Decision making as a random walk

Diego Hernández Jiménez

25/2/2022

Introduction

A binary choice problem is presented to a decision maker, e.g., choose between candidate A and candidate B. We assume that the decision making process involves sequentially examining data samples (Wang & Busemeyer, 2021). We don't know what kind of data is actually using the decision maker, but they serve as evidence favoring one or other alternative. In our voting example, we are assuming that the person is gathering information about the candidates, either explicitly and consciously (assessing the arguments given by each one) or implicitly and in a non-conscious way (being affected by peripheral cues such as attractiveness (see https://dictionary.apa.org/peripheral-cue)). It's important to note that, obviously, we can't observe the cognitive processes happening, so we cannot know what data the agent is assessing nor the value of the increment (incremented preference for A/B). However we can consider it as a process that varies randomly and model it probabilistically (later this will make more sense).

"The critical idea is that the decision-maker accumulates these evidence increments across time until the cumulative evidence becomes sufficiently strong (...)" (Wang & Busemeyer, 2021 p.122-23). Here, sufficiently strong means that the value representing the cumulative evidence reaches a threshold or decision criteria, so that a decision is finally made. After some time t our voter will have enough information favoring A (or B) to choose A (or B).

The decision making process as a random walk

How is this formalized? Over the years, many different models have been proposed; they are called sequential sampling models. I have no intention of reproducing any of these, my goal is to use one of the simplest versions, I am modeling the decision process as a *random walk* process in discrete time and discrete state space.

Why is that sensible? Well, "A random walk is defined as the running cumulative sum of a sequence of independent random variables, Z_j $j = 1, 2, \ldots$ In models of decision-making, the values of these variables are interpreted as the evidence in a sequence of discrete observations of the stimulus" (Smith & Ratcliff, 2015)

Note where it says "discrete observations". That means that time evolves in discrete timesteps, that's the reason why we are in discrete time. This discreteness also affects the state space, as said before, because the increments (the random variables of the random walk) take only discrete values, like, for example $+1$ or -1. It's possible to model the process as a stochastic process in continuous time and continuous space (*diffusion process*) but in this project I will stick with the discrete random walk.

And what makes this model "special"? Well, to be a bit more original, I added some extra parameters to the random walk that represent some psychological phenomena.

There is a deterministic part (it could also be random) given by V_0, which we can call initial valence or "favorability". Here we are assuming that a initial valence of 0 denotes neutrality, meaning that when $V_0 \neq 0$ there is an initial preference or bias towards one of the alternatives. The other 2 fixed values are the decision boundaries for each alternative. This values are arbitrary integer values. However the bigger their absolute value, the longer it will take to reach one of them. This can be used to model "reticence". Having a bigger criteria for A means that we need to accumulate more information to choose A than to choose B. Also, because we have two opposite alternatives, one decision criteria (*absorbing state* in the stochastic modeling literature) will be positive (for instance,the boundary for A), and the other will be negative.

The stochastic part of the model is given by Z, a random variable that represents the increment/decrement in the valence for each alternative at any time t. In the decision making process it's possible to encounter evidence favoring A, in that case the total valence goes up by 1. If the evidence favors B, the valence goes down by 1. But the probability of sampling evidence for A or B is not necessarily symmetric. There can be a sampling bias b_s that makes certain type of evidence be more often examined. This could represent a memory bias or availability bias (if you have been in a meeting for candidate A just before going to vote, it's possible that favorable arguments for choosing A will be more available) Here, b_s lies between -1 and 1 because it's the percentage of probability incremented. For instance, with $b_s = 0.2$, the probability of finding evidence for A is rises from 0.5 to $0.5 + 0.5 \times 0.2 = 0.6$. If b_s is negative, the bias works the same way but favoring B.

After having sampled an specific piece of information we can also imagine that another process takes place. Maybe the quality of evidence is low, and in that case it would not update the total value of valence. In other words, there is a possibility that at each timestep we end up with a increment of 0. The probability of that happening depends again on a new parameter, the acceptance bias b_a. To be honest, I haven't came up with a totally reasonable psychological interpretation of this value, but I see it as some sort of conservative bias. The bigger the value of it, the more likely the decision maker is to accept the new evidence. Again, as with b_s, it's the percentage of probability incremented. Negative values can be associated with a more conservative tendency because the probability of rejecting the new evidence presented (maintaining status quo?) is higher.

