Warmup 05: Functions

Stat 133, Spring 2019

The purpose of this assignment is to begin working on the programming concepts that are covered in week 7:

- writing simple functions
- documenting functions with Roxygen comments
- using conditionals

General Instructions

- Write your narrative and code in an `Rmd` (R markdown) file.
- Name this file as `warmup05-first-last.Rmd`, where `first` and `last` are your first and last names (e.g. `warmup05-gaston-sanchez.Rmd`).
- Include a code chunk at the top of your file like the one in the following screen capture:

````
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE, error = TRUE)
```
````

- Since you willl be writing a couple of functions with `stop()` statements, it is essential that you set up `error = TRUE`, otherwise `"knitr"` will stop knitting your `Rmd` file if it encounters an error.
- All your functions should include Roxygen comments: e.g.
  - `@title`
  - `@description`
  - `@param`
  - `@return`
- Submit your Rmd and html files to bCourses.

1) Gaussian Function

The Gaussian (Normal) function, given in the equation below, is one of the most widely used functions in science and statistics:

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

The parameters σ and μ are real numbers, where σ must be greater than zero. For more information, see the wikipedia entry:

https://en.wikipedia.org/wiki/Normal_distribution#/media/File:Normal_Distribution_PDF.svg

Function `gaussian()`

Write a function `gaussian()`—including roxygen comments—to compute the probability density based on the formula above.

- The function should take three arguments:
  - `x` = numeric vector for x (default 0)
  - `m` = numeric vector for μ (default 0)
  - `s` = numeric vector for σ (default 1)
- It should return the computed probability density. Do NOT use `print()` for the returned output. Use `return()` instead.
- If `s` is less than or equal to zero, raise an error: `"s must be greater than zero"`
- Include the following code in your `Rmd` file:

```
# test set 1
gaussian(x = 0, m = 0, s = 1)
gaussian(x = 1, m = 0, s = 2)

# test set 2
gaussian(x = 1, m = 0, s = 0)
gaussian(x = 1, m = 0, s = -1)
```

- Use your `gaussian()` function with a vector `seq(-4.5, 4.5, by = 0.1)`, and pass the values to `plot()` to get a normal curve:

```
# gaussian curve
x_values <- seq(from = -4.5, to = 4.5, by = 0.1)
y_values <- gaussian(x_values, m = 0, s = 2)
plot(x_values, y_values, las = 1, type = "l", lwd = 2)
```

- In addition to the above plot, you should also try to replicate—as much as possible—the following graph (original version displayed in the wikipedia entry of the normal distribution). You can use any plotting approach, but you must specify colors in hexadecimal notation.

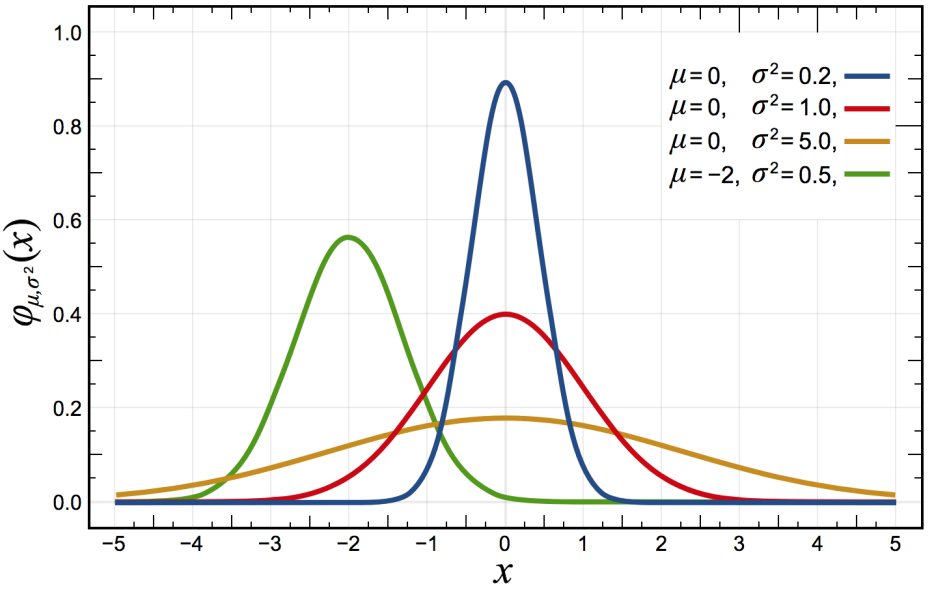

Figure 1: Normal probability density functions

Here are some resources about how to add mathematical symbols in R plots:

- https://andyphilips.github.io/blog/2017/08/16/mathematical-symbols-in-r-plots.html
- https://trinkerrstuff.wordpress.com/2018/03/15/2246/

More instructions on next page

2) Descriptive Statistics

Write a function `descriptive()`—including roxygen comments—that takes a numeric vector as input, and returns a **named vector** with the following descriptive statistics:

1) `min`: minimum
2) `q1`: first quartile (Q2)
3) `median`: median
4) `mean`: mean
5) `q3`: third quartile (Q3)
6) `max`: maximum
7) `range`: range or span (max - min)
8) `iqr`: interquartile range (IQR)
9) `sd`: standard deviation

- The function should take two arguments:
  - `x` = numeric vector for x (default 0)
  - `na.rm` = whether to remove missing values before computations (default `FALSE`)
- It should return a named vector with the computed summary statistics.
- The function should `stop()`—with a descriptive error message—when the input vector is not numeric.
- Test the function with the following code:

