# R Syntax 3: Functions

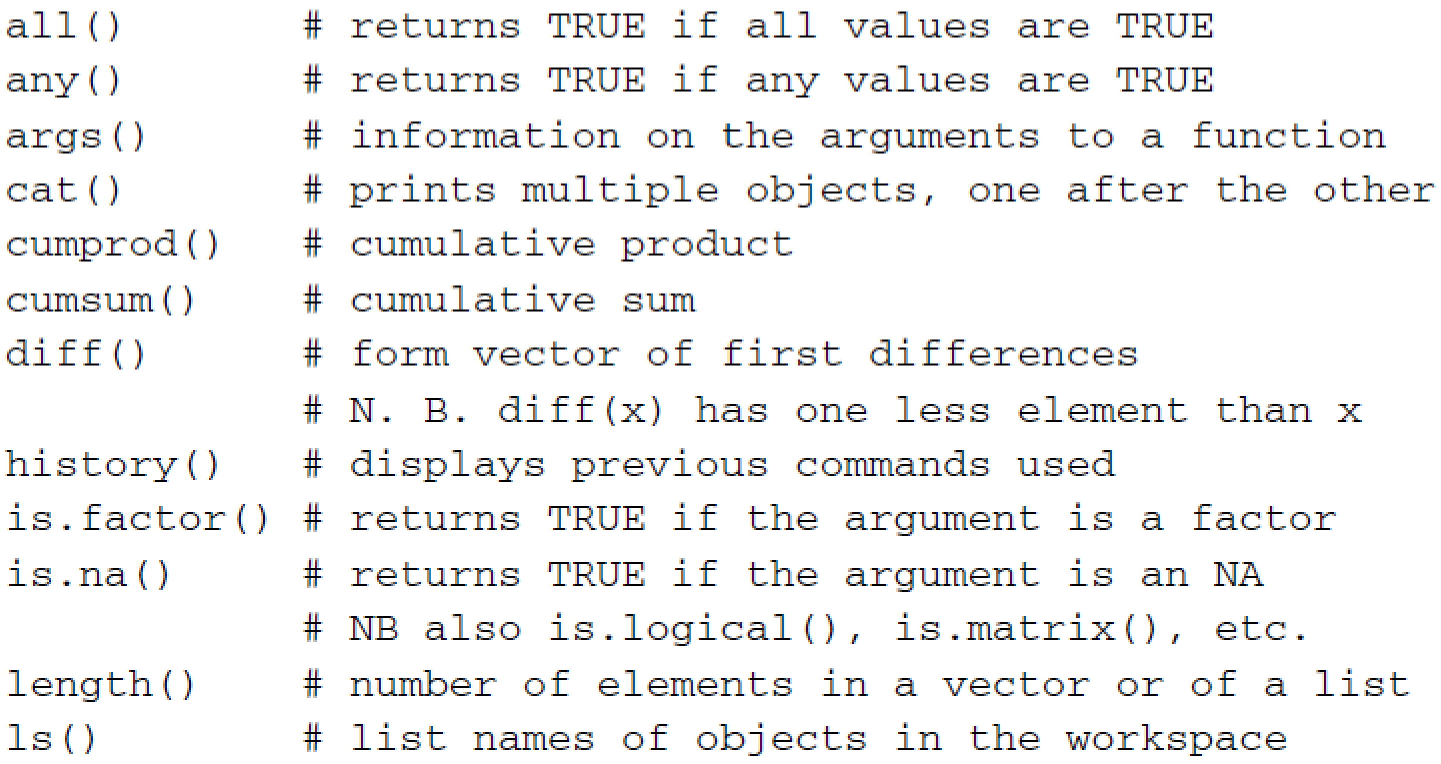
Pilsung Kang

School of Industrial Management Engineering

Korea University

# Function

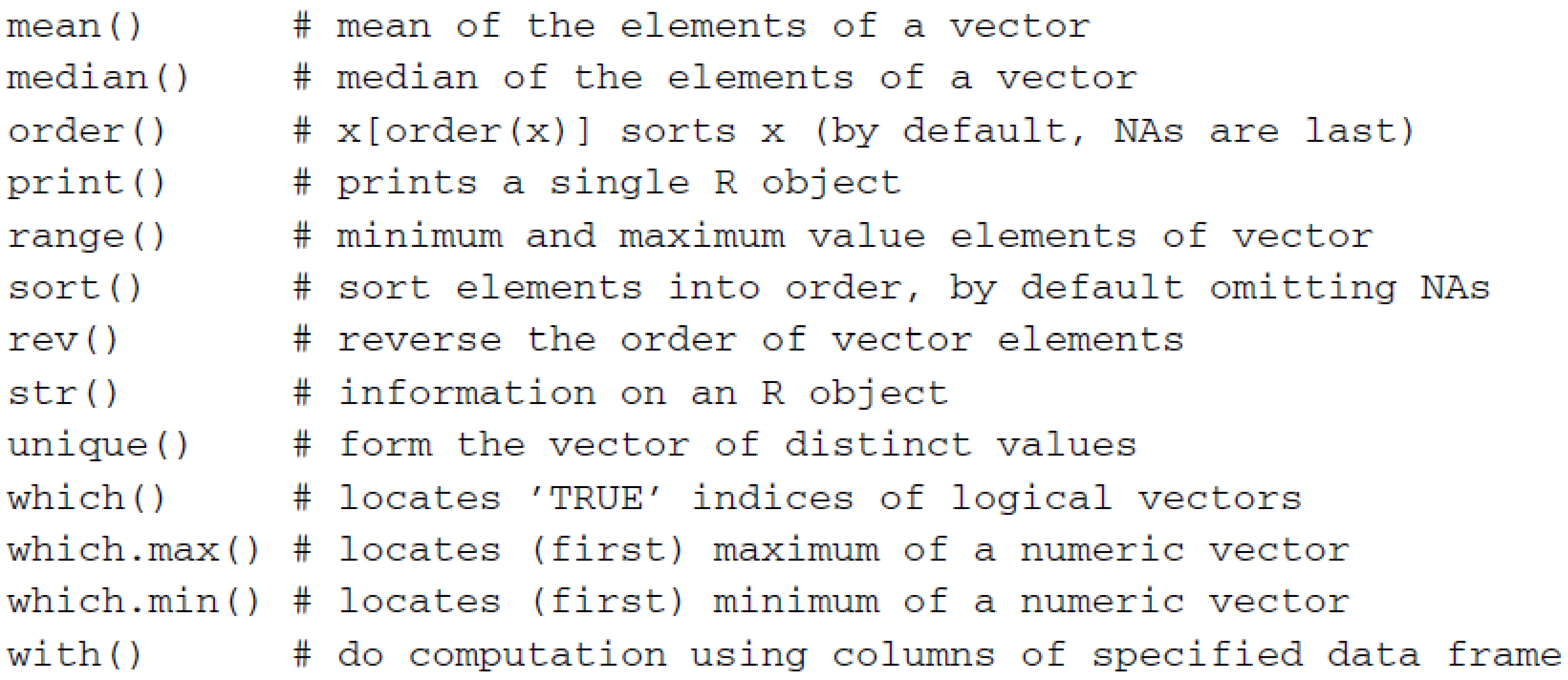
- Why functions?
- An incidental advantage of putting code into functions is that the workspace is not then cluttered with objects that are local to the function



```
all()         # returns TRUE if all values are TRUE
any()         # returns TRUE if any values are TRUE
args()        # information on the arguments to a function
cat()         # prints multiple objects, one after the other
cumprod()     # cumulative product
cumsum()      # cumulative sum
diff()        # form vector of first differences
              # N. B. diff(x) has one less element than x
history()     # displays previous commands used
is.factor()   # returns TRUE if the argument is a factor
is.na()       # returns TRUE if the argument is an NA
              # NB also is.logical(), is.matrix(), etc.
length()      # number of elements in a vector or of a list
ls()          # list names of objects in the workspace
```

