Density Estimation

STAT5003

Dr. Justin Wishart

Normal distribution

In the first part of the lecture this week, we revised various probability density distributions. Let's try to simulate some data from the Normal (Gaussian) distribution. R has functions to randomly generate samples from the normal distribution (`rnorm`), and to calculate the density function (`dnorm`) of the normal distribution.

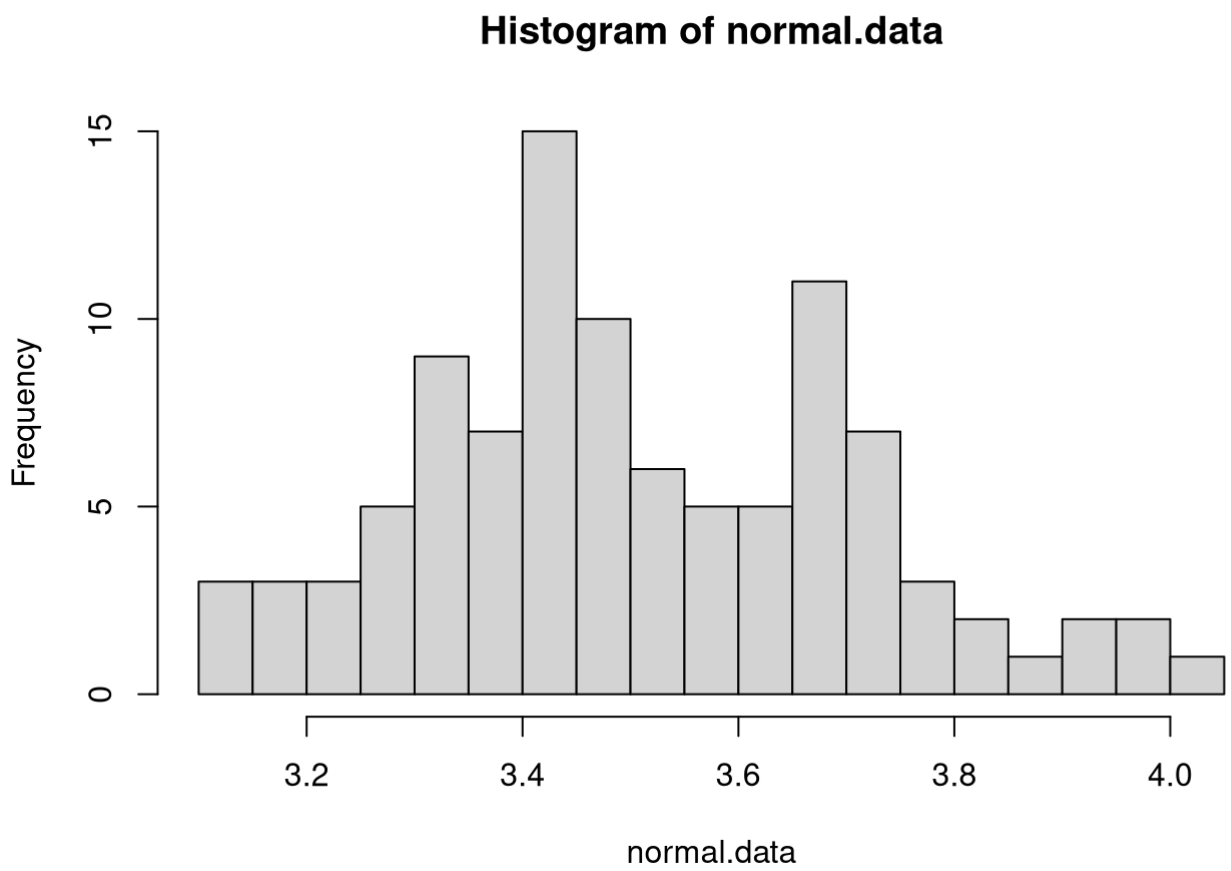
```
normal.data <- rnorm(100, mean = 3.5, sd = 0.2)
print(mean(normal.data))
```

```
## [1] 3.508041
```

