MT5763: Software for Data Analysis Assignment 2

Alex Ross

2022-10-29

GitHub repository: https://github.com/Magnifico1/MT5763_2_180008620

Problem A

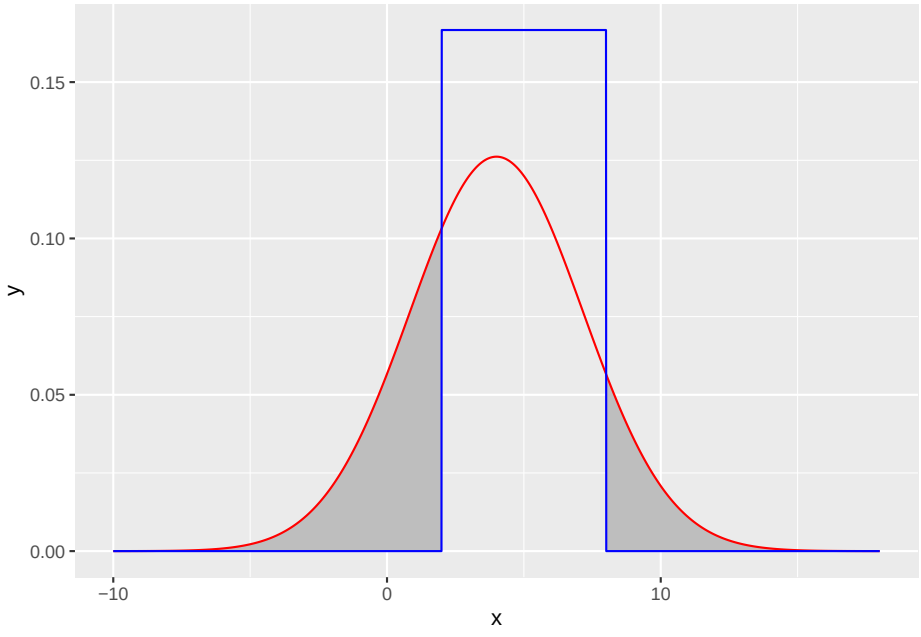
We consider the following independent random variables:

$X \sim \mathsf{Normal}(\mu = 4, \sigma^2 = 10)$

$Y \sim \mathsf{Uniform}(a = 2, b = 8)$.

When computing the probability $\Pr(X > Y)$, it is first helpful to have a visualisation of the two distributions. The distributions were overlaid in one plot in order to have some idea of the eventual expected probability.

```
library(ggplot2)
x=seq(-10,18,by=0.01)
dists<-data.frame(x=x,X=dnorm(x,mean=4,sd=sqrt(10)),Y=dunif(x,min=2,max=8))
data1<-dists[dists$x<2,]
data2<-dists[dists$x>8,]
ggplot(dists)+geom_ribbon(data=data1,aes(x=x,ymin=Y,ymax=X), fill="grey")+
  geom_ribbon(data=data2,aes(x=x,ymin=Y,ymax=X), fill="grey")+
  geom_line(aes(x=x,y=X),colour="red")+
  geom_line(aes(x=x,y=Y),colour="blue")+ylab("y")
```



The above graph shows the two distributions - X in red and Y in blue. The shaded region corresponds to the region in which the value of X is greater than Y. It can be seen that the exact value of $\Pr(X > Y)$ can therefore be calculated as follows:

```
P=pnorm(2,4,sqrt(10))+pnorm(8,4,sqrt(10),lower.tail=FALSE)
P
```

```
## [1] 0.3664962
```

In order to estimate this probability with Monte Carlo simulation, we first generated a number of random deviates from both distributions across a specified range of x values. This range of x-values was chosen with the mean of the X distribution as the mean and the bounds were chosen to include the vast majority of the X distribution; by looking at the plot of the distributions, it was noted that the range $x = (-10, 18)$ would achieve this. We then calculated the differences between the X and Y values and finally the proportion of the differences that were greater than zero (and thus the proportion of the X deviates greater than the Y deviates). This was achieved in R by the following:

```
#set seed for reproducibility
set.seed(2345,kind = "L'Ecuyer-CMRG")
#define range of x-values for which deviates will ge generated
xsize<-seq(-10,18,length=10000)
#define function
prob_func<-function(xsize){
  norm<-rnorm(xsize,mean=4,sd=sqrt(10))
  unif<-runif(xsize,min=2,max=8)
  diff<-norm-unif
  scs<-subset(norm,diff>0)
  p<-length(scs)/length(xsize)
  return(p)
}
p<-prob_func(xsize=xsize)
cat(p)
```

```
## 0.3854
```

Our estimate of the probability based on 10,000 random deviates from both distributions comes out as 0.386, which is very close to the true value of 0.366.

