Accurate calculation of combination TTKs

hubris_3sd

2022-05-01

Contents

Contact:

https://twitter.com/3sd_gaming

https://github.com/3sd-gaming

1 Introduction

1.1 Terminology and acronyms

- *TTK* : Time to kill, the amount of time it takes to deliver enough damage to kill or down a player
- *STK* : Shots to kill, the amount of shots needed to kill or down a player
- *shot_ratios* : The probability to hit at each location (sums to 1)
- *damage_vector* : The vector of damages at each location

- *hp* : The health points a player can have
- *firing_period* : Time between bullets. To find firing period in seconds from firing rate in RPM: $firing\_period = 60/firing\_rate$

1.2 TTK vs STK and probability

To compute time to kill we first need to know the probability of a given combination of shots occurring, and then translate that into time by multiplying by the firing period of the gun. Fundamentally the conversion from STK to TTK is a change in units, so the probabilities governing the STK distribution also apply to the TTK distribution.

$$TTK = (STK - 1) * firing\_period$$

2 Accurate calculation of shot combination probabilities

2.1 Case example

Lets start with a simple example where a player has **100** *hp* and there are only two shot locations for a gun with 600 RPM fire rate.

location	damage	probability
head	50	0.25
other	25	0.75

2.2 What is our chance to kill in two hits?

Intuitively there is only a single combination of shots capable of killing in two shots (two headshots), and given the probability of 0.25 then our chance of getting two headshots is (0.25)^2 = 0.0625, or 6.25% chance of killing in two shots.

We can draw out a table of the different possibilities and calculate their chances in the same way:

combination	head	other	probability
n_1	2	0	0.0625
n_2	1	1	0.1875
n_3	0	2	0.5625

And we see the first issue, our probabilities don't sum to 1!

```
sum(shot2_combs$probability)
```

```
## [1] 0.8125
```

The reason our probabilities don't sum to 1 is that there are actually two ways we can achieve combination n_2, either a headshot followed by an other shot OR an other shot followed by a headshot. When we correct that probability (multiply the raw probability by 2 to account for this) we will see that the probabilities now sum to 1.

```
shot2_combs <- data.frame(combination = paste0('n_', 1:3),
                          head = 2:0,
                          other = 0:2,
                          raw_probability = 0.25^(2:0)*0.75^(0:2))
```

```
2* shot2_combs$raw_probability[2]
```

```
## [1] 0.375
```

```
sum(shot2_combs$raw_probability[1],
    2* shot2_combs$raw_probability[2],
    shot2_combs$raw_probability[3])
```

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

2.3 How can we generalize this to more advanced situations?

In the toy example above we effectively used the probability mass function of the multinomial distribution to compute our chance to kill with each shot combination.

The multinomial distribution describes the probability of independent trials which lead to success in only one category. For example, the multinomial distribution describes the likelihood of rolling five die and getting the (order independent) result: {1,2,3,4,5,5} .

$$PMF(Multinomial\ distribution) = \frac{n!}{x_1!...x_k!} p_1^{x_1}...p_k^{x_k}$$

To apply this to our goal of computing TTKs:

- n = The number of bullets fired
- k = the different damage locations
- x_k = the number of bullets hit at each location
- p_k = the probability to hit at each location

2.4 Apply multinomal PMF to our example case

location	damage	probability
head	50	0.25
other	25	0.75

Using the same example situation with only two shot locations, we can go back and revisit our scenario of hitting one headshot and one other shot applied to the multinomial distribution.

