



# **6. More Loops, Control Structures, and Bootstrapping**

**Ken Rice  
Tim Thornton**

University of Washington

*Seattle, July 2019*

# In this session

---

We will introduce additional looping procedures as well as control structures that are useful in R. We also provide applications to bootstrapping.

- Repeat and While loops,
- If-Then and If-Then-Else structures
- Introduction to the bootstrap, with examples

# Repeat loops

---

The repeat loop is an infinite loop that is often used in conjunction with a **break** statement that terminates the loop when a specified condition is satisfied. The basic structure of the repeat loop is:

```
repeat{  
    expression  
    expression  
    expression  
    if(condition) break  
}
```

# Repeat loops

---

Below is a repeat loop for printing the square of integers from 1 to 10.

```
i <- 1
repeat {
  print(i^2)
  i <- i+1
  if(i > 10) break
}
```

# While loops

---

The while loop is often used for executing a set of commands or statements repeatedly until a specific condition is satisfied.

The structure of a while loop consists of a boolean condition and statements that are written inside while loop brackets, for which repetitive execution is to be carried out until the condition of interest is satisfied:

```
while (condition) {  
    expression  
    expression  
    expression  
}
```

It is important to note that the while loop will first check that the condition is satisfied prior to executing a first iteration of the commands.

# While loops

---

Below is a while loop for printing out the first few Fibonacci numbers: 0, 1, 1, 2, 3, 5, 8, 13,..., where each number is the sum of the previous two numbers in the sequence.

```
a <- 0
b <- 1
print(a)
while (b < 50) {
  print(b)
  temp <- a + b
  a <- b
  b <- temp
}
```

# While loops

---

Below is a while loop that creates a vector containing the first 20 numbers in the Fibonacci sequence

```
x <- c(0,1)
n <- 20
while (length(x) < n) {
  position <- length(x)
  new      <- x[position] + x[position-1]
  x       <- c(x,new)
}
```

# If-Then and If-Then-Else structures

---

Sometimes a block of code in a program should only be executed if a certain condition is satisfied. For these situations, *if-then* and *if-then-else* structures can be used:

The *if-then* structure has the following general form:

```
if (condition) {  
    expression  
    expression  
}
```

The *if-then-else* structure extends the same idea:

```
if (condition) {  
    expression  
    expression  
} else {  
    expression  
    expression  
}
```



# If-Then and If-Then-Else structures

---

An example: an *if-then-else* statement that takes the square root of the product of two numbers  $x$  and  $y$ , if the product is positive:

```
x <- 3
y <- 7
if( (x<0 & y<0) | (x>0 & y>0) ){
  myval <- sqrt(x*y)
} else{
  myval <- NA
}
```

And the value of `myval` when  $x=3$  and  $y=7$  is:

```
> myval
[1] 4.582576
```

What is `myval` if  $x=2$  and  $y=-10$ ?

```
> myval
[1] NA
```

# Introduction to bootstrapping

---

Bootstrapping is a very useful tool when the distribution of a statistic is unknown or very complex.

Bootstrapping is a *non-parametric* (i.e. assumption-lite) re-sampling method for estimating standard errors, computing confidence intervals, and hypothesis testing.

The method is often used when sample sizes are small and *asymptotic* (i.e. large- $n$ ) approximations, may be difficult to apply.

“The bootstrap is a computer-based method for assigning measures of accuracy to sample estimates.” [B. Efron and R. J. Tibshirani, An Introduction to the Bootstrap, Boca Raton, FL: CRC Press, 1994.]

# Introduction to bootstrapping

---

Bootstrapping uses three steps:

- Resample a given data set **with replacement** a specified number of times, where each *bootstrap sample* is the same size as the original sample
- Calculate a statistic of interest for each of the bootstrap samples.
- The distribution of the statistic from the bootstrap samples can then be used to estimate standard errors, create confidence intervals, and to perform hypothesis testing with the statistic.

