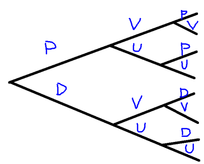
# Problem Set 1: Supplementary Thoughts

Michael Chirico

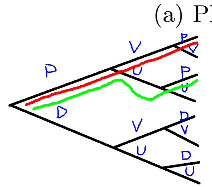
September 6, 2016

## 1 (f)

Perhaps a graph can help illustrate this problem:



(a) Plain Tree



(b) Tree with Valid Paths

Figure 1: Tree Representation of Tournament

To get the probability of a valid path, multiply all the probabilities of each branch along it. The total probability is the sum of valid path probabilities.

## 2 (a)

It's technically correct and perhaps easier to default to using 7-game series and simply count the number of combinations where Penn wins *at least* four games, but you must make the connection to why this approach is valid.

Anyway, to show the power of R, here's code to generate all the combinations and find the exact probability:

```
library(data.table)
TF <- c(TRUE, FALSE)
DF <-
  as.matrix(expand.grid(TF, TF, TF, TF, TF, TF, TF))
```

```r
series <-
  unique(as.data.table(t(apply(DF, 1L, function(x) {
    wins <- cumsum(x)
    last.game <- which(wins == 4)
    if (length(last.game)) {
      last.game <- last.game[1L]
      if (last.game < 7) x[(last.game + 1):7] <- NA
    }
    loss <- cumsum(!x)
    last.game <- which(loss == 4)
    if (length(last.game)) {
      last.game <- last.game[1L]
      if (last.game < 7) x[(last.game + 1):7] <- NA
    }
    x}))))
series[ , wins := rowSums(.SD, na.rm = TRUE)]
series[ , games := 7L - rowSums(is.na(.SD))]
series[ , prob := (.55)^wins * (.45)^(games - wins)]
series[wins == 4, sum(prob)]
```

It's much simpler in code to over-count:

```r
sum(sapply(4:7, choose, n = 7) * .55^(4:7) * .45^(3:0))
```

We could have also gotten an approximation by simulation:

```r
mean(replicate(1e6, sum(runif(7) <= .55) >= 4))
```

This runs on my machine in about 3 seconds and took about 30 seconds of coding, and gets the answer to 4 decimal places! Hurrah for simulation.

# 1 2 (c)

Here's barebones R code:

```r
# values for x-axis
p.seq <- seq(0, 1, length.out = 1000L)

p.win <- function(p)
  sum(sapply(4:7, choose, n = 7) *
        p^(4:7) * (1-p)^(3:0))

#approach 1: plot + lines
png("ps1_fig03.png")
plot(p.seq, p.seq^4, type = "l")
lines(p.seq, sapply(p.seq, p.win))
dev.off()
```

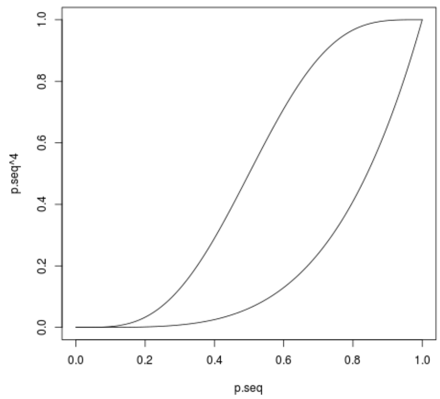



Figure 2: Minimal Plot for 2(c)

Producing this minimalist plot:
Bells and whistles bring this to being publication quality:

```
png("ps1_fig04.png")
plot(p.seq, p.seq^4, type = "l",
     main =
       paste0("Tournament Winning Probability by\n",
              "Underlying Advantage & ",
              "Tourney Structure"),
     xlab = "Single Game Win Probability",
     ylab = "Tournament Win Probability",
     #penn official colors!
     las = 1L, asp = 1L, col = "#A90533", lwd = 3L)
lines(p.seq, sapply(p.seq, p.win), lwd = 3L,
      col = "#004785")
legend("topleft",
       legend = c("Best-of-7", "Single-Elimination"),
       col = c("#004785", "#A90533"), lwd = 3L)
dev.off()
```