$$Prob(1,1) = \frac{n!}{x_1!...x_k!} p_1^{x_1}...p_k^{x_k}$$

$$Prob(1,1) = \frac{2!}{1! * 1!} 0.25^1...0.75^1$$

$$Prob(1,1) = \frac{2}{1} 0.1875 = 0.375$$

2.5 Applying this approach to every shot combination for n_bullets = {2:4}

```
# Define the potential shot combinations
shot2_combs <- data.frame(n_head = 2:0,
                          n_other = 0:2)
```

```
shot3_combs <- data.frame(n_head = 3:0,
                          n_other = 0:3)

shot4_combs <- data.frame(n_head = 4:0,
                          n_other = 0:4)

# Append them together
shot_combs <- rbind(shot2_combs,
                    shot3_combs,
                    shot4_combs)

# Add our shot_ratios and damage profile information
shot_combs$head_prob <- 0.25
shot_combs$other_prob <- 0.75
shot_combs$head_damage <- 50
shot_combs$other_damage <- 25

# Find the multinomial probability for each combination
shot_combs <- shot_combs %>%
  mutate(n = n_head + n_other) %>%
  mutate(prob = factorial(n)/(factorial(n_head)*factorial(n_other))* head_prob^n_head*other_prob^n_othe

knitr::kable(shot_combs)
```

n_head	n_other	head_prob	other_prob	head_damage	other_damage	n	prob
2	0	0.25	0.75	50	25	2	0.0625000
1	1	0.25	0.75	50	25	2	0.3750000
0	2	0.25	0.75	50	25	2	0.5625000
3	0	0.25	0.75	50	25	3	0.0156250
2	1	0.25	0.75	50	25	3	0.1406250
1	2	0.25	0.75	50	25	3	0.4218750
0	3	0.25	0.75	50	25	3	0.4218750
4	0	0.25	0.75	50	25	4	0.0039063
3	1	0.25	0.75	50	25	4	0.0468750
2	2	0.25	0.75	50	25	4	0.2109375
1	3	0.25	0.75	50	25	4	0.4218750
0	4	0.25	0.75	50	25	4	0.3164062

2.6 Converting shot combination probabilities to TTK

To convert the shot combination probabilities to TTK we need to:

1. Calculate the damage dealt by each combination
2. Evaluate whether this amount of damage will kill the target
3. Calculate the chance to kill in n bullets as the sum of the shot combination probabilities for all shot combinations which kill in n bullets
4. Convert STK to TTK using the fire period of the weapon

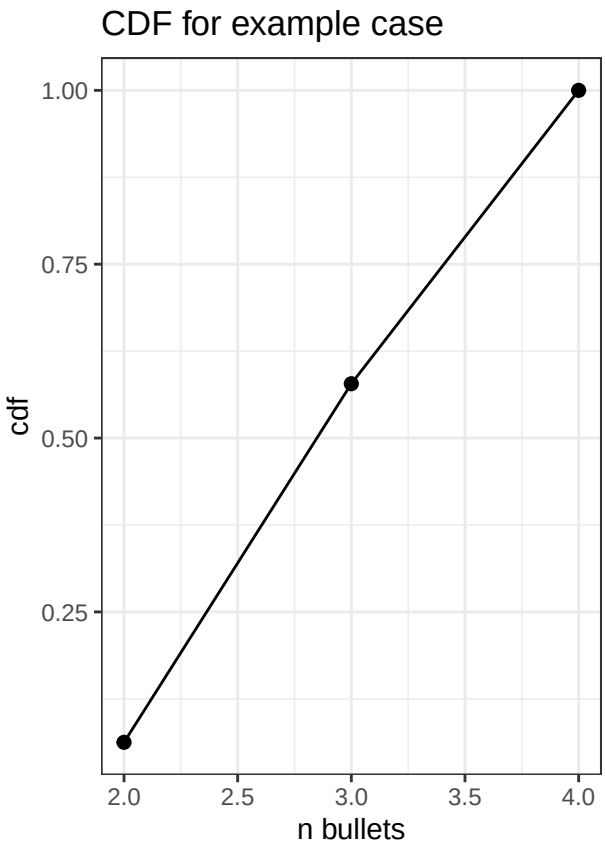
```
# Compute damage dealt by each combination and whether that would kill a target
shot_combs <- shot_combs %>%
  mutate(damage = n_head*head_damage + n_other*other_damage) %>%
  mutate(target_killed = damage >= 100)
```

