Lab 7

Code

AUTHOR
Maxim Dokukin

Remember, **follow the instructions below and use R Markdown to create a pdf document with your code and answers to the following questions on Gradescope.** You may find a template file by clicking "Code" in the top right corner of this page.

A. Random sampling in R

1. In your own words, explain the difference between `dnorm()`, `pnorm()`, `qnorm()`, and `rnorm()`.

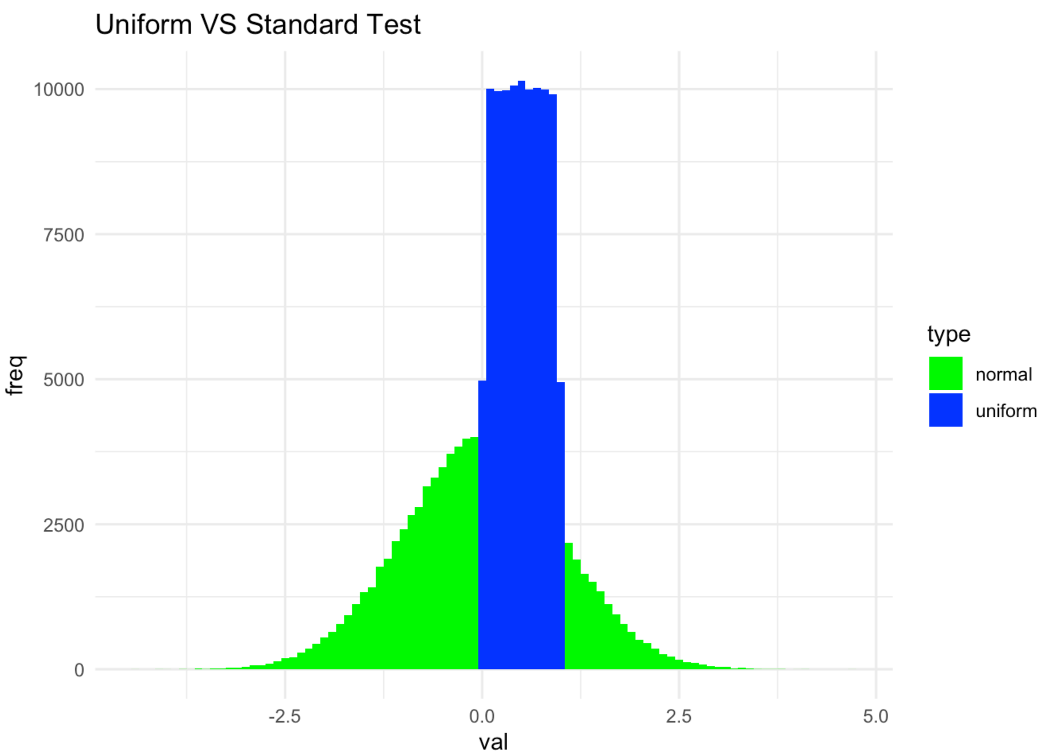
`dnorm()` - Normal distribution PDF `pnorm()` - Normal distribution CDF `qnorm()` - Normal distribution quantile `rnorm()` - Normal distribution random sample

2. Suppose we simulate `x <- runif(1)`. What is the distribution of `qnorm(x)`?

Normal? Because there is only one sample drawn by runif() its hard to identify any distribution.

```
library(ggplot2)

uniform <- runif(100000)
normal <- qnorm(uniform)

data <- data.frame(value = c(uniform, normal),
                   type = factor(rep(c("uniform", "normal"), ea

ggplot(data, aes(x = value, fill = type)) +
  geom_histogram(position = "identity", binwidth = 0.1) +
  scale_fill_manual(values = c("green", "blue")) +
  theme_minimal() +
  labs(title = "Uniform VS Standard Test",
       x = "val",
```

