[redacted]

STAT 540 fa 2025 Exam I

AUTHOR
Karl Gregory

Write answers (unless a question says otherwise) on blank sheets of paper. Hand in these question sheets along with your answer sheets with your name on both.

1. Give the output of this R code:

```
pets <- c("dog","cat","fish","rabbit","chinchilla","goldfish")
pets[c(1:6) %% 2 == 0]
```

$\underbrace{\texttt{c(1:6) \%\% 2 == 0}}_{\text{even indices}}$

"cat", "rabbit", "goldfish"

2. Give the output of the following R code:

[redacted]

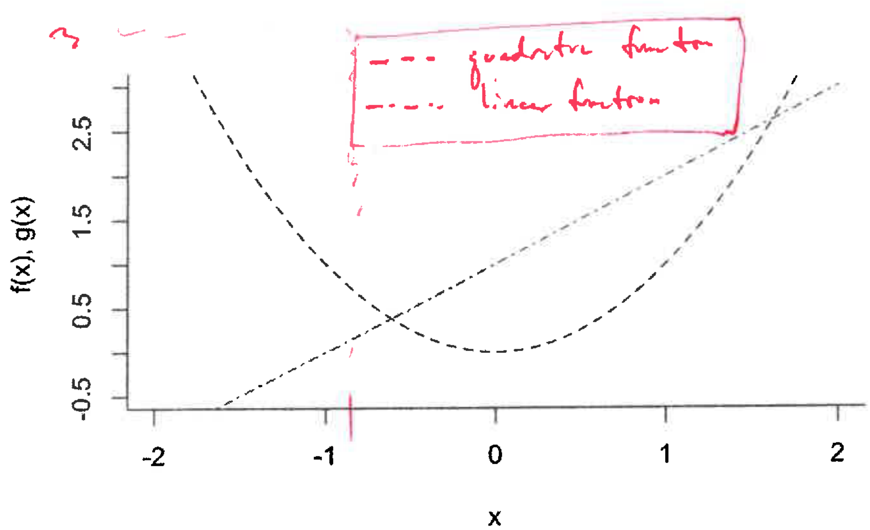
T F F F F F T T F T

```
x <- c(2.2, 3.2, 3.1, 3.5, 3.4, 1.8, 2.3, 2.3, -0.4, 0.0)
((x >= 0) & (x < 1)) | ((x >= 2) & (x < 3))
```

$x \in [0,1) \cup [2,3)$

3. Replace the question marks `???` in the code below the plot in order to add a legend labeling the lines as "Quadratic function" and "Linear function": Then *draw* your legend on the plot.

```
x <- seq(-2,2,length=201)
fx <- x^2
gx <- 1 + x

plot(fx ~ x,
     ylim = c(-1/2,3),
     type = "l",
     ylab = "f(x), g(x)",
     lty = 2,
     col = "black",
     bty = "l")
lines(gx ~ x, lty = 4, col = "blue")
```



$x = -0.8 \quad y = 3$

```
legend(x = ???,
       y = ???,
       lty = ???,
       col = ???,
       legend = ???)
```

~~let~~ lty = c(4, 2)

col = c("blue", "black")

legend = c("linear function", "quadratic function") [redacted] }

~~if(abs(x) <= gam*lam){~~

4. Write a function in R to evaluate the function given by

$$f(x,\lambda,\gamma) = \begin{cases} \lambda|x| - \dfrac{1}{2\gamma}x^2, & |x| \le \gamma\lambda \\ \dfrac{1}{2}\gamma\lambda^2, & |x| > \gamma\lambda, \end{cases}$$

~~}else{~~

}

such that under default settings the function is evaluated with $\lambda = 1, \gamma = 2$. Tip: Use `abs()` for the absolute value.

