Introduction to R and Statlab Server

Xuelong Wang

August 30, 2018

How to connect to the Statlab Server

- Fixed IP address 131.193.178.77
- hostname: statlab.math.uic.edu
- Mac users
  - Terminal.app
  - ssh `username@statlab.math.uic.edu`
- Windows users
  - PuTTY
  - open source SSH client for windows
  - `http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`

How to use the R.Studio

- R
  - Using Terminal or PuTTY
  - just type R
- Rstudio
  - Web browser e.g. Chrome
  - `http://statlab.math.uic.edu:8787/`
- some notes about using Rstudio
  - save all the documents
  - save and delete all objects in the Global Enviroment (rm(list=ls()))
  - using top and kill the terminate your R sessions

Basics concepts of R

Object-oriented

R is an object-oriented language

- Everything in R is an object and each object has a class
  e.g. matrix, list, data.frame,etc. . .
- For different classes, R has different operations which are called "Method"

An example of Linear Regression

Mathematical equation

$$Y = \beta_0 + \beta_1 X + \epsilon,$$

Where β_0 is the intercept and β_1 is the slope

Goal

Estimate the coefficients and prediction

Example Cars' Speed and distances to stop

```
?cars
```

Mathematical equation

$$Y_{dist} = \beta_0 + \beta_{speed} X_{speed} + \epsilon,$$

Where β_0 is the intercept and β_{speed} is the slope

Goal

Using a car's speed to predict the distance it will use to stop

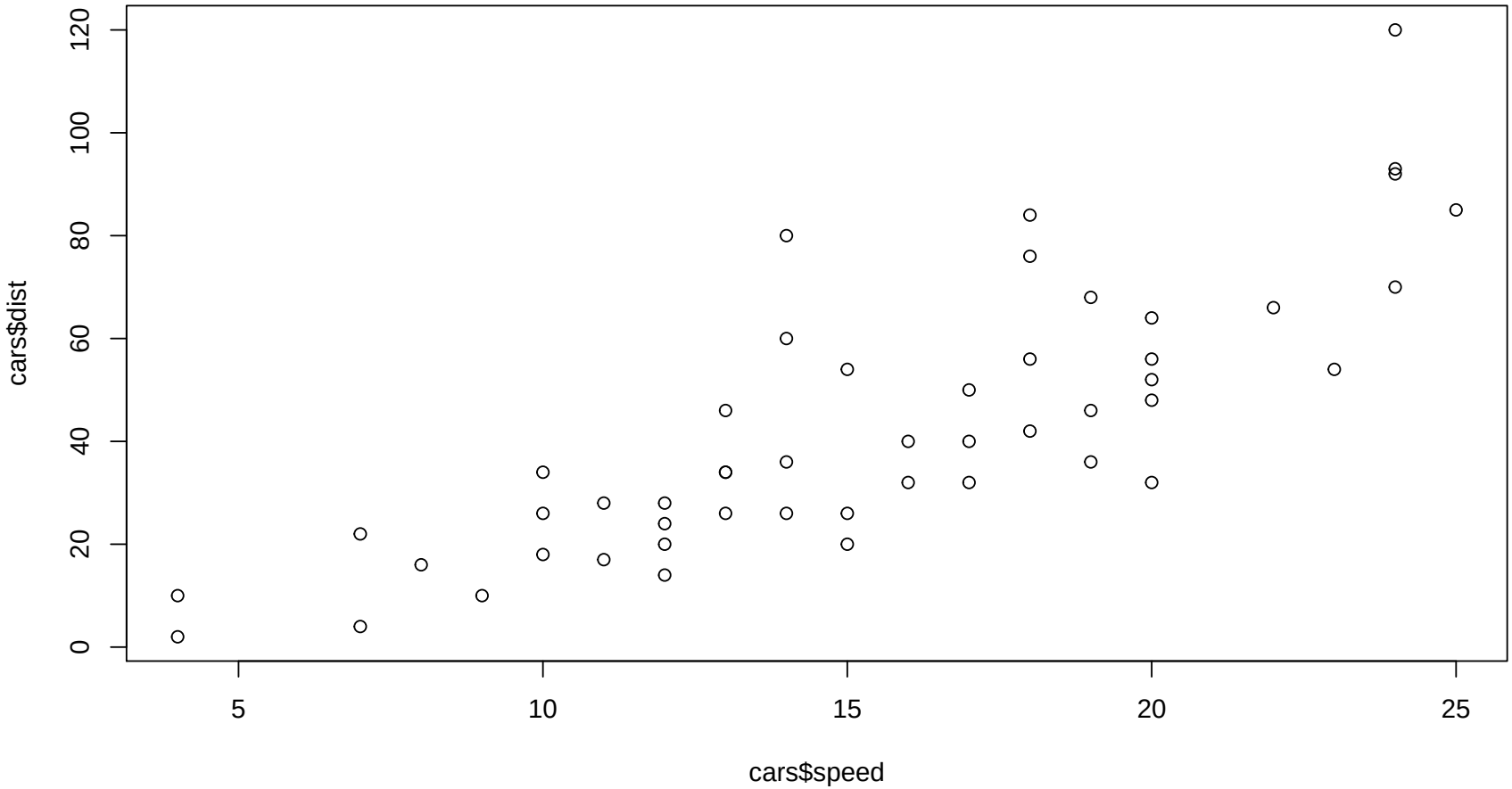
- esitmate the β_0 and β_{speed}

Check the data

```
head(cars)
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
class(cars)
## [1] "data.frame"
```

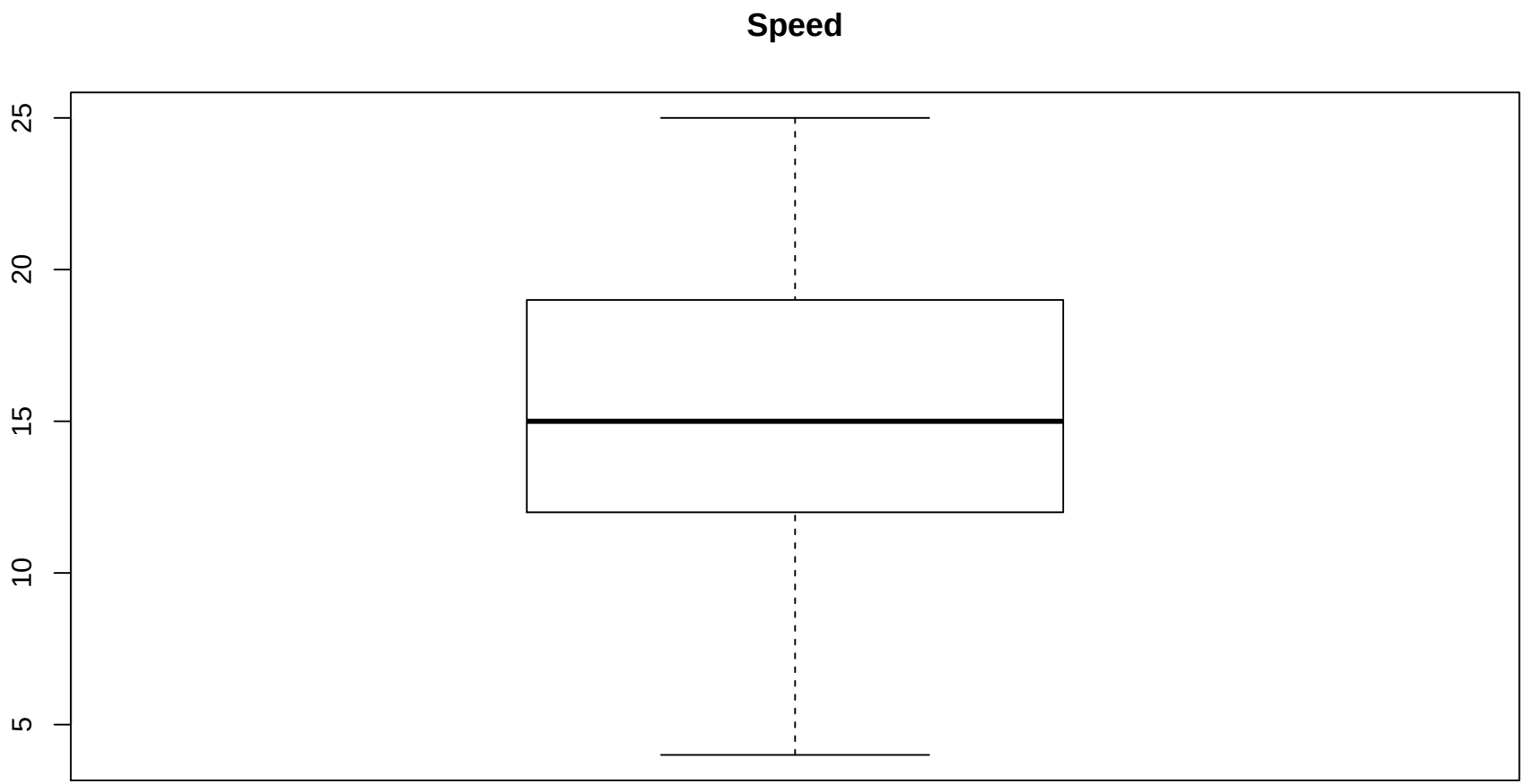
Plot X and Y together

```
plot(x = cars$speed, y = cars$dist)
```