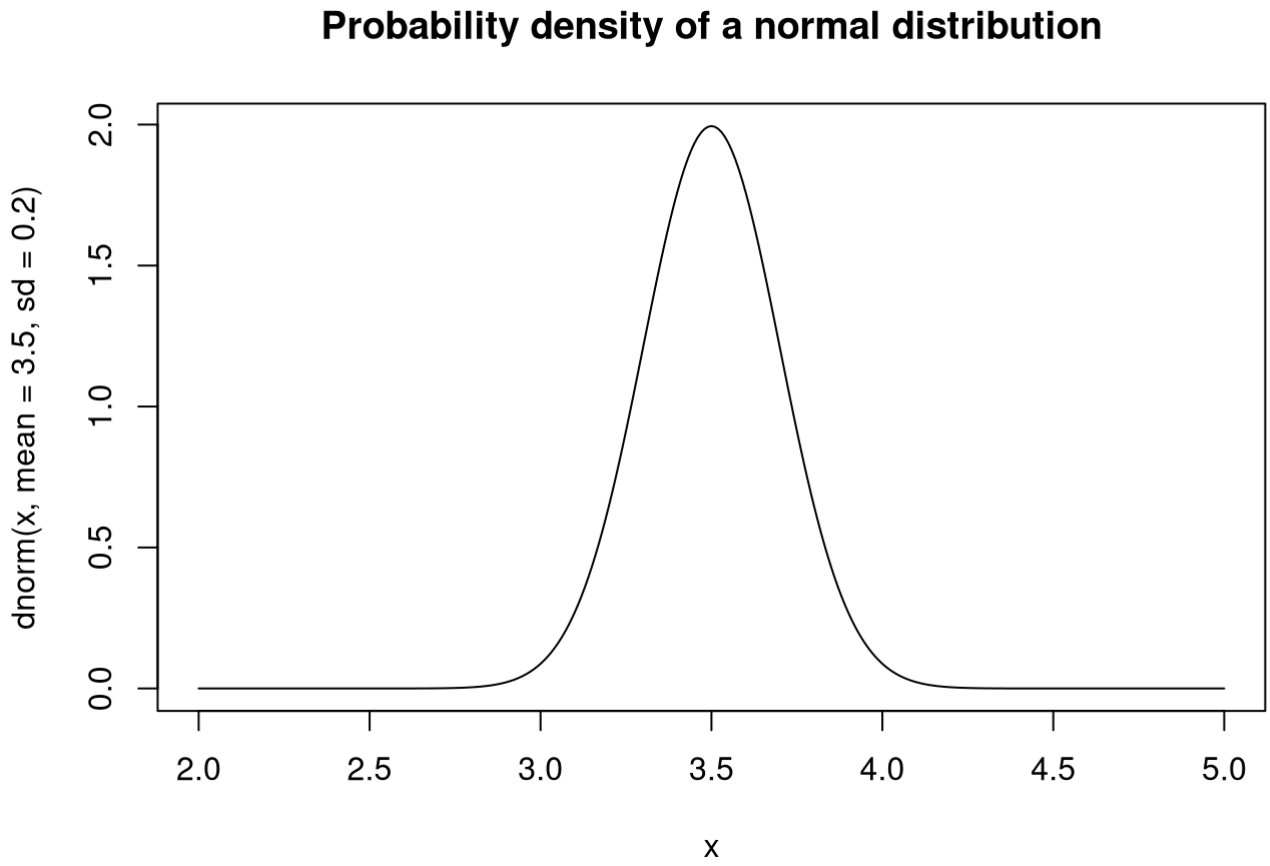
```
print(sd(normal.data))
```

```
## [1] 0.2056171
```

```
hist(normal.data, breaks = 30)
```



```
x <- seq(from = 2, to = 5, by = 0.01)
# Plotting the density function f(x) at {x=0,1,..10}
plot(x, dnorm(x, mean = 3.5, sd = 0.2), type = "l",
     main = "Probability density of a normal distribution")
```



Maximum likelihood Estimates

Here is code for the log-likelihood function of some data `normal.data` assumed to have been sampled from a Normal distribution with mean `mu` and variance `sigma`.

```
LL <- function(mu, sigma) {
    R = dnorm(normal.data, mean = mu, sd = sigma)
    return(-sum(log(R)))
}

# Let's try to calculate the likelihood for a few values of mu and sigma
LL(mu = 3, sigma = 1)
```

```
## [1] 106.8919
```

```
LL(mu = 3, sigma = 0.2)
```

```
## [1] 305.9012
```

```
LL(mu = 3.5, sigma = 0.2)
```

```
## [1] -16.64963
```

```
library(stats4)
mle.fit <- mle(LL, start = list(mu = 1, sigma = 1))
```

```
## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced

## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced

## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced

## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced

## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced

## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced

## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced

## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced

## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced

## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced

## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced

## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced

## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced

## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced

## Warning in dnorm(normal.data, mean = mu, sd = sigma): NaNs produced
```

```
mle.fit
```

```
##
## Call:
## mle(minuslogl = LL, start = list(mu = 1, sigma = 1))
##
## Coefficients:
##        mu     sigma
## 3.5080410 0.2045866
```