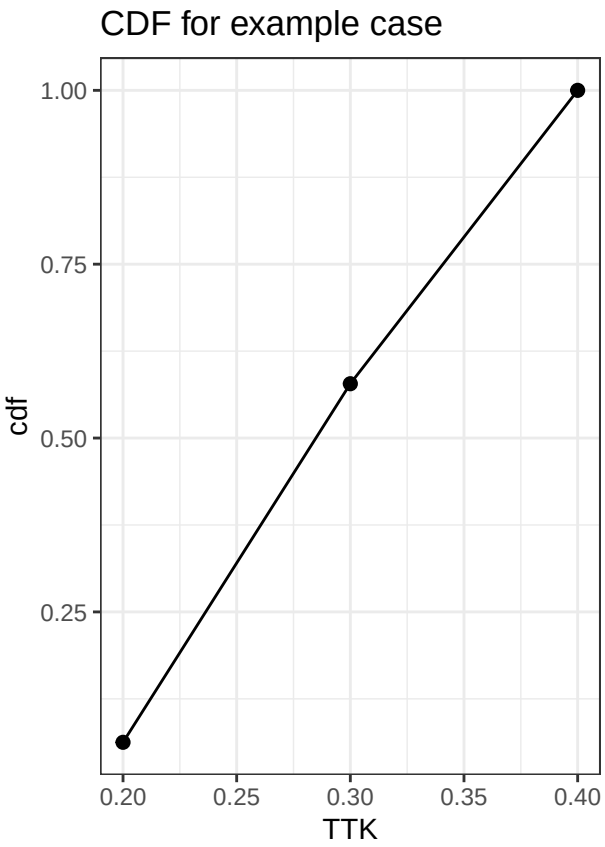
n_head	n_other	head_prob	other_prob	head_damage	other_damage	n	prob	damage	target_killed
2	0	0.25	0.75	50	25	2	0.0625000	100	TRUE
1	1	0.25	0.75	50	25	2	0.3750000	75	FALSE
0	2	0.25	0.75	50	25	2	0.5625000	50	FALSE
3	0	0.25	0.75	50	25	3	0.0156250	150	TRUE
2	1	0.25	0.75	50	25	3	0.1406250	125	TRUE
1	2	0.25	0.75	50	25	3	0.4218750	100	TRUE
0	3	0.25	0.75	50	25	3	0.4218750	75	FALSE
4	0	0.25	0.75	50	25	4	0.0039063	200	TRUE
3	1	0.25	0.75	50	25	4	0.0468750	175	TRUE
2	2	0.25	0.75	50	25	4	0.2109375	150	TRUE
1	3	0.25	0.75	50	25	4	0.4218750	125	TRUE
0	4	0.25	0.75	50	25	4	0.3164062	100	TRUE

```
# Find the cumulative distribution (chance to kill) in n bullets
stk <- shot_combs %>%
  group_by(n) %>%
  summarize(cdf = sum(prob*target_killed))
```

n	cdf
2	0.062500
3	0.578125
4	1.000000

From here we can simply multiply by the firing period to find TTK.



