Function Drills with Answers

Statistics 405

September 12, 2009

Write a simple function in R for each of the following tasks.

1. Return the circumference of a circle with the given radius.

```
c_circ <- function(r){
  2 * pi * r
}
```

2. Return the area of a circle with the given radius.

```
c_area <- function(r){
  pi * r ^ 2
}
```

3. Return the area of a circle with the given radius.

```
c_vol <- function(r){
  4 / 3 * pi * r ^ 3
}
```

4. Return the circumference, area (of the largest cross-section), and volume of a sphere with the given radius. Each should be labelled in the functions output.

```
c_stats <- function(r){
  c(circumference = c_circ(r),
    area = c_area(r),
    volume = c_vol(r)
  )
}
```

5. Given the coefficients of a quadratic polynomial, return the roots.

```
quad_formula <- function(a, b, c){
  stopifnot(b ^ 2 >= 4 * a * c)
  c((-b - sqrt(b ^ 2 - 4 * a * c)) / (2 * a),
    (-b + sqrt(b ^ 2 - 4 * a * c)) / (2 * a))
}
```

6. Return the lowest positive value of a vector

```
min_pos <- function(vec){
  pos <- vec[vec > 0]
  min(pos)
}
```

7. Return the second lowest positive value of a vector

```
min2_pos <- function(vec){
  pos <- vec[vec > 0]
  pos <- pos[-which(pos == min(pos))]
  min(pos)
}
```

8. Divide each element in a numeric vector by the vector's length.

```
div_vec <- function(vec){
  vec / length(vec)
}
```

9. Test whether a number is even.

```
is.even <- function(num){
  num %% 2 == 0
}
```

10. Test whether a number is odd.

```
is.odd <- function(num){
  num %% 2 == 1
}
```

11. If a number is odd add one to it.

```
one_to_odd <- function(num){
  if (is.odd(num))
    return(num + 1)
  return(num)
}
```

12. If any number in a numeric vector is odd, add one to it

```
one_to_odd2 <- function(vec){
  vec[is.odd(vec)] <- vec[is.odd(vec)] + 1
  vec
}

# note: this function is better than the previous one
# It works for numbers AND vectors, and it is simpler
```

13. Test whether a number is an integer.

```
integer <- function(a){
  trunc(a) == a
}
```

14. Find the range of a vector.

```
my_range <- function(vec){
  max(vec) - min(vec)
}
```

15. Find the sum of a numeric vector (without using `sum()`).

```
my_sum <- function(vec){
  total <- 0
  for (i in 1:length(vec)){
   total <- total + vec[i]
  }
  total
}
```

16. Find the mean of a numeric vector (without using `mean()`).

```
my_mean <- function(vec){
  my_sum(vec)/length(vec)
}
```

17. Find the mean of a vector that contains one or more NA's by ignoring any NA's (without using `mean()`).

```
my_mean2 <- function(vec){
  my_mean(na.omit(vec))
}
```

18. Find the variance of a numeric vector (without using `var()`).

```
my_var <- function(vec){
  xbar <- my_mean(vec)
  n <- length(vec)
  sum <- 0

  for (i in 1:n){
   sum <- sum + (vec[i] - xbar) ^ 2
  }

  # let's use n-1 so we can compare with R's var().
  # This is the sample variance
  sum / (n - 1)
 }
```

19. Automatically create a histogram of a vector.

```
my_hist <- function(vec){
  hist(vec)
}
```

20. Automatically create a histogram of a vector with the given number of bins.

```
better_hist <- function(vec, n){
  hist(vec, breaks = n)
}
```

21. Automatically create a scatterplot matrix with the variables in a given data frame.

```
scatterplot <- function(df){
  library(ggplot2)
  plotmatrix(df)
}
```

22. Find the least common multiple of two numbers.

```
LCM <- function(a,b){
  test <- a * c(1:b)
  multiples <- test[integer(test/b)]
  min(multiples)
}
```

23. Index a series of observations by the first observation (hint: express each observation as a percentage of the first observation).

```
how_to_index <- function(vec){
  vec / vec[1] * 100
}
```

24. Find the determinant of a four by four matrix.

```
determinant <- function(matrix){
  matrix[1,1] * matrix[2,2] - matrix[1,2] * matrix[2,1]
}
```

25. Separate the integer and decimal parts of a number, return them in a vector of length two.

```
split_num <- function(num){
  c(trunc(num), num - trunc(num))
}
```

26. Return the given vector with all NA's removed.

```
clean <- function(vec){
  na.omit(vec)
}
```

27. Return the row numbers of rows in a data frame that contain NA's.

```
get_NAs <- function(df){
  new <- na.omit(df)
  setdiff(row.names(df), row.names(new))
}
```

28. Return the actual rows of a data frame that contain NA's.

```
get_NAs2 <- function(df){
  new <- na.omit(df)
  rows <- setdiff(row.names(df), row.names(new))
  df[rows,]
}
```

29. Create a new vector by repeating a given vector a given number of times.

```
rep_vec <- function(vec, n){
  rep(vec, n)
}
```

30. Double each element in a vector (e.g., turn *a,b,c,...* into *a, a, b, b, c,...*).

```
rep_vec2 <- function(vec, n){
  vec[rep(1:length(vec), each = n)]
}
```