# Function

- Why functions?
- An incidental advantage of putting code into functions is that the workspace is not then cluttered with objects that are local to the function



```
mean()       # mean of the elements of a vector
median()     # median of the elements of a vector
order()      # x[order(x)] sorts x (by default, NAs are last)
print()      # prints a single R object
range()      # minimum and maximum value elements of vector
sort()       # sort elements into order, by default omitting NAs
rev()        # reverse the order of vector elements
str()        # information on an R object
unique()     # form the vector of distinct values
which()      # locates 'TRUE' indices of logical vectors
which.max()  # locates (first) maximum of a numeric vector
which.min()  # locates (first) minimum of a numeric vector
with()       # do computation using columns of specified data frame
```

# Function

- Writing a function

```
function_name <- function(arguments) {
statement 1
statement 2
…
return(object)
}
```

✓ function_name: name that the function is referred to

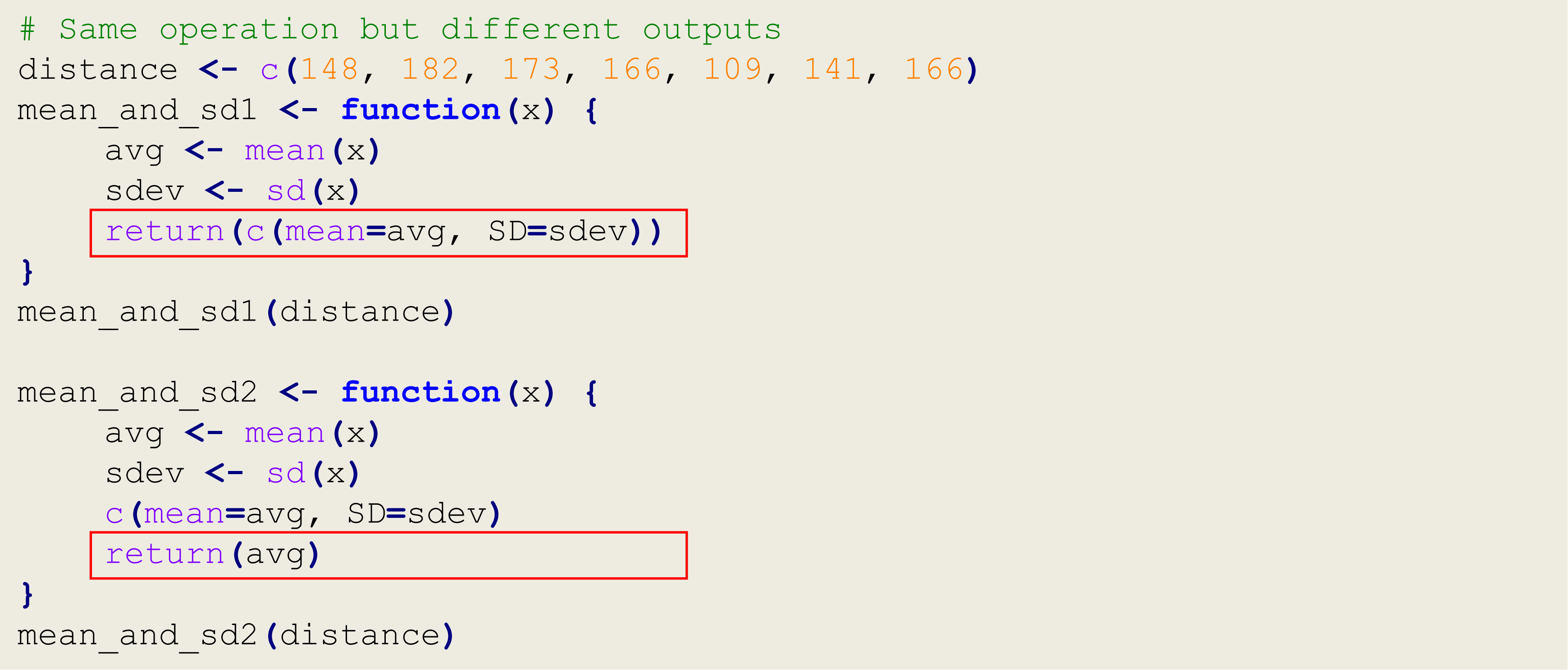
✓ arguments: inputs that a user should provide to run the function

