Statistics DDD Winter 2020

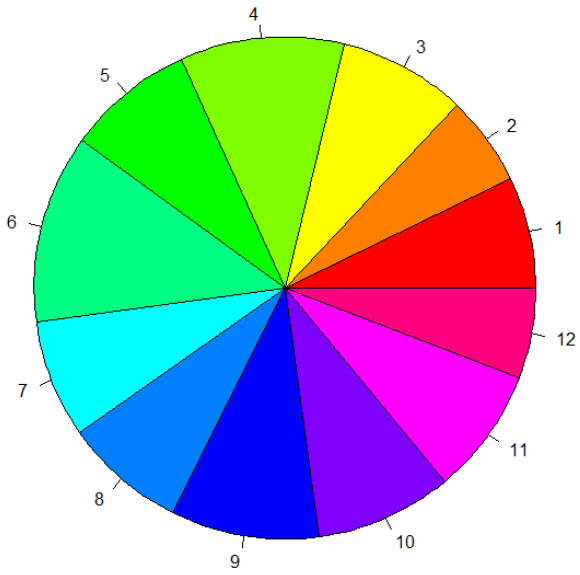
Nguyen Hoang Anh

Project Part 2

Instructor: Luiz K. Takei

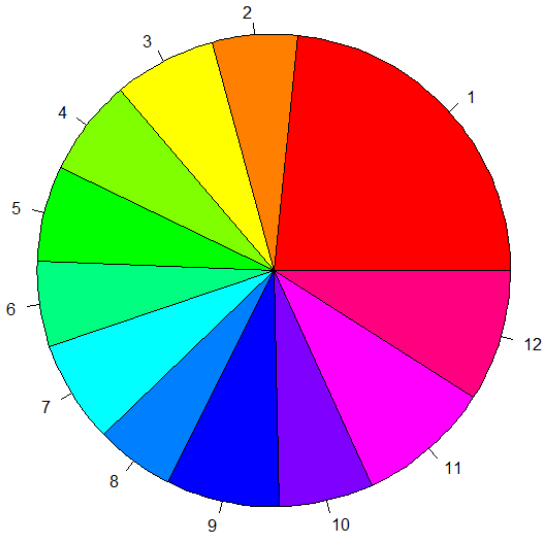
Exercise 1: Simulating Probabilities

```
# (a) Rolling a die

###  (i) Simulate rolling a fair 12-
sided die 500 times. Generate a pie chart of the results

roll_1 <- (sample(c(1:12), 500, replace=TRUE))
table_1 <- table(roll_1)
piepercent <- round(100*roll_1/sum(roll_1), 1)
pie(table_1, labels=(1:12), radius=1, col=rainbow(12))
```




Pie Chart 1ai. Rolling a fair 12-sided die 500 times (pseudo-random).

```r
### (ii) Simulate rolling an unfair 12-
sided die 500 times, where rolling a 1 is three times more likely to occur than the re
st of the outcomes. Generate a pie chart of the results.

p <- c(0.3, rep(0.1, 11))
roll_2 <- sample(c(1:12), 500, replace=TRUE, prob=p)
table_2 <- table(roll_2)
pie(table_2, labels=(1:12), radius=1, col=rainbow(12))
```




Pie Chart 1aii. Rolling an unfair 12-sided die 500 times with P(1) = 3/12.

```r
### (iii) Simulate rolling a pair of fair 12-
sided die 10000 times. From the results, estimate the probability that the sum of the
two dice is equal to 10. What is the exact theoretical probability?

roll_3_1 <- sample(c(1:12), 10000, replace=TRUE)
roll_3_2 <- sample(c(1:12), 10000, replace=TRUE)
count_sum <- 0
for (i in roll_3_1) {
    if (roll_3_1[i] + roll_3_2[i] == 10) {
        count_sum <- count_sum + 1
    }
}
cat('Estimated probability: ', count_sum/10000,'\n')
```

Estimated probability: 0.0623

Theoretical probability: 0.0625

```r
# (b) Playing the lottery

## 4 numbers
play_count <- 0
lotto_4 <- function() {
    n <- 0
    ticket <- sort(c(6,9,4,2))
    draw <- c(0,0,0,0)
    while (!all(ticket==sort(draw))) {
        n <- n+1
        draw <- sort(sample(1:49, 4, replace=FALSE))
        print(n)
    }
    return(list("play_count" = n, "ticket" = ticket))
}

draws <- lotto_4()
cat("Play counts: ", draws$play_count, "\n")
cat("Drawn ticket: ", draws$ticket, "\n")
cat("Number of years before winning: ", draws$play_count/52, '\n')
cat("Net earning: ", 16*10^6 - draws$play_count*3, '\n')
```