Because there are two joint events, sampling evidence and accepting/rejecting evidence (sequential, actually, but temporal order doesn't matter in classical probability theory), the probabilities for each possible outcome (increment valence by one, stay the same, decrement valence by one) are:

$p_{+1} = \mathbb{P}(\text{sampling } +1)\mathbb{P}(\text{accepting}) = \mathbb{P}(Z = +1) = 0.5(1 + b_s) \times 0.5(1 + b_a) = 0.25(1 + b_s)(1 + b_a)$

$p_0 = \mathbb{P}(\text{sampling } +1 \ \cup \ \text{sampling -1 })\mathbb{P}(\text{rejecting}) = \mathbb{P}(\text{rejecting}) = \mathbb{P}(Z = 0) = 1 - 0.5(1 + b_a) = 0.5(1 - b_a)$

$p_{-1} = \mathbb{P}(\text{sampling -1})\mathbb{P}(\text{accepting}) = \mathbb{P}(Z = -1) = \big(1 - 0.5(1 + b_s)\big) \times 0.5(1 + b_a) = 0.5(1 - b_s) \times 0.5(1 + b_a) = 0.25(1 - b_s)(1 + b_a)$

In summary

$$Z_t = \begin{cases} +1 & \text{with probability } \ p_{+1} = 0.25(1 + b_s)(1 + b_a) \\ \ \ 0 & \text{with probability } \ p_0 = 0.5(1 - b_a) \\ -1 & \text{with probability } \ p_{-1} = 0.25(1 - b_s)(1 + b_a) \end{cases}$$

After t timesteps, the valence will be $V_t = V_0 + \sum_{i=1}^{t} Z_i$. (That's the formula for the general random walk, see Jones $ Smith, (2018))

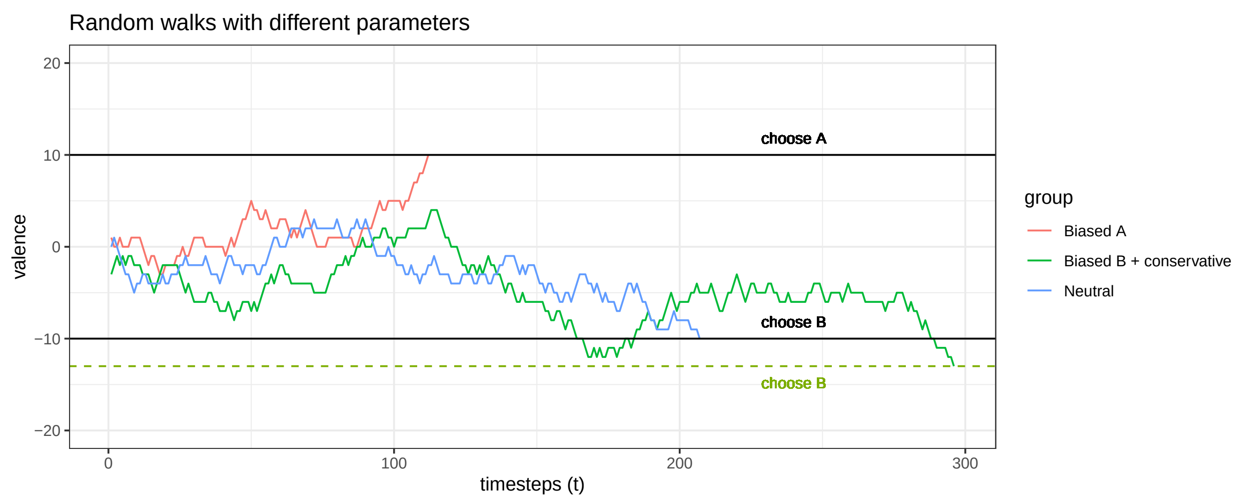
The algorithm in R

```r
randwalk <- function(V0,A_criteria,B_criteria,bs,ba){
   V <- V0
   t <- 0
   values <- integer()
   while ((V > B_criteria) && (V < A_criteria)){
      t <- t + 1

      # Sampling phase
      evidence_prob <- c(.5 +.5*bs,
                         1 -(.5 +.5*bs))
      Z <- sample(c(1,-1),size=1,prob=evidence_prob)

      # Acceptance phase
      acceptreject_prob <- c(.5 +.5*ba,
                             1 -(.5 +.5*ba))
      V <- sample(c(V+Z,V),size=1,prob=acceptreject_prob)

      # Updatinf phase
      values[t] <- V
```

```
  }
  return(values)
}
```

