SLLD - Module 1

Smoothing

L.Emer, F.Chiaromonte

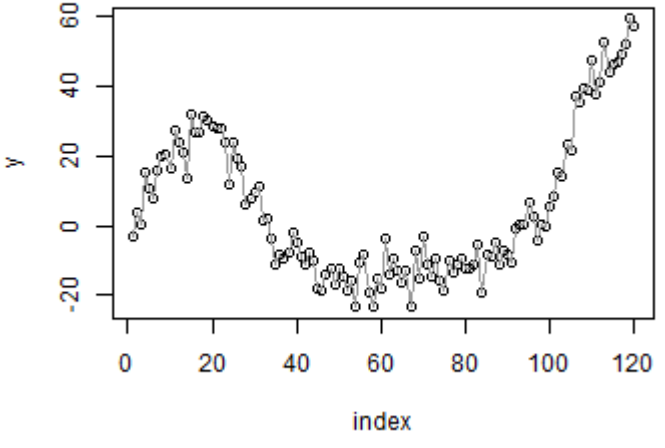
Sant'Anna School of Advanced Study - Pisa

13/2/2026

```
library(tidyverse) # for data manipulation and visualization
library(ggplot2) # for plots
```

Data

```
set.seed(1)
n <- 120
index <- 1:n
eps <- rnorm(n, 0, 5)
y <- as.vector(arima.sim(n=n-1,list(order=c(2,1,0),
                                    ar=c(0.5, 0.4))))+eps
plot(index, y)
lines(index, y, col='grey60')
```



LOWESS

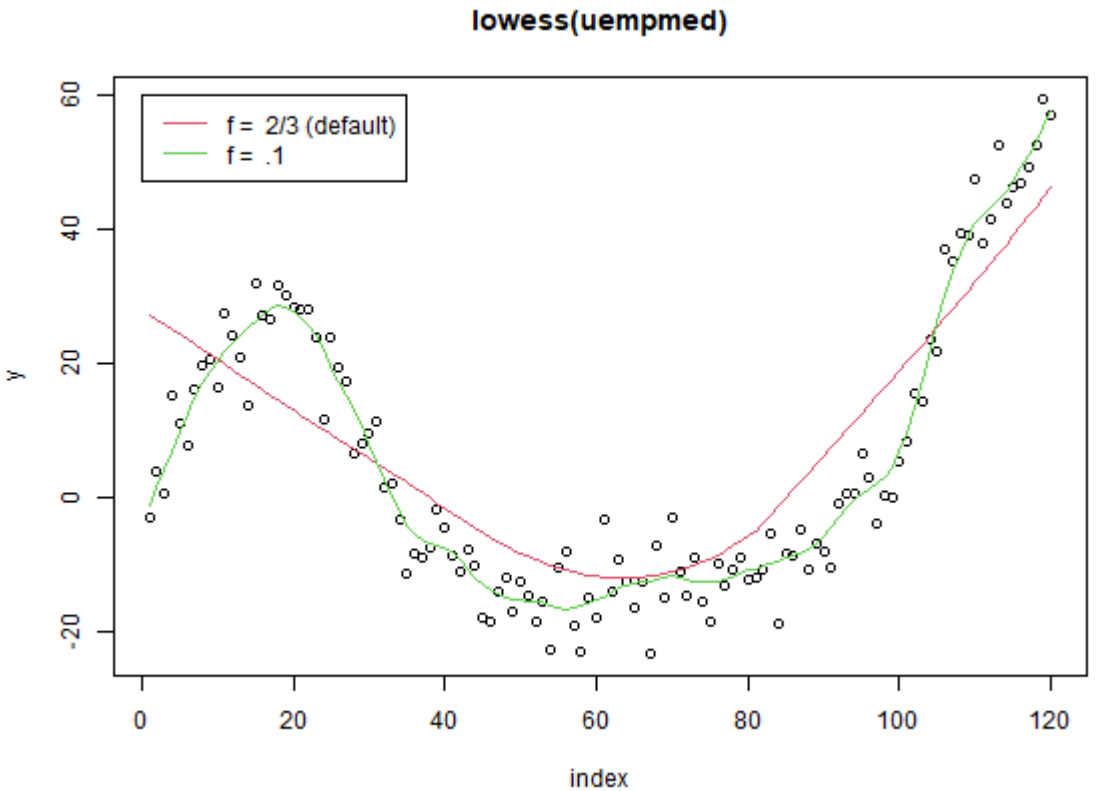
Perform LOWESS using the **lowess** function within the stats package. It takes as inputs:

- **x, y:** vectors giving the coordinates of the points in the scatter plot.
- **f:** smoother span. This gives the proportion of points in the plot which influence the smooth at each value. Larger values give more smoothness.