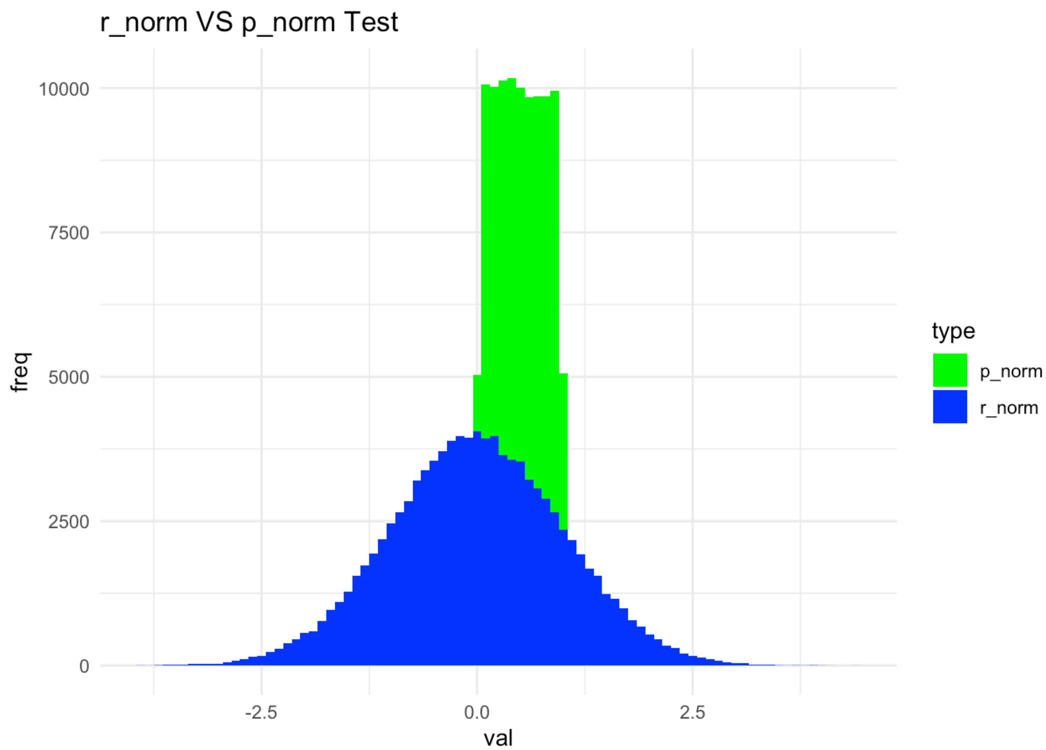
```
       y = "freq")
```



3. Suppose we simulate `x <- rnorm(1)`. What is the distribution of `pnorm(x)`?

Uniform? Because there is only one sample drawn by rnorm() its hard to identify any distribution.

```
r_norm <- rnorm(100000)
p_norm <- pnorm(r_norm)

data <- data.frame(value = c(r_norm, p_norm),
                   type = factor(rep(c("r_norm", "p_norm"), each

ggplot(data, aes(x = value, fill = type)) +
  geom_histogram(position = "identity", binwidth = 0.1) +
  scale_fill_manual(values = c("green", "blue")) +
  theme_minimal() +
  labs(title = "r_norm VS p_norm Test",
       x = "val",
       y = "freq")
```



B. Gambler's ruin

A and B are playing a coin flipping game. A starts with n_a pennies and B starts with n_b pennies. A coin is flipped repeatedly and if it comes up heads, B gives A a penny. If it comes up tails, A gives B a penny. The game ends when one player has no more pennies.

4. Write a function `run_one_sim(seed, n_a, n_b)` to simulate one game. Repeatedly use your code with different values of `seed` to estimate each player's probability of winning when $n_a = n_b = 10$.

```
run_one_sim <- function(seed, n_a, n_b){

  set.seed(seed)

  while(n_a > 0 & n_b > 0){

    if(sample(c('H', 'T'), size = 1) == 'H'){
      n_a = n_a + 1
      n_b = n_b - 1
```

```
    } else {
      n_a = n_a - 1
      n_b = n_b + 1
    }
  }

  return(ifelse(n_b == 0, 'A', 'B'))
}

runs <- 1000
results <- c()
for(i in 1:runs){

  results <- append(results, run_one_sim(i, 10, 10))
}
print(sprintf("A winrate: %.2f%%  |  B winrate: %.2f%%",
              sum(results == 'A') / runs * 100,
              sum(results == 'B') / runs * 100))
```

```
[1] "A winrate: 50.10%  |  B winrate: 49.90%"
```