We next needed to find the sampling distribution of this estimate based on a number of bootstrapped samples. To achieve this, it was necessary to create an observed dataset of the normal and uniform random deviates which we would be able to resample from with replacement, as is the method of bootstrapping. We could then loop over a chosen number of iterations, each of whose $\Pr(X > Y)$ could be calculated. This resulted in a vector of probabilities which were visualised with a histogram to show the sampling distribution of the estimate and the mean of the bootstrapped probabilities dashed in red.

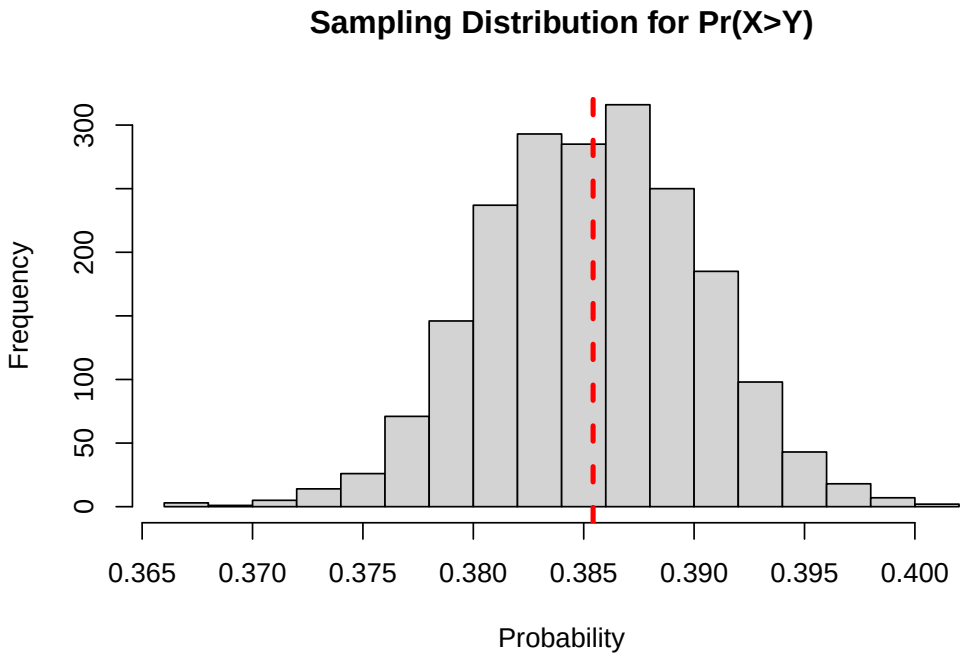
```
library(foreach)
set.seed(2345,kind = "L'Ecuyer-CMRG")
# no. of bootstrapped samples
NRepeat <- 2000
#distributions
norm<-rnorm(xsize,mean=4,sd=sqrt(10))
unif<-runif(xsize,min=2,max=8)
#observed data to be resampled
```

```
obsData <- data.frame(norm, unif)
newp<-c()
# loop across all samples
library(doParallel)
#create cluster
cl <- parallel::makeCluster(spec = 4/4, type= "PSOCK")
registerDoParallel(cl)

newp<- foreach(i=1:NRepeat, .combine = "c")%dopar%{
  # resample with replacement
  bootData <- obsData[sample(x = length(xsize), size = length(xsize), replace = T), ]

  # calculate probability in this alternative sample
  newdiff<-bootData[,1]-bootData[,2]
  newscs<-subset(newdiff,newdiff>0)
  length(newscs)/length(xsize)}

#visualisation
hist(newp, main = "Sampling Distribution for Pr(X>Y)",xlab="Probability",breaks=15)
abline(v = mean(newp), lty=2,lwd = 3, col="red")
```




```
cat(mean(newp))
```