Here are some examples simulating different type of subjects

```
library(ggplot2)
set.seed(1312)

# No preferences

neutral <- randwalk(V0=0,
                    A_criteria=10,B_criteria=-10,
                    bs=0,
                    ba=0)
neutral_df <- data.frame(list('valence'=neutral,
                              'timesteps'=1:length(neutral),
                              'group'=rep('Neutral',length(neutral))))

# It samples more info favoring A

biasedA <- randwalk(V0=0,
                    A_criteria=10,B_criteria=-10,
                    bs=.1,
                    ba=0)

biasedA_df <- data.frame(list('valence'=biasedA,
                              'timesteps'=1:length(biasedA),
                              'group'=rep('Biased A',length(biasedA))))


# initial preference for B
# It samples more info favoring B
# It doesn't easily change its mind
# However, higher decision threshold for B

biasedB_conserv <- randwalk(V0=-3,
                            A_criteria=10,B_criteria=-13,
                            bs=-.1,
                            ba=-.01)

biasedB_conserv_df <- data.frame(list('valence'=biasedB_conserv,
                                      'timesteps'=1:length(biasedB_conserv),
                                      'group'=rep('Biased B + conservative',
                                                  length(biasedB_conserv))))

paths_df <- rbind(neutral_df,biasedA_df,biasedB_conserv_df)

n <- 20
ggplot(paths_df,aes(x=timesteps,y=valence,colour=group)) +
  geom_line() +
  ylim(c(-n,n)) +
   geom_hline(yintercept = 10) +
   geom_text(aes(240,10,label = 'choose A', vjust = -1)
             ,color='black',size=3,show.legend=F ) +

   geom_hline(yintercept = -10) +
   geom_text(aes(240,-10,label = 'choose B', vjust = -1)
             ,color='black',size=3,show.legend=F ) +

   geom_hline(yintercept = -13,color='#7CAE00',linetype='dashed') +
   geom_text(aes(240,-13,label = 'choose B', vjust = 2)
             ,color='#7CAE00',size=3,show.legend=F ) +

   labs(title='Random walks with different parameters',
        x='timesteps (t)') +

   theme_bw()
```



Interesting predictions

Given our equation for the general random walk we can derive some predictions without the need of doing costly simulations. For example, we can obtain the expected value of the process. That's sort of the mean valence we would have if we extended the process of decision making t timesteps

$$\mathbb{E}(V_t) = \mathbb{E}(V_0) + \mathbb{E}\Big(\sum_{i=1}^{t} Z_i\Big)$$

V_0 is a constant and every Z_i is independent and identically distributed (iid assumption), so we rewrite it as

$$\mathbb{E}(V_t) = V_0 + t \times \mathbb{E}(Z_1)$$

The expected value of Z_1 can be obtained from its definition

$$
\begin{aligned}
\mathbb{E}(Z_1) = \sum_{z\in\{1,0,-1\}} z \times \mathbb{P}(Z_1 = z) &= (+1)p_{+1} + (0)p_0 + (-1)p_{-1} \\
&= p_{+1} - p_{-1} \\
&= 0.25(1+b_s)(1+b_a) - \Big[0.25(1-b_s)(1+b_a)\Big] \\
&= 0.25(1+b_a)\Big[(1+b_s) - (1-b_s)\Big] \\
&= 0.25(2b_s + 2b_s b_a) \\
&= \frac{b_s + b_s b_a}{2}
\end{aligned}
$$

The expected valence after t timesteps then is: $\mathbb{E}(V_t) = V_0 + t \times \frac{b_s + b_s b_a}{2}$

Taking the limit when $t \to \infty$ we get the expected value when we are gathering info indefinitely

$\lim_{t\to\infty} \mathbb{E}(V_t) = \lim_{t\to\infty} V_0 + \frac{1}{2} \times [t \times b_S(1+b_a)] = \infty\, b_S(1+b_a)$

There is no limit, the decision maker (unrealistically) accumulates evidence and updates the valence indefinitely.Because $b_s, b_a \in [-1,1]$, we know the preferred alternative (positive valence A or negative valence for B) will depend entirely on the sampling bias being positive or negative. And if $b_s = 0$ we have an expected valence of 0, our decision maker has no preference. It may be choose A or B at some point because the process is random, but after sufficient time she would arrive to the conclusion that she has no preference (same evidence for both alternatives).

