STT 301: Functions and conditional statements

Shawn Santo

September 24, 2018

Introduction

Learning objectives:

- Write functions that return a vector
- Write functions that return a list
- if-then statements
- `stopifnot` statements

R provides functions that return useful characteristics of many common statistical distributions. The naming convention for these functions is a prefix, which identifies what the function does, followed by an abbreviation of the statistical distribution's name. These prefixes are:

- `p` for "probability", the cumulative distribution function (CDF)
- `q` for "quantile", the inverse CDF
- `d` for "density", the density function (PDF)
- `r` for "random", a random variable having the specified distribution.

For the normal distribution, these functions are `pnorm`, `qnorm`, `dnorm`, and `rnorm`, where the norm portion reminds us this is for the normal distribution. For the binomial distribution, these functions are `pbinom`, `qbinom`, `dbinom`, and `rbinom`. Click here (https://stat.ethz.ch/R-manual/R-devel/library/stats/html/Distributions.html) for a list of statistical distributions in base R. By "base R" it means not part of a package and immediately available once R is open.

Pareto distribution

The Pareto distribution (https://en.wikipedia.org/wiki/Pareto_distribution) is not available in base R, so we're going to code it ourselves. For this in class assignment, we'll just code the quantile function, i.e., `qpareto`. Here's a bit of background on deriving the Pareto's quantile function (you don't need to understand the subsequent PDF or CDF details, but pay close attention to the definition of quantile function $Q(p)$).

The Pareto family of distributions is parameterized by α and x_0 and has probability density function

$$f(x) = \begin{cases} \frac{(\alpha-1)x_0^{\alpha-1}}{x^\alpha}, & x > x_0, \\ 0, & x \leq x_0. \end{cases}$$

From the PDF it is relatively easy to compute the CDF, which is given by

$$F(x) = \begin{cases} 0 & x < x_0 \\ 1 - \left(\frac{x_0}{x}\right)^{\alpha - 1} & x \geq x_0 . \end{cases}$$

The quantile function is defined for $0 \leq p \leq 1$, and it returns the value x_p such that $F(x_p) = p$. For the Pareto distribution, the quantile function is given by

$$Q(p) = Q(p, \alpha, x_0) = x_0 (1 - p)^{-\frac{1}{\alpha - 1}}.$$

Using the definition of $Q(p)$, we can compute the pth quantile for specific values of p. For example, here are the medians (0.5 quantiles) of Pareto distributions with $x_0 = 1, \alpha = 3.5$, $x_0 = 6 \times 10^8, \alpha = 2.34$, and the 0.92 quantile of the Pareto distribution with $x_0 = 1 \times 10^6, \alpha = 2.5$.

```
1 * (1 - 0.5)^(-1/(3.5 - 1))
```

```
[1] 1.319508
```

```
6e+08 * (1 - 0.5)^(-1/(2.34 - 1))
```

```
[1] 1006469227
```

```
1e+06 * (1 - 0.92)^(-1/(2.5 - 1))
```

```
[1] 5386087
```

It would be helpful to have a function that automated this process, both so we don't have to remember the form of the quantile function for the Pareto distribution and so we avoid making mistakes.

We will build our function, `qpareto`, in a sequence of steps.

Step 1

Write a function called `qpareto.1` that takes arguments `p`, `alpha`, and `x0` and returns $Q(p, \alpha, x_0)$ as defined above. Check to make sure your function returns the same answers as the three below.

```
qpareto.1(p = 0.5, alpha = 3.5, x0 = 1)
```

```
[1] 1.319508
```

```
qpareto.1(p = 0.5, alpha = 2.34, x0 = 6e+08)
```

```
[1] 1006469227
```

```
qpareto.1(p = 0.92, alpha = 2.5, x0 = 1e+06)
```

```
[1] 5386087
```

Step 2

Most of the quantile functions in R have an argument `lower.tail` that is either `TRUE` or `FALSE`. If `TRUE`, the function returns the pth quantile. If `FALSE`, the function returns the $(1-p)$th quantile, i.e., returns the value x_p such that $F(x_p) = 1 - p$.

Create a function `qpareto.2` that has an additional argument `lower.tail` which is by default set to `TRUE`. Your `qpareto.2` function should test whether `lower.tail` is `FALSE`. If it is `FALSE`, the function should replace p by $1-p$. Then pass either p or $1-p$ to `qpareto.1` to compute the appropriate quantile, i.e., `qpareto.1` is called from inside of `qpareto.2`. Check to make sure your function returns the same answers as the two below.

```
qpareto.2(p = 0.5, alpha = 3.5, x0 = 1)
```

```
[1] 1.319508
```

```
qpareto.2(p = 0.08, alpha = 2.5, x0 = 1e+06, lower.tail = FALSE)
```

```
[1] 5386087
```

There is a downside to writing the function the way we have. We need `qpareto.1` to be in the workspace when `qpareto.2` is called. But there is a big advantage. If we discover a better way to calculate quantiles of the Pareto distribution we can rewrite `qpareto.1` and the new version will automatically be used in `qpareto.2`.