✓ statements: operations running inside the function

✓ object: function output

# Function

- Same operations but different outputs



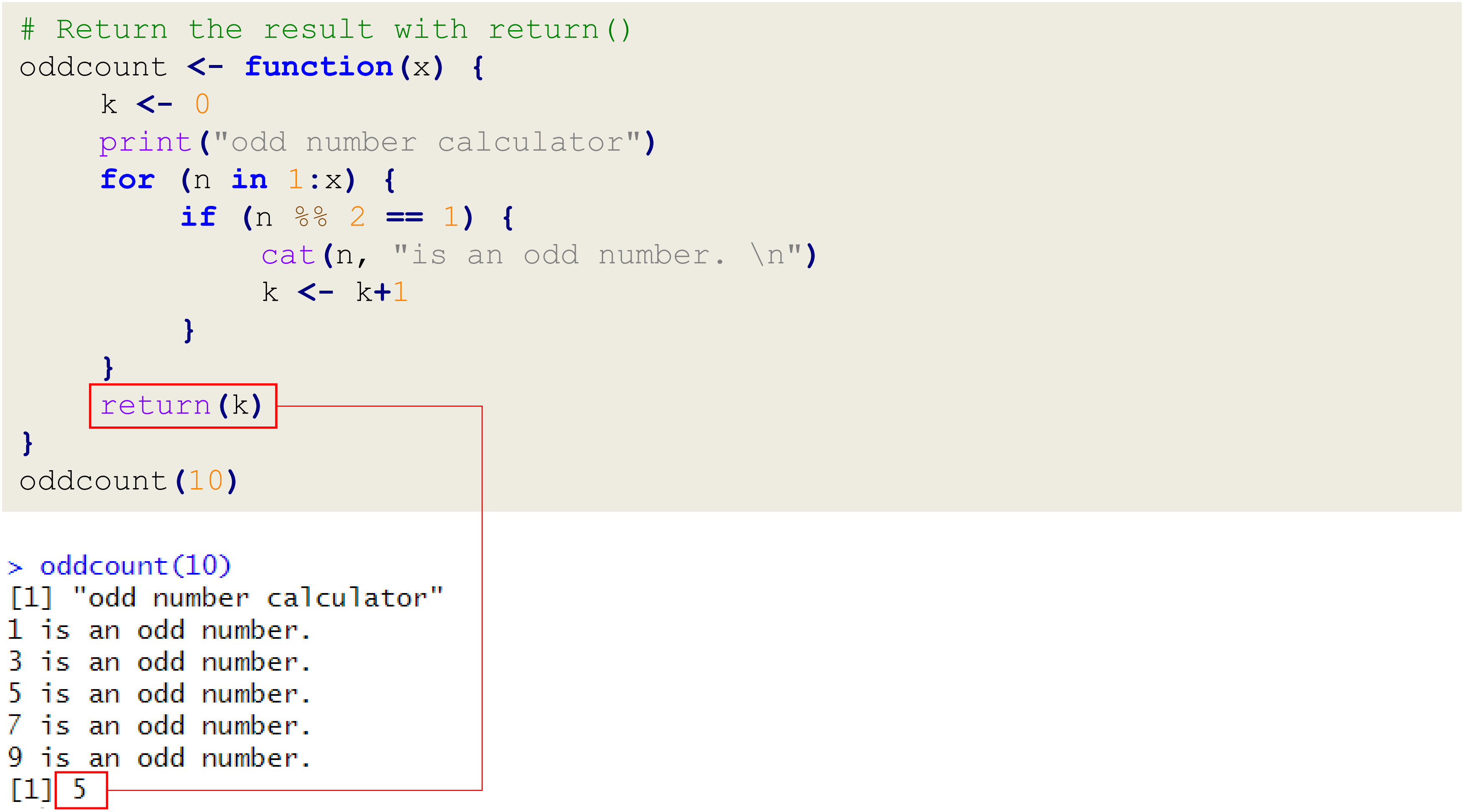
```r
# Same operation but different outputs
distance <- c(148, 182, 173, 166, 109, 141, 166)
mean_and_sd1 <- function(x) {
    avg <- mean(x)
    sdev <- sd(x)
    return(c(mean=avg, SD=sdev))
}
mean_and_sd1(distance)

mean_and_sd2 <- function(x) {
    avg <- mean(x)
    sdev <- sd(x)
    c(mean=avg, SD=sdev)
    return(avg)
}
mean_and_sd2(distance)
```