Kernel density estimation

To demonstrate how to perform kernel density estimation, we will be using the same dataset of Melbourne house prices as last week. Reminder that this dataset was downloaded from Kaggle and the data was released under the CC BY-NC-SA 4.0 license.

```
## melbdata <- read.csv("Melbourne_housing_FULL.csv", header = TRUE)
dim(melbdata)
```

```
## [1] 34857    21
```

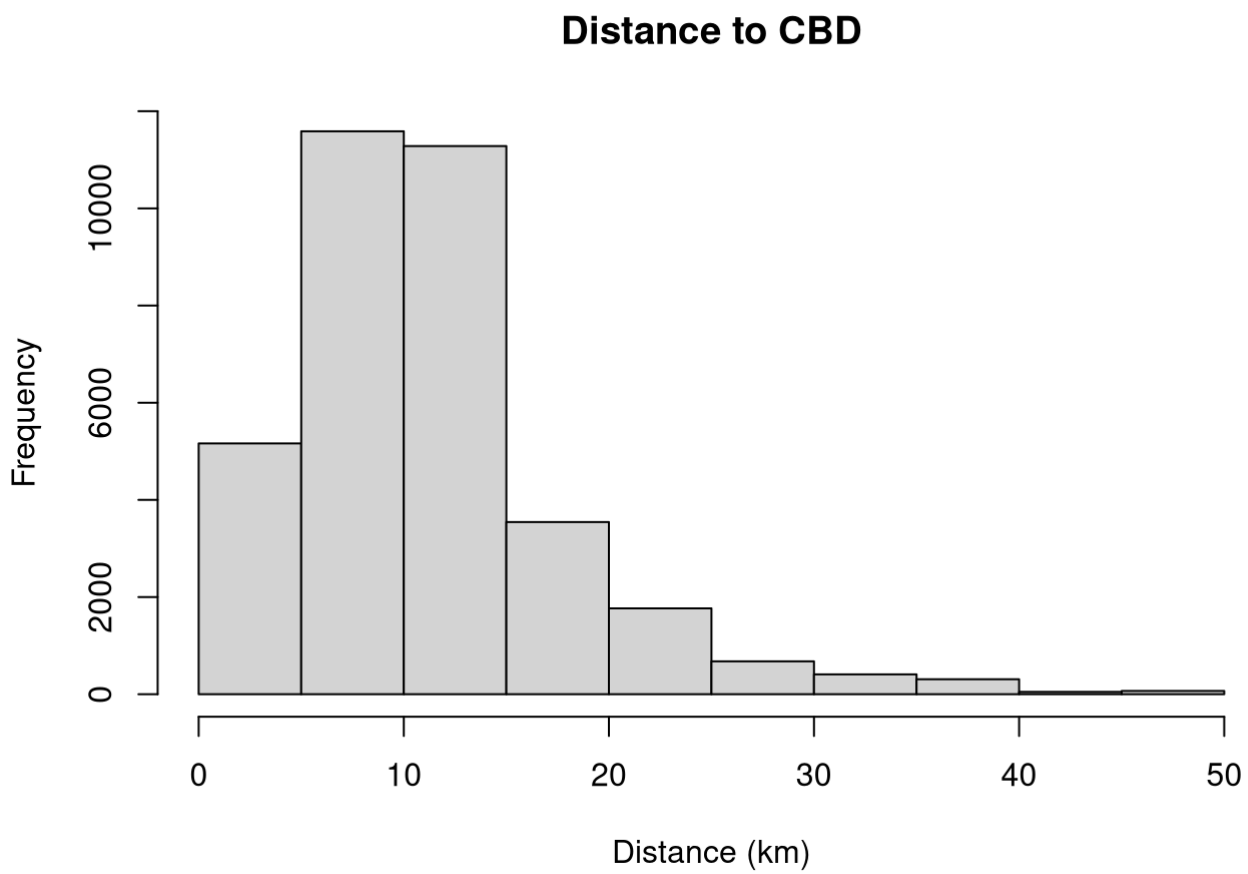
```
names(melbdata)
```

```
##  [1] "Suburb"        "Address"       "Rooms"         "Type"
##  [5] "Price"         "Method"        "SellerG"       "Date"
##  [9] "Distance"      "Postcode"      "Bedroom2"      "Bathroom"
## [13] "Car"           "Landsize"      "BuildingArea"  "YearBuilt"
## [17] "CouncilArea"   "Lattitude"     "Longtitude"    "Regionname"
## [21] "Propertycount"
```