Play counts: 48267, Drawn ticket: 2 4 6 9 (for debugging), Number of years before winning: 928.2115, Net earning: 15855199.

```r
## 6 numbers

play_count2 <- 0
lotto_6 <- function() {
    k <- 0
    ticket2 <- sort(c(6,9,4,2,24,7))
    draw2 <- c(0,0,0,0,0,0)
    while (!all(ticket2==sort(draw2))) {
        k <- k+1
        draw2 <- sort(sample(1:49, 6, replace=FALSE))
        print(k)
    }
    return(list("play_count" = k, "ticket" = ticket2))
}

draws2 <- lotto_6()
cat("Play counts: ", draws2$play_count2, "\n")
cat("Drawn ticket: ", draws2$ticket2, "\n")
cat("Number of years before winning: ", draws2$play_count2/52, '\n')
cat("Net earning: ", 16*10^6 - draws2$play_count2*3, '\n')
```

Run 1:

Play counts: 715973, Drawn ticket: 2 4 6 9 7 24, Number of years before winning: 13768.71, Net earning: 13852081.

Run 2:

Play counts: 1674213, Drawn ticket: (same), Number of years before winning: 32196.4, Net earning: 10977361.

Exercise 2: Moran Process

```
# Moran process

n <- 1000

simulate_moran <- function(n) {
    population <- c(rep(0, n-1), 1)
    # print(population) # for debugging only
    k <- 0
    while (!(all(population == c(rep(0, n))) | all(population == c(rep(1, n))))) { # r
un/stop logic
        k <- k+1

        # replace that individual with another randomly selected one with respect to I
NDEX LOCATION
        population[sample(c(1:n), 1, replace=FALSE)] <- population[sample(c(1:n), 1, r
eplace=FALSE)]

        # print(population) # for debugging only
    }
    return(list("generations" = k, "dominant" = population[1]))
}

dom0 <- 0
dom1 <- 0
gen0 <- 0
gen1 <- 0

for (i in 1:1000) {
    print(i) # print i-th simulation
    moran <- simulate_moran(n)
    if (moran$dominant == 0) {
        dom0 <- dom0 + 1 # number of simulations ending up in type 0
        gen0 <- gen0 + moran$generations
```

```
    }
    else if (moran$dominant == 1) {
        dom1 <- dom1 + 1 # number of simulations ending up in type 1
        gen1 <- gen1 + moran$generations # sums the generations
    }
}

cat("Type 0 dominance: ", dom0, "\n")
cat("Type 0 average gen: ", gen0/dom0, "\n")

cat("Type 1 dominance: ", dom1, "\n")
cat("Type 1 average gen: ", gen1/dom1, "\n")
```

Results: (parameters: n = 100, iterations = 1500)

```
[1] 1498
[1] 1499
[1] 1500
> cat("Type 0 dominance: ", dom0, "\n")
Type 0 dominance:  1483
> cat("Type 0 average gen: ", gen0/dom0, "\n")
Type 0 average gen:  400.5448
> cat("Type 1 dominance: ", dom1, "\n")
Type 1 dominance:  17
> cat("Type 1 average gen: ", gen1/dom1, "\n")
Type 1 average gen:  8921.353
> cat("Percentage of Type 1 dominance: ", dom1/i*100,"%", "\n")
Percentage of Type 1 dominance:  1.133333 %
```