5. Consider the R function defined below. Describe what the function does if `x` is a numeric vector of length `10` and one uses `alpha = 0.10`.

```
trm <- function(x,alpha){

  n <- length(x)
  x <- sort(x)
  tr <- floor(alpha*n)

  l <- tr + 1
  u <- n - tr

  val <- mean(x[l:u])

  return(val)

}
```

6. Explain exactly what the function below computes:

```
mdf <- function(x){

  n <- length(x)
  x <- sort(x)

  if(n %% 2 == 0){

    val <- (x[n/2] + x[n/2+1])/2

  } else{

    val <- x[floor(n/2)+1]

  }

  return(val)

}
```

7. Give the entries the vector `a` will contain after the following code runs:

```
n <- 10
a <- numeric(n)
a[1] <- 0
a[2] <- 1
for(i in 3:n){

  a[i] <- a[i-1] + a[i-2]

}
```

~~0, 1, 1, 2, 3, 5, 8, 13, 21, 34~~

8. Suppose you borrow an amount P_0 at annual interest rate r, compounded monthly, and you make monthly payments in the amount p. If P_n is the amount you owe at the beginning of month n, the amount you owe at the beginning of month $n+1$ is given by

$$P_{n+1} = P_n(1 + r/12) - p.$$

Write a for loop to compute the amount you will owe after making 36 monthly payments in the amount $\$200$ on a loan of $\$10000$ at annual interest rate 0.056, compounded monthly.

9. Give the output of the following code:

```
x <- c(2,6,-2,7,8)
n <- length(x)
val <- 0
i <- 1
while(i <= n){

  val <- val + x[i]
  i <- i + 1

}
val
```

10. An algorithm for obtaining an approximation to the square root of a number S is to make an initial guess x_0 and update the guess according to the recursion

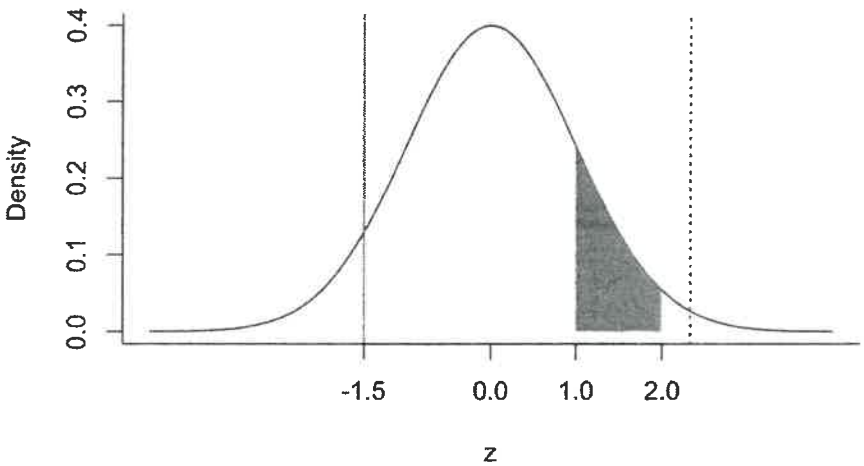
$$x_{n+1} = (x_n + S/x_n)/2$$

"until convergence", meaning until the update from x_n to x_{n+1} becomes small enough to be negligible. Write a loop which will make the update above until the convergence criterion $S(x_{n+1} - x_n)^2 < \epsilon$ is satisfied for $\epsilon = 10^{-6}$. Set the initial value x_0 equal to S. You may follow the template below, replacing each `???` with one or more commands.

```
???

conv <- F
while(!conv){

  ???
  conv <- ???

}
```

11. Below is depicted the probability density function (pdf) of the standard normal distribution. Give R code which returns:

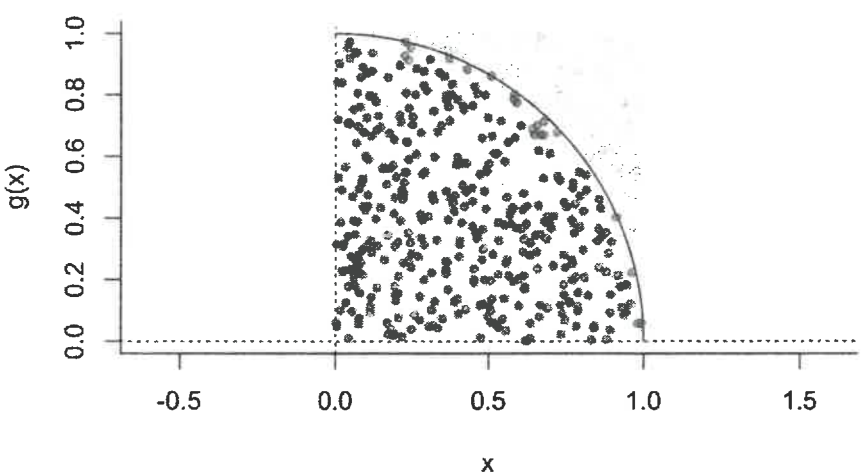
i. The area of the shaded region.
ii. The height of the density when it crosses the solid vertical line.
iii. The horizontal position of the vertical dotted line, given that the area under the curve to the right of this line is equal to 0.01.
iv. 100 independent realizations of a random variable having this pdf.