# Example: bootstrapping the median

---

We can bootstrap in R by going round a loop, using the `sample(x, size, replace, prob)` function at each iteration:

- `x` is a vector containing the items to be resampled.
- `size` specifies how many resamples to take: the default is the length of `x`
- `replace` determines if the sample will be drawn with or without replacement. The default value, `FALSE` i.e. sampling is performed without replacement
- `prob` lets us specify unequal probabilities of resampling each element of `x` – not needed here

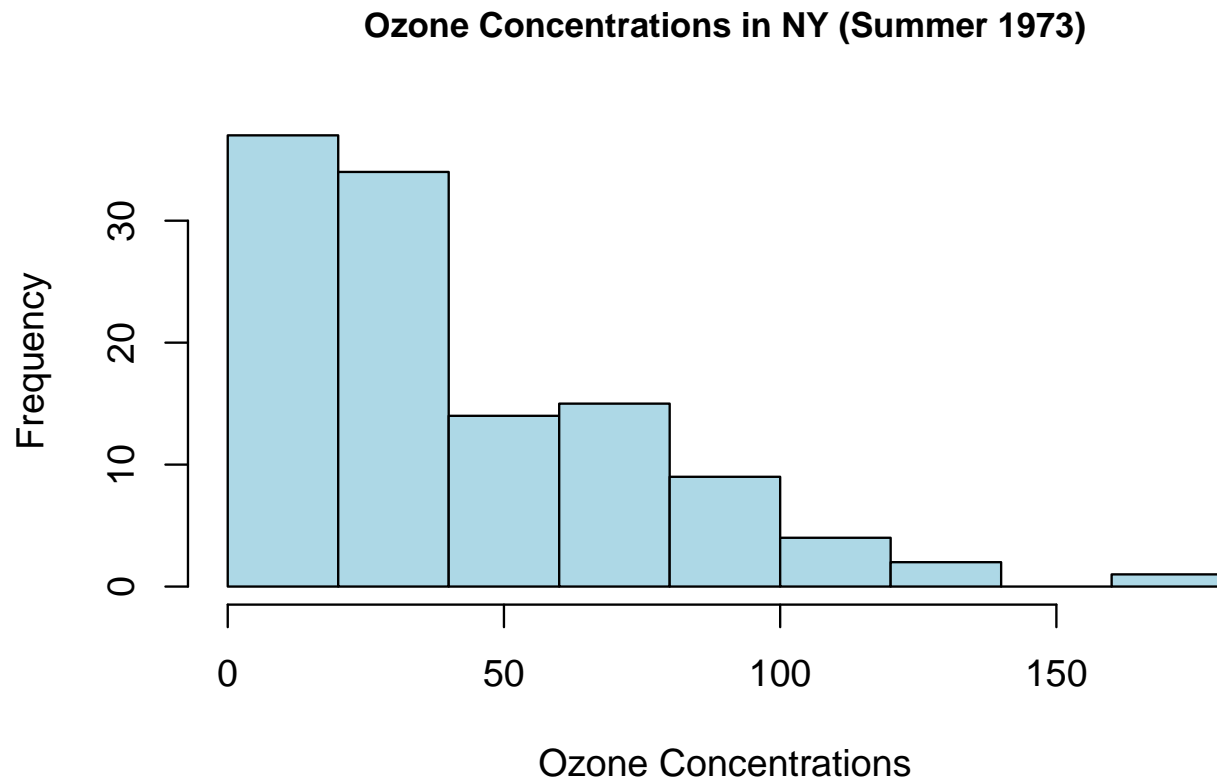
Bootstrapping uses resamples of the same size as the original data, sampling with replacement – so `sample(x, replace=TRUE)`

# Example: bootstrapping the median

---

Let's consider the *airquality* dataset again. Below is a histogram of the daily ozone concentrations in New York, summer 1973.

```
hist(airquality$Ozone,col="lightblue",xlab="Ozone Concentrations",  
main="Ozone Concentrations in NY (Summer 1973)")
```



What's the median ozone level?

