Today we practice with functions. This document is adapted from a handout from Cozma Shalizi.

For many common distributions, R contains functions which return the density, the CDF, the quantiles, and randomly generated values. For example `dnorm`, `pnorm`, `qnorm`, and `rnorm` are such functions for the normal family of distributions. Today we will create our own quantile function for a distribution which is not built in to R, the Pareto Distribution.

The Pareto family of distributions is parameterized by $\alpha$ 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 $Q$ is defined for $0 < p < 1$ by

$$Q(p) = \inf\{x: F(x) \geq p\}.$$ 

The quantile function returns the value $x_p$ such that $F(x_p) = p$, i.e., the value with area $p$ to its left. For the Pareto distribution the quantile function is given by

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

It is easy to compute the $p$th 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
```
Of course it would be helpful to have a function which 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.

**Exercise 1.** Write a function called `qpareto.1` which takes arguments `p`, `alpha`, and `x0` and which returns $Q(p, \alpha, x_0)$. So for example the function should do the following:

```r
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
```

END OF EXERCISE

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

**Exercise 2.** Create a function `qpareto.2` which has an additional argument `lower.tail` which is by default set to `TRUE`. The function should test whether `lower.tail` is `TRUE`. If so, the function should replace $p$ by $1 - p$. *Then the function should call `qpareto.1` to compute the appropriate quantile.* So for example the function should do the following:

```r
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
```

Note that there is a downside to writing the function the way we have. We need `qpareto.1` to be in the workspace when `qpareto.3` 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`.  

END OF EXERCISE
Next, adding checks to ensure that arguments to the function are reasonable would be helpful. 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. Here is an example.

```r
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: p > 0 is not TRUE
ff(p = -1, y = 3, z = 2)
Error: p > 0 is not TRUE
ff(p = 2, y = 3, z = 5)
Error: p < 1 is not TRUE
ff(p = 0.5, y = 3, z = 2)
Error: y < z is not TRUE
```

**Exercise 3.** Write a function qpareto which adds a stopifnot statement to the qpareto.2 function. The stopifnot statement should check the validity of the three arguments \( p, x_0, \) and \( \alpha. \) Here are some examples of this function in action:

```
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)
```

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

Error: p <= 1 is not TRUE

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

Error: alpha > 1 is not TRUE

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

Error: x0 > 0 is not TRUE

The qpareto functions returned a length one vector. Often functions should return more complex R objects such as lists. Recall that the maximum likelihood estimators of the mean and variance of a normal distribution are the sample mean and a scaled version of the sample variance:

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

Exercise 4. Write a function named normal.mle which takes as input a vector x of data values and which returns a two-component list. One component should be named mean_hat and should be the estimate of the mean. The other component should be named var_hat and should be the estimate of the variance. The function should check whether the argument x is numeric, and whether the length of x is at least two. Here is the function in action.

```
normal.mle(c(1, 2, 1, 4))
```

$mean_hat

[1] 2

$var_hat

[1] 1.5

```
normal.mle(c("a", "b"))
```

Error: is.numeric(x) is not TRUE

```
normal.mle(1)
```

Error: length(x) >= 2 is not TRUE