```
# input vectors
set.seed(100)
x <- rnorm(100)
y <- x
y[sample(1:100, size = 10)] <- NA

# test set 1
descriptive(x)
descriptive(y)
descriptive(y, na.rm = TRUE)
descriptive(letters)

# test set 2
a <- descriptive(x)
class(a)
length(a)
names(a)
```

More instructions on next page

3) Minkowski Distance

For a point $x = (x_1, x_2, \ldots, x_n)$ and a point $y = (y_1, y_2, \ldots, y_n)$, the Minkowski distance of order p (p-norm distance) is defined as:

$$\text{1-norm distance} = \sum_{i=1}^{n} |x_i - y_i|$$

$$\text{2-norm distance} = \left(\sum_{i=1}^{n} |x_i - y_i|^2 \right)^{1/2}$$

$$\text{p-norm distance} = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

$$\text{infinity norm distance} = max(|x_1 - y_1|, |x_2 - y_2|, \ldots, |x_n - y_n|)$$

where p need not be an integer, but it cannot be less than 1, because otherwise the triangle inequality does not hold. In case you're curious, you can find more information about this type of distance in the following wikipedia entry:

https://en.wikipedia.org/wiki/Distance

Function `minkowski()`

Write a function `minkowski()`—including roxygen comments—for the Minkowski distances.

- The function should take three arguments:
  - `x` = numeric vector for one point
  - `y` = numeric vector for the other point
  - `p` = either a numeric value greater than 1, or a character string `"max"` (default 1)
- It should return the computed distance. Do NOT use `print()` for the returned output. Use `return()` instead.
- Check that `x` and `y` have the same length, otherwise raise an error: `"x and y have different lengths"`.
- Give `p` a default value of 1.
- If `p` is numeric, check that `p` is greater than 1, otherwise raise an error: `"p cannot be less than 1"`
- If `p` is character, check that `p` equals `"max"`, otherwise raise an error: `"invalid character value for p"`

Informal Tests for `minkowski()`

Include the following code in your `Rmd` file:

```
# some points
point1 <- c(0, 0)
point2 <- c(1, 1)
point3 <- sqrt(c(2, 2))
point4 <- c(0, 1)
point5 <- c(1, 1, 1)

# test set 1
minkowski(point1, point2, p = 1)
minkowski(point1, point3, p = 2)
minkowski(point1, point2, p = 'max')

# test set 2
minkowski(point4, point5, p = 1)
minkowski(point1, point2, p = 0.5)
minkowski(point1, point2, p = 'min')
```