Let's say A and B play 7 games and A wins 4 of them. ***What would be a reasonable estimate for A's win probability (over B)?*** A natural estimate would be 4/7. More generally, if A wins x games, then the estimator $\hat{p} = x/7$ is an unbiased estimator of the true win probability p. This is not difficult to show: since $x \sim \text{Binom}(7, p)$, we have

$$\mathbb{E}[x/7] = \mathbb{E}[x]/7 = (7p)/7 = p.$$

Now, let's say that A and B play a *best-of-7 series (or equivalently, first to 4 wins)*, and A wins 4 of them. It seems natural to estimate A's win probability by 4/7 again. If we define the estimator

$$\hat{p} = \frac{\#\text{ games A wins}}{\#\text{ games}},$$

does this estimator have any nice properties? It's hard to say anything about this estimator (if someone knows something, please let me know!). In fact, it's not even unbiased!

Estimator is biased: a simulation

First, let's set up two functions. The first function, `SampleSeries`, simulates a series which ends when either player 1 reaches `n_wins1` wins or when player 2 reaches `n_wins2` wins. (The true win probability for player 1 is `p`.) The second function, `EstimateWinProb`, samples `n_sim` series. For each series, it estimates player 1's win probability as in $\hat{p}$ above, then returns the mean of $\hat{p}$ over the `n_sim` simulations.

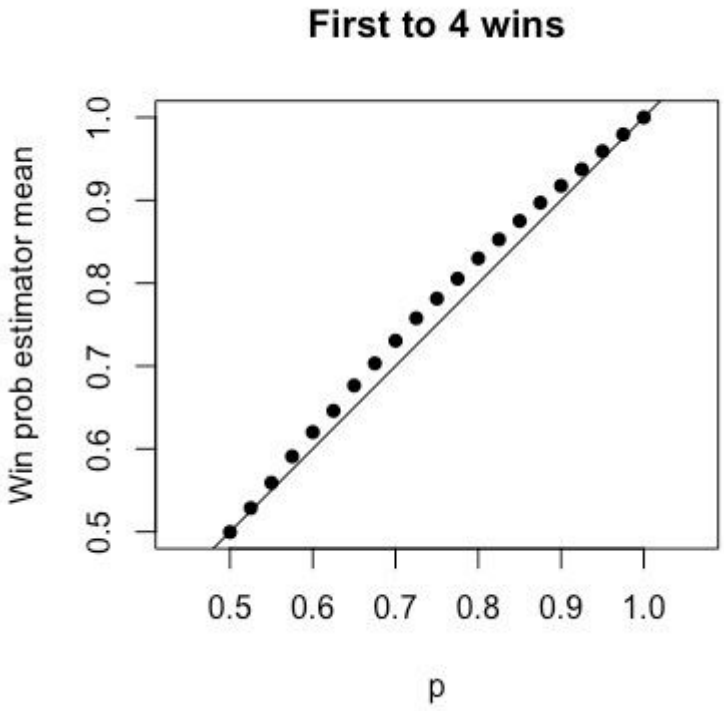
```
# Player 1 has win probability p. Simulates a series of games which
ends when
# player 1 reaches `n_wins1` wins or when player 2 reaches `n_wins2`
wins.
SampleSeries <- function(p, n_wins1, n_wins2 = n_wins1) {
  game_results <- rbinom(n_wins1 + n_wins2 - 1, size = 1, prob = p)
  if (sum(game_results) >= n_wins1) {
    n_games <- which(game_results == 1)[n_wins1]
  } else {
    n_games <- which(game_results == 0)[n_wins2]
  }
  player1_wins <- sum(game_results[1:n_games])
  return(c(player1_wins, n_games - player1_wins))
}

# Run SampleSeries `n_sim` times and for each run, estimate p by player
1's
# win percentage in that series. Returns the estimated probabilities.
EstimateWinProb <- function(p, n_sim, n_wins1, n_wins2 = n_wins1) {
  win_data <- replicate(n_sim, SampleSeries(p, n_wins1, n_wins2))
  prob_estimates <- apply(win_data, 2, function(x) x[1] / (x[1] +
x[2]))
  return(mean(prob_estimates))
}
```

Next, we run a Monte Carlo simulation (20,000 simulations) to compute the mean of $\hat{p}$ for a range of values of p. (Because of the symmetry inherent in the setup, we only consider $p \geq 0.5$

.)

```
set.seed(1)
p <- 20:40 / 40
n_wins1 <- 4
n_sim <- 20000
mean_phat <- sapply(p, function(p) EstimateWinProb(p, n_sim, n_wins1))
plot(p, mean_phat, asp = 1, pch = 16, ylab = "Win prob estimator mean",
     main = "First to 4 wins")
abline(0, 1)
```



If $\hat{p}$ were unbiased, we would expect the black dots to lie on the black line $y = x$; it's clear that they don't. In this setting, $\hat{p}$ is biased upwards.