How accurately do we know the median?

## Example: bootstrapping the median

---

First, let's work out the median;

```
> median(airquality$Ozone)
[1] NA
```

Several ozone concentration values are missing, but if we take the median of the 116 observed values;

```
> median(airquality$Ozone,na.rm=TRUE)
[1] 31.5
```

How might this value differ, in other similar experiments? We will use the bootstrap to estimate its distribution, and to provide a 95% confidence interval for the median.

## Example: bootstrapping the median

---

To make the code easier to read, make a vector of the ozone concentrations with missing values excluded:

```
ozone <- airquality$Ozone[ !is.na(airquality$Ozone) ]
```

Using a `for()` loop, we can create 10,000 bootstrap samples and calculate the median for each sample:

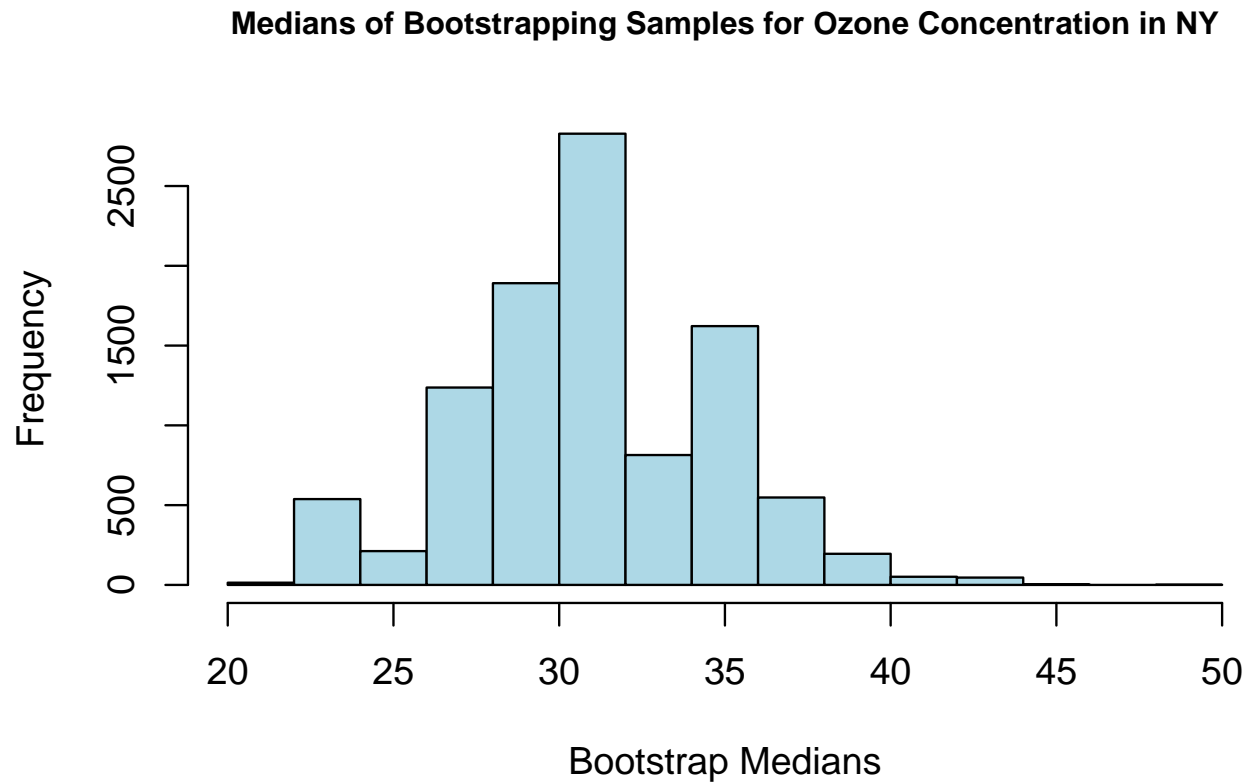
```
nboot <- 10000      # number of bootstrap samples
bootstrap.medians <- rep(NA, nboot)
set.seed(10)
for(i in 1:nboot){
  bootstrap.medians[i] <- median(sample(ozone,replace=TRUE))
}
```