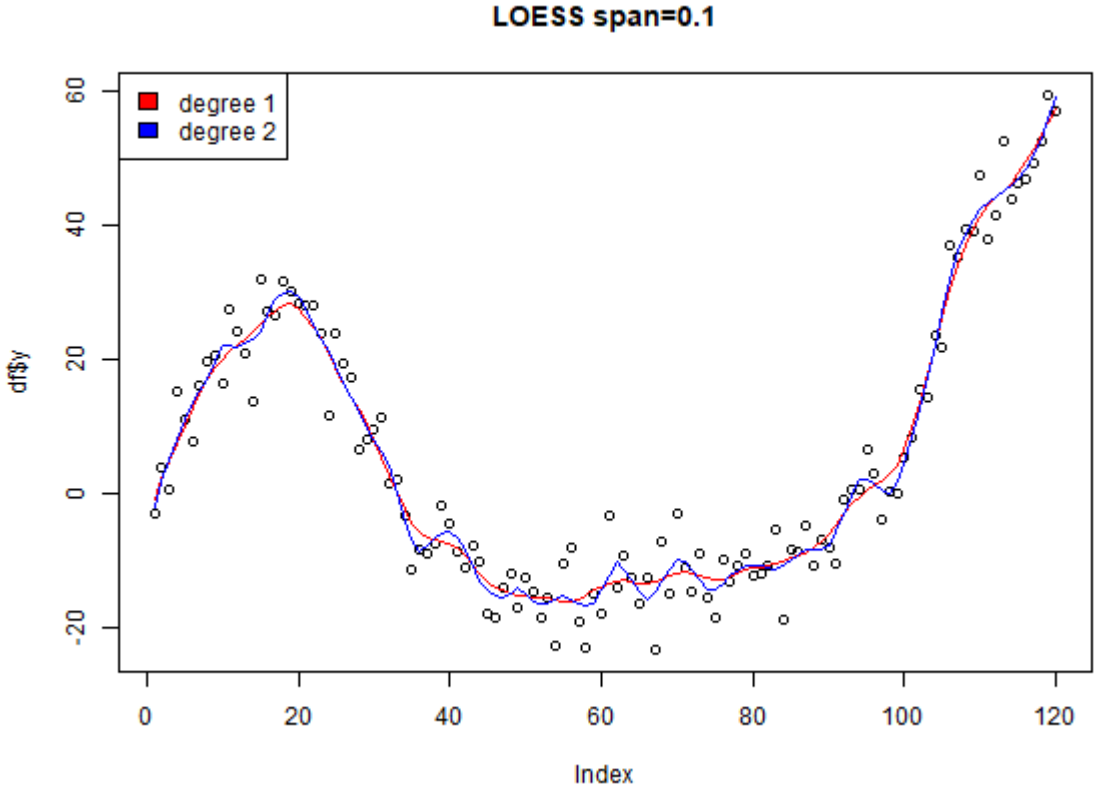
You can also perform LOWESS through the **loess** command in the stats package. Let us focus on the following arguments:

- **formula:** a formula specifying the numeric response and one to four numeric predictors
- **data:** the dataframe
- **span:** the parameter which controls the degree of smoothing
- **degree:** the degree of the polynomials to be used, normally 1 or 2

