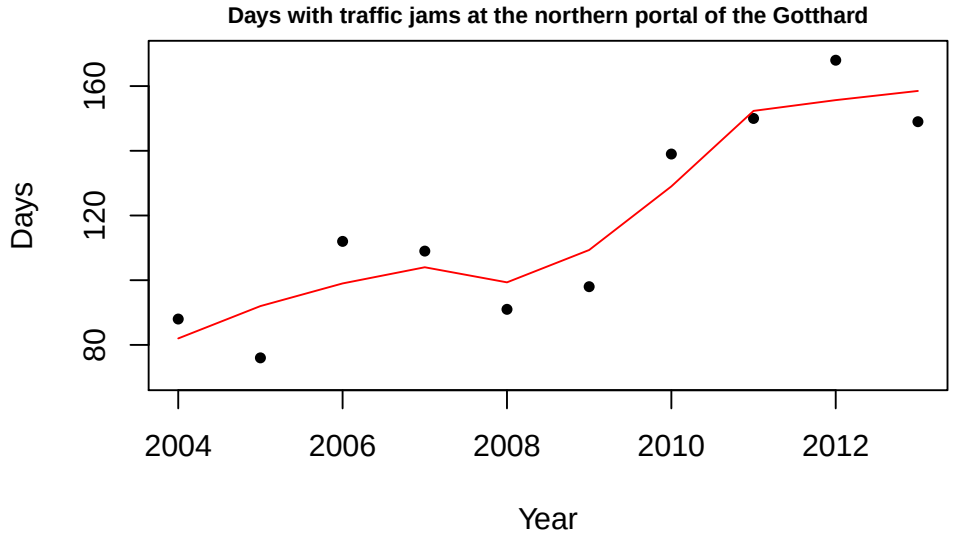
Solution to Series 2

1. a)

```
> ## Running Mean with window width of 3 years
> rng.mean <- c(mean(traffic_jam[1:2]), mean(traffic_jam[1:3]), mean(traffic_jam[2:4]),
                mean(traffic_jam[3:5]), mean(traffic_jam[4:6]), mean(traffic_jam[5:7]),
                mean(traffic_jam[6:8]), mean(traffic_jam[7:9]), mean(traffic_jam[8:10]),
                mean(traffic_jam[9:10]))
> ## Add to plot
> #par(pin=c(4,2))
> plot(year, traffic_jam, xlab="Year", ylab="Days", pch=20, ylim=c(70,170))
> title("Days with traffic jams at the northern portal of the Gotthard", cex.main = 0.75)
> lines(year, rng.mean, col="red")
```

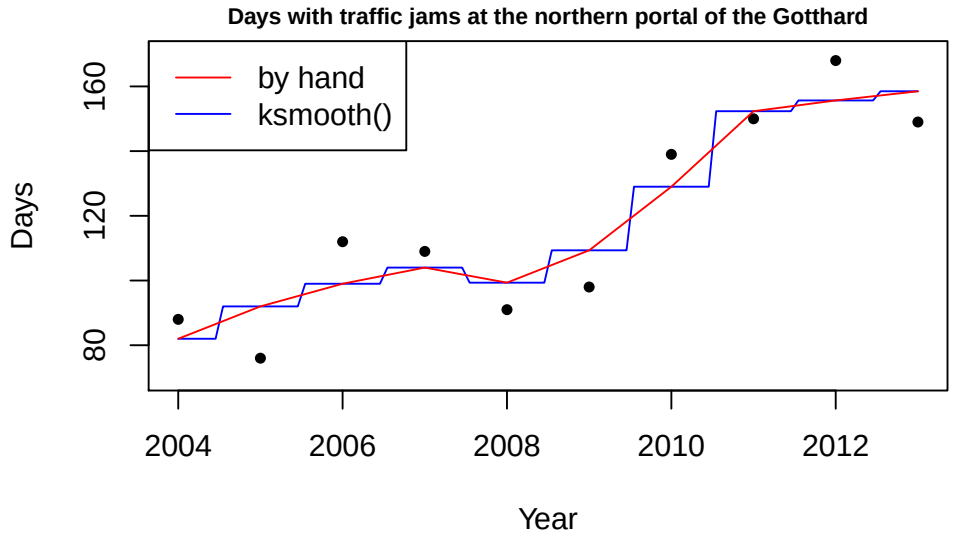


Alternatively, you can use the function `ksmooth()` with the argument `x.points`. This evaluates the running mean smoother at the same points as in the manual calculation.