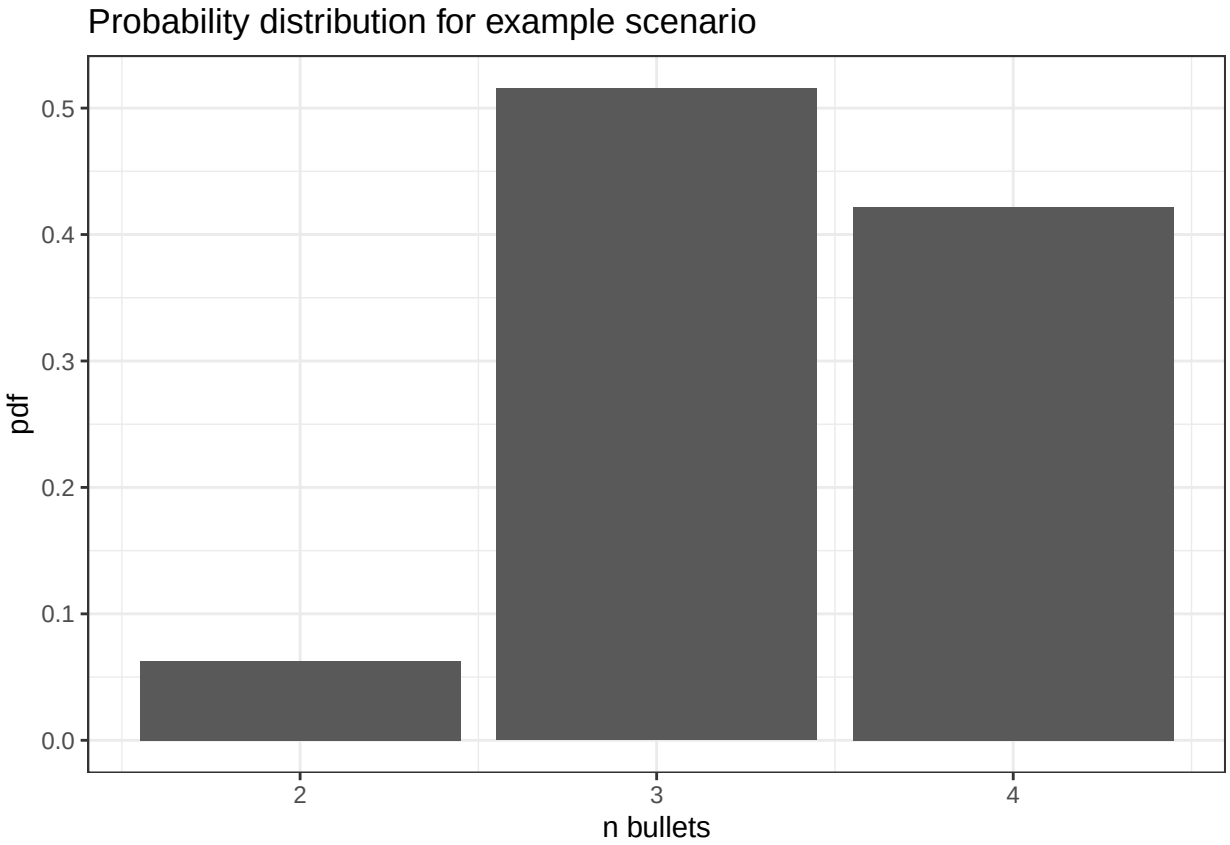


2.7 Summarizing this distribution

2.7.1 Probability Density Function from CDF

The distribution of possible TTKs is probably better visualized by the probability density function than CDF. The PDF is simply the rate of change in the CDF. For discrete functions this means $PDF[x] = CDF[x] - CDF[x-1]$

```
stk <- stk %>%
  mutate(pdf = cdf - lag(cdf, default = 0))
```



2.7.2 Summary Statistics

Mean_STK : Is the mean of the distribution. It can be found with: $mean(STK) = \sum STK * PDF$

Mode_STK : Is the most likely time to kill, this is simply the maximum probability

Median_STK: Is the the lowest shot value where CDF >= 0.5

```
stk_summary <- stk %>%
  summarize(mean_stk = sum(n*pdf),
            mode_stk = n[which.max(pdf)])

stk_summary$median_stk <- stk %>%
  filter(cdf >= 0.5) %>%
  filter(cdf == min(cdf)) %>%
  dplyr::select(n) %>%
  pull()

knitr::kable(stk_summary)
```

mean_stk	mode_stk	median_stk
3.359375	3	3

3 Why the expected damage model (shot_ratios * damage_profile = average damage) cannot be used to calculate combination TTK

3.1 The expected damage model to find 'average' bullet damage

The simplest way you might estimate combination TTK is to compute the expected damage, which would simply be the sum of the product of *shot_ratios* and *damage_vector* . This will return the expected damage for firing a single bullet.

$$expected\_damage = \sum_{i}^{n} damage\_vector_i * shot\_ratios_i$$

The expected STK is then computed as:

$$expected\_STK = \frac{hp}{expected\_damage}$$

And expected TTK is then a function of expected_STK and the firing period of the weapon

$$expected\_TTK = (expected\_STK - 1) * firing\_period$$

3.2 Fractions of a bullet - continuous vs discrete data

The expected_TTK, as calculated from the expected damage of a bullet, fails to accurately estimate the actual TTK distribution because the TTK distribution is a collection of discrete probabilities. Simply put - you can't hit with a fraction of a bullet. While you could round up to the nearest, that will often result in an even greater error.