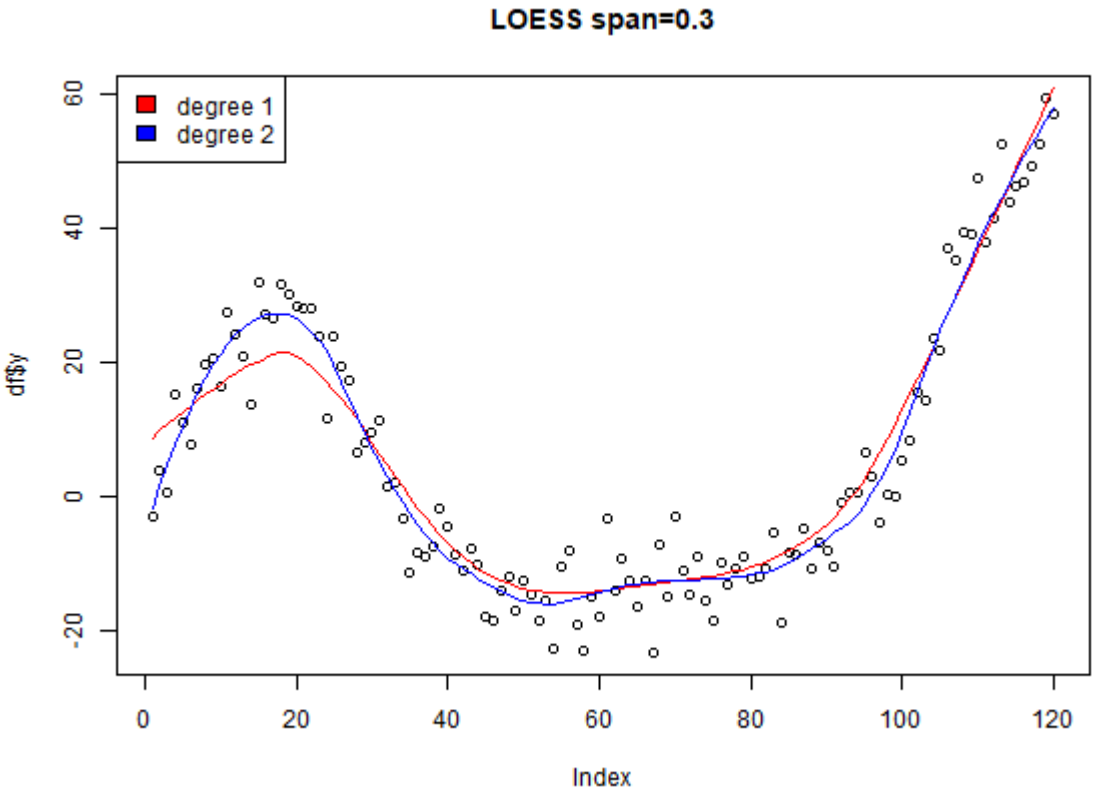
```
plot(y ~ index, main = "lowess(uempmed)")
lines(lowess(y), col = 2)
lines(lowess(y, f = .1), col = 3)
legend(0, 60, c(paste("f = ", c("2/3 (default)", ".1"))),
       lty = 1, col = 2:3)
```



```
df = data.frame(cbind(y, index))
loess1_10 <- loess(y ~ index, data = df, span = 0.1, degree=1)
loess2_10 <- loess(y ~ index, data = df, span = 0.1, degree=2)
plot(df$y, main="LOESS span=0.1")
lines(predict(loess1_10), col='red')
lines(predict(loess2_10), col='blue')
legend("topleft", fill = c("red","blue"),
legend = c("degree 1", "degree 2"))
```



```
loess1_3 <- loess(y ~ index, data = df, span = 0.3, degree=1)
loess2_3 <- loess(y ~ index, data = df, span = 0.3, degree=2)
plot(df$y, main="LOESS span=0.3")
lines(predict(loess1_3), col='red')
lines(predict(loess2_3), col='blue')
legend("topleft", fill = c("red","blue"),
legend = c("degree 1", "degree 2"))
```



Kernel Smoothing

The command that we are going to use is **ksmooth**.

If the mean is computed giving equal weights to the points belonging to each window, we talk about “box” kernel. The result is a list with the original x and the new smoothed values y.

```
window <- 5
box_smooth <- ksmooth(df$index, df$y, kernel='box',
                      bandwidth = window)
```