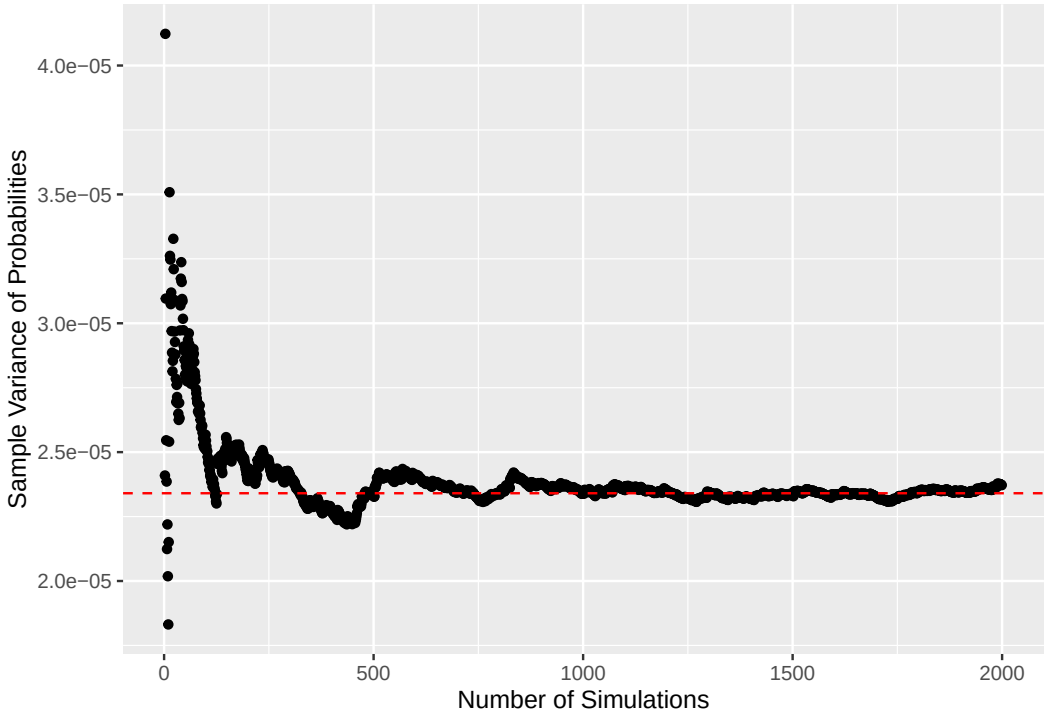
```
## 0.3854245
```

The distribution clearly shows a characteristic Normal shape as we would expect as a result of the Central Limit Theorem. The mean of the distribution is displayed above, and is quite close to the true value of 0.366.

Finally, we investigate how the variance of the sampling distribution changes as a function of the number of Monte Carlo simulations. When the number of simulations is low, the number of probabilities is also low, so we would expect the variance to be most changeable in this region. As the number of simulations increases, each subsequent addition of a simulation has a decreasing effect on the overall variance of the sampling distribution, so we expect the variance to converge on some value.

The following R code achieves the calculation of the variance for increasing numbers of simulations and plots the result.

```
#define vector
variance <- foreach(i=2:length(newp),.combine = "c")%dopar%{
  var(newp[1:(i+1)])
}
result<-data.frame(vars=variance,n_sim=seq(2,2000))
#plot variances
ggplot(result)+geom_point(aes(x=n_sim,y=vars))+xlab("Number of Simulations")+
  ylab("Sample Variance of Probabilities")+
  geom_hline(yintercept=mean(result$vars[1000:1998]), linetype="dashed", color = "red")
```




```
cat(mean(result$vars[1000:1998]))
```