Using the formula of the expected value we can solve for t and obtain the mean number of timesteps needed to reach some decision boundary. This boundary is equal to the expected value, it's the value the subject reaches on average after t timesteps.

$$
\begin{aligned}
\mathbb{E}(V_t) &= V_0 + t \times \frac{b_s + b_s b_a}{2} \\
C &= V_0 + t \times \frac{b_s + b_s b_a}{2} \\
t &= \frac{2(C - V_0)}{b_s + b_s b_a}
\end{aligned}
$$

We need to take into account the signs for the variables if we want a value of t that makes sense.

Here's an example with simulated data and with the result derived from the formula.

```r
set.seed(1414)

time_needed <- function(C,V0,bs,ba){

   return( (2*(C - V0))/(bs + bs*ba) )
}

C <- 10; V0 <- 0; bs <- .4; ba <- 0
params <- list('C'=10,'V0'=0,'bs'=c(.3,.4),'ba'=c(0,.1))

expected_time_bias <- time_needed(C,V0,bs,ba)

iters <- 1000

sim_times_bias <- replicate(iters,randwalk(V0,
                                          A_criteria=C,B_criteria=-C,
                                          bs,
                                          ba)) |> rapply(f=length)



rbind('expected time'=expected_time_bias,'simulated mean time'=mean(sim_times_bias))
```

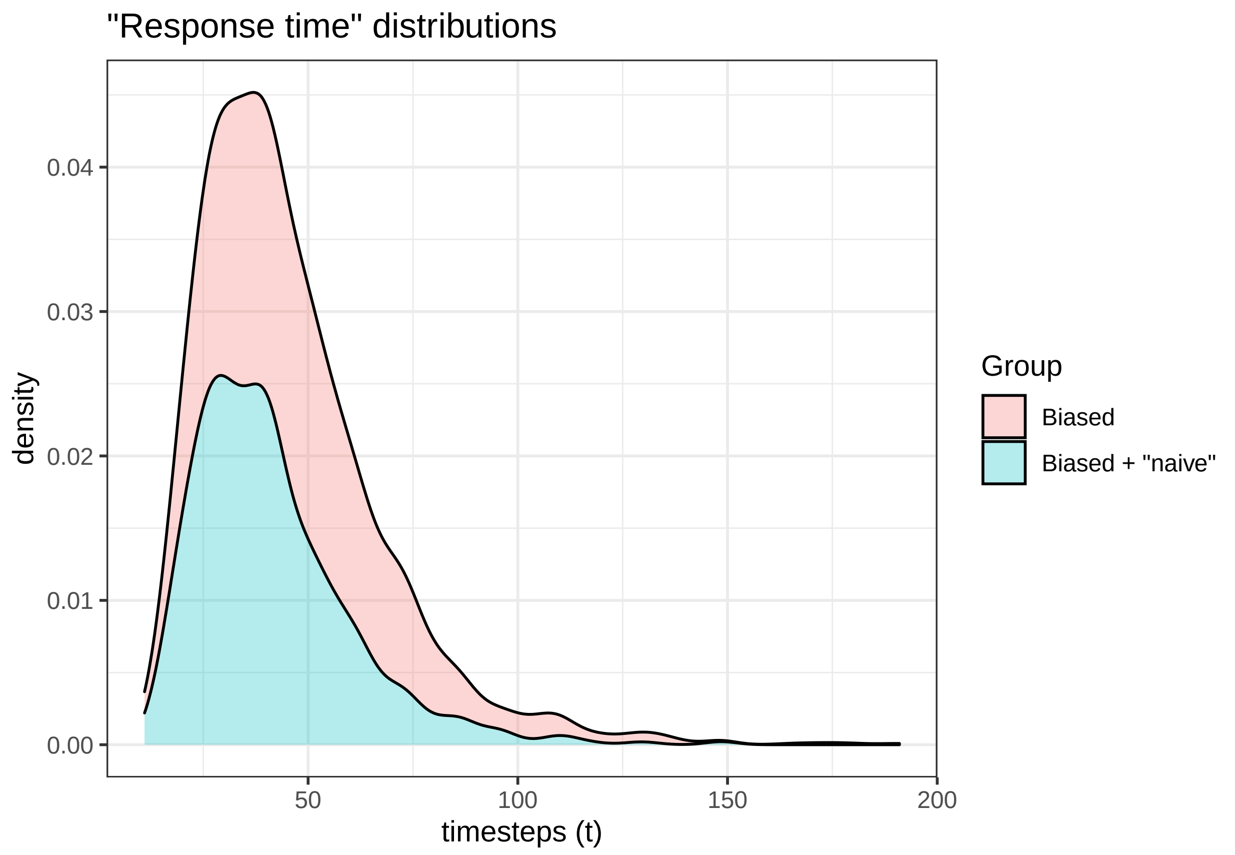
```
##                       [,1]
## expected time       50.000
## simulated mean time 49.918
```