✓ Both functions take a vector and compute its mean and standard deviation

- First function returns both mean and standard deviation
- Second function only returns the mean

# Function

- Function output with return( ) instruction

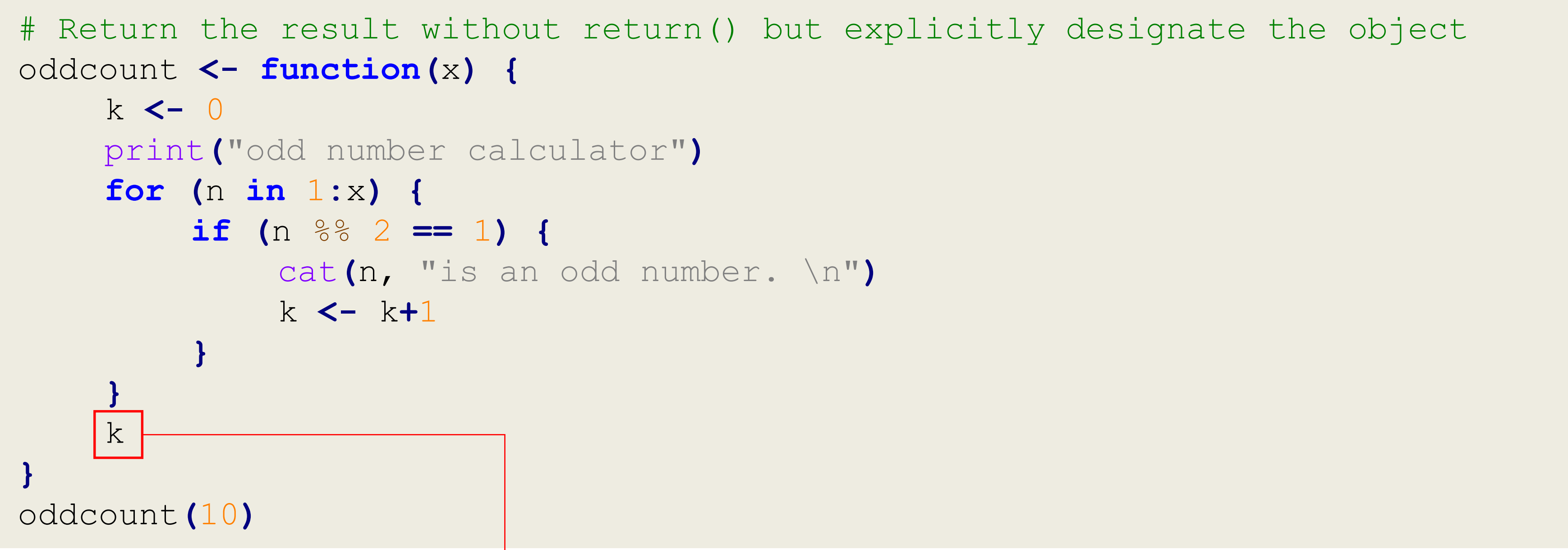


```r
# Return the result with return()
oddcount <- function(x) {
     k <- 0
     print("odd number calculator")
     for (n in 1:x) {
          if (n %% 2 == 1) {
               cat(n, "is an odd number. \n")
               k <- k+1
          }
     }
     return(k)
}
oddcount(10)
```

