0 = function(x) {
    c(1, x, x^2, x^3)
}

#1st deriv
d1 = function(x) {
    c(0, 1, 2*x, 3*x^2)
}

#2nd deriv
d2 = function(x) {
    c(0, 0, 2, 6*x)
}

zero = function(n) {
    rep(0, n)
}

# The equation to be solved
# lhs = m %*% coefs

# Construct lhs
lhs = c(0, y[1], y[2], 0, 0, y[2], y[3], 0)

# Construct and fill m
m = matrix(nrow=4*n, ncol=4*n)

m[1,] = c( d1(x[1]),  zero(4)  ) # first derivative at first point = 0
```

```
m[2,] = c( d0(x[1]),  zero(4)  ) # y[1] = d0(x[1])
m[3,] = c( d0(x[2]),  zero(4)  ) # y[2] = d0(x[2])
m[4,] = c( d1(x[2]), -d1(x[2]) ) # first deriv for coefs 1 = first deriv coefs 2
m[5,] = c( d2(x[2]), -d2(x[2]) ) # second deriv for coefs 1 = second deriv for coefs 2
m[6,] = c( zero(4) ,  d0(x[2]) ) # y[2] = d0(x[2])
m[7,] = c( zero(4) ,  d0(x[3]) ) # y[3] = d0(x[3])
m[8,] = c( zero(4) ,  d1(x[3]) ) # first derivative at last point = 0

# Solve the equation
coefs = solve(m) %*% lhs
zapsmall(coefs)
```

```
##       [,1]
## [1,]  1.00
## [2,]  0.00
## [3,] -0.75
## [4,]  0.25
## [5,]  1.00
## [6,]  0.00
## [7,] -0.75
## [8,]  0.25
```

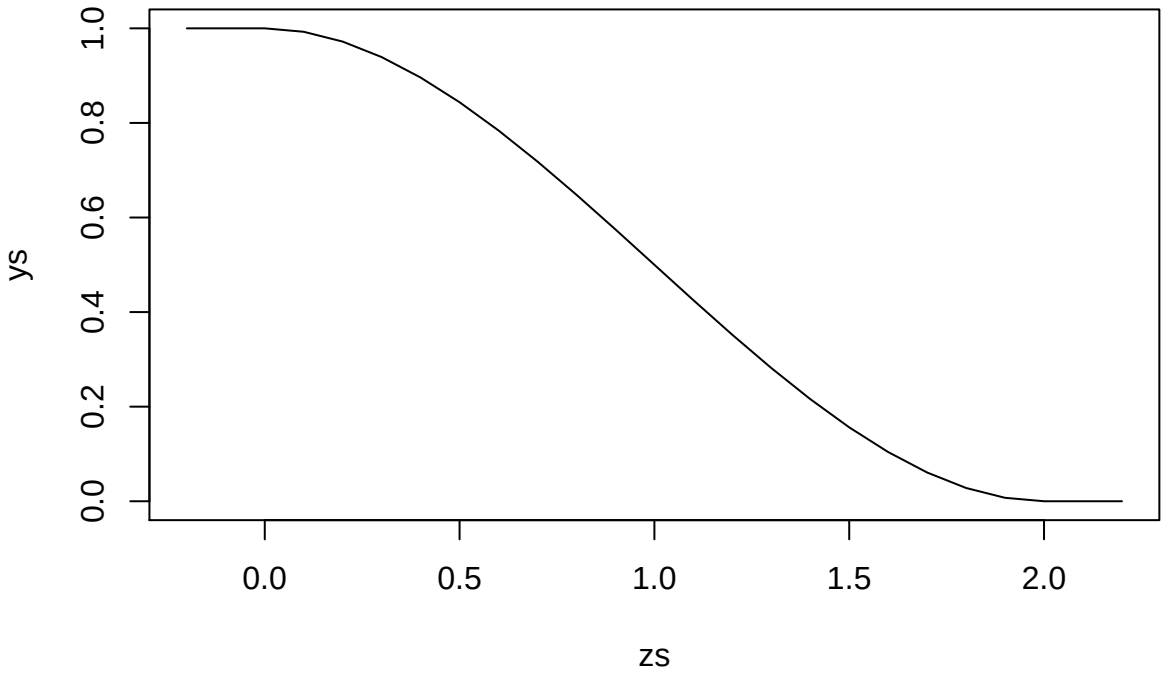
The coefficients repeat twice, so the coefficients are 1, 0, -0.75, and 0.25. The zero derivative cubic spline function is thus

$$dZDCS(z) = 1 - 0.75z^2 + 0.25z^3$$

where z is analogous to a z-score for a normal in that the inflection point of the spline happens at $z = 1$. In the package, this function is `dZeroDerivSpline`.

```
zs = seq(-0.2, 2.2, 0.1)
ys = CatContModel::dZeroDerivSpline(zs)

plot(zs, ys, type='l')
```



Note that as indicated by the plot, for reasons of convenience `dZeroDerivSpline` gives a value of 1 for $z < 0$ and a value of 0 for $z > 2$.

$$dSpline(z, \sigma) = \begin{cases} 1 & \text{if } z \leq 0 \\ 0 & \text{if } z \geq 2 \\ 1 - 0.75z^2 + 0.25 * z^3 & 0 < z < 2 \end{cases}$$

$$dPS(x|\mu, \rho, \sigma) = \begin{cases} 1 & \text{if } |x - \mu| < \rho \\ 1 & \text{if } n = 2 \end{cases}$$

```
#xs =

#CatContModel:::dPlatSplineFull_R()
```

Total Weight Less Than 1

The PlatSpline distribution can have areas with total of 0 weight, which means that studied values in those areas are not assigned to any category.

For the between-item model variant, the options are that an item is 1) encoded with continuous information, 2) encoded with categorical information, or 3) not encoded.

In 0 weight areas, values cannot be assigned to a category based on the weights distribution. What to do with them?

One option is to assign them to a random category. This follows the logic that an item which is to be remembered categorically must be assigned to a category. Random category assignment in 0 weight areas is straightforward.

I think there may be a better solution, however, which involves feedback from the weights distribution to `pContBetween`. In particular, in 0 weight areas, `pContBetween` could be set to 1, with the meaning that values which cannot be categorized must be stored continuously. (By the mixture model logic, items that are not encoded at all correspond to `pMem = 0` for that item. Feedback from the weights could also be used to affect `pMem`.)

Failures to Categorize

When using the PlatSpline distribution, some study values are less likely to be put into any of the categories than other study values. That is, the sum of the category weights can be less than 1 for some study values. When this happens, it is a failure to categorize.

In the model, if the participant reaches the categorical memory node but fails to categorize, what is stored in memory?