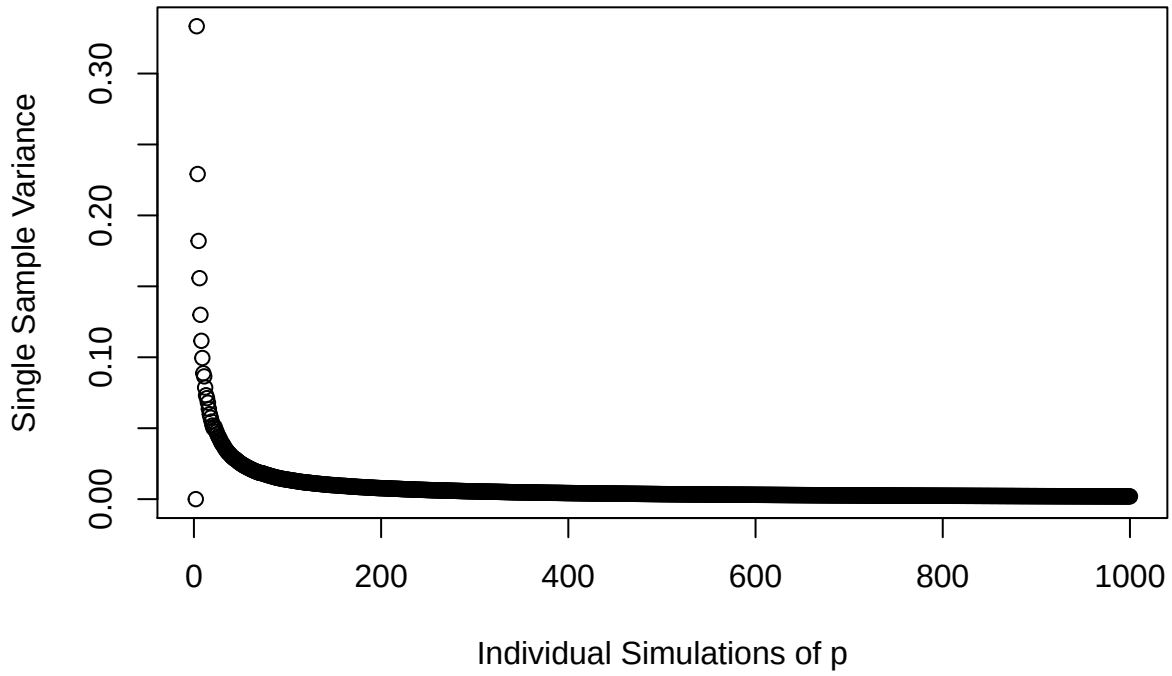
```
## 2.340456e-05
```

The resulting plot shows the variance climbing as the number of simulations increases from 0. It then begins to stabilise, and after about 750 simulations, the variance appears to be largely constant, converging on the value printed above. This is in line with our expectation.

Another interpretation of this question is to investigate how the variance of the probabilities changes as the number of deviates in a single sample, i.e. without bootstrapping increases. In this case, we would expect the variance to start at some relatively high value (since the variance of a small number of generated deviates

will be greater than fora larger sample) and subsequently decline as the number of deviates from which the value of p is calculated increases. To visualise this relationship, we calculated the variances of an increasing number of probabilities, themselves generated from an increasing number of deviates, and plotted them.

```
#generate values of p from samples with increasing number of deviates
p<-foreach(i=1:1000,.combine="c")%dopar%{
    norm<-rnorm(i,mean=4,sd=sqrt(10))
    unif<-runif(i,min=2,max=8)
    diff<-norm-unif
    scs<-subset(diff,diff>0)
    length(scs)/i}
#define variances
varp<-foreach(i=1:1000-1,.combine = "c")%dopar%{
  varp<-var(p[1:(i+1)])
}
#plot variances
plot(x=seq(1:1000),y=varp,xlab="Individual Simulations of p",
     ylab="Single Sample Variance")
```



This confirms our expectation that the variance would continue to decline from its starting value, which represents the variance of only the first two probabilities, and have a increasing shallow curve as the number of simulations increases. This can be explained by the fact that the addition of one more deviate to the sample from which p is calculate has a large effect on p when the number of deviates in the sample is small, but has an increasingly small effect as the number of deviates in the sample rises.

Problem B

We consider the following football tournament format: a team keeps playing until they accrue 7 wins or 3 losses (whichever comes first - no draws allowed). Assume a fixed win rate $p \in [0, 1]$ across all rounds.