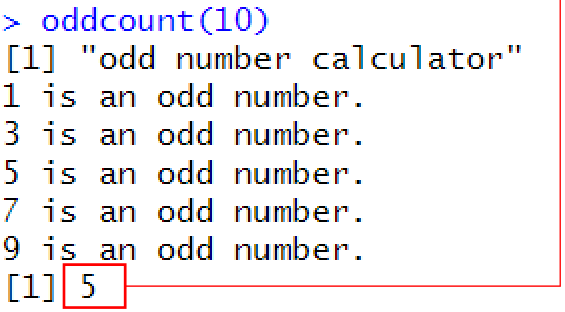
```
> oddcount(10)
[1] "odd number calculator"
1 is an odd number.
3 is an odd number.
5 is an odd number.
7 is an odd number.
9 is an odd number.
[1] 5
```

# Function

- Function output without return( ) instruction but explicitly designate the object



```
# Return the result without return() but explicitly designate the object
oddcount <- function(x) {
     k <- 0
     print("odd number calculator")
     for (n in 1:x) {
          if (n %% 2 == 1) {
               cat(n, "is an odd number. \n")
               k <- k+1
          }
     }
     k
}
oddcount(10)
```



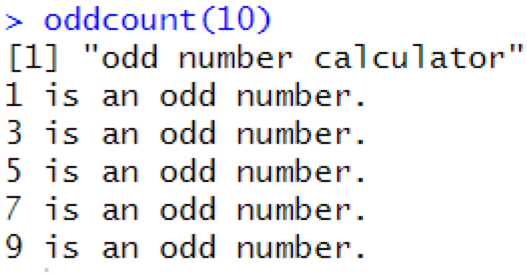
```
> oddcount(10)
[1] "odd number calculator"
1 is an odd number.
3 is an odd number.
5 is an odd number.
7 is an odd number.
9 is an odd number.
[1] 5
```

If return( ) is not used,
the final object inside the function is returned
(not recommended)

# Function

- Function output without return( ) instruction and object designation

```r
# Return the result without return() and explicit designation
oddcount <- function(x) {
     k <- 0
     print("odd number calculator")
     for (n in 1:x) {
          if (n %% 2 == 1) {
               cat(n, "is an odd number. \n")
               k <- k+1
          }
     }
}
oddcount(10)
```



```
> oddcount(10)
[1] "odd number calculator"
1 is an odd number.
3 is an odd number.
5 is an odd number.
7 is an odd number.
9 is an odd number.
```

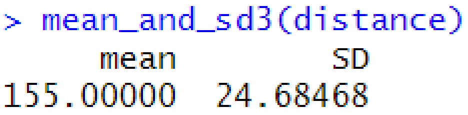
This function returns nothing
because the condition for the last if statement (when n == 10) is not true

# Function

- Function arguments: default arguments

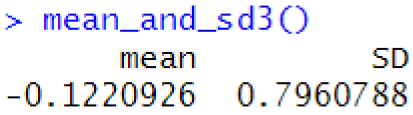
```
mean_and_sd3 <- function(x = rnorm(10)) {
    avg <- mean(x)
    sdev <- sd(x)
    return(c(mean=avg, SD=sdev))
}

mean_and_sd3(distance)
mean_and_sd3()
```

✓ If the argument is provided by a user, function statements run with the provided argument



```
> mean_and_sd3(distance)
     mean        SD
155.00000  24.68468
```