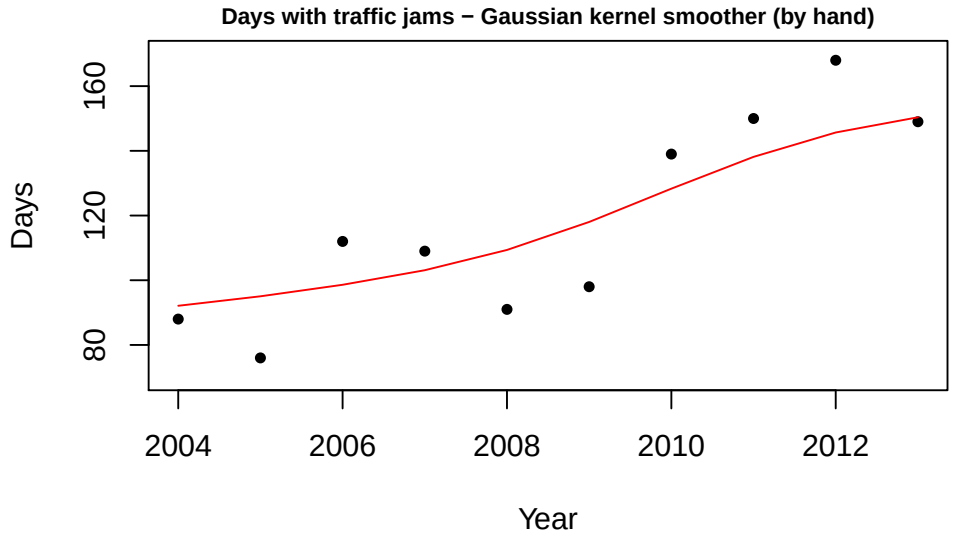
```
> fit <- ksmooth(year, traffic_jam, bandwidth=3, x.points=2004:2013)
```

b) If one uses the function `ksmooth()` with the default values, it is evaluated at more than the ten turns of the year. This yields a step function which is not nicely interpretable.

```
> plot(year, traffic_jam, xlab="Year", ylab="Days", pch=20, ylim=c(70,170))
> title("Days with traffic jams at the northern portal of the Gotthard", cex.main = 0.75)
> ## Smoothing with ksmooth()
> fit <- ksmooth(year, traffic_jam, bandwidth=3)
> lines(fit$x, fit$y, col="blue")
> ## Reproduce manual solution:
> fit_manual <- ksmooth(year, traffic_jam, bandwidth=3, x.points=2004:2013)
> lines(fit_manual, col="red")
> ## Add legend
> legend("topleft", lty=1, col=c("red", "blue"),
        legend=c("by hand", "ksmooth()"))
```



c)
```
> plot(year, traffic_jam, xlab="Year", ylab="Days", pch=20, ylim=c(70,170))
> title("Days with traffic jams - Gaussian kernel smoother (by hand)", cex.main = 0.75)
> gew  <- dnorm(seq(-10,10,by=1),0,2)
> smoo <- c(gew[11:20]%*%traffic_jam/sum(gew[11:20]),
            gew[10:19]%*%traffic_jam/sum(gew[10:19]),
            gew[ 9:18]%*%traffic_jam/sum(gew[ 9:18]),
            gew[ 8:17]%*%traffic_jam/sum(gew[ 8:17]),
            gew[ 7:16]%*%traffic_jam/sum(gew[ 7:16]),
            gew[ 6:15]%*%traffic_jam/sum(gew[ 6:15]),
            gew[ 5:14]%*%traffic_jam/sum(gew[ 5:14]),
            gew[ 4:13]%*%traffic_jam/sum(gew[ 4:13]),
            gew[ 3:12]%*%traffic_jam/sum(gew[ 3:12]),
            gew[ 2:11]%*%traffic_jam/sum(gew[ 2:11]))
> lines(year, smoo, col="red")
```