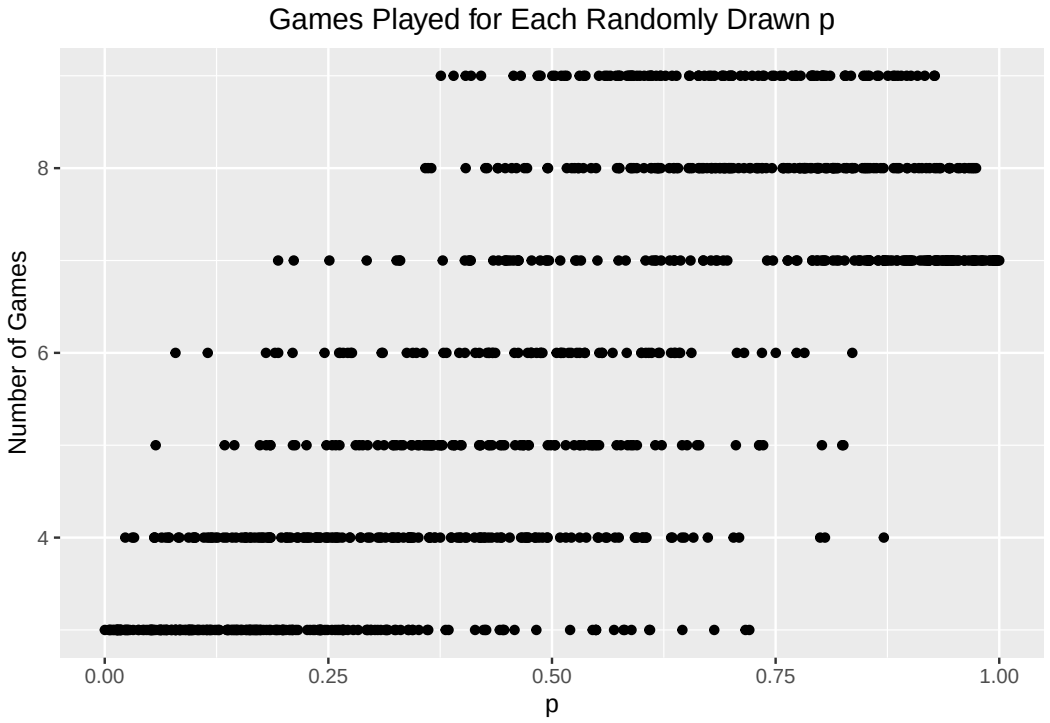
From this information, it can be observed that the minimum of the total number of games played by any team is three (three consecutive losses) and that the maximum is nine (seven total wins and two losses, with the final round resulting in a win). To ascertain the relationship between win probability p and number of games played, we first defined a function which took win p as its input and returned the simulated tournament results.

```
tourn_func<-function(winp){
  #loss probability
  lossp=1-winp
  #possible outcomes
  outcome<-c("win","loss")
  #define empty vector
  tournres<-c()
  #loop over number of possible rounds
  for(i in 1:9){
    matchres<-sample(outcome,1,replace=T,prob=c(winp,lossp))
    tournres<-c(tournres,matchres)
    if(length(tournres[tournres=="loss"])==3){
      break}
    if(length(tournres[tournres=="win"])==7){
      break}
    }
  #total number of games played
  n_games<-length(tournres)
  #winrate
  winrate<-length(tournres[tournres=="win"])/length(tournres)
  #dataframe of results
  statdf<-data.frame(n=n_games,winrate=winrate)
}
```

Now we wanted to simulate the tournament results for a large number of p values in the specified range $[0, 1]$ to visualise the pattern between p and number of games played. One thousand deviates from a uniform distribution in the specified range were generated and passed into the aforedescribed function, resulting in a thousand data points which could be plotted.

```
#number of deviates
np=1000
#for reproducibility
set.seed(30)
#vector of deviates from uniform distribution
pvec<-sort(runif(np,min=0,max=1))
#define empty vectors for the outcome statistics
nvec<-c()
winrvec<-c()
#loop over all deviates
for(i in seq(np)){
  tourn<-tourn_func(pvec[i])
  nvec<-c(nvec,tourn$n)
  winrvec<-c(winrvec,tourn$winrate)
```

```
}
#collect results in a dataframe
tourndf<-data.frame(p=pvec,n_games=nvec,obs_winrate=winrvec)
#plot number of games against p
ggplot(tourndf)+
  geom_point(aes(x=pvec,y=nvec))+
  xlab("p")+ylab("Number of Games")+
  ggtitle("Games Played for Each Randomly Drawn p")+
  theme(plot.title = element_text(hjust = 0.5))
```