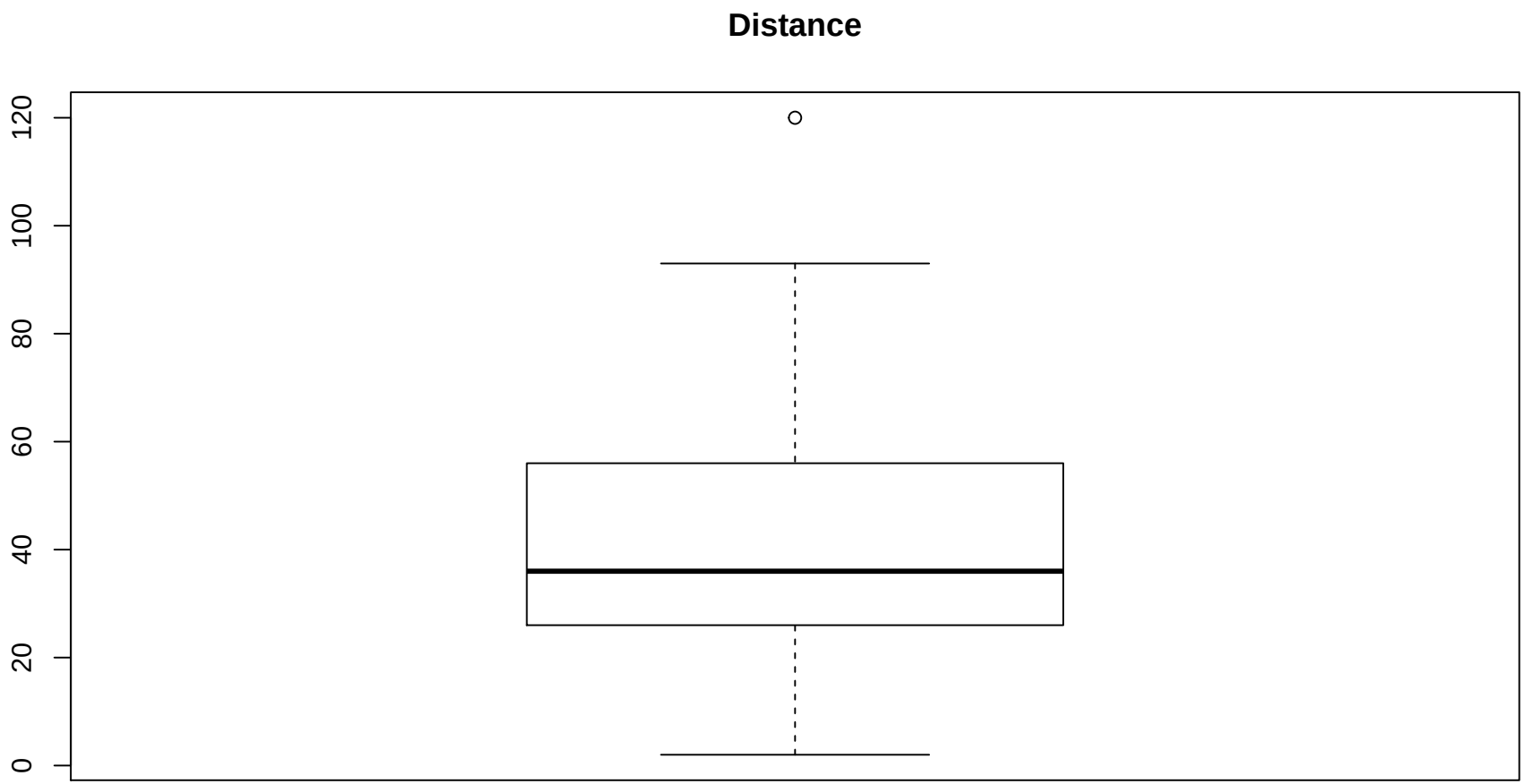
Plot a box-plot of speeds for outliers

```
boxplot(cars$speed, main="Speed")
```