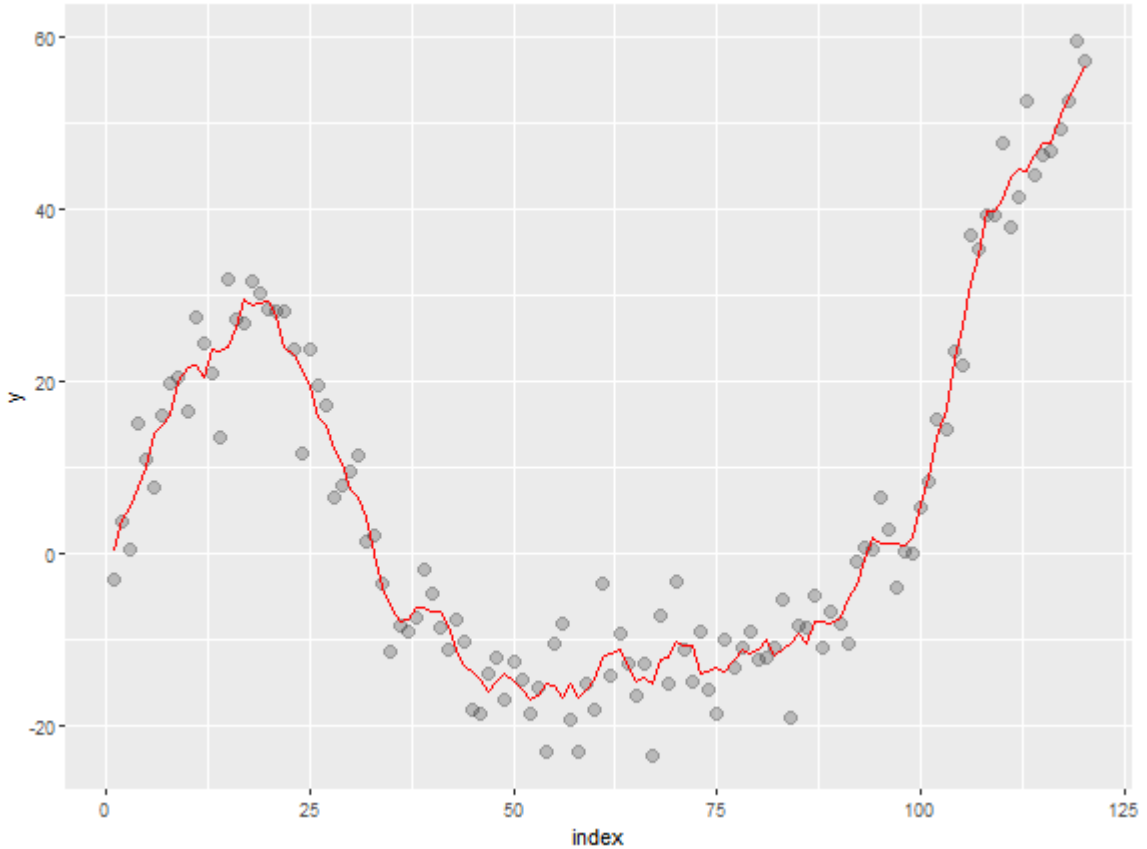
We can change units’ weights by giving larger weights to the ones in the “central” portion of the window, thus the points at the edges will receive very little weights.

Here we will use a weighted average, where weights are provided by a normal density.