Estimator is biased: exact computation

I was not able to find a nice closed-form expression for $\mathbb{E}[\hat{p}]$: for a best-of-7 series, WolframAlpha "simplified" my expression to give

$$\mathbb{E}[\hat{p}] = \frac{4x}{5} + \frac{2x^2}{15} + \frac{4x^3}{105} + \frac{101x^4}{21} - \frac{1252x^5}{105} + 10x^6 - \frac{20x^7}{7}.$$

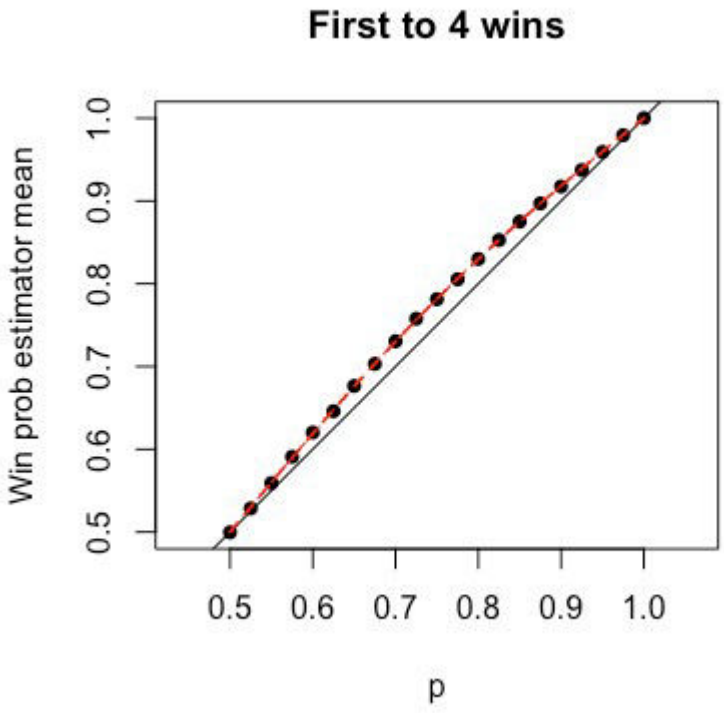
However, it is possible to write a function that computes $\mathbb{E}[\hat{p}]$ from first principles:

```
# Return expected value of the "series win percentage" estimator
GetEstimatorMean <- function(p, n_wins1, n_wins2 = n_wins1) {
  # first line is for when player 1 wins, second line is for when
player 2 wins
  summands1 <- sapply(0:(n_wins2 - 1),
                      function(x) n_wins1 / (n_wins1 + x) * p^n_wins1 *
(1 - p)^x *
                        choose(x + n_wins1 - 1, x)
```

```
  )
  summands2 <- sapply(0:(n_wins1 - 1),
                      function(x) x / (x + n_wins2) * p^x * (1 -
p)^n_wins2 *
                        choose(x + n_wins2 - 1, x)
  )
  return(sum(c(summands1, summands2)))
}
```

Let's plot these exact values on the previous plot to make sure we got it right:

```
lines(p, sapply(p, function(x) GetEstimatorMean(x, n_wins1)),
      col = "red", lty = 2, lwd = 1.5)
```



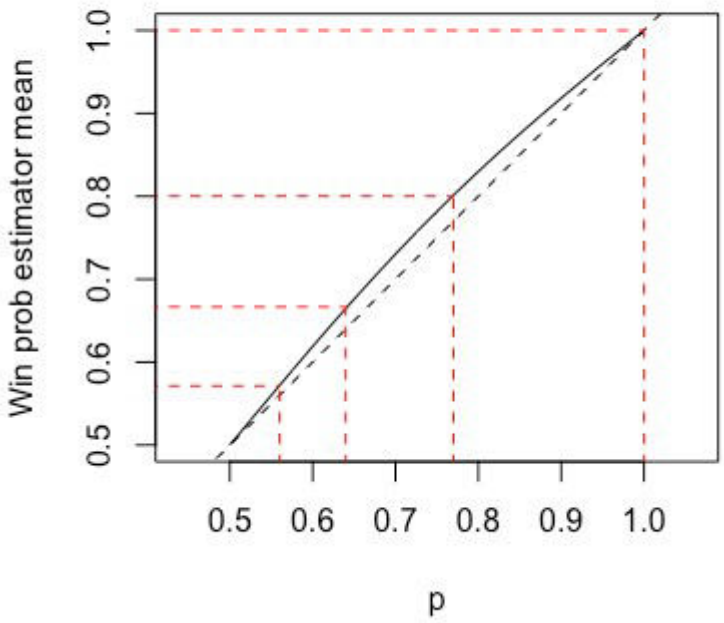
What might be a better estimator?

