Sporting events host witness to a wide range of human emotion. The emotional ups and downs are especially clear among invested fans. Fans experience the joy and excitement of a triumphant comeback, or the anxiety and disappointment of a loss. It is particularly interesting to see how emotions differ from two opposing fan groups watching the same match.

I decided to perform some simulations on how a crowd of fans would react during a basketball game. Why basketball? Two reasons: First is the frequency of scoring (more baskets = more *reactive* simulation), and second is that the NBA finals are in swing at time of writing.

Below I’ve described the parameters of the simulation in detail and provided reproducible code for all examples, but here’s the quick and dirty. I’m simulating fans’ happiness and nerves (valence and arousal). Happiness reflects baskets scored (up for team basket and down for enemy basket) and overall team performance. Nerves reflects baskets scored, time remaining in game, and the difference between score (higher for closer game). There are other parameters involved that are described in more detail below.

Let’s run the simulation! I’m simulating a short game that’s about 16 minutes long, but the simulation is sped up. There are 25 fans per team with each dot representing a fan. Blue dots root for the blue team and red dots for red. In this game, blue starts ahead, but red makes a rallying comeback.

What are we seeing here? As the game progresses, fans move into different emotional states. I’ve used the terms *content*, *disappointed*, *nervous*, and *excited* as helpful placeholders, but we should think of these in terms of varying along dimensions of *valence* (e.g., happiness) and *arousal* (e.g., nerves, excitedness). For example, when their team is doing well, fans are happier, but they’re only excited when the score is close. For each basket, there’s a brief spike in arousal and happiness. As the game draws closer to an end, fans get more excited/anxious, especially if the score is close.

So how would this play out if one team decimated the other? Let’s give the red team a slight edge over blue:

Notice how ‘excited’ becomes ‘content’ and ‘nervous’ turns to ‘disappointment’ because the difference becomes insurmountable for the blue team. We can plug in any numbers for this function to produce different results. However, the algorithm is highly sensitive to small changes in scoring chance because this is calculated for every *second* of the game.

**Going Behind the Simulation**

Skip this if you have no interest in the psychology of emotions. What I’m striving to simulate are the laws of emotion dynamics (Kuppens & Verduyn, 2017). Emotions change from moment to moment, but there’s also some stability from one moment to the next. Apart from when a basket is scored, most fans cluster around a particular state (this is called an *attractor state*). Any change is attributable to random fluctuations (e.g., one fan spills some of their beer, maybe another fan sees an amusing picture of a cat on their phone). When a basket *is* scored, this causes a temporary fluctuation away from the attractor state, after which people resort back to their attractor. More gradual situational factors result in small changes in attractors too. This is why arousal tends to increase with closer games and when the game is approaching the end.

**Detailed Aspects of Simulation and Code**

There are two parts to this simulation. Part 1 simulates a basketball game for a specified duration of time (*n\_seconds*) and specified probabilities of scoring. The simulation is also designed with a small post game period – sort of like a cool down after the game. During the post game, fans’ arousal returns to baseline.

The second and more complicated part of the simulation is the emotional part. Each fan has a score on valence and arousal at each second. These can be further broken down into fixed effects and random effects. The fixed effect for valence is described as follows:

\[Valence = 3 + \beta\_{1overall.difference}(.20) + \beta\_{2recent.score(1.5 + \frac{current.time}{total.time})} – \beta\_{3recent.enemy.score}\]

The fixed effect for arousal looks like this:

\[Arousal = 3 + \beta\_{1 (\frac{current.time}{total.time})(.75)(post.game)} + \beta\_{2(1 – \sqrt{|difference|})} + \beta\_{3|recent.basket|}\]

The exact values for these coefficients are somewhat arbitrary. They resulted from a lot of trial and error to identify which best simulated emotion dynamics.

Each fan’s valence and arousal is calculated using the above equations + variability. The variability is added by applying random variation to the first constant in the above equations and white noise with a simulated ARIMA model. On top of this, I used a kernel regression smoother to smooth out each fan’s trajectory. To simulate delayed reaction time to baskets, I added some *lags* at random intervals – otherwise the simulated fans appear to react quicker than the score board. Here’s what all of this looks like for one fan’s valence over time:

library(tidyverse)

game\_long %>%

filter(fan == 1,

team == "team1") %>%

ggplot(aes(x = second, y = valence)) +

geom\_line(color = "red", size = 1) +

theme\_minimal(base\_size = 16) +

labs(title = "Fan One's Valence over Time",

x = "Time",

y = "Valence") +

theme(legend.position = "none",

axis.title.x = element\_text(size = 18),

axis.title.y = element\_text(size = 18),

plot.title = element\_text(size = 18, hjust = .5))

We can see this across all fans as well:

game\_long %>%

ggplot(aes(x = instance, color = team)) +

geom\_line(aes(y = valence), alpha = .65) +

labs(title = "Valence over Time",

x = "Time",

y = "Valence") +

theme\_minimal() +

theme(legend.position = "none",

axis.title.x = element\_text(size = 18),

axis.title.y = element\_text(size = 18),

plot.title = element\_text(size = 18, hjust = .5))

