DSSH 6301 - Midterm Solutions

Problem 1

```
is_prime <- function(x) {
  if (x == 2)
    return(TRUE)

  # If there is a number between 2 and x-1 that leaves no remainder, then the number
  # is not prime.
  for (i in 2:(x-1)) {
    if (x %% i == 0)
      return(FALSE)
  }

  return(TRUE)
}

primes <- c()

# Loop over all numbers from 2 to 100.
for (num in 2:100) {
  # Add number to prime list if it meets the definition of prime.
  if (is_prime(num))
    primes <- c(primes, num)
}

primes
```

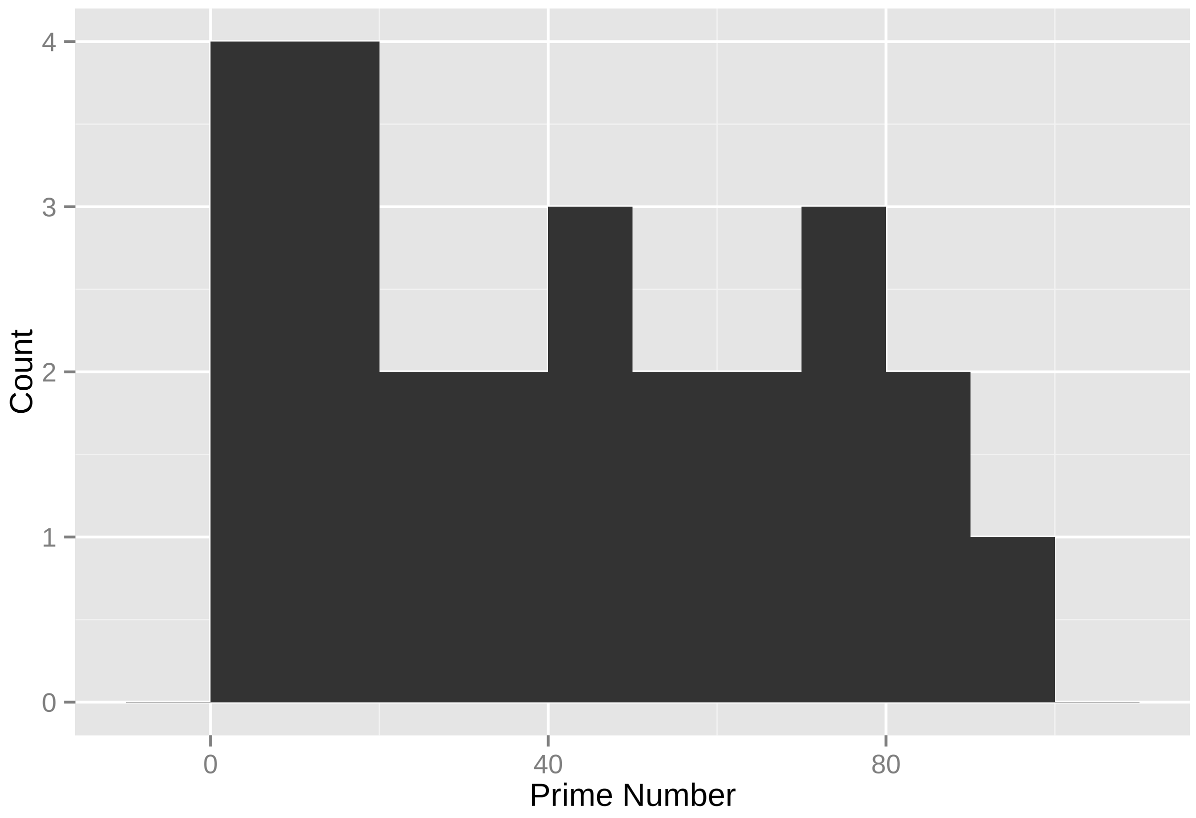
```
##  [1]  2  3  5  7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83
## [24] 89 97
```

Problem 2

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
ggplot(data=data.frame(x=primes)) + geom_histogram(aes(x), binwidth=10) +
  xlab("Prime Number") + ylab("Count")
```



Problem 3

Part a

There are 5 events, each with 2 possible outcomes. This gives us a total of 2ˆ5 events. We can manually count up the 8 of those sequences that have 3 heads in a row:

HHHHH

HHHHT

HHHTH

HHHTT

THHHH

THHHT

TTHHH

HTHHH

```
8 / 2^5
```

```
## [1] 0.25
```

Part b

If the first event is an H, there are 2^4 remaining possible outcomes. 5 of those sequences have 3 heads in a row.

(H)HHHH

(H)HHHT

(H)HHTH

(H)HHTT

(H)THHH

```r
5 / 2^4
```

```
## [1] 0.3125
```

Problem 4

Define a function using these relationships.

$$\Pr(+) = \Pr(+|A)\Pr(A) + \Pr(+|NA)(1 - \Pr(A))$$

$$\Pr(A|+) = \frac{\Pr(+|A) * \Pr(A)}{\Pr(+)}$$