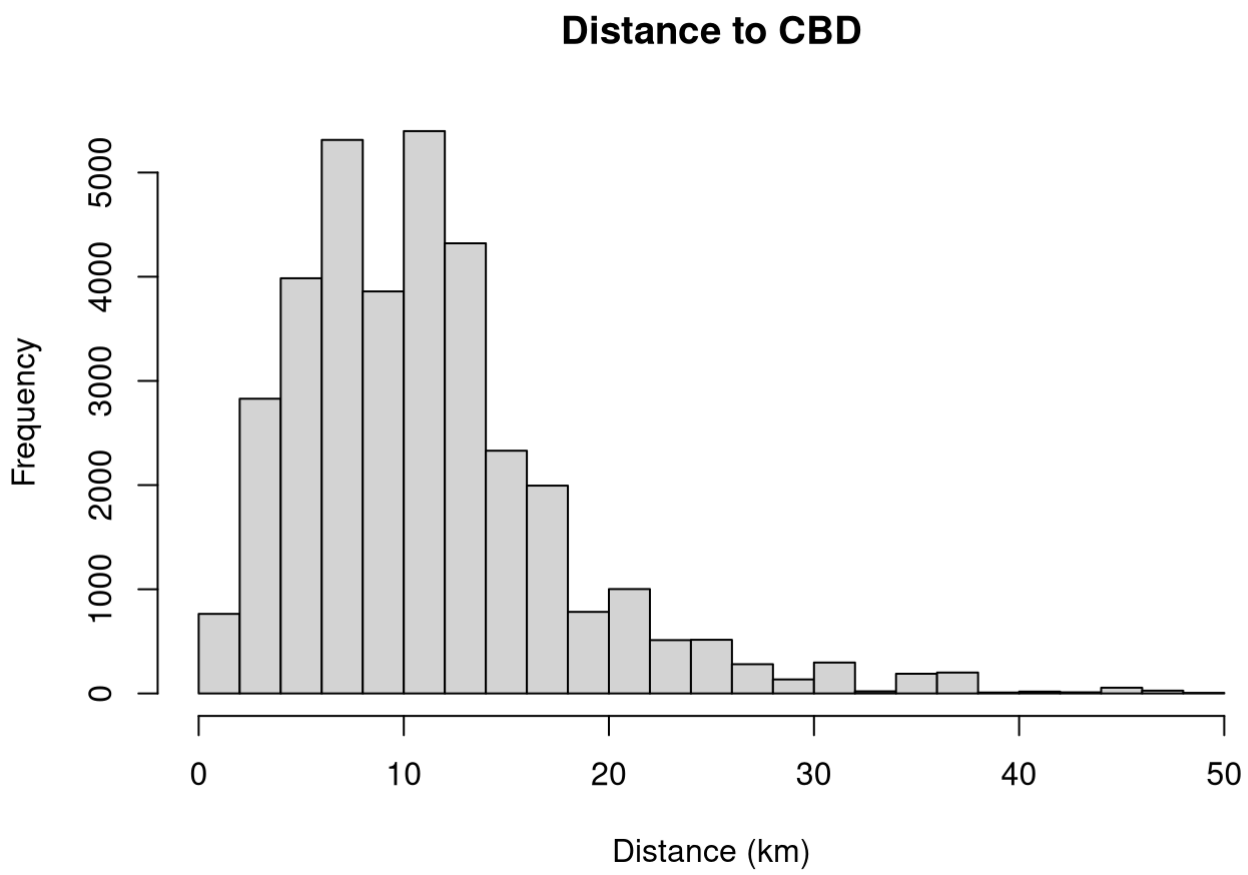
Histogram

Let's look at the histogram of the distance from the city centre variable. We can plot histogram as frequency or as density.