To be clear, the expected damage model is an appropriate and valid way to estimate the average damage dealt by a bullet over many samples. The error of the expected damage model approaches zero as the number of samples approaches infinity. This means that the expected damage model is more appropriate when applied to scenarios where many bullets are fired (damage per magazine calculations) than combination TTK with the assumption of high accuracy.

3.3 Comparison of expected damage model (wrong) versus multinomial probabilities (correct)

Lets apply the expected damage model to our simple model of a 100 hp player being hit by a gun with two shot locations firing at 600 RPM:

location	damage	probability
head	50	0.25
other	25	0.75

```
expected_damage = sum(ex$damage*ex$probability)
expected_damage
```

```
## [1] 31.25
```

```
expected_stk <- 100 / expected_damage
expected_stk
```

```
## [1] 3.2
```

```
expected_ttk <- (3.2 - 1)*60/600 # Fire rate of 600 RPM
expected_ttk
```

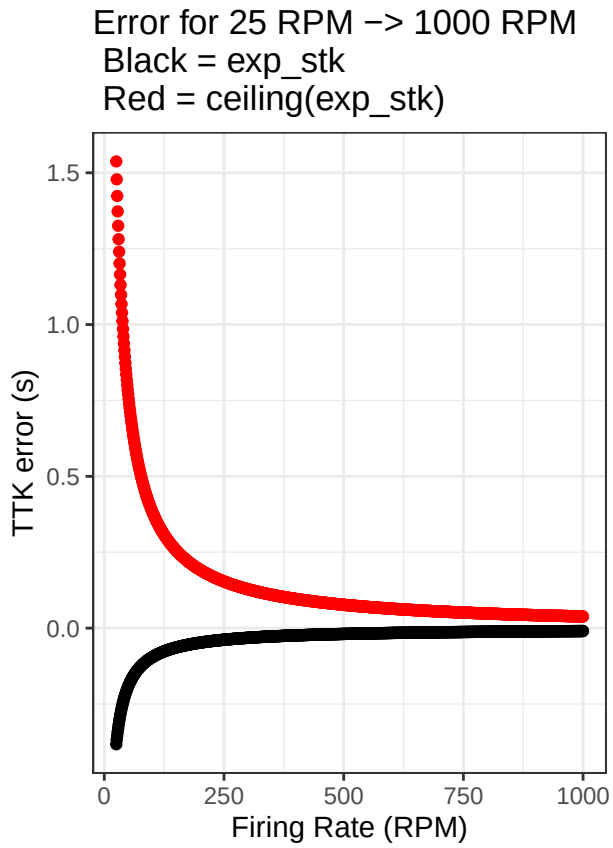
```
## [1] 0.22
```

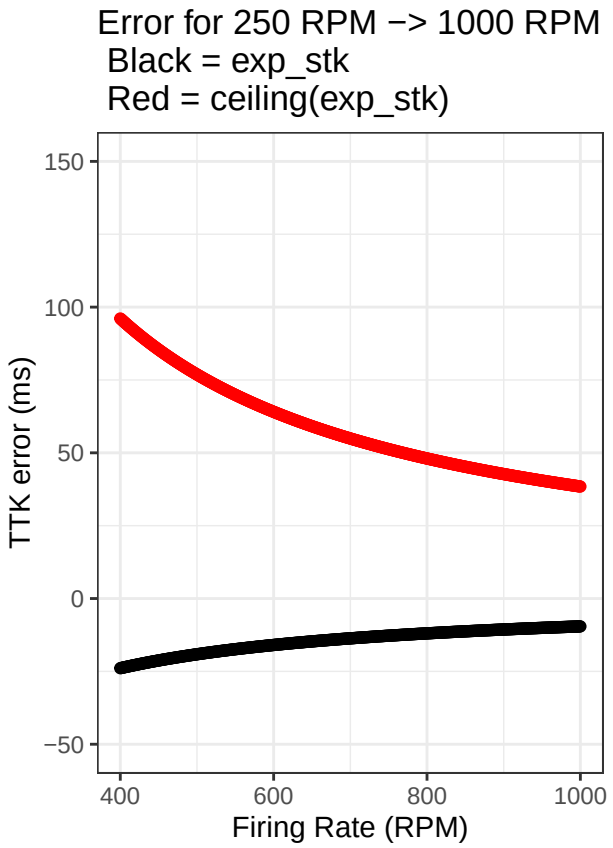
When we compare the expected STK to mean STK we found with the multinomial distribution we see that they don't agree, with an error of ~0.16 .

```
stk_summary$mean_stk - expected_stk
```

```
## [1] 0.159375
```

