4. Calling Functions in R

CT1100 - J. Duggan

Functions

- A function is a group of instructions that:
  - takes input,
  - uses the input to compute other value, and
  - returns a result (Matloff 2009).
- Functions are a fundamental building block of R
- Users of R should adopt the habit of creating simple functions which will make their work more effective and also more trustworthy (Chambers 2008).
- Two approaches:
  - Use R functions
  - Write your own functions

R - A function "ecosystem"

- John Chambers: "To understand computations in R, two slogans are helpful:
  - Everything that exists is an object.
  - Everything that happens is a **function call.**"

```
x <- 1:6
sqrt(x)
```

```
## [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490
```

```
min(x)
```

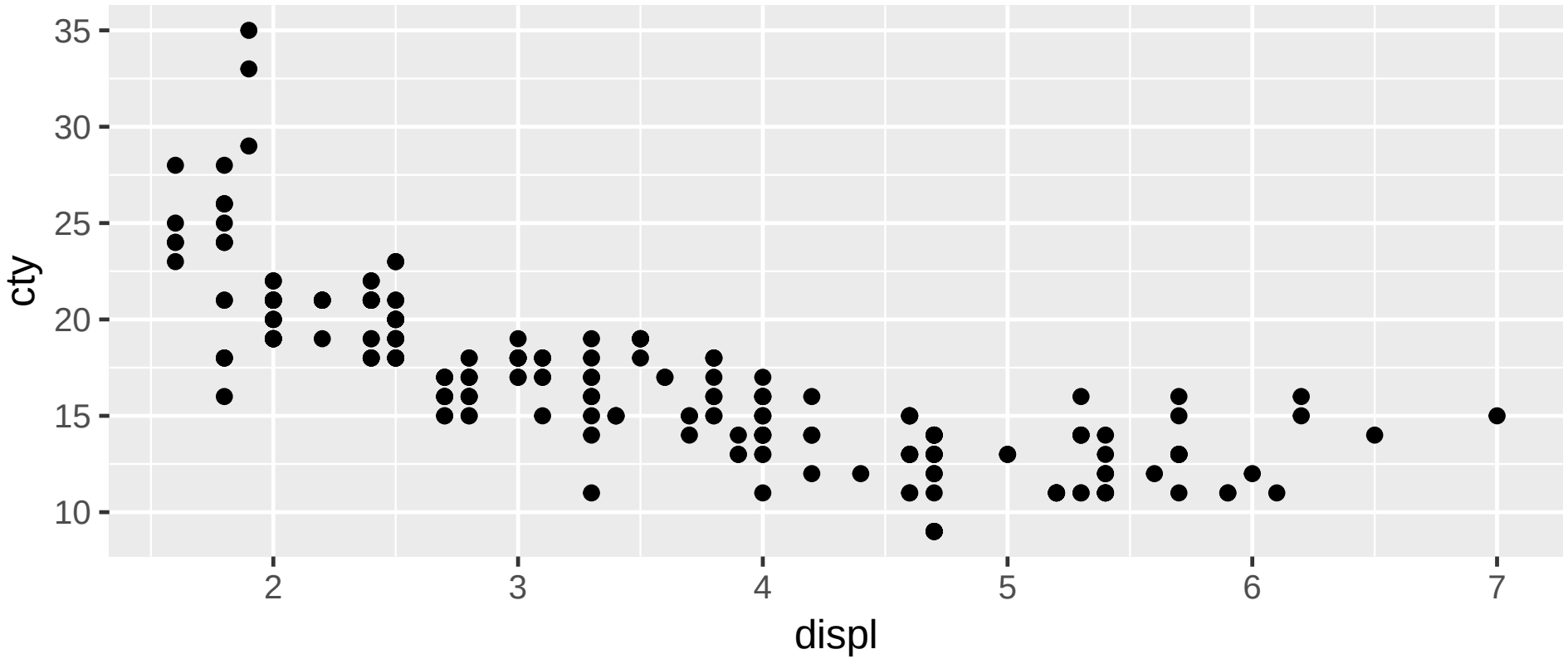
```
## [1] 1
```

```
mean(x)
```

```
## [1] 3.5
```