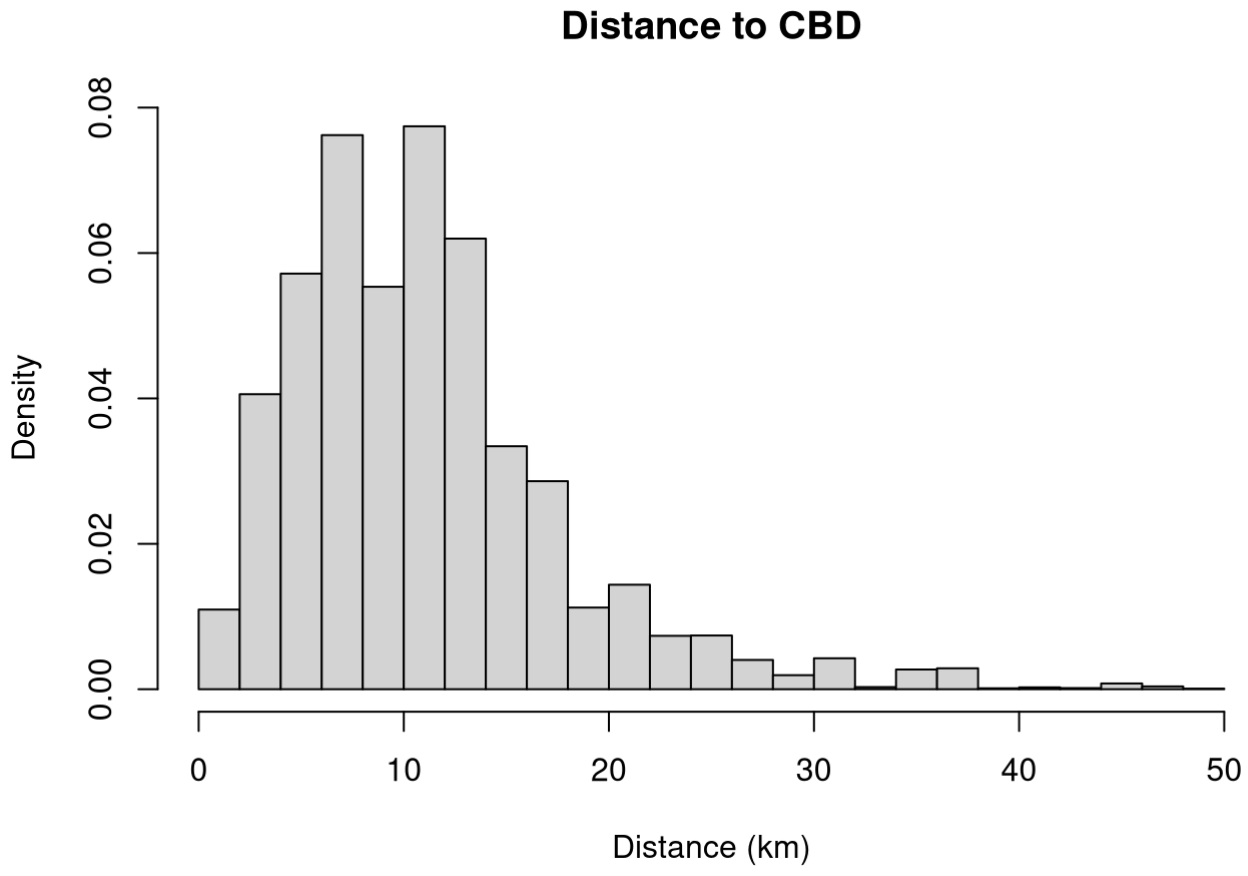
```
hist(na.omit(melbdata$Distance), main = "Distance to CBD", xlab = "Distance (km)")
```




```
hist(na.omit(melbdata$Distance), breaks = 30,
     main = "Distance to CBD", xlab = "Distance (km)" )
```