Step 3

Next, let's add some check with regards to the function's arguments. In the case of the Pareto quantile function, we need $0 \leq p \leq 1$, $\alpha > 1$, and $x_0 > 0$. We can use several `if` statements and `stop` functions to check arguments and to display error messages. Another option is to use the `stopifnot` function. This function evaluates each of the expressions given as arguments. If any are not `TRUE`, the `stop` function is called, and a message is printed about the first untrue statement. Below is an example.

```
ff <- function(p, y, z) {
    stopifnot(p > 0, p < 1, y < z)
    return(c(p, y, z))
}

ff(p = 0.5, y = 3, z = 5)
```

```
[1] 0.5 3.0 5.0
```

```
ff(p = -1, y = 3, z = 5)
```

```
Error in ff(p = -1, y = 3, z = 5): p > 0 is not TRUE
```

```
ff(p = -1, y = 3, z = 2)
```

```
Error in ff(p = -1, y = 3, z = 2): p > 0 is not TRUE
```

```
ff(p = 2, y = 3, z = 5)
```

```
Error in ff(p = 2, y = 3, z = 5): p < 1 is not TRUE
```

```
ff(p = 0.5, y = 3, z = 2)
```

```
Error in ff(p = 0.5, y = 3, z = 2): y < z is not TRUE
```

Write a function named `qpareto` that adds a `stopifnot` statement to the `qpareto.2` function. The `stopifnot` statement should check the validity of the three arguments `p`, `x0`, and `alpha`.

R Markdown will not compile if your R function stops due to the `stopifnot` function. You can, and should, tell the offending R code chunks to ignore the stop call, by including `error=TRUE` as a chunk option.

Check to make sure your function returns the same answers as the five below. Remember to set the option `error=TRUE` so your document will knit.

```
qpareto(p = 0.5, alpha = 3.5, x0 = 1)
```

```
[1] 1.319508
```

```
qpareto(p = 0.08, alpha = 2.5, x0 = 1e+06, lower.tail = FALSE)
```

```
[1] 5386087
```

```
qpareto(p = 1.08, alpha = 2.5, x0 = 1e+06, lower.tail = FALSE)
```

```
Error in qpareto(p = 1.08, alpha = 2.5, x0 = 1e+06, lower.tail = FALSE): p <= 1 is not T
RUE
```

```
qpareto(p = 0.5, alpha = 0.5, x0 = -4)
```

```
Error in qpareto(p = 0.5, alpha = 0.5, x0 = -4): alpha > 1 is not TRUE
```

```
qpareto(p = 0.5, alpha = 2, x0 = -4)
```

```
Error in qpareto(p = 0.5, alpha = 2, x0 = -4): x0 > 0 is not TRUE
```

Maximum likelihood estimates

The `qpareto` functions returned a vector of length one. However, many functions return more complex R objects such as lists. Consider a random sample of data from a population known to follow a normal distribution. Let x_i $(i = 1 \dots n)$ be the observed random sample from a normal distribution. The maximum likelihood estimators of the population mean and population variance of a normal distribution are defined as

$$\hat{\mu} = \frac{\sum_{i=1}^{n} x_i}{n} \quad \text{and} \quad \hat{\sigma}^2 = \frac{\sum_{i=1}^{n} (x_i - \hat{\mu})^2}{n}.$$

Write a function named `normal.mle` that takes as input a vector `x` of data values and returns a three-component list. One component should be named `mean.hat` and should be the maximum likelihood estimate of the population mean. The next component should be named `var.hat` and should be the maximum likelihood estimate of the population variance. The last component should be names `sample.data` and be the data vector, `x` . The function should check whether the argument `x` is numeric and whether the length of `x` is at least two.

Check to make sure your function returns the same answers as the three shown below. Remember to set the chunk option `error=TRUE` so your document will knit.

```
# show correctly
normal.mle(c(1, 2, 1, 4))
```

```
$mean.hat
[1] 2

$var.hat
[1] 1.5

$sample.data
[1] 1 2 1 4
```

```
# check that error shows when a character vector is input
normal.mle(c("a", "b"))
```

```
Error in normal.mle(c("a", "b")): is.numeric(x) is not TRUE
```

```
# check that error shows when a vector of length 1 is input
normal.mle(1)
```

```
Error in normal.mle(1): length(x) >= 2 is not TRUE
```