Many other ways to do this, here are two:

```
#approach 2: matplot
png("ps1_fig05.png")
matplot(p.seq, cbind(sapply(p.seq, p.win), p.seq^4),
        type = "l", col = c("#004785", "#A90533"),
        lty = 1L, lwd = 3L, asp = 1L, las = 1L,
        xlab = "Single Game Win Probability",
        ylab = "Tournament Win Probability",
        main =
```

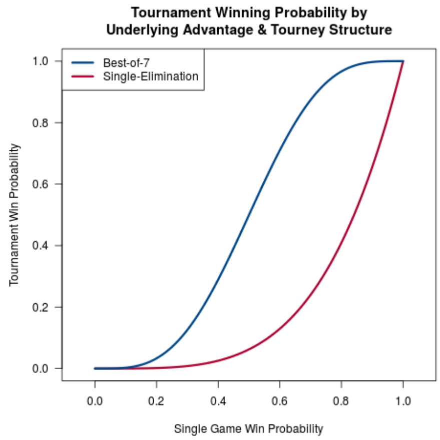



Figure 3: Minimal Plot for 2(c)

```
          paste0("Tournament Winning Probability by\n",
                 "Underlying Advantage & ",
                 "Tourney Structure"))
legend("topleft",
       legend = c("Best-of-7", "Single-Elimination"),
       col = c("#004785", "#A90533"), lwd = 3L)
dev.off()

#approach 3: ggplot
#  warning: I don't really know anything about ggplot2.
#  But it's the plotting device of choice for many
#  data folks, including over at fivethirtyeight
library(ggplot2)
png("ps1_fig06.png")
ggplot() +
  geom_point(aes(p.seq, p.seq^4), color = "#A90533") +
  geom_point(aes(p.seq, sapply(p.seq, p.win)),
             color = "#004785") +
  coord_fixed() +
  xlab("Single Game Win Probability") +
  ylab("Tournament Win Probability") +
  ggtitle(paste0(
    "Tournament Winning Probability by\n",
    "Underlying Advantage & Tourney Structure"
  ))
dev.off()
```

## 3

Simulating the problem in R

```
doors = 1:4
games <-
  #depending on your computer, this may take
  #  a while... maybe go with 1e5 if impatient.
  replicate(1e6, {
    actual.door = sample(doors, 1)
    chosen.door = sample(doors, 1)
    correct.first = actual.door == chosen.door
    exposed.door =
      sample(setdiff(doors,
                     c(actual.door, chosen.door)), 1)
    switch.doors =
      setdiff(doors, c(chosen.door, exposed.door))
    switch.to = sample(switch.doors, 1)
    correct.switched = switch.to == actual.door
    c(correct.first, correct.switched)})

#first row is a), second row is b)
rowMeans(games)
```

## 4

Especially useful to draw trees for problems like this.

Here is the main tree for all but part (d):

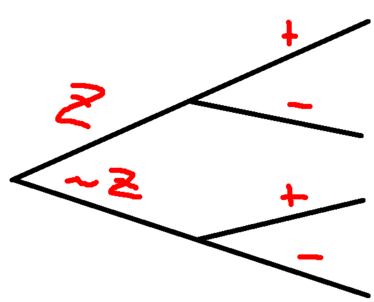



Figure 4: Minimal Plot for 2(c)

In part (d), we would extend this by one more level.

As always, we can just simulate answers in R to know if we're on the right track:

```r
population <- data.table(indiv = runif(1e6))

population[ , zika := indiv > .98]

#test results depend on zika status
population[(zika), paste0("test", 1:2) :=
              .(runif(.N) > .05, runif(.N) > .05)]

population[(!zika), paste0("test", 1:2) :=
              .(runif(.N) > .97, runif(.N) > .97)]

#part a)
population[ , mean(test1)]

#part b)
population[(test1), mean(zika)]

#part c)
population[(!test1), mean(zika)]

#part d)
population[test1 & test2, mean(zika)]
```