And for arousal…

game\_long %>%

ggplot(aes(x = instance, color = team)) +

geom\_line(aes(y = arousal), alpha = .65) +

labs(title = "Arousal over Time",

x = "Time",

y = "Arousal") +

theme\_minimal() +

theme(legend.position = "none",

axis.title.x = element\_text(size = 18),

axis.title.y = element\_text(size = 18),

plot.title = element\_text(size = 18, hjust = .5))

For those interested, here’s the full code for the simulation and first animation. Have fun with it! I’m also looking for ways to improve it. Ironically, I don’t watch a ton of sports, so if you think I’m open to input on how to make it more realistic!

library(tidyverse)

library(forecast)

library(gganimate)

library(extrafont)

game\_simulation <- function(n\_seconds = 1000, n\_fans = 25, team1\_prob = 0.012, team2\_prob = 0.012) {

score\_board <- data.frame(matrix(nrow = n\_seconds, ncol = 5))

colnames(score\_board) <- c("second", "team1\_score", "team2\_score", "difference", "end\_game")

score\_board$second <- c(1:n\_seconds)

score\_board$team1\_score <- cumsum(sample(c(0, 1, 1.5), n\_seconds, replace = TRUE, prob = c(1 - team1\_prob, team1\_prob \* .90, team1\_prob \* .10)))

score\_board$team2\_score <- cumsum(sample(c(0, 1, 1.5), n\_seconds, replace = TRUE, prob = c(1 - team2\_prob, team2\_prob \* .90, team2\_prob \* .10)))

score\_board$difference <- score\_board$team1\_score - score\_board$team2\_score

score\_board$end\_game = 1

total\_time <- n\_seconds + n\_seconds \* .05

end\_time <- data.frame(max(score\_board$second + 1):(total\_time))

colnames(end\_time) <- "second"

end\_state <- score\_board[nrow(score\_board), 2:5]

post\_game <- data.frame(matrix(nrow = n\_seconds \* .05, ncol = 0),

"team1\_score" = end\_state$team1\_score,

"team2\_score" = end\_state$team2\_score,

"difference" = end\_state$difference,

"end\_game" = 0)

post\_game <- cbind(end\_time, post\_game)

score\_board <- rbind(score\_board, post\_game)

valence <- function(time, overall\_difference, recent\_team\_score, recent\_enemy\_score) (

(overall\_difference \* .20) + (recent\_team\_score \* (1.5 + time/length(time))) - (recent\_enemy\_score)

)

arousal <- function(time, difference, recent\_score, end\_game) (

((time/n\_seconds) \* .75 \* end\_game) + (1 - sqrt(abs(difference))) + abs(recent\_score \* 1.5)

)

team1\_valence <- data.frame(matrix(nrow = total\_time, ncol = 0))

team1\_arousal <- data.frame(matrix(nrow = total\_time, ncol = 0))

team2\_valence <- data.frame(matrix(nrow = total\_time, ncol = 0))

team2\_arousal <- data.frame(matrix(nrow = total\_time, ncol = 0))

for(i in 1:n\_fans) {

team1\_valence[[i]] <- ksmooth(x = score\_board$second,

y = rnorm(n = 1, mean = 3, sd = .25) + (arima.sim(list(order = c(1, 0, 1), ma = -.1, ar = c(.9)), n = total\_time) \* .25) + valence(time = score\_board$second,

overall\_difference = score\_board$difference,

recent\_team\_score = lag(score\_board$team1\_score, n = sample(c(1:5), 1), default = 0) - lag(score\_board$team1\_score, n = sample(c(6:8), 1), default = 0),

recent\_enemy\_score = lag(score\_board$team2\_score, n = sample(c(1:5), 1), default = 0) - lag(score\_board$team2\_score, n = sample(c(6:8), 1), default = 0)),

bandwidth = total\_time/100, kernel = "normal")$y

}

colnames(team1\_valence) <- paste0("team1\_", "valence\_", 1:n\_fans)

team1\_valence[team1\_valence > 6] <- 6

team1\_valence[team1\_valence < 0] <- 0

for(i in 1:n\_fans) {

team1\_arousal[[i]] <- ksmooth(x = score\_board$second,

y = rnorm(n = 1, mean = 3, sd = .25) + (arima.sim(list(order = c(1, 0, 1), ma = -.1, ar = c(.9)), n = total\_time) \* .25) + arousal(time = score\_board$second,

difference = score\_board$difference,

recent\_score = lag(score\_board$difference, n = sample(c(1:3), 1), default = 0) - lag(score\_board$difference, n = sample(c(4:6), 1), default = 0),

end\_game = score\_board$end\_game),

bandwidth = total\_time/100, kernel = "normal")$y

}

colnames(team1\_arousal) <- paste0("team1\_", "arousal\_", 1:n\_fans)