Exercise 3: Probability Distributions

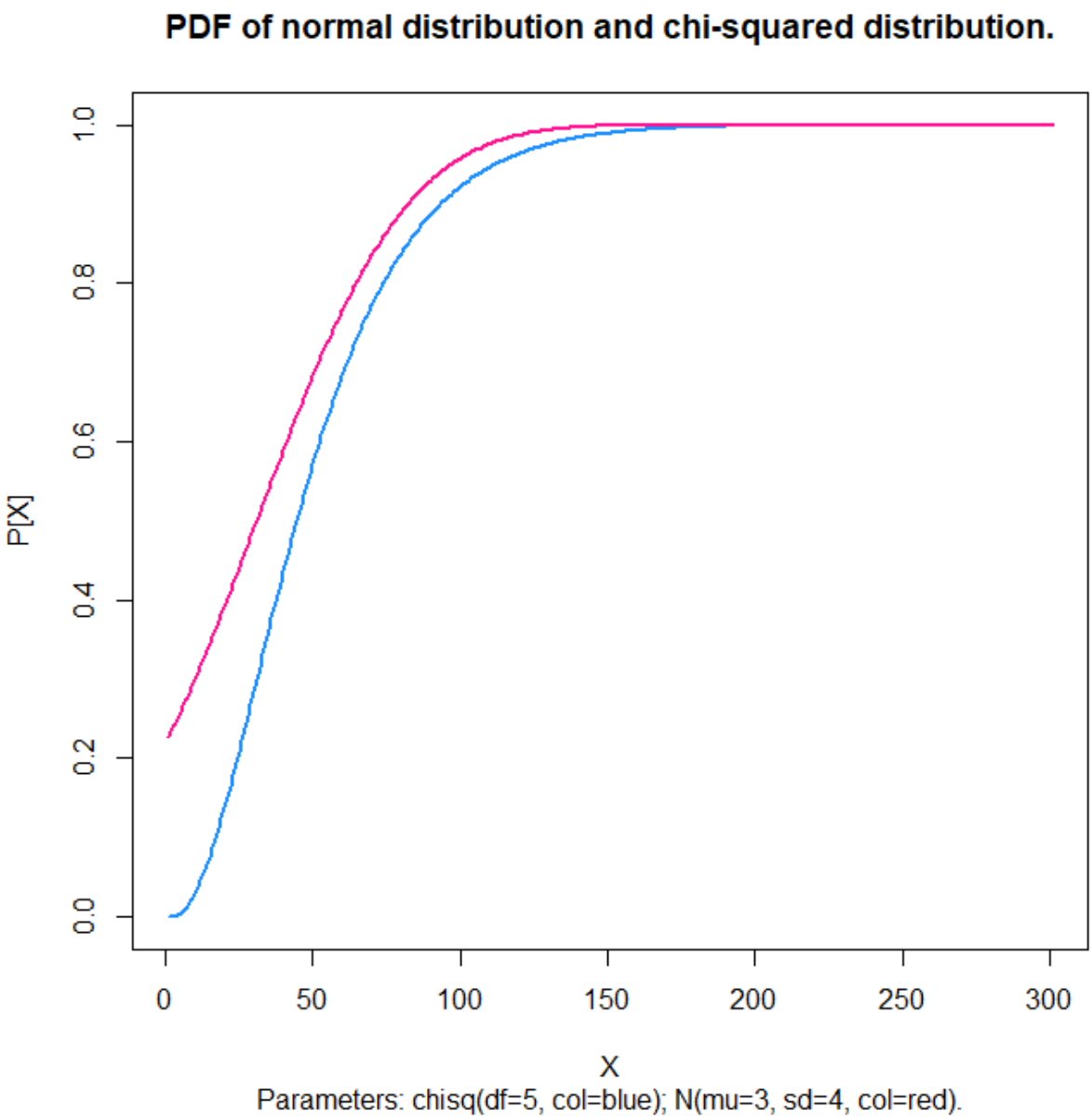
```
# (a) Tim Horton's

n = 33357600 # cups
p = 1/6 # winning
q = 5/6 # normal
k = 8000 # subpop to kirkland
x = 1300 # X < 1300, as in x <= 1299

pdf_hyper = phyper(x-1, n*p, n*q, k)
cat("P[X<=1299] = ", pdf_hyper, "\n")

# (b) Hockey

mu = 29 # average shots per game
n = 82 # sample size of games
```

```r
x = 2400 # P[X>2400], as in P[X>=2401]

# set lower.tail = TRUE for maximum meme power
pdf_poisson = ppois(2401, lambda = 29*82, lower.tail = FALSE)

cat("P[X>=2401] or P[X>2400] = ", pdf_poisson, '\n')

# (c) cdf (or pdf) for 2 different cont dist.

xlabel = "X"
ylabel = "P[X]"
title = "PDF of normal distribution and chi-squared distribution."
subtitle = "Parameters: chisq(df=5, col=blue); N(mu=3, sd=4, col=red)."
# change by param in seq(...) for line smoothing resolution

plot(pchisq(seq(0, 30, by=0.1), df=5), type="l", col="dodgerblue", xlab=xlabel, ylab=y
label, main=title,sub=subtitle, font.sub=1, font.main=2, lwd=2)

lines(pnorm(seq(0, 30, by=0.1), mean = 3, sd = 4), col="deeppink", lwd=2)
```

Results:

```
[1] "Tim Horton's cup distribution with hypergeometric"
> n = 33357600 # cups
> p = 1/6 # winning
> q = 5/6 # normal
> k = 8000 # subpop to kirkland
> x = 1300 # X < 1300, as in x <= 1299
> pdf_hyper = phyper(x-1, n*p, n*q, k)
> cat("P[X<=1299] = ", pdf_hyper, "\n")
P[X<=1299] =  0.1549954
```

```
[1] "Hockey poisson distributions"
> mu = 29 # average shots per game
> n = 82 # sample size of games
> x = 2400 # P[X>2400], as in P[X>=2401]
> pdf_poisson = ppois(2401, lambda = 29*82, lower.tail = FALSE)
> cat("P[X>=2401] or P[X>2400] = ", pdf_poisson, '\n')
P[X>=2401] or P[X>2400] =  0.3140058
```


PDF of normal distribution and chi-squared distribution.
P[X]
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
200
250
300
X
Parameters: chisq(df=5, col=blue); N(mu=3, sd=4, col=red).