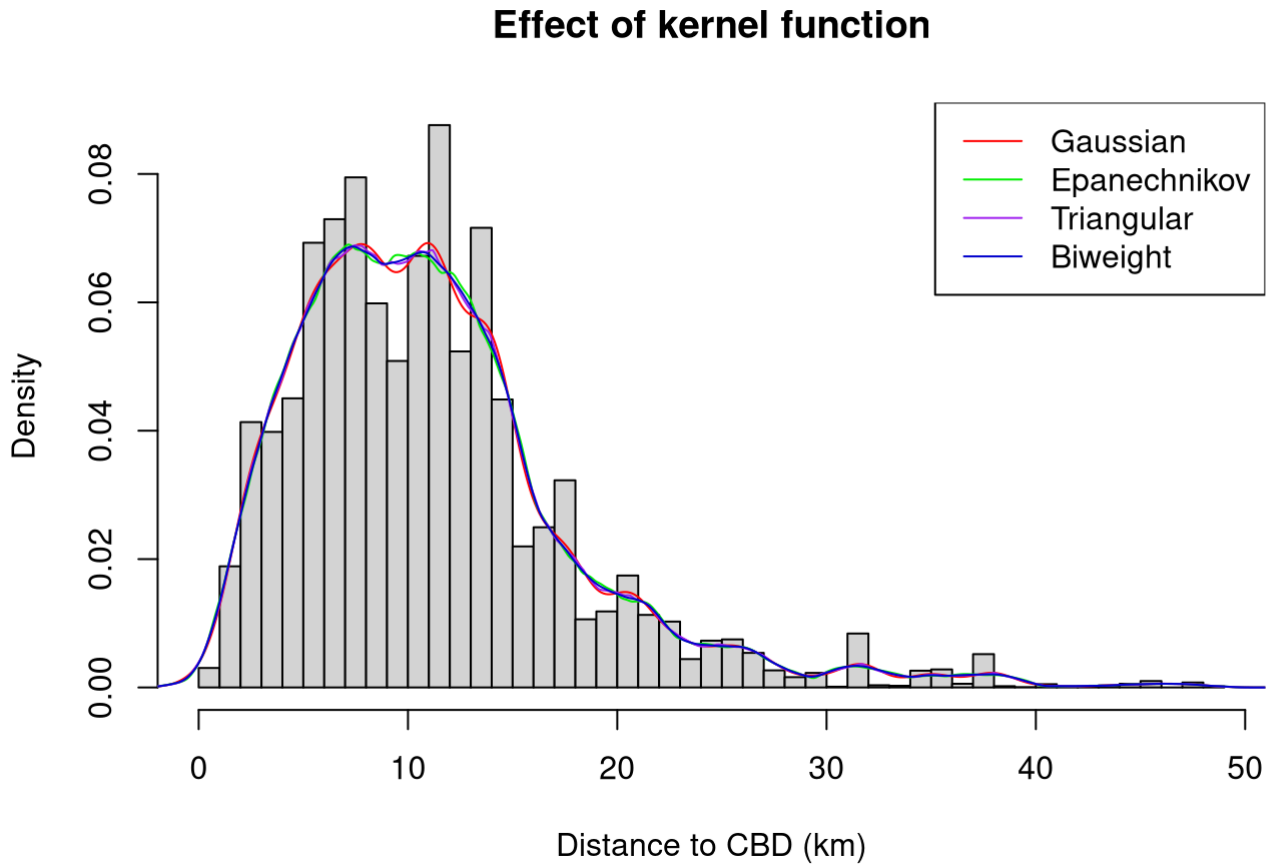



```
hist(na.omit(melbdata$Distance), breaks = 30, freq = FALSE,
     main = "Distance to CBD", xlab = "Distance (km)")
```