By the way, the resulting time distributions in sequential sampling models usually fit very well empirical response time distributions (Smith & Ratcliff, 2015), a reason explaining the success of this models.

```
rbind(summary(sim_times_bias),
      summary(sim_times_bias2)) # computed but not shown. The difference is that ba = .2
```

```
##      Min. 1st Qu. Median   Mean 3rd Qu. Max.
## [1,]   13      33     46 49.918      61  191
## [2,]   11      28     38 41.289      50  149
```

```
df <- data.frame(list('Response_time'=c(sim_times_bias,sim_times_bias2),
                      'Group'=rep(c('Biased','Biased + "naive"'),each=iters)))

ggplot(df, aes(x = Response_time, fill = Group)) +
   geom_density(position='stack',alpha = 0.3) +
   labs(title='"Response time" distributions',
        x='timesteps (t)') +
   theme_bw()
```



The positive skewness observed is typical of human response time distributions.

Markov chain approach

The general random walk equation $V_t = V_0 + \sum_{i=1}^{t} Z_i$ implies that $V_t = \left(V_0 + \sum_{i=1}^{t-1} Z_i\right) + Z_t = V_{t-1} + Z_t$.The value of the valence depends only on the previous state (fixed) and on the value of the increment at time t

$$\mathbb{P}(V_t = v_t \mid V_{t-1} = v_{t-1}, V_{t-2} = v_{t-2}, \ldots, V_0 = v_0,) = \mathbb{P}(V_t = v_t \mid V_{t-1} = v_{t-1})$$

This memoryless property is the *Markov property*. Because it holds here, it means that our random walk process is in fact a *Markov chain*. We can go further. Let $v_{t-1} = i$, then $v_t = j = i + z$, with z being the value of the increment, which varies according to $\mathbb{P}(Z_t = z)$. Because j is a constant, then $\mathbb{P}(V_t = i + z \mid V_{t-1} = i) = \mathbb{P}(Z_t = z)$. In our specific case we had:

$$\mathbb{P}(V_t = i + 1 \mid V_{t-1} = i) = 0.25(1 + b_s)(1 + b_a)$$
$$\mathbb{P}(V_t = i + 0 \mid V_{t-1} = i) = 0.5(1 - b_a)$$
$$\mathbb{P}(V_t = i - 1 \mid V_{t-1} = i) = 0.25(1 - b_s)(1 + b_a)$$

This probabilities are called transition probabilities because they explicitly describe the probability of transitioning from state i to state $j = i + z$. Now imagine a situation with decision boundaries set at -2 and 2. There 5 possible states and 25 possible transitions. They can be nicely put in matrix form, the *transition matrix* $\mathbf{T}$ where each entry $\mathbf{T}_{ij}$ represents the probability of changing from state i to state j. Note that here the rows are probability vectors (non negative entries that sum up to 1), but we could also put them as columns. Returning to our specific model, the transition matrix would be

$$\mathbf{T} = \begin{array}{c|ccccc} & -2 & -1 & 0 & 1 & 2 \\ \hline -2 & 1 & 0 & 0 & 0 & 0 \\ -1 & \frac{(1-b_s)(1+b_a)}{4} & \frac{(1-b_a)}{2} & \frac{(1+b_s)(1+b_a)}{4} & 0 & 0 \\ 0 & 0 & \frac{(1-b_s)(1+b_a)}{4} & \frac{(1-b_a)}{2} & \frac{(1+b_s)(1+b_a)}{4} & 0 \\ 1 & 0 & 0 & \frac{(1-b_s)(1+b_a)}{4} & \frac{(1-b_a)}{2} & \frac{(1+b_s)(1+b_a)}{4} \\ 2 & 0 & 0 & 0 & 0 & 1 \end{array}$$

Also, where we had a single value to represent the initial valence, now we have a probability vector representing the probability of being at each state. Here we are assuming the preference is certain and located in one valence value. For instance, $V_0 = 0$ translates to

$$\mathbf{v}_0^\intercal = \begin{array}{ccccc} -2 & -1 & 0 & 1 & 2 \\ \hline 0 & 0 & 1 & 0 & 0 \end{array}$$

Among other things, with $\mathbf{T}$ and $\mathbf{v}_0$ we can make predictions about the long term behavior of the model. We can compute the probability of being in each state after t timesteps (we love probability/density distributions in psychology). With our previous model we could simulate the evolution of the psychological states (the random path itself), but with every simulation we had a different trajectory, a different specific and unique evolution of the mental states. Now, instead of being able to simulate a specific sequence of valences, we can simulate a sequence of probability distributions. At each moment t we can predict the probability of being in each state.