# Example: bootstrapping the median

---

What do the medians look like? How do they compare with original 'raw' data?

```
hist(bootstrap.medians,col="lightblue",xlab="Bootstrap Medians",  
main="Bootstrap Medians for Ozone Concentrations in NY",cex.main=.8)
```



10,000 of them,  
not 116

Much less skewed  
than raw data

Much less variable  
than raw data



## Example: bootstrapping the median

---

The 95% confidence interval is given by the .025 and .975 quantiles of those bootstrap medians;

```
> quantile(bootstrap.medians, c(0.025, 0.975) )  
2.5% 97.5%  
23.5  39.0
```

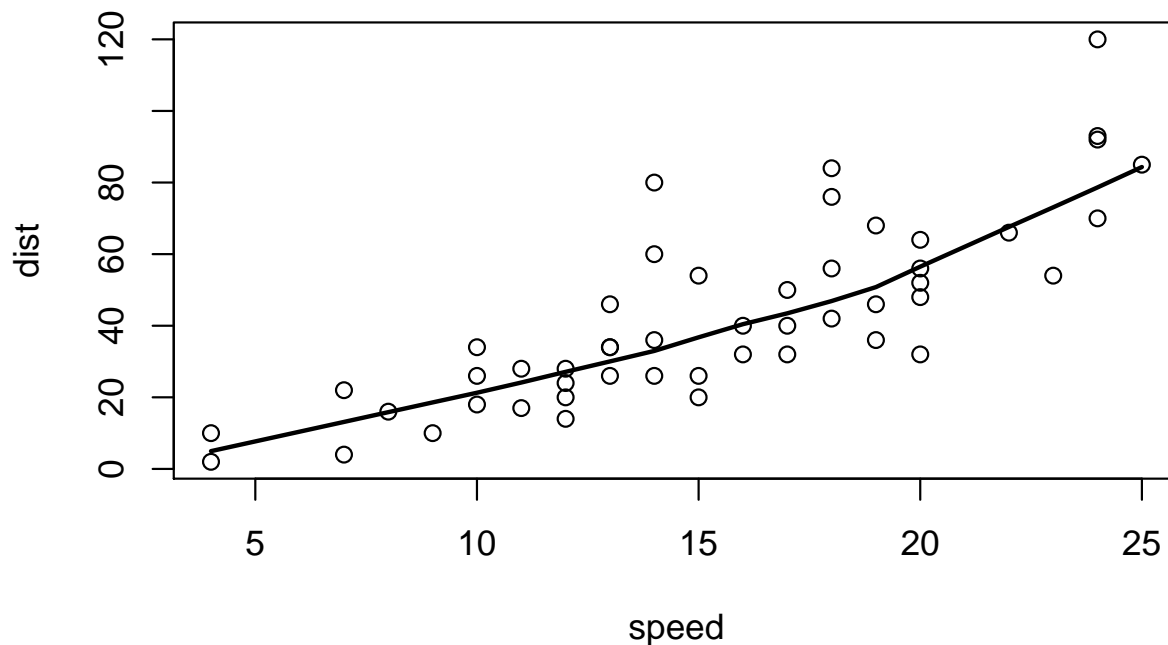
- Could read off from the previous graph
- (23.5, 39.0) is a range of median values we might expect to see (i.e. the uncertainty in the medians) if repeating the experiment many times
- This method does assume that the ozone measurement on different days is independent so probably understates uncertainty, here!