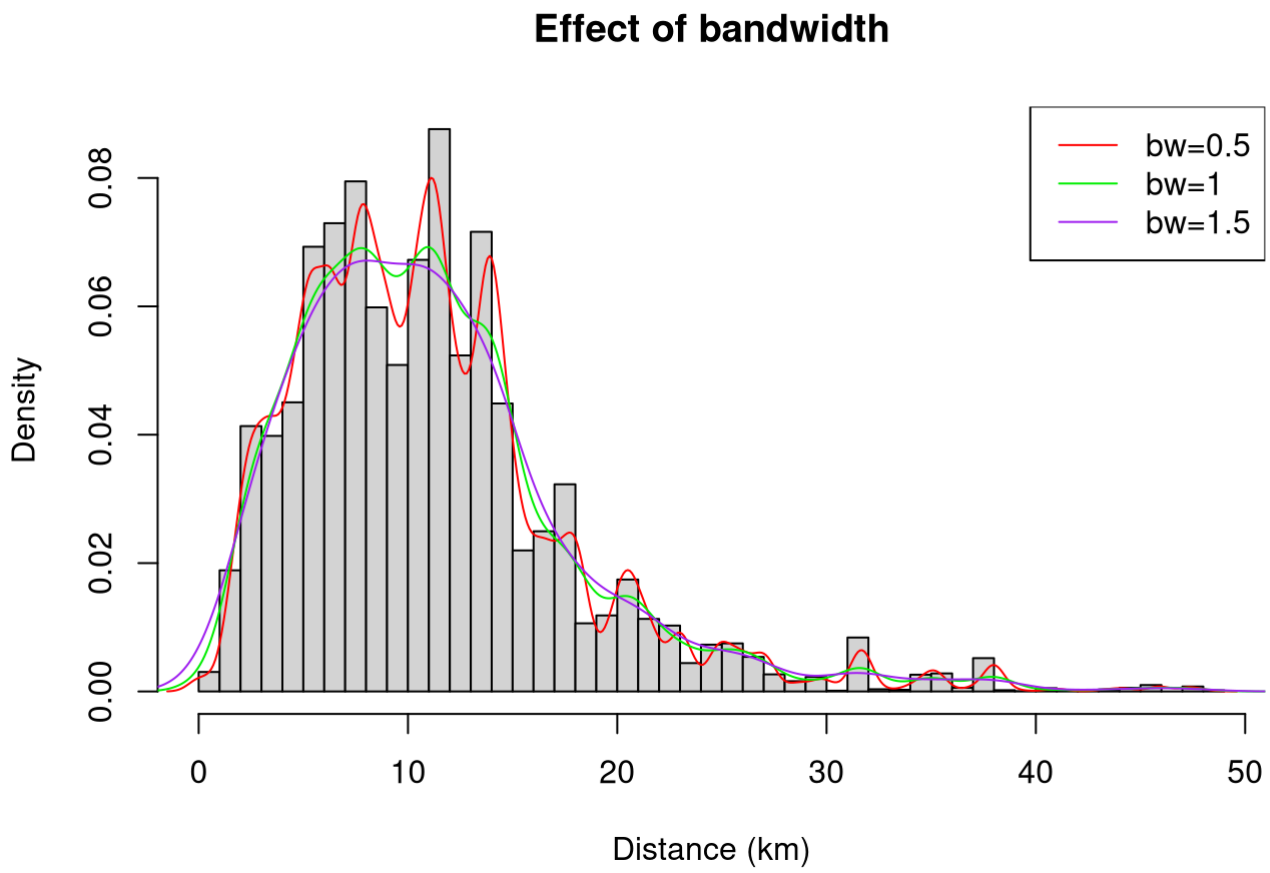
Let's now do some density estimation with different kernels.

```
d1 <- density(na.omit(melbdata$Distance), bw = 1, kernel = "gaussian")
d2 <- density(na.omit(melbdata$Distance), bw = 1, kernel = "epanechnikov")
d3 <- density(na.omit(melbdata$Distance), bw = 1, kernel = "triangular")
d4 <- density(na.omit(melbdata$Distance), bw = 1, kernel = "biweight")

hist(na.omit(melbdata$Distance), freq = FALSE, breaks = 50, main = "Effect of kernel
 function", xlab = "Distance to CBD (km)")
lines(d1$x, d1$y, col = "red")
lines(d2$x, d2$y, col = "green2")
lines(d3$x, d3$y, col = "purple")
lines(d4$x, d4$y, col = "blue3")
legend("topright",c("Gaussian","Epanechnikov", "Triangular", "Biweight"),
      lty = c(1, 1, 1, 1), col = c("red", "green2", "purple", "blue3"))
```



Now let's look at the effect of bandwidth size on smoothness and accuracy.

```
d1 <- density(na.omit(melbdata$Distance), bw = 0.5, kernel = "gaussian")
d2 <- density(na.omit(melbdata$Distance), bw = 1, kernel = "gaussian")
d3 <- density(na.omit(melbdata$Distance), bw = 1.5, kernel = "gaussian")

hist(na.omit(melbdata$Distance), freq = FALSE, breaks = 50, main = "Effect of bandwid
th", xlab = "Distance (km)")
lines(d1$x, d1$y, col = "red")
lines(d2$x, d2$y, col = "green2")
lines(d3$x, d3$y, col = "purple")
legend("topright",c("bw=0.5", "bw=1", "bw=1.5"),
      lty=c(1,1,1,1), col=c("red", "green2", "purple"))
```



Extracting information from density estimates

Let's use the density estimation to answer the question - what's the probability of finding a house for sale between 8 and 10km from the CBD?