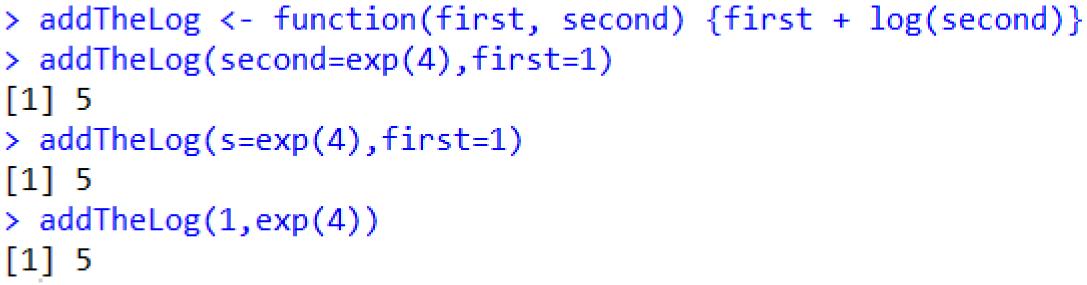
✓ If the argument is not provided, default function argument is activated



```
> mean_and_sd3()
      mean         SD
-0.1220926  0.7960788
```

# Function

- Function arguments
  - ✓ Each argument has its own name
  - ✓ Name is used to access the corresponding argument within function
  - ✓ Three possible ways to assign the argument
    - ▪ Exact name
    - ▪ Partially matching names (not recommended)
    - ▪ Argument order



```
> addTheLog <- function(first, second) {first + log(second)}
> addTheLog(second=exp(4),first=1)
[1] 5
> addTheLog(s=exp(4),first=1)
[1] 5
> addTheLog(1,exp(4))
[1] 5
```

# Function

- Function example 1
  - ✓ Question: from a vector consisting of only 0 and 1, return the indices from which 1 repeatedly appears k times

| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

- ▪ If k = 2, the answer is (1,2,8,11,12)

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | **1** | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **1** | **1** | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | **1** | **1** | 0 | 1 | 1 | 1 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | **1** | **1** | 1 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | **1** | **1** |

# Function

- Function example 1

  ✓ Question: from a vector consisting of only 0 and 1, return the indices from which 1 repeatedly appears k times

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| Value | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

  - If k = 3, the answer is (1,11)

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | **1** | **1** | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | **1** | **1** | **1** |

  - If = 4, the answer is NULL

# Function

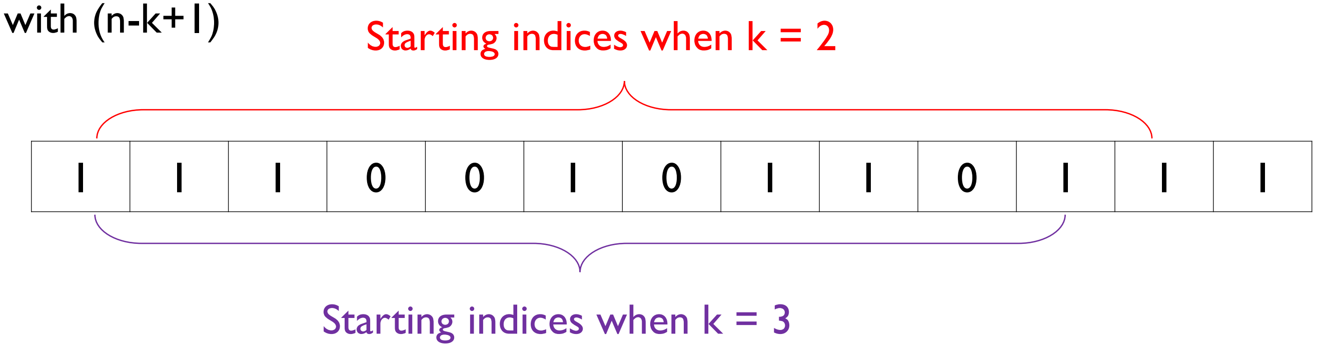
- Function example 1

```r
# Function example 1
findrepeats <- function(x, k) {
    n <- length(x)
    repeats <- NULL
    for (i in 1:(n-k+1)) {
        if(all(x[i:(i+k-1)] == 1)) repeats <- c(repeats, i)
    }
    return(repeats)
}
```