## Example: bootstrap for lowess curve

---

The bootstrap is a very powerful idea. For a more sophisticated example, recall the cars data, and the line we put through it;

```
data(cars)
plot(dist~speed,data=cars)
with(cars, lines(lowess(speed, dist), col="tomato", lwd=2))
```

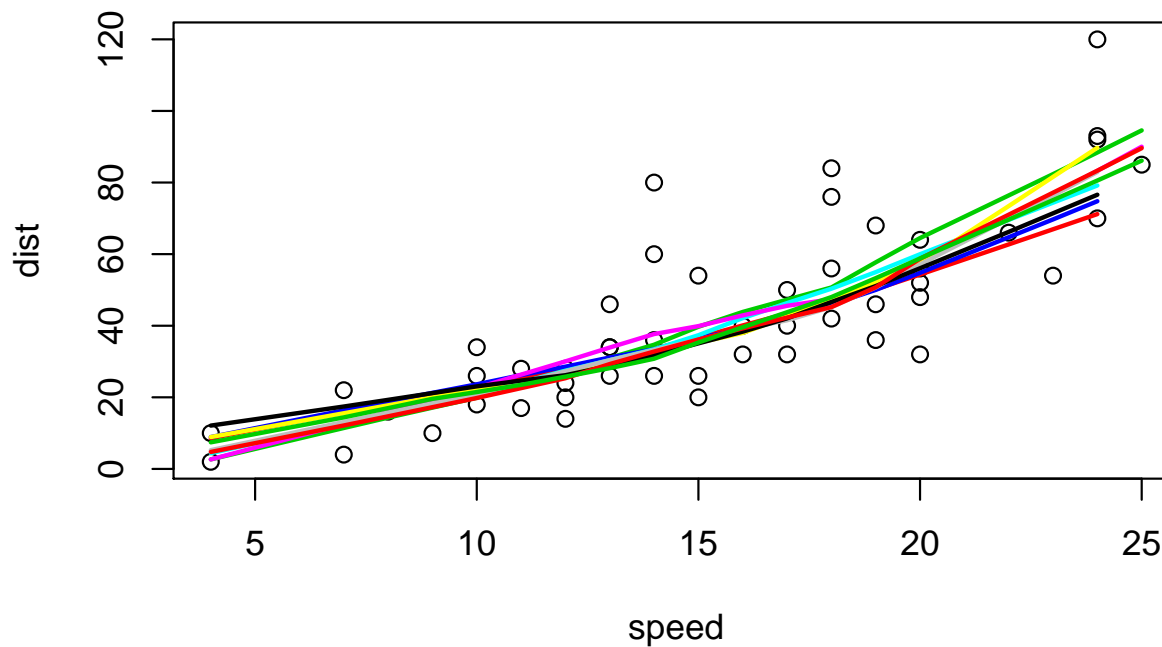


# Example: bootstrap for lowess curve

---

To bootstrap the *curve*, we resample entire observations;

```
m <- dim(cars)[1]      # obtain the sample size
nboot <- 20
for(i in 1:nboot){
  mysample <- sample(1:m, replace=TRUE) # i.e. which rows are resampled?
  with(cars[mysample,],
    lines(lowess(speed, dist), col=(i+1), lwd=2)
  )}
}
```



## Example: bootstrap for lowess curve

---

For a smoother version, note `lowess()` only produces output at the *sampled* points – so we extrapolate to the others using `approx()`;

```
nboot <- 1000
boot.speed <- matrix(NA, 1000, m) # for storing the curve's value at all m points
set.seed(1314) # Battle of Bannockburn
for(i in 1:nboot){
  mysample <- sample(1:m,replace=TRUE)
  low1 <- with(cars, lowess(speed[mysample], dist[mysample]))
  low.all <- approx(low1$x, low1$y, xout=cars$speed, rule=2)
  boot.speed[i,] <- low.all$y
}
```

Now work out the lower and upper ranges of the lines, at all  $m$  values of speed;