It can be shown that $\mathbf{v}_{t+1} = \mathbf{v}_t^\intercal \mathbf{T}$ which means (if we follow the recursion process) the following

$$\mathbf{v}_t = \mathbf{v}_0^\intercal \mathbf{T}^t$$

If we calculate the product $\mathbf{v}_0^\intercal \mathbf{T}^t$ we get the desired state probabilities. The "trouble" here is $\mathbf{T}^t$. Many packages can compute powers of matrices, but here I will take another approach that also has other benefits. We can diagonalize the transition matrix: $\mathbf{T} = \mathbf{P}^{-1}\mathbf{D}\mathbf{P}$ and take advantage of the result $\mathbf{T}^n = \mathbf{P}^{-1}\mathbf{D}^n\mathbf{P}$. Here $\mathbf{P}$ is a matrix with the eigenvectors of $\mathbf{T}$ as columns. On the other hand, $\mathbf{D}$ is a diagonal matrix with the eigenvalues of $\mathbf{T}$ as diagonal entries (Singh, 2013).

This strategy is legit in our case because a necessary and sufficient condition for a $n \times n$ matrix to be diagonalizable is that it has n linearly independent eigenvectors (Singh,2013). We can check this by computing the rank of the transposed eigenvector matrix. Full rank would mean that the matrix has n linearly independent rows (in this case the rows are the eigenvectors). The fact that the matrix is transposed doesn't affect because $rank(\mathbf{A}^T) = rank(\mathbf{A})$

This code shows that the condition holds, so the transition matrix is diagonalizable

```r
transition_matrix <- function(A_criteria,B_criteria,bs,ba){

   # re-scale to have B boundary on 1
   Ac <- A_criteria - B_criteria + 1
   Bc <- B_criteria - B_criteria + 1

   # build transition matrix
   Tr <- diag(1-0.5*(1+ba),nrow=length(Bc:Ac),ncol=length(Bc:Ac))
   dimnames(Tr) <- list(B_criteria:A_criteria,B_criteria:A_criteria)

   # set prob. for absorbing states (decision boundaries)
   Tr[Bc,Bc] <- 1
   Tr[Ac,Ac] <- 1
   for (state_now in 2:(nrow(Tr)-1)){
      for (state_next in 1:ncol(Tr)){

         if (state_next - state_now == 1){
            Tr[state_now,state_next] <- 0.5*(1+bs) * 0.5*(1+ba)
         }
         if (state_next - state_now == -1){
            Tr[state_now,state_next] <- (1-0.5*(1+bs)) * 0.5*(1+ba)
         }
      }
   }

   return(Tr)
}

Tr <- transition_matrix(A_criteria=2,B_criteria=-2,bs=0,ba=0)
Tr
```

```
##      -2   -1    0    1    2
## -2 1.00 0.00 0.00 0.00 0.00
## -1 0.25 0.50 0.25 0.00 0.00
## 0  0.00 0.25 0.50 0.25 0.00
## 1  0.00 0.00 0.25 0.50 0.25
## 2  0.00 0.00 0.00 0.00 1.00
```

```
eigenvec_matrix <- eigen(Tr)$vector
Matrix::rankMatrix(eigenvec_matrix)[1] == nrow(eigenvec_matrix)
```