While this doesn't seem like too large of an error, it is important to note that this error represents the difference in bullets required to kill, meaning the error will inflate when using slower firing weapons. This TTK error is especially concerning when using combination TTK to optimize attachments which modify weapon characteristics such as fire rate or damage per location.



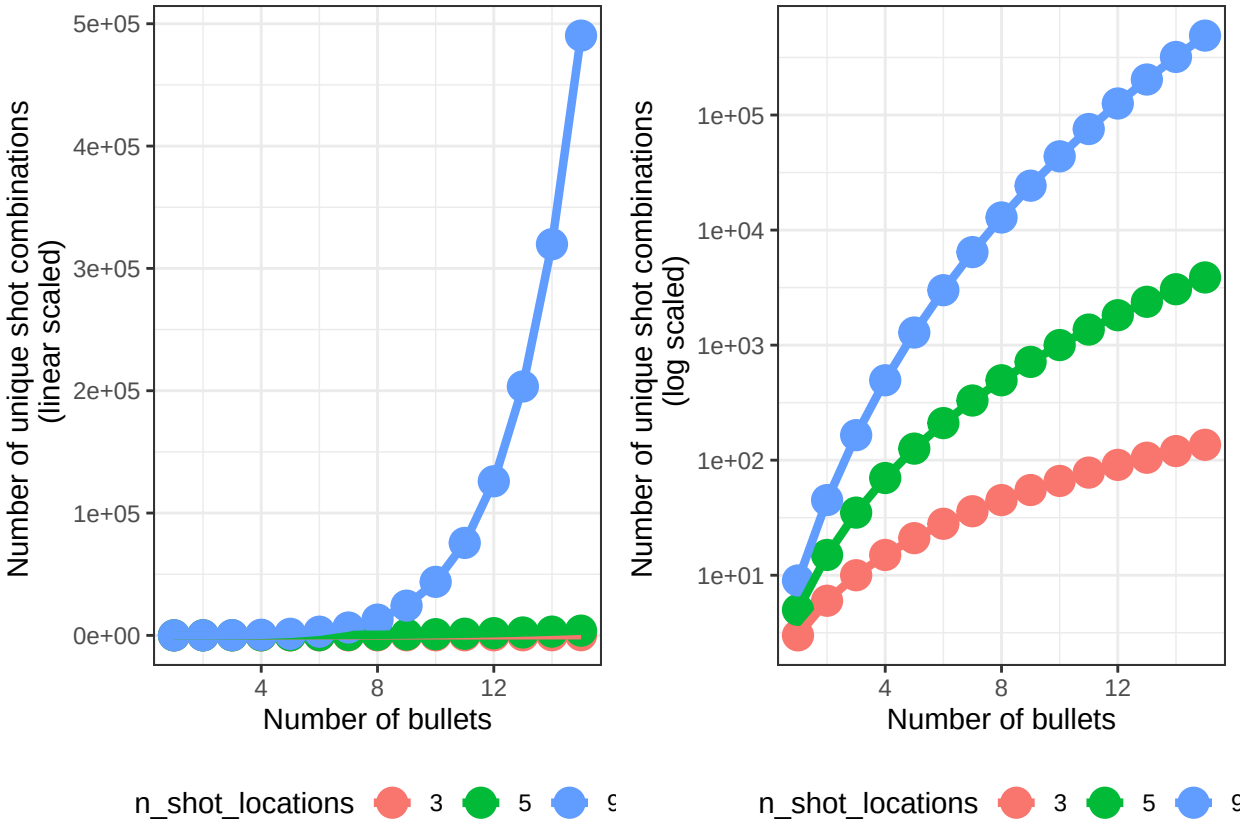


4 Implementation, alternative approaches, and caveats

4.1 Computational intensity

Due to the combinatorial nature of the multinomial distribution, the number of potential shot combinations increases exponentially with the number of shots fired. The number of unique combinations which can be made from sampling (with replacement) r samples from n objects is:

$$C^r(n,r) = \frac{(n+r-1)!}{r!(n-1)!}$$



Some approaches to simplify the computation:

- Merge shot locations with the same damage value and take the joint probability as the sum of individual pobabilities.
- Filter shot combinations to only those capable of killing before computing probablity. (Potentially a gradient-descent like approach could be used to propagate from the maximum damage combination)

