Probability Theory Shown by Simulation

This code is adapted from *Simulation for Data Science with R*.

```
# message=FALSE makes this chunk silent

# call required packages
library(tidyverse) # always
library(gridExtra) # for grid.arrange
library(car) # for prestige data
```

Weak Law of Large Numbers

First, we will develop a coin flipping simulation.

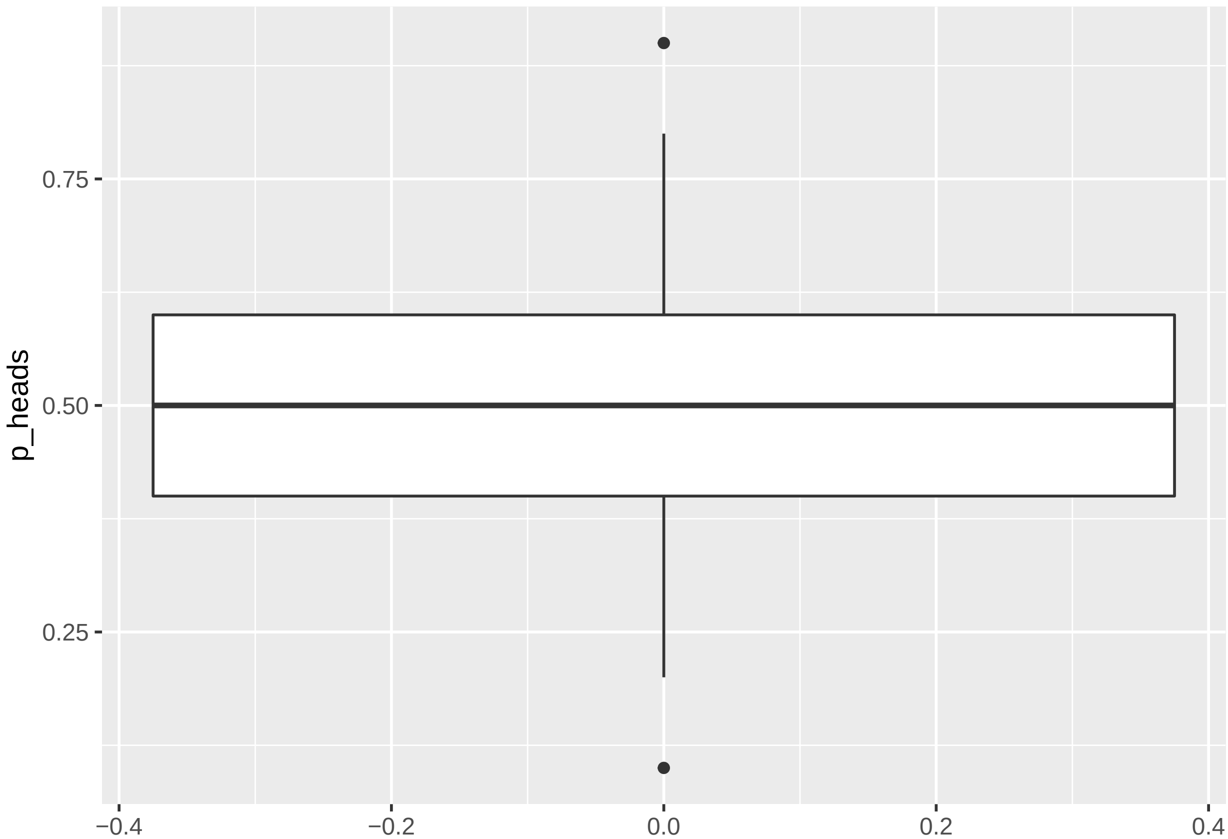
```
# set seed for reproducibility
set.seed(143)

coin_flip_function <- function(n, outcomes, p, trials) {
  # create an empty tibble
  coin_flip_sim <- tibble(trial = rep(NA, trials),
                          n_heads = rep(NA, trials),
                          p_heads = rep(NA, trials),
                          error = rep(NA, trials))
  # loop over experiment
  for (i in 1:trials){
    tosses <- rbinom(n = n, size = outcomes, prob = p)
    n_heads <- sum(tosses)
    p_heads <- n_heads/n
    error <- p_heads - p
    coin_flip_sim[i,] <- c(i, n_heads,p_heads,error)
  }
  return(coin_flip_sim)
}
```

Now we can replicate a single experiment.

```
# coin toss experiment parameters
n <- 10 # number of tosses per trial
outcomes <- 1 # number of outcomes - 1
p <- 0.5 # probability of outcome A (heads)
trials <- 100 # number of trials
coin_flip_outcomes <- coin_flip_function(n, outcomes, p, trials)

# plot outcomes
ggplot(coin_flip_outcomes, aes(y = p_heads)) +
  geom_boxplot()
```



Vary n

```
n_vary <- seq(10,100,10)
coin_flips_vary_n <- tibble(n = n_vary,
                            outcomes = map(n_vary,
                                           function(x) coin_flip_function(x,
                                                                          outcomes,
                                                                          p,
                                                                          trials)))
# unnest the tibble
coin_flips_vary_n <- coin_flips_vary_n %>%
  unnest() %>%
  mutate(n = as.factor(n))

# plot outcomes per value of paramter
ggplot(coin_flips_vary_n, aes(x = n, y = p_heads)) +
  geom_boxplot() + ggtitle("Varying Value of n") +
  xlab("Value of n") + ylab("Probability of Heads")
```

Q1. What happens as n increases?