team1\_arousal[team1\_arousal > 6] <- 6

team1\_arousal[team1\_arousal < 0] <- 0

for(i in 1:n\_fans) {

team2\_valence[[i]] <- ksmooth(x = score\_board$second,

y = rnorm(n = 1, mean = 3, sd = .25) + (arima.sim(list(order = c(1, 0, 1), ma = -.1, ar = c(.9)), n = total\_time) \* .25) + valence(time = score\_board$second,

overall\_difference = -score\_board$difference,

recent\_team\_score = lag(score\_board$team2\_score, n = sample(c(1:5), 1), default = 0) - lag(score\_board$team2\_score, n = sample(c(6:8), 1), default = 0),

recent\_enemy\_score = lag(score\_board$team1\_score, n = sample(c(1:5), 1), default = 0) - lag(score\_board$team1\_score, n = sample(c(6:8), 1), default = 0)),

bandwidth = total\_time/100, kernel = "normal")$y

}

colnames(team2\_valence) <- paste0("team2\_", "valence\_", 1:n\_fans)

team2\_valence[team2\_valence > 6] <- 6

team2\_valence[team2\_valence < 0] <- 0

for (i in 1:n\_fans) {

team2\_arousal[[i]] <- ksmooth(x = score\_board$second,

y = rnorm(n = 1, mean = 3, sd = .25) + (arima.sim(list(order = c(1, 0, 1), ma = -.1, ar = c(.9)), n = total\_time) \* .25) + arousal(time = score\_board$second,

difference = score\_board$difference,

recent\_score = lag(score\_board$difference, n = sample(c(1:3), 1), default = 0) - lag(score\_board$difference, n = sample(c(4:6), 1), default = 0),

end\_game = score\_board$end\_game),

bandwidth = total\_time/100, kernel = "normal")$y

}

colnames(team2\_arousal) <- paste0("team2\_", "arousal\_", 1:n\_fans)

team2\_arousal[team2\_arousal > 6] <- 6

team2\_arousal[team2\_arousal < 0] <- 0

game\_affect <- cbind(score\_board, team1\_valence, team1\_arousal, team2\_valence, team2\_arousal)

return(game\_affect)

}

set.seed(060519)

game <- game\_simulation(n\_seconds = 1000, n\_fans = 25)

game\_long <- game %>%

mutate(instance = row\_number(),

team1\_score = team1\_score \* 2,

team2\_score = team2\_score \* 2) %>%

select(instance, everything()) %>%

select(-end\_game) %>%

gather(key = "team", value = "score", c(team1\_valence\_1:length(game))) %>%

separate(col = team, into = c("team", "dimension", "fan"), sep = "\_") %>%

spread(dimension, score)

p <- game\_long %>%

ggplot() +

annotate("rect", xmin = 0, xmax = 3, ymin = 0, ymax = 3, alpha = 0.5, fill = "lightblue") +

annotate("rect", xmin = 3, xmax = 6, ymin = 0, ymax = 3, alpha = 0.5, fill = "lightgreen") +

annotate("rect", xmin = 0, xmax = 3, ymin = 3, ymax = 6, alpha = 0.2, fill = "red") +

annotate("rect", xmin = 3, xmax = 6, ymin = 3, ymax = 6, alpha = 0.2, fill = "yellow") +

annotate("text", x = c(1.5, 4.5, 1.5, 4.5), y = c(1.5, 1.5, 4.5, 4.5), label = c("Disappointed", "Content", "Nervous", "Excited"),

size = 10, alpha = .50, family = "Verdana") +

geom\_text(aes(x = 1.5, y = 5.5, label = paste("Red:", team1\_score)), size = 8, color = "red", family = "Verdana") +

geom\_text(aes(x = 4.5, y = 5.5, label = paste("Blue:", team2\_score)), size = 8, color = "blue", family = "Verdana") +

geom\_point(aes(x = valence, y = arousal, color = team), alpha = .70, size = 4) +

scale\_x\_continuous(limits = c(0, 6)) +

scale\_y\_continuous(limits = c(0, 6)) +

scale\_color\_manual(values = c("red", "blue")) +

labs(title = 'Time: {round((max(game\_long$second) - frame\_time)/60, 2)}',

x = "Happiness",

y = "Nerves") +

theme\_minimal(base\_size = 16) +

transition\_time(second) +

theme(legend.position = "none",

axis.title.x = element\_text(size = 18),

axis.title.y = element\_text(size = 18),

plot.title = element\_text(size = 18, hjust = .5),

text = element\_text(family = "Verdana"))

game1 <- animate(p, nframes = 1062, width = 500, height = 500, fps = 14, end\_pause = 12)

game1

**References**

Kuppens P, Verduyn P. (2017). Emotion dynamics. *Current Opinion in Psychology. 17*, 22–26. doi: 10.1016/j.copsyc.2017.06.004.