A higher-order function is a function that takes one or more functions as arguments, and\or returns a function as its result. This can be super handy in programming when you want to tilt your code towards readability and still keep it concise.  
  
Consider the following code:

# Generate some fake data

> eps <- rnorm(10, sd= 5)

> x <- c(1:10)

> y <- 2+2\*x + eps

# Load libraries required

> library(quantreg)

> library(magrittr)

> eps <- rnorm(10, sd= 5)

> x <- c(1:10)

> y <- 2+2\*x + eps

# create a higher order function

> higher\_order\_function <- function(func){

+ func(y ~ x) %>% summary

+ }

>

# Give as an argument the function "lm"

> higher\_order\_function(lm)

Call:

func(formula = y ~ x)

Residuals:

Min 1Q Median 3Q Max

-12.0149 -0.7603 1.0969 2.7483 4.2373

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 1.3214 3.3338 0.396 0.70219

x 2.1690 0.5373 4.037 0.00375 \*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.88 on 8 degrees of freedom

Multiple R-squared: 0.6708, Adjusted R-squared: 0.6296

F-statistic: 16.3 on 1 and 8 DF, p-value: 0.003751

# Now give as an argument the function rq (for regression quantile)

> higher\_order\_function(rq)

Call: func(formula = y ~ x)

tau: [1] 0.5

Coefficients:

coefficients lower bd upper bd

(Intercept) 3.80788 -1.26475 6.15759

x 1.83968 1.59747 2.98423

It’s also quite safe to use in that if you provide a non-existent function it would not default to some unknown behavior but will return an error:

> higher\_order\_function(mm)

Error in eval(lhs, parent, parent) : object 'mm' not found

However, this function can be also written as a sequence of if statements, like so

> if\_function <- function(x,y, which\_reg){

+ if (which\_reg== "OLS") { lm(y~x) %>% summary }

+ else if (which\_reg== "LAD") { rq(y~x) %>% summary }

+ }

> if\_function(x,y, which\_reg= "OLS")

Call:

lm(formula = y ~ x)

Residuals:

Min 1Q Median 3Q Max

-12.0149 -0.7603 1.0969 2.7483 4.2373

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 1.3214 3.3338 0.396 0.70219

x 2.1690 0.5373 4.037 0.00375 \*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.88 on 8 degrees of freedom

Multiple R-squared: 0.6708, Adjusted R-squared: 0.6296

F-statistic: 16.3 on 1 and 8 DF, p-value: 0.003751

> if\_function(x,y, which\_reg= "LAD")

Call: rq(formula = y ~ x)

tau: [1] 0.5

Coefficients:

coefficients lower bd upper bd

(Intercept) 3.80788 -1.26475 6.15759

x 1.83968 1.59747 2.98423

Using higher-order functions does not seem to create any additional computational cost:

> library(microbenchmark)

> microbenchmark( higher\_order\_function(rq), if\_function(x, y, "LAD") )

Unit: milliseconds

expr min lq mean median uq

higher\_order\_function(rq) 1.463210 1.498967 1.563553 1.527253 1.624969

if\_function(x, y, "LAD") 1.468262 1.498464 1.584453 1.618997 1.644462

max neval

2.280419 100

2.082765 100

> microbenchmark( higher\_order\_function(lm), if\_function(x, y, "OLS") )

Unit: microseconds

expr min lq mean median uq max

higher\_order\_function(lm) 916.858 928.8825 946.9838 935.3930 955.791 1025.575

if\_function(x, y, "OLS") 918.674 928.1260 953.2587 938.0465 958.284 1433.167

neval

100

100

So you can make your code more concise with little computational overhead.