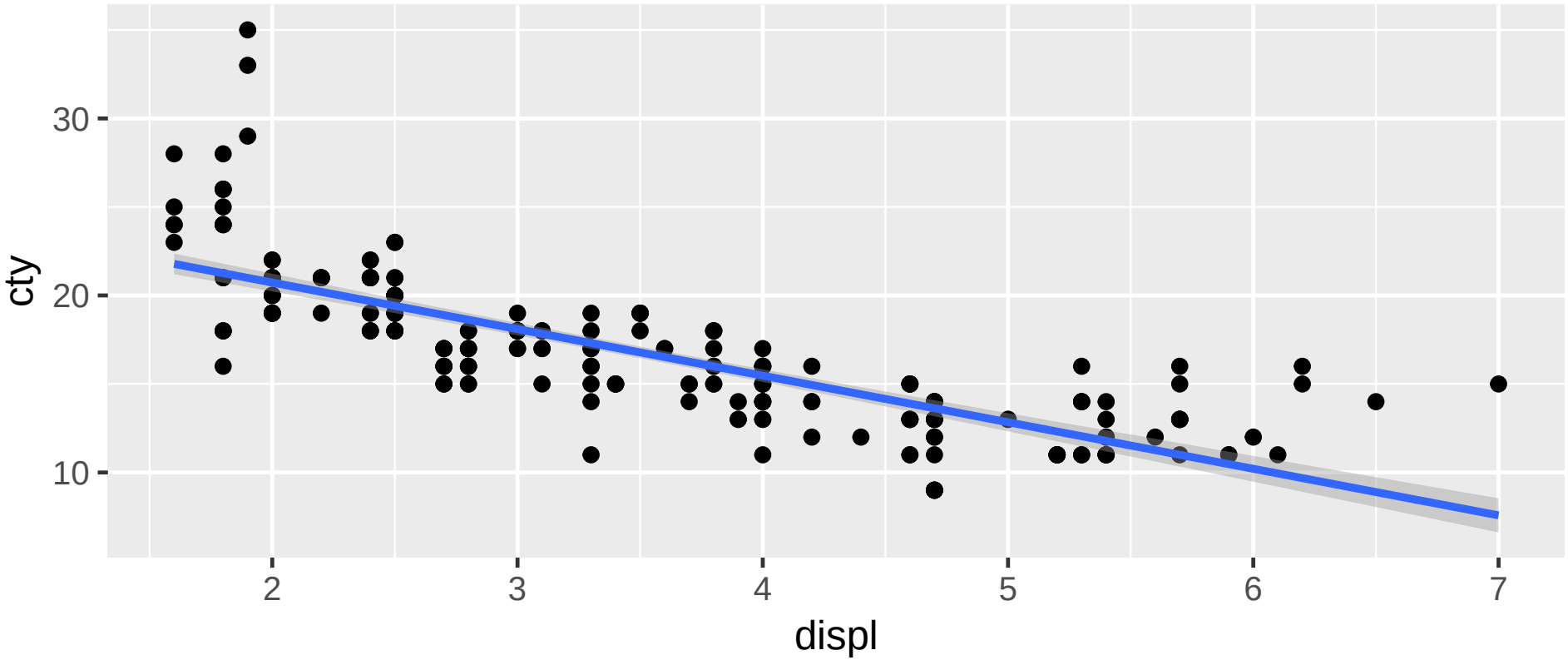
Calling a function to create a plot (from package ggplot2)

```
library(ggplot2)
ggplot(data=mpg)+
  geom_point(mapping=aes(x=displ,y=cty))
```



Adding information to the plot...

```
library(ggplot2)
ggplot(data=mpg,mapping=aes(x=displ,y=cty))+
  geom_point()+geom_smooth(method = "lm")
```



Writing Functions

- **function** (*arguments*) *expression*
- arguments gives the arguments, separated by commas.
- Expression (body of the function) is any legal R expression, usually enclosed in { }
- Last evaluation is returned
- return() can also be used, but usually for exceptions.

```
f <- function(x)x^2 # this function squares a vector
f(1:3)
```

```
## [1] 1 4 9
```

Challenge 1.4

Write a function that takes in a vector and returns a vector with no duplicates. Make use of the R function duplicated().

```
x <- c(1, 2, 3, 4, 5, 1)
duplicated(x)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE  TRUE
```

Function Arguments

- It is useful to distinguish between formal arguments and the actual arguments
  - Formal arguments are the property of the function
  - Actual arguments can vary each time the function is called.
- When calling functions, arguments can be specified by
  - Complete name
  - Partial name
  - Position
- Guidelines (Wickham 2015)
  - Use positional mapping for the first one or two arguments (most commonly used)
  - Avoid using positional mapping for less commonly used attributes
  - Named arguments should always come after unnamed arguments

Function Arguments - Example

```
f1 <- function(arg1, arg2, arg3) arg1 * arg2  + arg3
f1(2, 3, 4) # positional
```

```
## [1] 10
```

```
f1(2, arg3=4,3) # name for arg3
```

```
## [1] 10
```

```
f1(arg3=4, arg2=3, 2) # name for arg2, arg3
```

```
## [1] 10
```

Functions are objects

- Functions are first class objects, so they can be passed to other functions
- Provides flexibility, and widely used in R

```
f1 <- function(f,v)f(v) # f is a function object

f1(min,c(2,4,6,7))
```

```
## [1] 2
```

```
f1(max,c(2,4,6,7))
```

```
## [1] 7
```

Challenge 1.4

Write a function that takes in a vector and returns a vector with no duplicates. Make use of the R function duplicated().

```
x <- c(1, 2, 3, 4, 5, 1)
duplicated(x)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE  TRUE
```

Summary

- Functions are a fundamental building block of R
- Functions:
  - are declared using the function reserved word
  - are objects
- Functions can access variables within the environment where they are created
- Functionals are functions that takes a function as an input and returns a vector as output (can be used as a looping structure)
- The apply family in R are functionals (**apply**, **sapply**, **lapply**)
- The package **purrr** is now being used instead of **apply**