✓ This function takes two arguments: x (target vector) and k (number of repeats)

✓ We need to determine the search candidates

  - Since we have to check k consecutive numbers, the starting index begins with 1 and ends with (n-k+1)



Starting indices when k = 2

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

Starting indices when k = 3

# Function

- Function example 1

```
# Function example 1
findrepeats <- function(x, k) {
    n <- length(x)
    repeats <- NULL
    for (i in 1:(n-k+1)) {
        if(all(x[i:(i+k-1)] == 1)) repeats <- c(repeats, i)
    }
    return(repeats)
}
```

✓ If statement:

- if all k consecutive values starting from ith value are 1
- add the starting index to the variable `repeats`

✓ Example A: i = 2, k= 3 (condition is not satisfied)

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **1** | **1** | **0** | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

✓ Example B: i=11, k=3 (condition is satisfied)

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | **1** | **1** | **1** |

# Function

- Function example 2: Kendall's tau
  - ✓ Raw data: temperature and pressure recorded every hour

| Time | 10:00 | 11:00 | 12:00 | 13:00 | 14:00 |
|---|---|---|---|---|---|
| Temperature | 10 | 15 | 13 | 17 | 20 |
| Pressure | 900 | 920 | 890 | 940 | 920 |

  - ✓ What to do
    - ▪ Determine whether each indicator increases or decreases
    - ▪ Return the proportion of the events in which the change directions of the two indicators are the same

# Function

- Function example 2: Kendall’s tau

```r
# Example 2: Kendall's tau
findud <- function(v) {
    vud <- v[-1] - v[-length(v)]
    return(ifelse(vud >0, 1, -1))
}
```

✓ Inner function: determine whether the variable is increased or decreased

- For temperature

| Temperature | 10 | 15 | 13 | 17 | 20 |
|---|---|---|---|---|---|
| `v[-1]` | | 15 | 13 | 17 | 20 |
| `v[-length(v)]` | | 10 | 15 | 13 | 17 |
| `vud` | | 5 | -2 | 4 | 3 |
| `return(ifelse(vud >0, 1, -1))` | | 1 | -1 | 1 | 1 |

# Function

- Function example 2: Kendall's tau

```r
# Example 2: Kendall's tau
findud <- function(v) {
    vud <- v[-1] - v[-length(v)]
    return(ifelse(vud >0, 1, -1))
}
```

✓ Inner function: determine whether the variable is increased or decreased

- For pressure

| Pressure | 900 | 920 | 890 | 940 | 920 |
|---|---|---|---|---|---|
| `v[-1]` | | 920 | 890 | 940 | 920 |
| `v[-length(v)]` | | 900 | 920 | 890 | 940 |
| `vud` | | 20 | -30 | 50 | -20 |
| `return(ifelse(vud >0, 1, -1))` | | 1 | -1 | 1 | -1 |

# Function

- Function example 2: Kendall’s tau

```
udcorr <- function(x,y) {
    ud <- lapply(list(x,y), findud)
    return(mean(ud[[1]] == ud[[2]]))
}

temp <- c(10, 15, 13, 17, 20)
pressure <- c(900, 920, 890, 940, 920)
udcorr(temp,pressure)
```

| | | | | |
|---|---|---|---|---|
| ud[[1]] | 1 | -1 | 1 | 1 |
| ud[[2]] | 1 | -1 | 1 | -1 |
| ud[[1]] == ud[[2]] | 1 | 1 | 1 | 0 |

- The final output = 0.75 (3/4)

ANY
questions ?