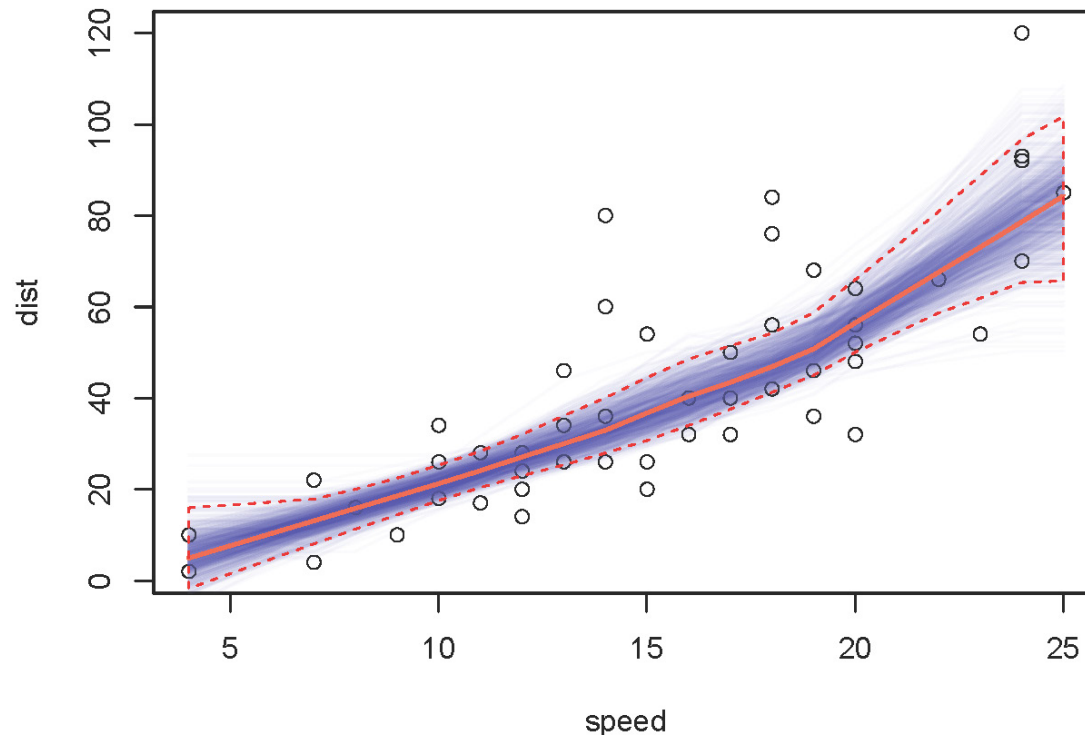
```
upper <- rep(NA, m)
lower <- rep(NA, m)
for(j in 1:m){
  upper[j] <- quantile(boot.speed[,j], 0.975)
  lower[j] <- quantile(boot.speed[,j], 0.025)}
```

# Example: bootstrap for lowess curve

---

Finally, make a cool blue picture, using transparency;

```
plot(dist~speed,data=cars)
for(i in 1:nboot){
  lines(x=cars$speed, y=boot.speed[i,], col= adjustcolor("blue", alpha.f=0.02)) }
with(cars, lines(lowess(speed, dist), col="tomato", lwd=2)) # raw data lowess
polygon(x=c(cars$speed, rev(cars$speed)), y=c(upper, rev(lower)),
        density=0, col="red", lty=2) # pointwise 95% conf ints
```



# Summary

---

- `while{}` and `repeat{}` are useful tools for looping until a condition is satisfied
- *if-then* and *if-then-else* structures allow blocks of code to be executed under different specified conditions
- Bootstrapping is a powerful statistical technique for expressing accuracy/inaccuracy. (*Almost* all other methods used for this can be thought of as approximations to some form of bootstrap)
- Bootstrapping can be implemented in a few lines of R, using loops and the `sample()` function