Let's go back to the setup where A and B play a *best-of-7 series (or equivalently, first to 4 wins)*, and A wins 4 of them. Instead of estimating A's win probability by 4/7, we could use the plot above to find which value of p would have given $\mathbb{E}[\hat{p}] = 4/7$, and use that instead. This is essentially a ***method of moments estimator*** with a single data point, and so has some nice properties (well, at least it's consistent).

The picture below depicts this method for obtaining estimates for when A wins 4 out of 4, 5, 6 and 7 games.

```
p <- 20:40 / 40
original_estimate <- c(4 / 7:4)
mm_estimate <- c(0.56, 0.64, 0.77, 1)  # by trial and error
plot(p, sapply(p, function(x) GetEstimatorMean(x, n_wins1 = 4)), type =
"l",
     asp = 1, ylab = "Win prob estimator mean")
abline(0, 1, lty = 2)
```

```
segments(x0 = 0, x1 = mm_estimate, y0 = original_estimate, col = "red",
lty = 2)
segments(y0 = 0, y1 = original_estimate, x0 = mm_estimate, col = "red",
lty = 2)
```

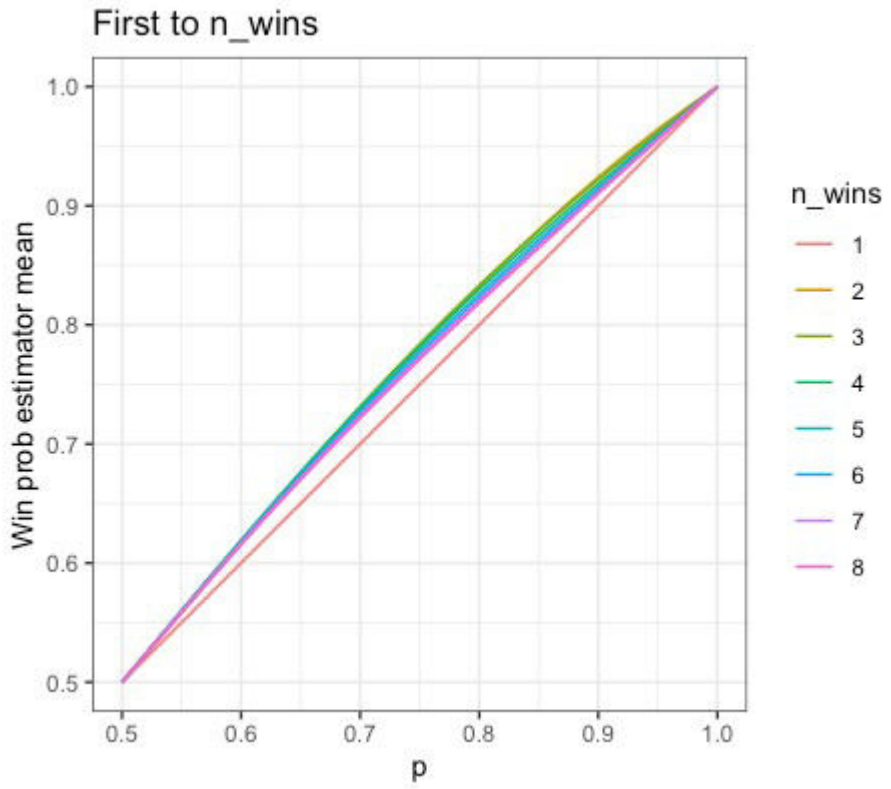


What are some other possible estimators in this setting?