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

The state probability vector at time t is then given by

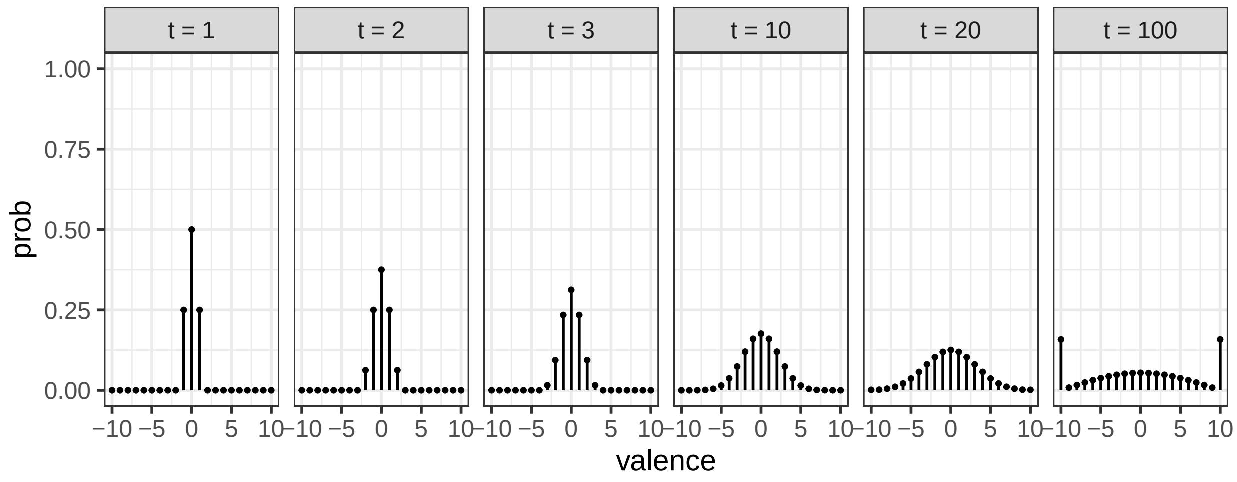
$$\mathbf{v}_t = \mathbf{v}_0^\intercal(\mathbf{P}^{-1}\mathbf{D}^t\mathbf{P})$$

The construction of $\mathbf{T}$, its decomposition and the final formula for $\mathbf{v}_t$ can be put in a single function

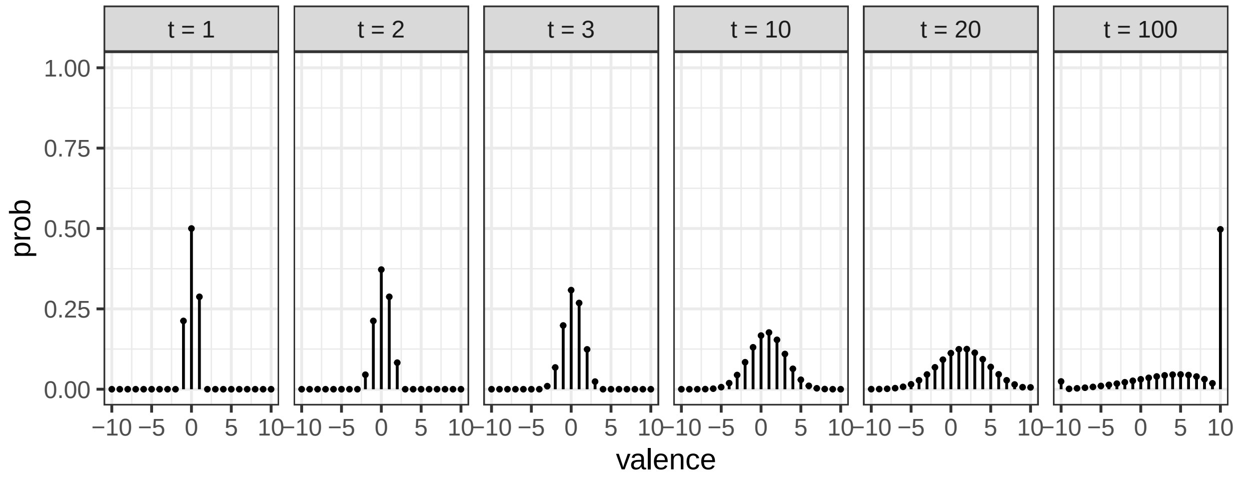
```
markov_probs <- function(V0,A_criteria,B_criteria,bs,ba,t){

   # re-scale to have B boundary on 1
   Ac <- A_criteria - B_criteria + 1
   Bc <- B_criteria - B_criteria + 1

   # build transition matrix
   Tr <- diag(1-0.5*(1+ba),nrow=length(Bc:Ac),ncol=length(Bc:Ac))

   # set prob for absorbing states (decision boundaries)
   Tr[Bc,Bc] <- 1
   Tr[Ac,Ac] <- 1

   for (state_now in 2:(nrow(Tr)-1)){
      for (state_next in 1:ncol(Tr)){
        if (state_next - state_now == 1){
          Tr[state_now,state_next] <- 0.5*(1+bs) * 0.5*(1+ba)
        }
        if (state_next - state_now == -1){
          Tr[state_now,state_next] <- (1-0.5*(1+bs)) * 0.5*(1+ba)
        }
      }
   }

   # diagonalize Tr to compute matrix powers easily
   D <- diag(eigen(Tr)$values)
   P <- eigen(Tr)$vectors

   # compute probs after t timesteps
   state_probs_after_t <- t(V0) %*% (P %*% D^t %*% solve(P))

   df <- data.frame(list('valence'=B_criteria:A_criteria,
                         'prob'=t(state_probs_after_t)))

   return(df)
}
```