12. Write R code to obtain a Monte Carlo approximation to the integral

$$I = \int_0^1 \sqrt{1 - x^2} dx$$

using the hit-or-miss method. The method is depicted below.



13. Suppose we are interested in the distribution of the random variable $X = \sum_{i=1}^{k} |Z_i|^{3/2}$, where $Z_1, \ldots, Z_k$ are independent $\text{Normal}(0, 1)$ random variables for some $k \geq 1$. Choose some value of k and write a Monte Carlo simulation in R for obtaining an approximation to the expected value of the random variable X.

14. Consider the following experiment: Place two decks of 52 cards side by side and flip over the top card of each deck; record whether the cards match and flip over the next card in each deck. Continue flipping over the top card of each deck and recording whether the cards match until all 52 cards in each deck have been flipped. Of interest is the probability of obtaining at least one match. Write a Monte Carlo simulation in R code for obtaining an approximation to the probability of getting at least one match in this experiment. Tip: The code `sample(1:52,52,replace = F)` gives a random permutation of the numbers 1 through 52.

15. The code below creates a data set and reshapes it with the `reshape()` function. Write down in detail the complete data frame `toy_data_2`.

```
toy_data <- data.frame(Unit = c(1,1,1,1,2,2,2,2),
                       Observation = c(1,2,3,4,1,2,3,4),
                       Value = round(rnorm(8),2))

toy_data
```

```
  Unit Observation Value
1    1           1 -0.59
2    1           2 -1.44
3    1           3 -1.96
4    1           4  0.15
5    2           1 -0.45
6    2           2 -0.70
7    2           3 -0.28
8    2           4 -1.41
```

```
toy_data_2 <- reshape(toy_data,
                      idvar = "Unit",
                      timevar = "Observation",
                      sep = " ",
                      direction = "wide",
                      new.row.names = 1:2)
```

① "cat", "rabbit", "goldfish"

② T F F F F F T T F T

③ On ~~paper~~ question sheet.

④ fn <- function (x, lam = 1, gam = 2) {

if (abs(x) <= gam * lam) {

if (abs (x) <= gam * lam) {

val <- lam * abs(x) - x ** 2 / (2 * gam)

} else {

gam * lam ** 2 / 2

}

⑤ This function removes ~~the~~ a number of the smallest and largest values in x and takes the mean of the middle values. With n = 10, alpha = 0.1, it will return the mean of the middle 8 values.

⑥ If x has an even # of values, it returns the midpoint between the middle two; if an odd # of values, the middle value.

In short, the median.

(7) 0, 1, 1, 2, 3, 5, 8, 13, 21, 34

(8)

```
P <- 10000
p <- 200
n <- 36
r <- 0.056

for (i in 1:n) {
        P <- P*(1 + r/12) - p
    }

P
```

(9) This is the sum of the entries in x.

$2 + 6 - 2 + 7 + 8 = 21$

(10)

```
S <- a number
x <- S
tol <- 1e-6
conv <- F
while (!conv) {
      x0 <- x
      x <- (x + S/x)/2
      conv <- S(x-x0)** < tol
}
```

(11)

(i) pnorm(2) - pnorm(1)

(ii) dnorm(-1.5)

(iii) qnorm(0.99)

(iv) rnorm(100).

(12)

```
g <- function(x) { sqrt (1- x**2) }
X = runif(0,1)
Y = runif(0,1)
mean (Y <= g(X))
```

(13)

```
k <- 16
N <- 1000   (large)
X <- numeric (N)
for (i in 1:N) {
        Z ~ rnorm(k)
        X[i] <- ~~mean~~ sum ( abs (Z)**(3/2))
      }
mean (X)
```

14

```
N <- 1000
match <- logical(N)
for (i in 1:N) {

    f1 <- sample(1:52, 52, replace = F)
    f2 <- sample(1:52, 52, replace = F)
match[i] <- any(f1 == f2)

}
mean(match)
```

15

Unit	Observation 1	Observation 2	Observation 3	Observation 4
1	-0.59	-1.44	-1.96	0.15
2	-0.45	-0.70	-0.28	-1.41