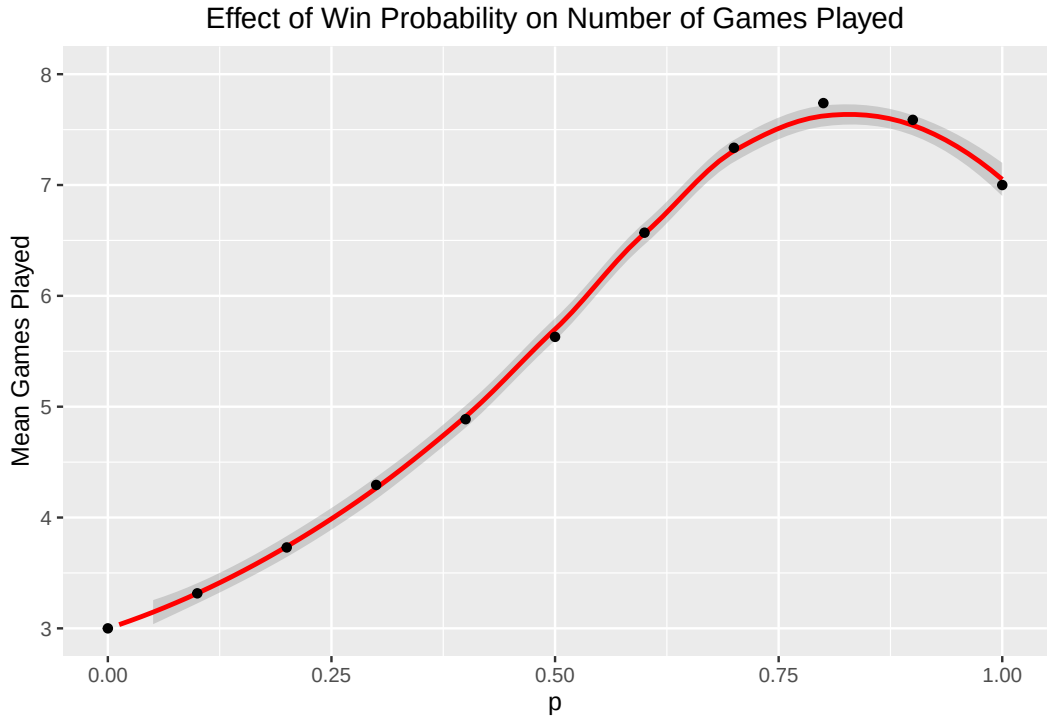
From this preliminary plot, we see a general increases in the number of games played as the win probability increases from 0 to 1. However, at very large p values, there a few tournament outcomes in which the number of games played is other than seven. Thus, the average seems to drop off towards the upper bound of p. To determine the linear relationship between p and games played, a sequence of equidistant p values were chosen and, for each value, a thousand simulations were done. The mean number of games was then calculated for each p and plotted.

```
#number of simulations for each p
prep=1000
#define vector of p
sq<-seq(0,1,by=0.1)
pvec<-rep(sq,each=prep)
#create empty vectors to store statistics
nvec<-c()
winrvec<-c()
#loop over all p
for(i in seq(length(pvec))){
  tourn<-tourn_func(pvec[i])
  nvec<-c(nvec,tourn$n)
```

```
  winrvec<-c(winrvec,tourn$winrate)
}
#collect results in dataframe
tourndf<-data.frame(p=pvec,n_games=nvec,obs_winrate=winrvec)
#calculate statistics for each different p
mean_gam<-tapply(tourndf$n_games, pvec, mean)
mean_obswin<-tapply(tourndf$obs_winrate, pvec, mean)
#collect averaged statistics in dataframe
means_res<-data.frame(p=sq,n_games=mean_gam,obs_winrate=mean_obswin,discrepancy=sq-mean_obswin)
#print dataframe
row.names(means_res)=NULL
knitr::kable(means_res[,c(1,2)])
```

p	n_games
0.0	3.000
0.1	3.316
0.2	3.730
0.3	4.294
0.4	4.887
0.5	5.630
0.6	6.570
0.7	7.336
0.8	7.739
0.9	7.588
1.0	7.000

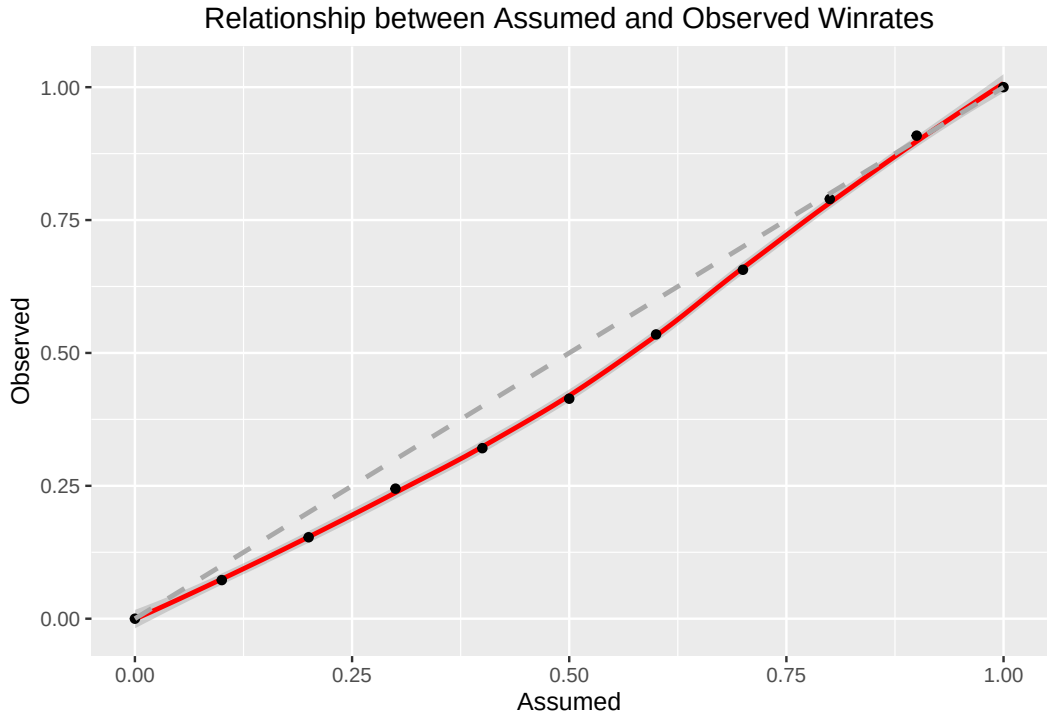
```
#plot results
ggplot(means_res)+
  geom_smooth(aes(x=p,y=n_games),colour="red",method="loess")+
  geom_point(aes(x=p,y=n_games))+
  xlab("p")+ylab("Mean Games Played")+
  scale_y_continuous(limits = c(3, 8))+
  ggtitle("Effect of Win Probability on Number of Games Played")+
  theme(plot.title = element_text(hjust = 0.5))
```