5. Use your function to estimate each player's probability of winning when $n_a = 1, \ldots, 5$ and $n_b = 1, \ldots, 5$, testing every combination. Organize your results in a 5 by 5 matrix and print it out. What do you notice?

```
run_mult_sim <- function(runs, n_a, n_b){

  results <- c()
  for(i in 1:runs){

    seed <- sample(1:1000000, size = 1)
    results <- append(results, run_one_sim(seed, n_a, n_b))
  }

  return(paste('A:', sum(results == 'A') / runs * 100, '% B:', s
}


mat <- matrix(0, 5, 5)

for(i in 1:5){
```

```
  for(j in 1:5){
    mat[i, j] <- run_mult_sim(10000, i, j)
  }
}
mat
```

```
     [,1]                        [,2]                        [,3]
[1,] "A: 50.33 % B: 49.67 %" "A: 35 % B: 65 %"       "A: 24.82
% B: 75.18 %"
[2,] "A: 70.6 % B: 29.4 %"   "A: 48.64 % B: 51.36 %" "A: 42.56
% B: 57.44 %"
[3,] "A: 78.01 % B: 21.99 %" "A: 61.12 % B: 38.88 %" "A: 51.38
% B: 48.62 %"
[4,] "A: 77.89 % B: 22.11 %" "A: 66.8 % B: 33.2 %"   "A: 57.02
% B: 42.98 %"
[5,] "A: 83.66 % B: 16.34 %" "A: 71.01 % B: 28.99 %" "A: 61.96
% B: 38.04 %"
     [,4]                        [,5]
[1,] "A: 20.31 % B: 79.69 %" "A: 15.5 % B: 84.5 %"
[2,] "A: 31.12 % B: 68.88 %" "A: 26.63 % B: 73.37 %"
[3,] "A: 42.87 % B: 57.13 %" "A: 44.59 % B: 55.41 %"
[4,] "A: 51.28 % B: 48.72 %" "A: 42.13 % B: 57.87 %"
[5,] "A: 57.66 % B: 42.34 %" "A: 50.88 % B: 49.12 %"
```

The bigger the starting capital, the higher chances for player to win. On the diagonal of the matrix (same capital) odds are approx 50/50 for both.

C. One-dimensional random walks

In this part, you will simulate a one-dimensional random walk. Suppose you are at the point x at time t. At time $t+1$, the probability of moving forwards to $x+1$ is p and the chance of moving backwards to $x-1$ is $1-p$. Assume that at time $t=1$, you are at $x_1=0$.

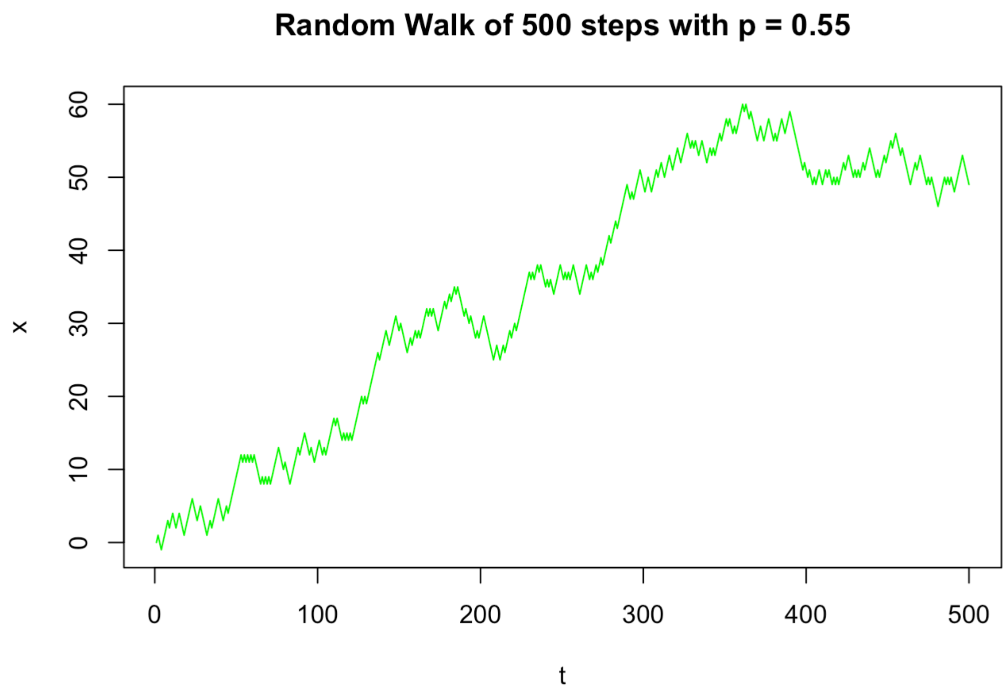
6. Write a function `random_walk()` that takes as input a numeric `n_steps` and a numeric p and simulates `n_steps` steps of the one-dimensional random walk with forward probability p. You may have other input arguments if desired. The output should be a length vector of length `n_steps` starting with 0 where the ith entry represents the location of the random walker at time $t=i$. For example, `random_walk(5, .5)` may return the vector $(0, 1, 2, 1, 2)$.

```r
random_walk <- function(n_steps, p){

  output <- c(0)
  x_pos <- 0

  for(step in 1:(n_steps-1)){

    if(runif(1) <= p){
      x_pos = x_pos + 1
    } else {
      x_pos = x_pos - 1
    }
    output <- append(output, x_pos)
  }
  return(output)
}

random_walk(5, 0.5)
```

```
[1]  0  1  0 -1  0
```