4.2 Using the multinomial function to generate a simulation of combination TTKs

As described above, the number of unique shot location combinations quickly increases to an unfeasible level. To counter this, we can take advantage of generative functions which take a vector of probabilities and return random samples from the multinomial distribution. These results can then be aggregated and used in place of direct computation of probabilities. While this does effectively return a bootstrapped estimate of the distribution instead of absolute probabilities, it can be orders of magnitude faster to generate with an error rate much lower than 1ms. Example below.

4.3 Accounting for missed shots

You can account for missed shots by simply adding an additional shot location which corresponds to zero damage dealt and expanding the *shot_ratio* vector by one additional value corresponding to this miss rate.

$$damages_{new} = \{damages_{orig},\ 0\}$$

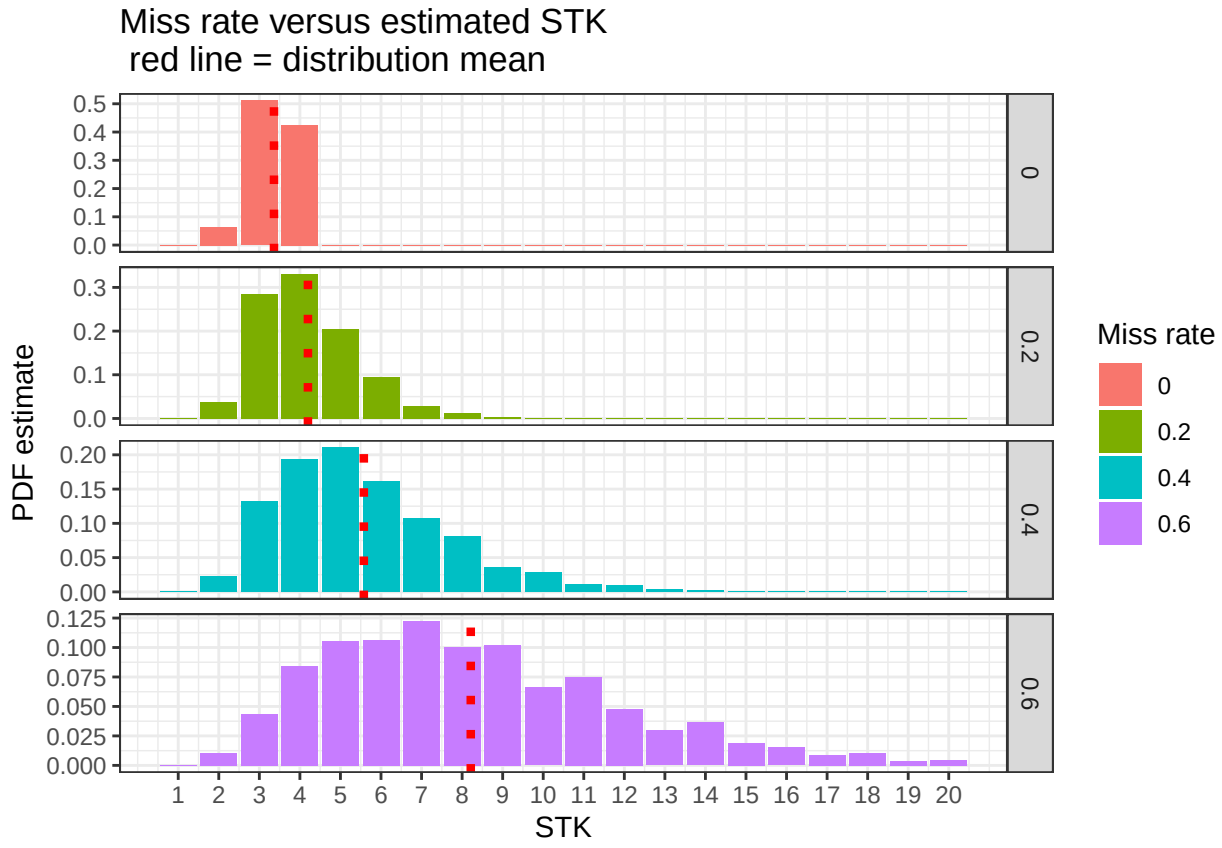
$$shot\_ratios_{new} = \{shot\_ratios_{orig} * (1 - miss\_rate),\ miss\_rate\}$$