Vary Trials

Q2. First, make a prediction. What will happen as the number of trials increases? Write down your prediction.

```
trials_vary <- seq(100,1000,100)
coin_flips_vary_trials <- tibble(trials = trials_vary,
                                 outcomes = map(trials_vary,
                                                function(x) coin_flip_function(n,
                                                                               outcomes,
                                                                               p,
                                                                               x)))
# unnest the tibble
coin_flips_vary_trials <- coin_flips_vary_trials %>%
  unnest() %>%
  mutate(trials = as.factor(trials))

# plot outcomes per value of paramter
ggplot(coin_flips_vary_trials, aes(x = trials, y = p_heads)) +
  geom_boxplot() + ggtitle("Varying Value of Trials") +
  xlab("Value of Trials") + ylab("Probability of Heads")
```

Q3. What happens as trials increases?

Central Limit Theorem

First, we need to write a function that will draw samples from distributions.

```
CLT_function <- function(n, trials) {
  # create an empty tibble
  CLT_sim <- tibble(trial = rep(NA, trials),
                    uniform_mean = rep(NA, trials),
                    normal_mean = rep(NA, trials),
                    exp_mean = rep(NA, trials),
                    beta_mean = rep(NA, trials))
  # loop over experiment
  for (i in 1:trials){
    uniform_mean <- runif(n) %>% mean()
    normal_mean <- rnorm(n) %>% mean()
    exp_mean <- rexp(n) %>% mean()
    beta_mean <- rbeta(n, shape1 = 0.35, shape2 = 0.25) %>% mean()
    CLT_sim[i,] <- c(i, uniform_mean, normal_mean, exp_mean, beta_mean)
  }
  return(CLT_sim)
}
```

Let's start by making 1 observation for each distribution with 1000 trials. Then let's experiment with increasing n.

```
n = 1
trials = 1000
CLT_outcomes <- CLT_function(n,trials)

uniform_plot <- ggplot(CLT_outcomes, aes(x = uniform_mean)) +
  geom_histogram() + ggtitle("Uniform")

normal_plot <- ggplot(CLT_outcomes, aes(x = normal_mean)) +
  geom_histogram() + ggtitle("Normal")
```

```r
exp_plot <- ggplot(CLT_outcomes, aes(x = exp_mean)) +
  geom_histogram() + ggtitle("Exponential")

beta_plot <- ggplot(CLT_outcomes, aes(x = beta_mean)) +
  geom_histogram() + ggtitle("Beta")

grid.arrange(uniform_plot, normal_plot,
             exp_plot, beta_plot,
             nrow=2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Q4. What happens as n increases?

Estimators

Here, we are going to investigate the properties of confidence intervals and the implications for interpreting p-values.

```r
# data that we are starting from
data("airquality")
hist(airquality$Wind)
```

```r
# define distribution
n_wind <- length(airquality$Wind)
mean_wind <- mean(airquality$Wind)
sd_wind <- sd(airquality$Wind)

# generate simulated wind data
trials = 100
n_obs = 100 # number of observations in each trial
Wind_outcomes <- replicate(trials, rnorm(n_obs, mean_wind, sd_wind))

# set alpha for defining confidence intervals
alpha = .01
critical_value <- -qnorm(alpha/2)
Wind_CI <- tibble(trial = seq(1,trials),
                  mean = colMeans(Wind_outcomes),
                  sd = apply(Wind_outcomes, 2, sd),
                  n = apply(Wind_outcomes, 2, length)) %>%
  mutate(se = sd/sqrt(n),
         CI_lower = mean - critical_value*se,
         CI_upper = mean + critical_value*se,
         missing_true_in_CI = ifelse(CI_lower > mean_wind,1,
                                     ifelse(CI_upper < mean_wind, 1, 0))) %>%
  mutate(missing_true_in_CI = as_factor(as.character(missing_true_in_CI)))

# plot confidence intervals
ggplot(Wind_CI, aes(x=trial, y=mean, color = missing_true_in_CI)) +
  geom_point() + geom_errorbar(aes(ymin=CI_lower, ymax=CI_upper)) +
```

```r
  geom_hline(yintercept = mean_wind) + theme_bw() +
  scale_colour_manual(values=c("black","red"))
```

Q5. What happens when you change n_obs? What about alpha?