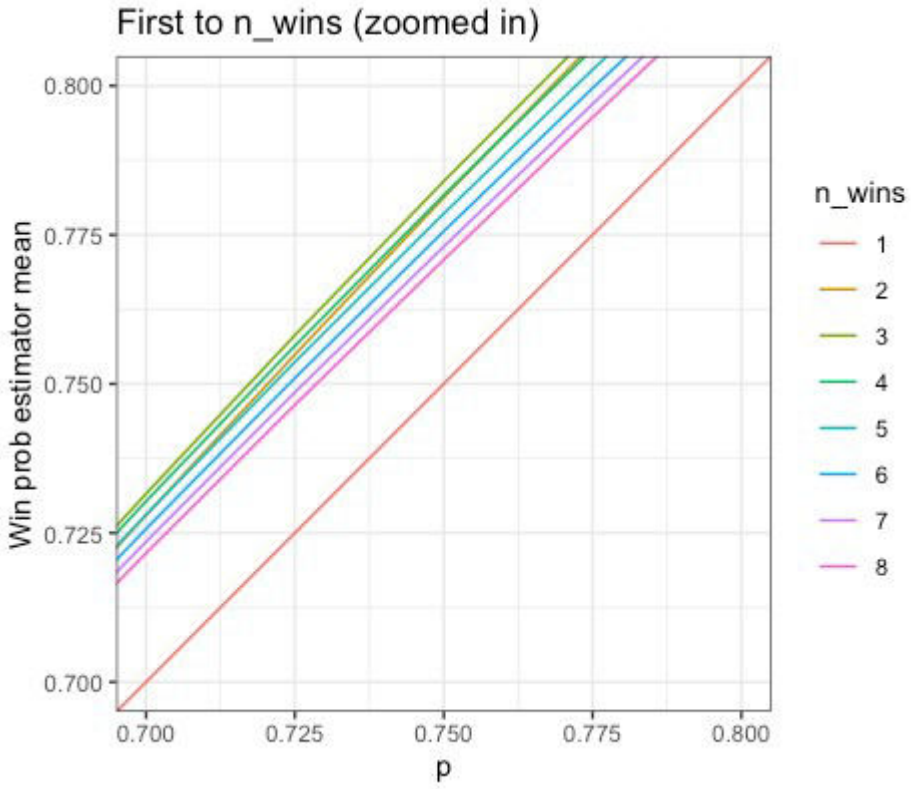
First to *n* wins: generalizing the above

Since we've coded up all this already, why look at just first to 4 wins? Let's see what happens to the estimator $\hat{p}$ when we vary the number of wins the players need in order to win the series. The code below makes a plot showing how $\mathbb{E}[\hat{p}]$ varies as we change the number of wins needed from 1 to 8 (inclusive).

```
library(tidyverse)
theme_set(theme_bw())
df <- expand.grid(p = 20:40 / 40, n_wins1 = 1:8)
df$mean_phat <- apply(df, 1, function(x) GetEstimatorMean(x[1], x[2]))
ggplot(df, aes(group = factor(n_wins1))) +
  geom_line(aes(x = p, y = mean_phat, col = factor(n_wins1))) +
  coord_fixed(ratio = 1) +
  scale_color_discrete(name = "n_wins") +
  labs(y = "Win prob estimator mean", title = "First to n_wins")
```



Past some point, the bias decreases as `n_wins1` increases. Peak bias seems to be around `n_wins1 = 3`. The picture below zooms in on a small portion of the picture above:



Asymmetric series: different win criteria for the players

Notice that in the code above, we maintained separate parameters for player 1 and player 2 (`n_wins1` and `n_wins2`). This allows us to generalize even further, to explore what happens when players 1 and 2 need a different number of wins in order to win the series.

The code below shows how $\mathbb{E}[\hat{p}]$ varies with p for different combinations of `n_wins1` and `n_wins2`. Each line within a facet corresponds to one value of `n_wins1`, while each facet of the plot corresponds to one value of `n_wins2`. The size of $\hat{p}$'s bias increases as the difference between `n_wins1` and `n_wins2` increases.

```
df <- expand.grid(p = 20:40 / 40, n_wins1 = 1:8, n_wins2 = 1:8)
df$mean_phat <- apply(df, 1, function(x) GetEstimatorMean(x[1], x[2],
x[3]))
ggplot(df, aes(group = factor(n_wins1))) +
  geom_line(aes(x = p, y = mean_phat, col = factor(n_wins1))) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed") +
  coord_fixed(ratio = 1) +
  scale_color_discrete(name = "n_wins1") +
  facet_wrap(~ n_wins2, ncol = 4, labeller = label_both) +
  labs(y = "Win prob estimator mean", title = "Player 1 to n_wins1 or
Player 2 to n_wins2")
```