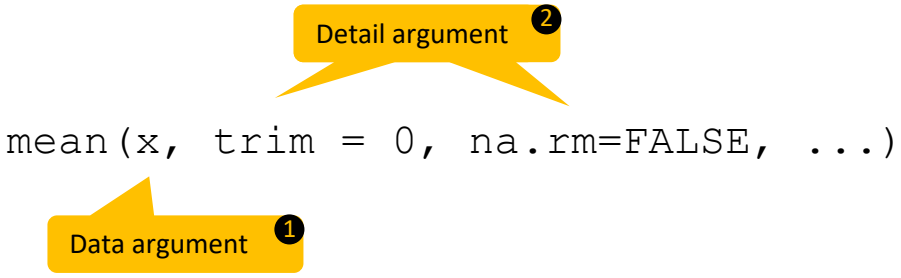
Advanced techniques for creating functions

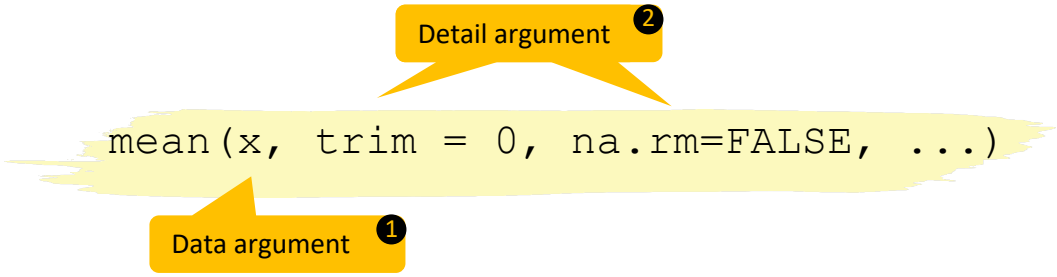
Functions can have two types of arguments

1. **Data arguments**
   - Supply data to compute the function.
   - Make sure data arguments are listed first.
2. **Detail arguments**
   - Control the details of the computation.
   - Detail arguments should have a default.



Functions can have two types of arguments

1. **Data arguments**
   - Supply data to compute the function.
   - Make sure data arguments are listed first.
2. **Detail arguments**
   - Control the details of the computation.
   - Detail arguments should have a default.



Advanced option: Return multiple objects

```
KPIs <- function(x,r,c){
            x <- as.data.frame(x)
            profit <- sum(x[,r],na.rm=T) - sum(x[,c],na.rm=T)
            roi <- profit/    sum(x[,c],na.rm=T)*100
            return(list(profit, roi))
               }

KPIs(myData, "PurchAmount", "Cost")
```

1 Use a list to return multiple objects

OUTPUT:

```
[[1]]
[1] 10077368

[[2]]
[1] 115.7332
```

2 Profit

3 ROI

Advanced option: Return multiple objects

```
KPIs <- function(x,r,c){
            x <- as.data.frame(x)
            profit <- sum(x[,r],na.rm=T) - sum(x[,c],na.rm=T)
            roi <- profit/    sum(x[,c],na.rm=T)*100
            return(list(profit, roi))
               }

KPIs(myData, "PurchAmount", "Cost")
```

1 Use a list to return multiple objects

OUTPUT:

```
[[1]]
[1] 10077368

[[2]]
[1] 115.7332
```

2 Profit

3 ROI

Sidenote: Extract components from lists

```
KPI_result <- KPIs(myData,"PurchAmount","Cost")
```

```
KPI_result[[1]]
```

Use double square brackets to get the object stored in a certain list slice

OUTPUT:

```
[1] 10077368
```

```
KPI_result[[2]]
```

OUTPUT:

```
[1] 1.157332
```