The smoothed non-parametric line shows the relationship between win probability and the mean number of games played over the thousand simulations for each p. It shows that the number of games increases with slightly increasing rate from $p = 0$ to $p = 0.5$. The slope of the curve then begins to increase with decreasing rate, however. It peaks at around $p = 0.8$ and the number of games subsequently decreases for higher values of p. This confirms our suspicion that the number of games played declined for p close to 1.

The means_res data frame also contains the mean observed win rate in addition to the mean of the total number of games. This was calculated by dividing the number of wins by the total number of games played for each simulation, and averaging over each p value. The observed win rate is plotted against the win probability, or assumed win rate p, below.

```
ggplot(means_res)+
  geom_smooth(aes(x=p,y=obs_winrate),colour="red",method="loess")+
  geom_point(aes(x=p,y=obs_winrate))+
  geom_line(aes(x=p,y=p),colour="darkgrey",size=1,linetype="dashed")+
  xlab("Assumed")+
  ylab("Observed")+
  ggtitle("Relationship between Assumed and Observed Winrates")+
  theme(plot.title = element_text(hjust = 0.5))
```




```
knitr::kable(means_res[,c(1,3,4)])
```

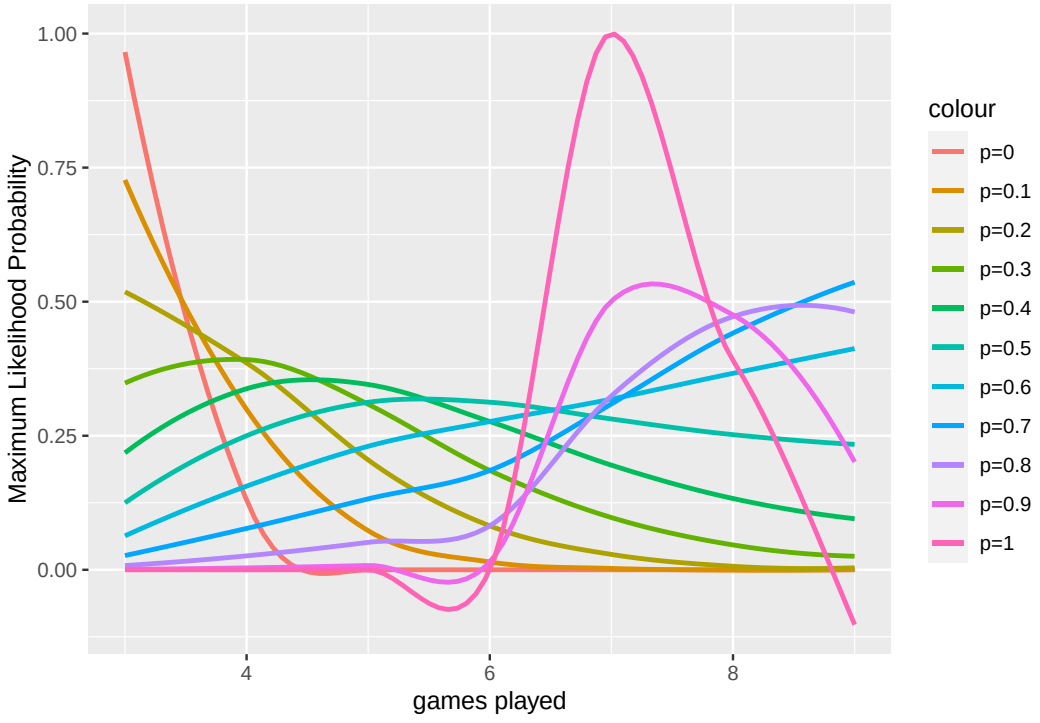
p	obs_winrate	discrepancy
0.0	0.0000000	0.0000000
0.1	0.0726464	0.0273536
0.2	0.1532393	0.0467607
0.3	0.2447143	0.0552857
0.4	0.3208452	0.0791548
0.5	0.4139004	0.0860996
0.6	0.5348377	0.0651623
0.7	0.6565004	0.0434996
0.8	0.7896679	0.0103321
0.9	0.9088488	-0.0088488
1.0	1.0000000	0.0000000