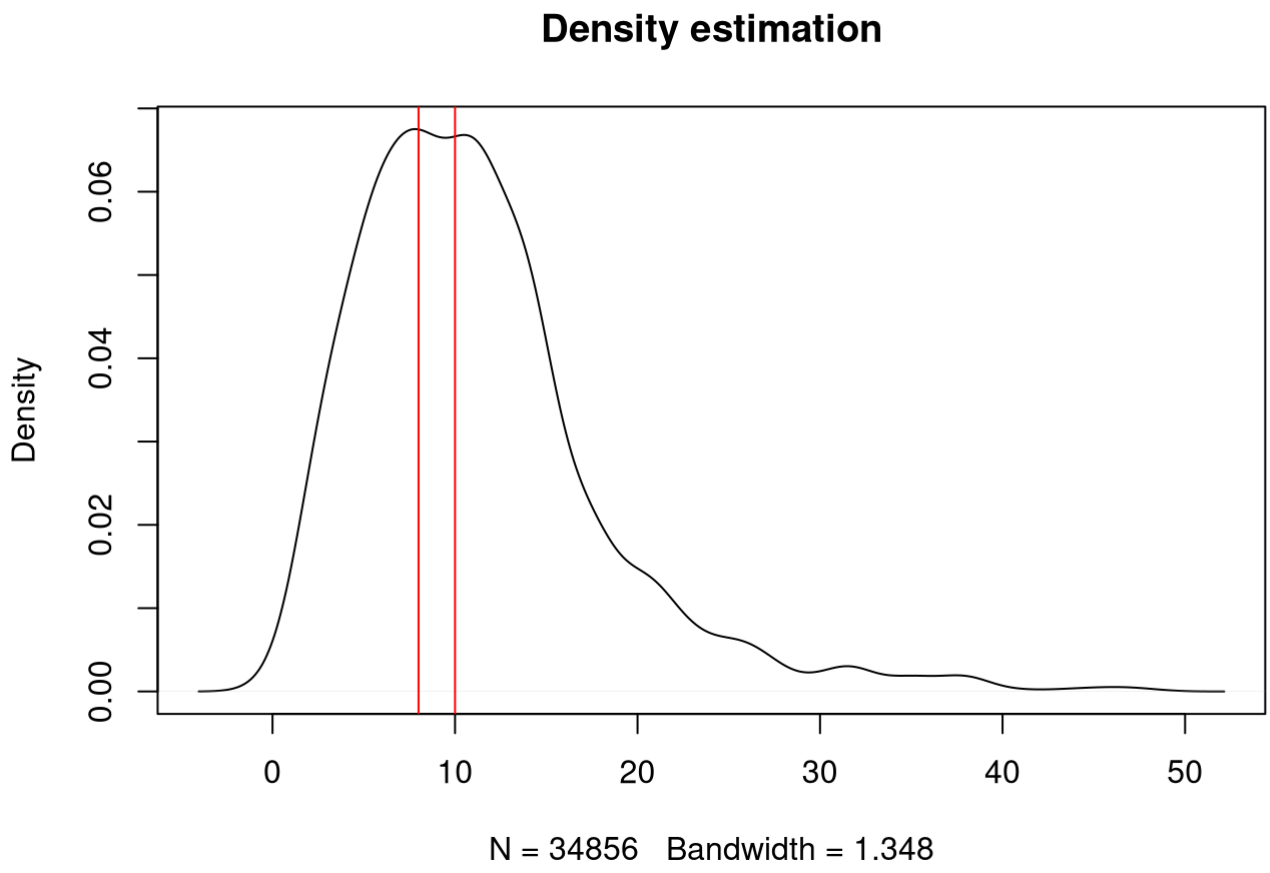
```
bw <- bw.bcv(na.omit(melbdata$Distance), lower = 0.01, upper = 2)
d2 <- density(na.omit(melbdata$Distance), bw = bw, kernel = "gaussian")
f <- approxfun(d2)
class(f)
```

```
## [1] "function"
```

```
p <- integrate(f, 8, 10)
p
```

```
## 0.1336237 with absolute error < 5.8e-06
```

```
plot(d2, main = "Density estimation")
abline(v=8, col = "red")
abline(v=10, col = "red")
```


Density estimation
Density
0.00
0.02
0.04
0.06
0
10
20
30
40
50
N = 34856 Bandwidth = 1.348