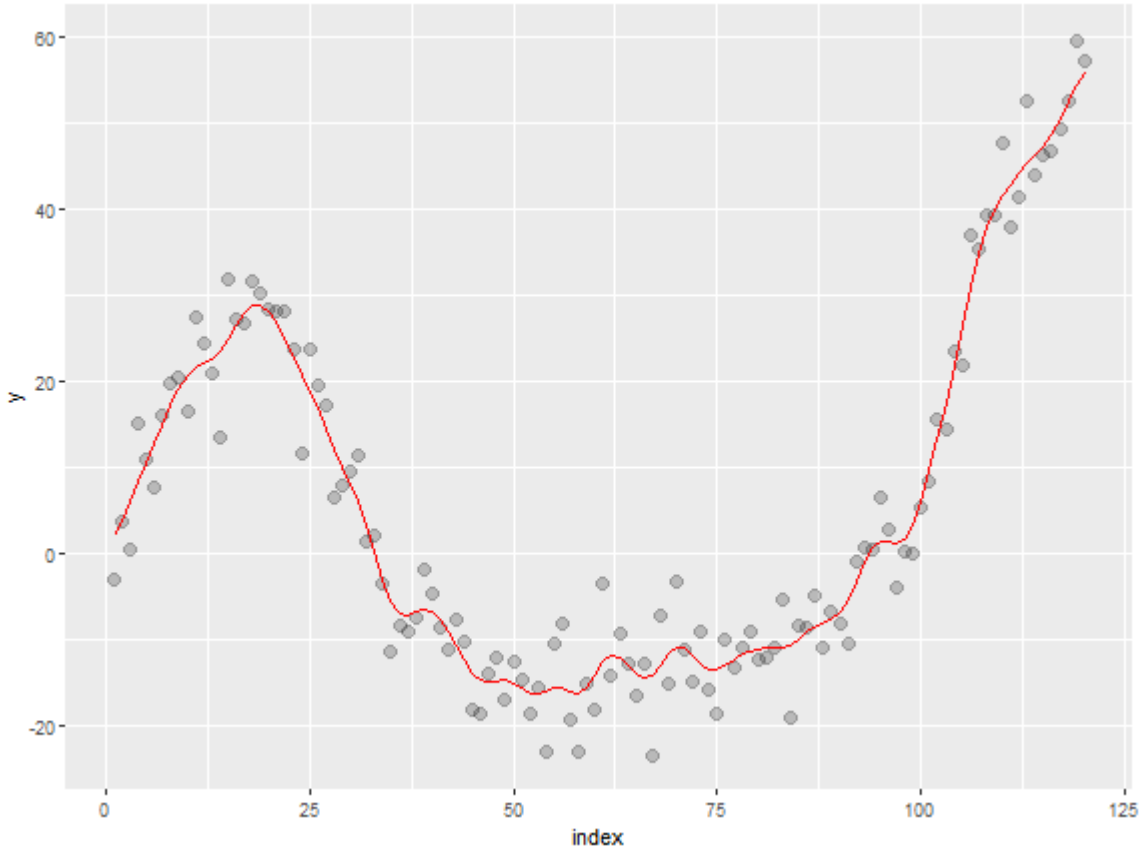
```
window <- 5
norm_smooth <- ksmooth(df$index, df$y, kernel='normal',
                      bandwidth = window)
```

Let’s plot our result using ggplot (unlike base R plots).

```
df %>% mutate(smooth = box_smooth$y) %>% ggplot(aes(index, y)) +
geom_point(size=3, alpha=0.2, color='black') +
  geom_line(aes(index, smooth), color='red')
```



```
df %>% mutate(smooth = norm_smooth$y) %>% ggplot(aes(index, y)) +
geom_point(size=3, alpha=0.2, color='black') +
  geom_line(aes(index, smooth), color='red')
```



Kernel Density Estimator

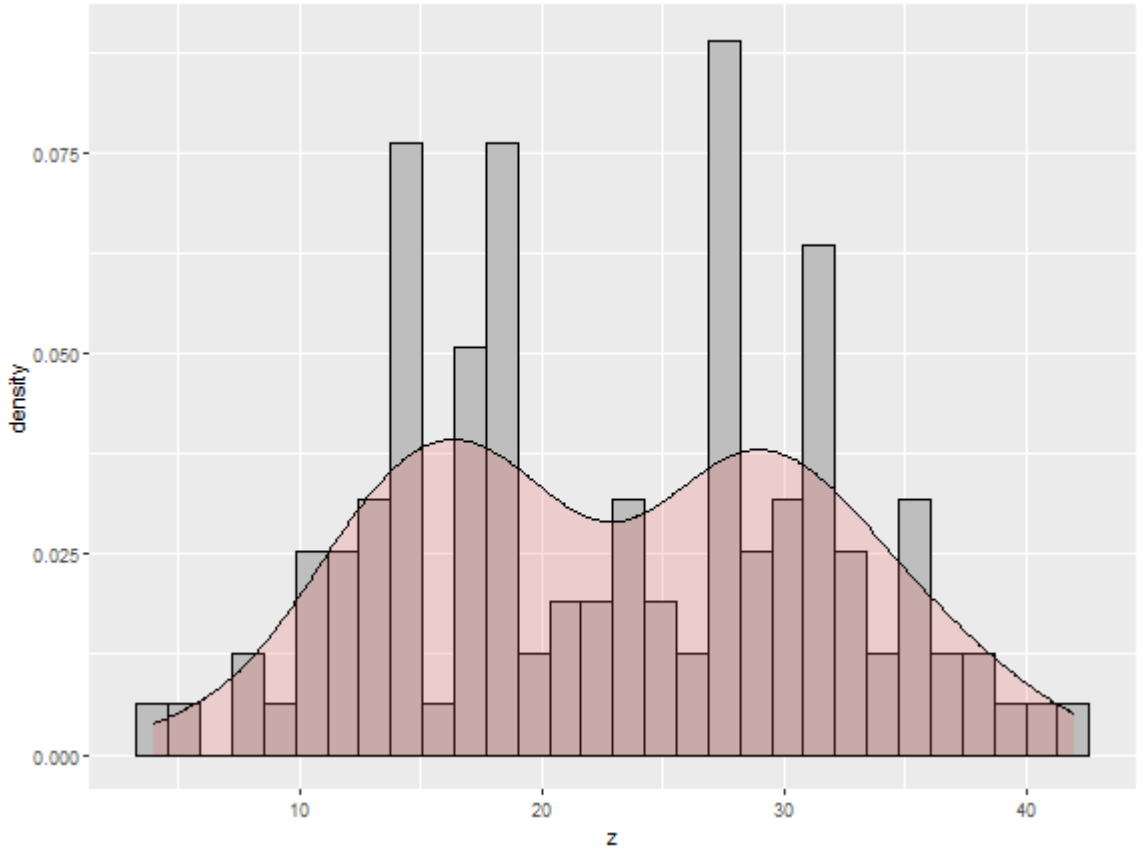
Let's simulate a new variable and produce a histogram and its density through ggplot.

```
set.seed(1)
df2 <- data.frame(z = round(c(rnorm(60, mean=15, sd=5),
rnorm(60, mean=30, sd=5))), index)
```