```r
p_strike_given_pos <- function(p_strike, sensitivity, fp_rate) {
  sensitivity*p_strike / (sensitivity*p_strike + fp_rate*(1-p_strike))
}

p_strike_given_pos(1/100000, 99/100, 1/100)
```

```
## [1] 0.0009890307
```

Problem 5

```r
lambda <- 1
ppois(4, lambda, lower.tail=F)
```

```
## [1] 0.003659847
```

Problem 6

$$H_0 : \mu = 7$$
$$H_a : \mu \neq 7$$

```
data <- c(7, 6, 5, 8, 6, 6, 4, 5, 8, 7)

n <- length(data)
mu <- 7
x_bar <- mean(data)
s <- sd(data)

stand_err <- s / sqrt(n)

# Threshold values
alpha <- 0.05
thres <- qt(c(alpha/2, 1-alpha/2), n-1)
thres
```

```
## [1] -2.262157  2.262157
```

```
# CI
x_bar + thres * stand_err
```

```
## [1] 5.258189 7.141811
```

```
# Test statistic:
t <- (x_bar - mu) / stand_err
t
```

```
## [1] -1.921538
```

```
# p-value:
p_val <- pt(t, n-1)*2
p_val
```

```
## [1] 0.08684229
```

We fail to reject the null. We cannot say the true mean is different from 7 hours.

Problem 7

Changing n affects both the degrees of freedom for the t-distribution and the standard error. A simple way to find the number of people necessary considering both these metrics is to increase the population size until the upper CI threshold is below the mean you are considering.

```
for (i in 11:1000) {
  # Recalculate the standard error and CI
  stand_err <- s / sqrt(i)
  ci <- x_bar + c(qt(alpha/2, i-1), qt(1-alpha/2, i-1))*stand_err

  if (ci[2] < mu)
    break # condition met, leave the for loop
}

i
```

```
## [1] 13
```

```r
i-n
```

```
## [1] 3
```

Thus you should sample 13 people, 3 more than in the original sample.

```r
# New CI
ci
```

```
## [1] 5.40441 6.99559
```

```r
# New p-value
new_t <- (x_bar - mu) / stand_err

new_p_val <- pt(new_t, i-1)*2
new_p_val
```

```
## [1] 0.04892973
```

Problem 8

```r
data_resample <- c(5, 4, 5, 7, 5, 4, 5, 4, 6, 5)

data_test <- data_resample - data

n <- length(data_test)
x_bar <- mean(data_test)
s <- sd(data_test)

stand_err <- s / sqrt(n)

# Threshold values
alpha <- 0.05
thres <- qt(c(alpha/2, 1-alpha/2), n-1)
thres
```

```
## [1] -2.262157  2.262157
```

```r
# CI
x_bar + thres * stand_err
```

```
## [1] -1.9388174 -0.4611826
```

```r
# Test statistic:
t <- x_bar / stand_err
t
```

```
## [1] -3.674235
```

```
# p-value:
p_val <- pt(t, n-1)*2
p_val
```

```
## [1] 0.005121073
```

We reject the null. We can claim that people get significantly less sleep during finals with this data.

Problem 9

```
table <- matrix(c(4, 8 , 11, 7), nrow=2)

rownames(table) <- c("treatment", "control")
colnames(table) <- c("live", "die")

table
```

```
##           live die
## treatment    4  11
## control      8   7
```

```
expected <- function(margin_cell1, margin_cell2, total)
  return(margin_cell1*margin_cell2/total)

fe <- sapply(margin.table(table, 1), expected, margin.table(table, 2),
             margin.table(table))

# Test statistic
nelem <- nrow(table)*ncol(table)
chi_sq <- sum((t(table) - fe)^2 / fe)
chi_sq
```

```
## [1] 2.222222
```

```
# Degrees of freedom
df <- (nrow(table)-1) * (ncol(table)-1)
df
```

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

```
# Threshold
qchisq(0.95, df=df)
```

```
## [1] 3.841459
```

```
# p-value
pchisq(chi_sq, df=df, lower.tail=F)
```

```
## [1] 0.1360371
```

We fail to reject the null. The variables may all be independent.

Problem 10

```r
table <- as.data.frame(matrix(c(50, 45, 55, 10, 7, 4, 20, 10, 10), nrow=3))

rownames(table) <- c("water", "vodka", "coffee")
colnames(table) <- c("mean", "sd", "n")

table
```

```
##        mean sd  n
## water    50 10 20
## vodka    45  7 10
## coffee   55  4 10
```

```r
mu <- 50

G <- nrow(table)
N <- sum(table$n)

btween_var <- sum((table$mean - mu)^2*table$n) / (G-1)
within_var <- sum((table$n - 1)*table$sd^2) / (N-G)

# Test statisitic
f <- btween_var / within_var
f
```

```
## [1] 3.722334
```

```r
# Degrees of Freedom
df1 <- G-1
df2 <- N-G

df1
```

```
## [1] 2
```

```r
df2
```

```
## [1] 37
```

```r
# Threshold
qf(0.95, df1=df1, df2=df2)
```

```
## [1] 3.251924
```

```r
# p-value
pf(f, df1=df1, df2=df2, lower.tail=F)
```

```
## [1] 0.03365643
```

We reject the null hypothesis. The means are probably not all the same.