Plot a box-plot of distances for outliers

```
boxplot(cars$dist, main="Distance")
```

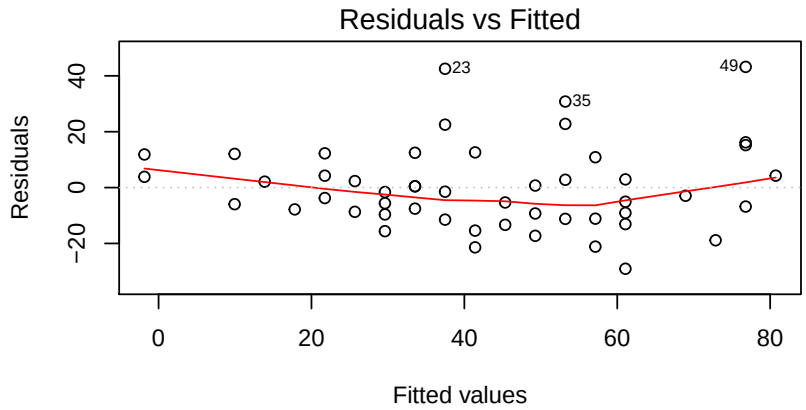


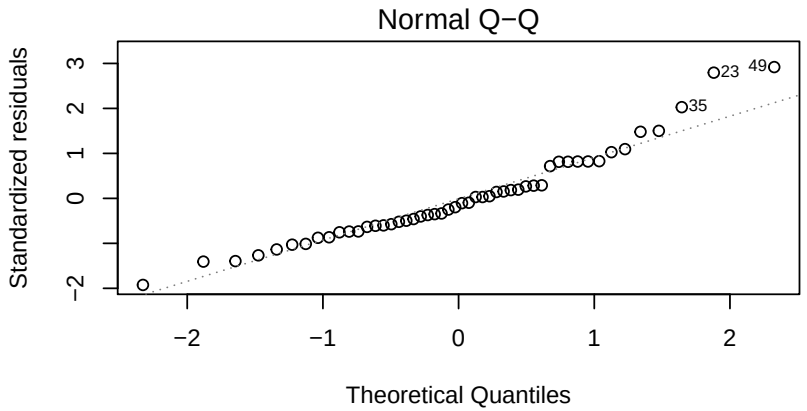
Fit the model

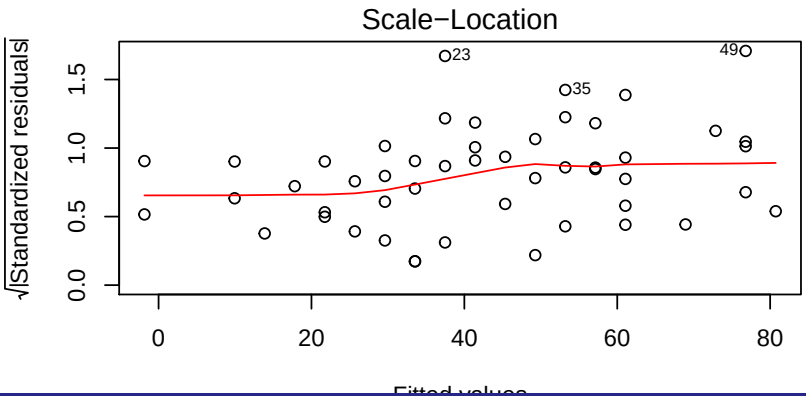
```
help(lm)
class(lm)
## [1] "function"
linearMod <- lm(dist ~ speed, data = cars)
class(linearMod)
## [1] "lm"
print(linearMod)
##
## Call:
## lm(formula = dist ~ speed, data = cars)
##
## Coefficients:
## (Intercept)        speed
##     -17.579        3.932
```

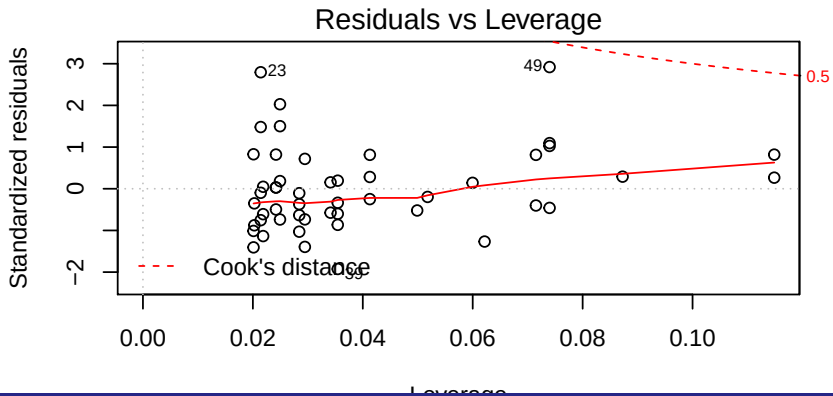
plot the fitted model

```
par(mfrow = c(2, 2))
plot(linearMod)
```