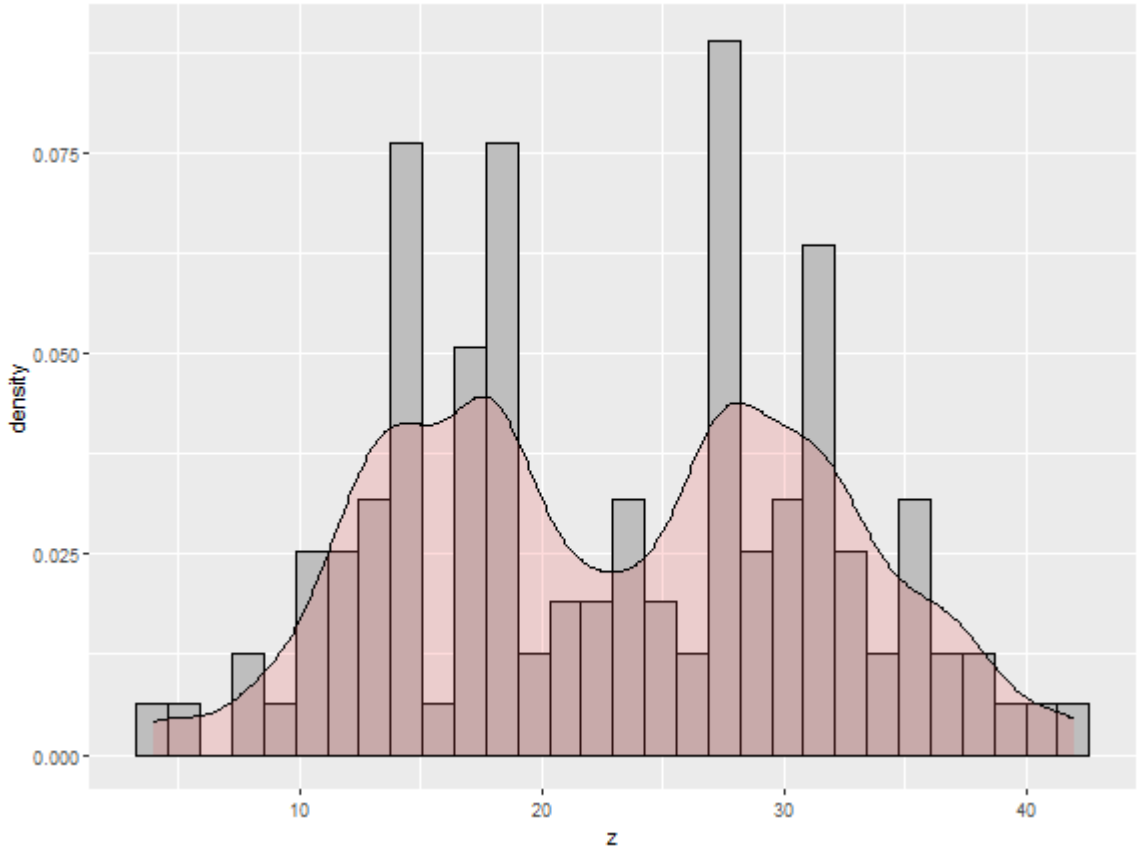
The functon **geom_density** computes and draws kernel density estimate, which is a smoothed version of the histogram. This is a useful alternative to the histogram for continuous data that comes from an underlying smooth distribution.