This approach combined with using multinomial generative functions is the recommended implementation for high bullet calculations such as expected damage per mag.

4.4 Example: Generative Multinomial function to compare TTK across various miss rates

```
# Write function to simulate samples for X bullets
multinom_gun_sim <- function(X,
                             n_samples = 10000,
                             shot_probs = c(0.25,0.75),
                             damage = c(50, 25),
                             health = 100,
                             miss = 0){

   multinom_results <- rmultinom(n = n_samples,
                              size = X,
                              prob = c(shot_probs*(1-miss), miss))

   multinom_damages <- multinom_results*c(damage, 0)

   cdf_estimate = sum(colSums(multinom_damages) >= health)/n_samples

   return(tibble(n_bullets = X,
                 cdf_est = cdf_estimate,
                 miss_rate = miss))
}

# Use function to assess STK at different miss ratios
miss_ratios <- seq(from = 0, to = 0.6, by = 0.2)

for(i in 1:length(miss_ratios)){
  curr_probs <- sapply(X = 1:20,
                FUN = multinom_gun_sim,
                health = 100,
                damage = c(50,25),
                shot_probs = c(.25,.75),
                miss = miss_ratios[i])

  curr_probs_tibble <- tibble(n_bullets = unlist(t(curr_probs)[,1]),
                      cdf_est = unlist(t(curr_probs)[,2]),
                      miss_rate = unlist(t(curr_probs)[,3])) %>%
    mutate(pdf_est = cdf_est - lag(cdf_est, default = 0))

  if(i == 1){
    probs_tibble <- curr_probs_tibble
  }else{
    probs_tibble <- rbind(probs_tibble,
                          curr_probs_tibble)
  }
}
```

```r
probs_mean <- probs_tibble %>%
  group_by(miss_rate) %>%
  summarize(mean_stk = sum(n_bullets * pdf_est))

ggplot(probs_tibble, aes(x = n_bullets, y = pdf_est, fill = as.factor(miss_rate)))+
  geom_col()+
  geom_vline(data = probs_mean, aes(xintercept = mean_stk), linetype = 'dotted', color = 'red', size = 
  facet_grid(vars(miss_rate), scales = 'free_y')+
  theme_bw()+
  ylab('PDF estimate')+
  xlab('STK')+
  ggtitle('Miss rate versus estimated STK \n red line = distribution mean')+
  scale_x_continuous(breaks = 1:20)+
  labs(fill = 'Miss rate')
```



4.5 Open bolt delay

Open bolt delay (OBD) is the time delay between the initial trigger press and the firing of the first bullet. OBD does not change the probability distribution of STK, and for TTK calculations OBD can simply be added to the TTK.

$$TTK_{OBD} = (STK - 1) * firing\_period + OBD$$