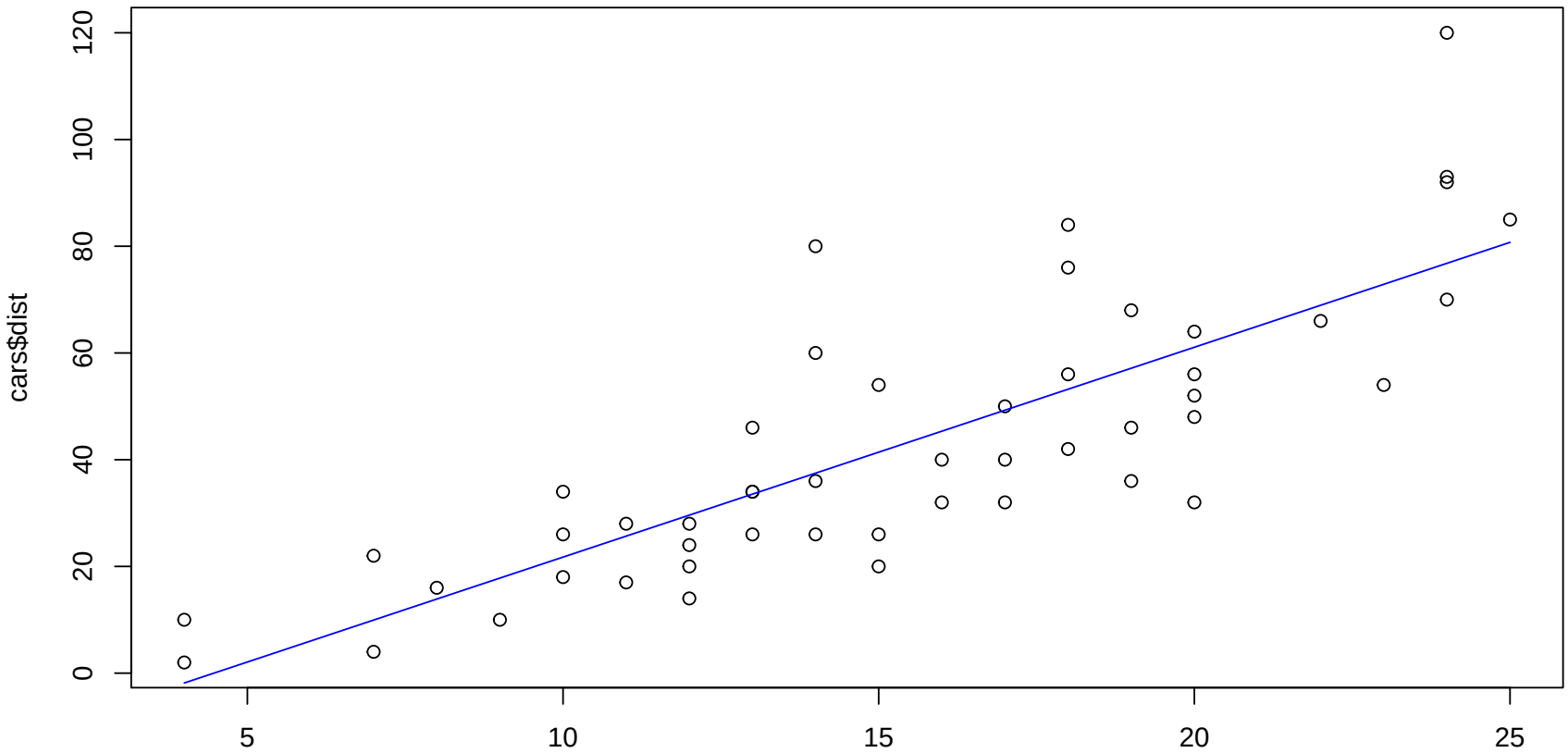


Using the fitted model to predict

```
help(predict)
help(predict.lm)
# esimation
est <- predict(linearMod)
est[1:5]
##         1         2         3         4
## -1.849460 -1.849460  9.947766  9.947766
##         5
## 13.880175
# prediction
new_data <- data.frame(speed = 200)
pre <- predict(linearMod, newdata = new_data)
pre
##         1
## 768.9027
```

plot the predicted value vs true

```
plot(x = cars$speed, y = cars$dist)
lines(x = cars$speed, linearMod$fitted.values,
    type = "l", col = "blue")
```



foreach package

An easy and starndard way of parallel comuptation

- Can run a for-loop task as a set of of parallel tasks
- Take care of the communication between the tasks (cores)

Getting start Example

Calculate the sum of the square

$$\sum_{i=1}^{10000} \sum_{j=1}^{i} j^2$$

There is a warning saying the loop ran sequentially
To run the loop parallelly, we need to register parallel backends.

```
system.time(foreach(i = 1:10000) %do% sum((1:i)^2))[3]
## elapsed
##   3.255
system.time(foreach(i = 1:10000) %dopar% sum((1:i)^2))[3]
## Warning: executing %dopar% sequentially:
## no parallel backend registered
## elapsed
##   3.064
```

Parallel backends

```
registerDoParallel()
getDoParWorkers()
## [1] 2
```

registerDoParallel() is used to register cores to parallel computation

```
system.time(foreach(i = 1:10000) %do% sum(sqrt(1:i)))[3]
## elapsed
##   3.439
system.time(foreach(i = 1:10000) %dopar% sum(sqrt(1:i)))[3]
## elapsed
##   3.138
```

Bootstraping example

```
dim(x)
## [1] 100   2
r <- foreach(i = 1:10000) %dopar% {
    ind <- sample(100, 100, replace = TRUE)
    result1 <- glm(x[ind, 2] ~ x[ind, 1], family = binomial
    coefficients(result1)
}
```

- Escaped time for using 2 cores is 16.599 seconds
- Escaped time for using single core is 26.126 seconds