We can adjust the default density through the **adjust** argument (default is 1). It indicates a multiplicate bandwidth adjustment. For example, adjust = 1/2 means use half of the default bandwidth

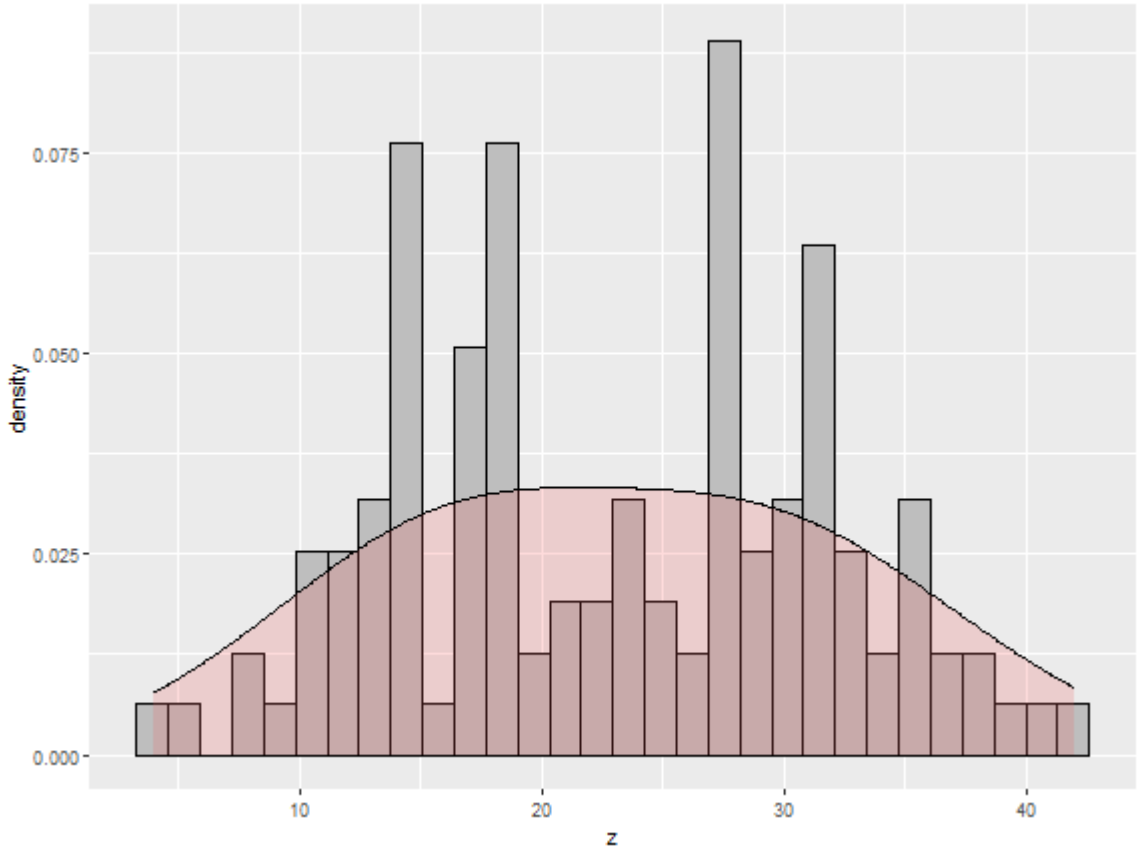
```
ggplot(df2, aes(x=z)) + geom_histogram(aes(y=..density..),
    colour="black", fill="grey")+
  geom_density(alpha=.2, fill="#FF6666")
```



```
ggplot(df2, aes(x=z)) +
  geom_histogram(aes(y=..density..), colour="black", fill="grey")+
  geom_density(alpha=.2, fill="#FF6666", adjust=1/2)
```



```
ggplot(df2, aes(x=z)) +
  geom_histogram(aes(y=..density..), colour="black", fill="grey")+
  geom_density(alpha=.2, fill="#FF6666", adjust=2)
```



Now it's your turn!!!