31. Randomly return one of the following phrases, "Ace", "King" or "Queen" with equal probability of returning each.

```
random <- function(){
  sample(c("Ace", "King", "Queen"), 1)
}
```

32. Randomly return one of the following phrases, "Ace", "King" or "Queen" with twice as much probability of returning "Ace" as either "King" or "Queen."

```
 random2 <- function(){
  sample(c("Ace", "King", "Queen"), 1, prob = c(2, 1, 1))
}
```

33. Take any character string and add "...in Stat 405" to the end.

```
fortune_cookie <- function(fortune){
  paste(fortune, "...in Stat405.")
}
```

34. Take any character string and add "...in Stat 405" to the end. Check that the input is a character string. Return an error if it is not.

```
fortune_cookie <- function(fortune){
  stopifnot(is.character(fortune))
  paste(fortune, "...in Stat405.")
}
```

35. Save the current graph with width = 6 and height = 6 as a pdf with the inputted name.

```
save_plot <- function(name){
  filename <- paste("name", "pdf", sep = ".")
  ggsave(filename, width = 6, height = 6)
}
```

36. Save the current graph with a given width and height as a pdf with the inputted name.

```
save_plot2 <- function(name, width, height){
  filename <- paste("name", "pdf", sep = ".")
  ggsave(filename, width = width, height = height)
}
```

37. Save a copy of a data frame as a comma separated values file whose filename is the name of the data frame plus ".csv"

```
save_file <- function(df){
  filename <- paste(substitute(df), "csv", sep = ".")

  # best method to avoid adding row numbers
  write.table(file, filename, sep = ",", row = F)
}
```

38. Identify whether an object is a logical, character, or numeric object.

```
type <- function(obj){
  mode(obj)
}
```

39. Display the number of groups of size n can be made from the inputted vector of length k.

```
n_choose_k <- function(n, k){
  factorial(n) / (factorial(k) * factorial(n - k))
}
```

40. Return the number of unique permutations that can be from a given vector (caution: don't use large vectors).

```
num_seqs <- function(vec){
  n <- length(vec)
  factorial(n)
}
```

41. Return the number of unique sets that can be made from an inputted vector.

```
num_sets <- function(vec){
  vec <- unique(vec)
  n <- length(vec)
  my_sum(2 ^ c(0:(n-1)))
}
```

42. Return whichever the entered number is closest to: 0 or 1000.

```
closest <- function(a){
  if (a > 1000){
   a <- 1000
  }

  round(a, -3)
}
```

43. Given a data frame with two columns, return all of the combinations of the two variables that occur once or more.

```
combos <- function(df){
  combinations <- table(df[,1], df[,2])
  df_counts <- as.data.frame(combinations)
  names(df_counts) <- c(names(df), "count")
  df_counts <- subset(df_counts, count > 0)
  df_counts
}
```

44. Automatically plot the above results with each variable on an axis and the number of occurrences (counts) represented by color.

```
colorful_counts <- function(df){
  df_counts <- combos(df)
  df_counts <- within(df_counts, {
    x <- as.numeric(as.character(df_counts[,1]))
    y <- as.numeric(as.character(df_counts[,1]))
  })
  qplot(x, y, data = df_counts, colour = count)
}
```

45. Create a new vector where each *ith* element is the sum of the first *i* elements of the given vector.

```
cumsum <- function(vec){
  new <- vector(length = length(vec))
  for(i in 1:length(vec)){
   new[i] <- my_sum(vec[1:i])
  }
  new
}
```

46. Select the number in a vector that is the greatest distance from the first element of the vector

```
select <- function(vec){
  distance <- abs(vec - vec[1])
  vec[which(distance == max(distance))]
}
```

47. Return whether a vector of numbers is right skewed or left skewed by comparing its mean and median.

```
# note: this concept only works for large amounts of skew - Garrett
skew <- function(vec){
  if (mean(vec) < median(vec))
    return("left-skewed")
  if (mean(vec) > median(vec))
    return("right-skewed")
  return("symmetric")
}
```

48. Find the (statistical) mode of a vector.

```
mode_vec <- function(vec){
  counts <- as.data.frame(table(vec))$Freq
  vec[which(counts == max(counts))]
}
```

49. Given a numeric vector of length 100, determine which element occurs at the 70th percentile.

```
ptile100 <- function(vec){
  vec[order(vec)]
  vec[70]
}
```

50. Given a numeric vector of length 10, determine which element occurs at the 70th percentile.

```
ptile10 <- function(vec){
  vec[order(vec)]
  vec[7]
}
```

51. Given a numeric vector of length 10, determine which element occurs at the 70th percentile.

```
ptile <- function(vec){
  vec[order(vec)]
  vec[round(.7 * length(vec), 1)]
}
```

52. Return a vector with its elements reordered in a random manner.

```
shuffle <- function(vec){
  sample(vec, length(vec), replace = F)
}
```

53. Return a vector with its elements ordered from smallest to largest.

```
order1 <- function(vec){
  vec[order(vec)]
}
```

54. Return a vector with its elements ordered largest to smallest.

```
order2 <- function(vec){
  vec[order(-vec)]
}
```