The dashed grey line represents hypothetical equality between the observed and assumed win rates. The data points, however, do not lie on this line - instead the smoothed curve stays below the line $y = x$. For lower values of the assumed win rate p, the curve has shallower slope than the line of equality, and its slope subsequently increases to be greater than that of the line of equality for $p > 0.5$ such that the two lines eventually converge. Thus it appears that the assumed win rate is always at least as large as the observed win rate.

The table above shows the discrepancies between the assumed and observed win rates for each of the p values plotted. The greatest discrepancy occurs when $p = 0.4$, giving a value of 0.0737, and the discrepancies decline either side of this assumed win rate. This can be seen to follow from the format of the tournament. The tournament stops when three losses or seven wins are acquired, thus with an assumed probability (of 0.5 for example giving easiest understanding), it is nevertheless more likely that the tournament stops after three losses, rather than seven wins, simply due to the much higher number of wins required. Therefore, the observed win rates tend to be less than the assumed ones.

To observe the most likely assumed win rate resulting in each of the possible resultant number of games, we calculated the maximum likelihood probabilities for each number of games played and plotted them.

```
no_games=c(3:9)
#function for maximum likelihood probabilities for each no. games played
ml<-function(w){
  l=1-w
  probs<-c(l^3,choose(4,3)*l^3*w,choose(5,3)*l^3*w^2,choose(6,3)*l^3*w^3,
           choose(7,3)*l^3*w^4+w^7,choose(8,3)*l^3*w^5+choose(8,1)*l*w^7,
           choose(9,3)*l^3*w^6+choose(9,2)*l^2*w^7)
  return(probs)
}
#collect maximum likelihood probabilities for each win rate
data<-data.frame(no_games=no_games,ml0=ml(0),ml.1=ml(.1),ml.2=ml(.2),ml.3=ml(.3),
                 ml.4=ml(.4),ml.5=ml(.5),ml.6=ml(.6),ml.7=ml(.7),ml.8=ml(.8),
                 ml.9=ml(.9),ml1=ml(1))
#plot games played against maximum likelihood probabilities for each assumed win rate w"
ggplot(data)+stat_smooth(aes(x=no_games,y=ml0,colour="p=0"),se=FALSE)+
     geom_smooth(aes(x=no_games,y=ml.1,colour="p=0.1"),se=FALSE)+
     geom_smooth(aes(x=no_games,y=ml.2,colour="p=0.2"),se=FALSE)+
     geom_smooth(aes(x=no_games,y=ml.3,colour="p=0.3"),se=FALSE)+
     geom_smooth(aes(x=no_games,y=ml.4,colour="p=0.4"),se=FALSE)+
     geom_smooth(aes(x=no_games,y=ml.5,colour="p=0.5"),se=FALSE)+
     geom_smooth(aes(x=no_games,y=ml.6,colour="p=0.6"),se=FALSE)+
     geom_smooth(aes(x=no_games,y=ml.7,colour="p=0.7"),se=FALSE)+
     geom_smooth(aes(x=no_games,y=ml.8,colour="p=0.8"),se=FALSE)+
     geom_smooth(aes(x=no_games,y=ml.9,colour="p=0.9"),se=FALSE)+
     geom_smooth(aes(x=no_games,y=ml1,colour="p=1"),se=FALSE)+
     xlab("games played")+ylab("Maximum Likelihood Probability")
```



Ignoring where there are lines below the x axis (probability must be positive), we see the peak of the curve representing each assumed win rate drift from x=3 for p=0, to x=7 for p=1. This explains the shape of the curve in the graph of the effect of win probability on the number of games played, as the x-value of the peak of the curves above decreases above p=0.8.