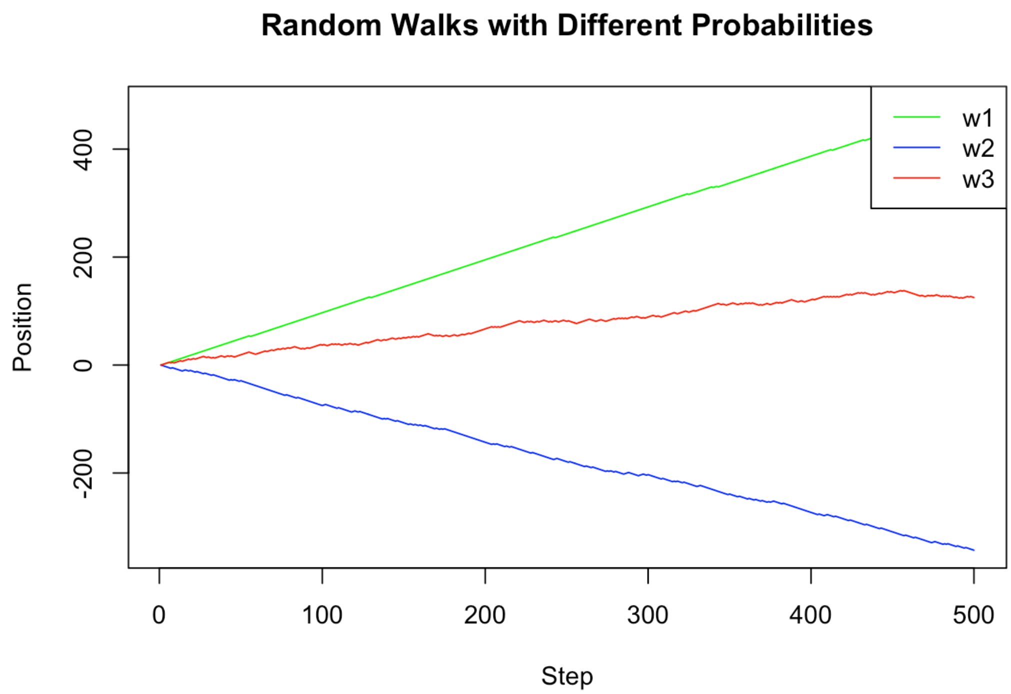
7. Use your function to generate a random walk of 500 steps with probability $.55$ and generate a line graph with $t = 1, \ldots, 500$ on the x-axis and $x_1, \ldots, x_{500}$ on the y-axis.

```r
plot(random_walk(500, 0.55),
     type = 'l',
     main = "Random Walk of 500 steps with p = 0.55",
     xlab = "t",
     ylab = "x",
     col = "green"
     )
```



8. Use your function to generate two more random walks of 500 steps with probability p, where $p \sim \mathrm{Unif}(0, 1)$ and create a line graph with all three of your random walks, using different colors for each walk.

```
alks <- data.frame(w1 = random_walk(500, runif(1)),
                   w2 = random_walk(500, runif(1)),
                   w3 = random_walk(500, runif(1))
                   )

atplot(walks,
       type = 'l',
       lty = 1,
       col = c("green", "blue", "red"),
       xlab = "Step",
       ylab = "Position",
       main = "Random Walks with Different Probabilities"
       )

egend("topright", legend = c("w1", "w2", "w3"), col = c("green",
```


Random Walks with Different Probabilities
w1
w2
w3
Position
Step
400
200
0
-200
0
100
200
300
400
500