The fancy stuff goes now. We can iteratively compute this vector $\mathbf{v}_t$ for many values of t and then see the change in the distribution of probability over time.

```
A_criteria <- 10
B_criteria <- -10
V0 <- integer(length(B_criteria:A_criteria))
V0[ceiling(length(V0)/2)] <- 1 # starts at 0

df <- data.frame()
for (t in c(1,2,3,10,20,100)){
   # start with a bias towards A
   mc <- markov_probs(V0,A_criteria=10,B_criteria=-10,bs=0,ba=0,t=t)
   mc['t'] <- paste0('t = ',as.character(t))
   df <- rbind(df,mc)
}
df$t <- factor(df$t,levels=paste0('t = ',c(1,2,3,10,20,100)),ordered=T)

ggplot(df,aes(x=valence,y=prob)) +
   geom_point(size=0.5) +
   coord_cartesian(ylim=c(0,1)) +
   geom_segment(aes(x=valence,xend=valence,y=0,yend=prob)) +
   facet_grid(.~ t) +
   theme_bw()
```



Now look what happens when there is a sampling bias ($b_s = 0.15$)



```r
gifski::save_gif(
   for (t in c(rep(1,20),1:100)){
      # start with a bias towards A
      mc <- markov_probs(V0,A_criteria=10,B_criteria=-10,bs=0.15,ba=0,t=t)
      g <- ggplot(mc,aes(x=valence,y=prob)) +
            geom_point(size=1) +
            coord_cartesian(ylim=c(0,1)) +
            geom_segment(aes(x=valence,xend=valence,y=0,yend=prob)) +
            labs(title=paste0('t = ',t)) +
            theme_bw()
      print(g)
}
,delay=.15,width=1280,height=720,res=144)
```

The distribution shifts to the right, where the positive values are. By $t = 100$ the preference is clear (what I haven't figure out is the fact that the probability of having a valence of -10, which implies choosing B, is so high)

The extra advantage of the diagonalization mentioned before has to do with the eigenvectors and eigenvalues obtained. Thanks to them we can obtain the steady-state vector for the transition matrix. It's also called the equilibrum vector because it's the state probability vector to which the Markov process converges as $t \to \infty$. Yes, we can predict the expected behavior of the process without making any simulations with big values of t. The problem is that the interpretation of this vector as a vector of long run probabilities is possible only when all entries of our transition matrix are strictly positive (Lay et. al, 2020)…so I'll leave this here.

References and other sources used

Diederich, A., & Busemeyer, J.R. (2003). Simple matrix methods for analyzing diffusion models of choice probability, choice response time, and simple response time. *Journal of Mathematical Psychology*, 47(3), 304-322.

Jones, P.W., & Smith, P. (2018). Random walks. In P.W. Jones & P. Smith, *Stochastic processes. An introduction* (pp.49-63). Chapman and Hall/CRC.

Lay, D.C., Lay, S.R., & McDonald, J.J. (2020). Finite-State Markov Chains. In D.C. Lay, S.R. Lay, & J.J. McDonald, *Linear algebra and its applications. Global edition* (pp. 596-658). Pearson.

Singh, K. (2013). Diagonalization. In K. Singh, *Linear algebra: step by step* (pp.519-533). OUP Oxford.

Smith, P.L., & Ratcliff, R. (2015). An introduction to the diffusion model of decision making. In P.L. Smith & R. Ratcliff, *An introduction to model-based cognitive neuroscience* (pp. 49-70). Springer.

Wang, Z.J., & Busemeyer, J.R. (2021). Sequential sampling models. In Z.J. Wang & J.R. Busemeyer, *Cognitive choice modeling* (pp. 121-148). MIT Press.