

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.

```
mean(x, trim = 0, na.rm=FALSE, ...)
```

Data argument ¹

Detail argument ²

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.

The diagram shows the function call `mean(x, trim = 0, na.rm=FALSE, ...)` on a yellow brushstroke background. A yellow callout box labeled 'Data argument' with a circled '1' points to the `x` parameter. Another yellow callout box labeled 'Detail argument' with a circled '2' points to the `trim = 0` parameter.

```
mean(x, trim = 0, na.rm=FALSE, ...)
```

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))  
}
```

Use a list to return multiple objects ¹

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

OUTPUT:

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

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

Profit ²

ROI ³

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))  
}
```

Use a list to return multiple objects ¹

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

OUTPUT:

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

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

Profit ²

ROI ³

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
```