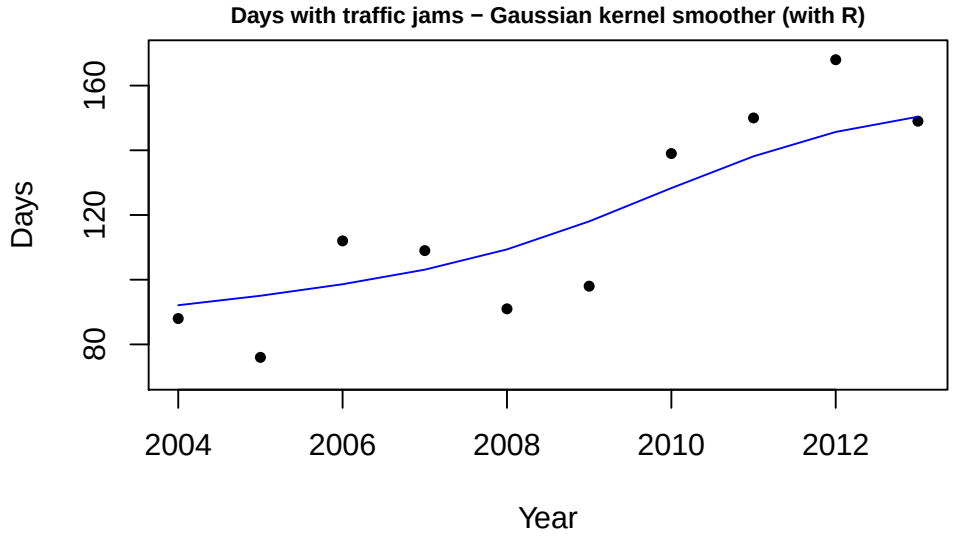
The weights are determined with the function `dnorm()` in R. Then, we compute the dot product between the weight vector and the y-values and we normalize with the sum of the weights.

d) If we want to reproduce the solution computed by hand with R, we need to ensure that the standard deviation used in the R-function is also two. The argument `bandwidth` in R is such that the absolute value of the 25% quantile (or the 75% quantile) of the distribution corresponds to 0.25*`bandwidth`. The 25% quantile is

```
> qnorm(0.25, 0, 2)
[1] -1.34898
```

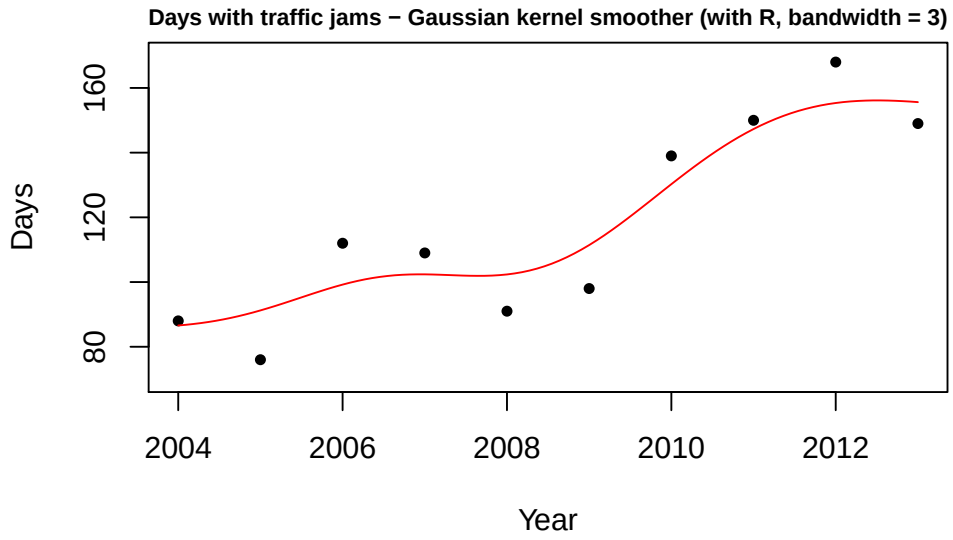
The bandwidth we need to choose is then exactly four times the absolute value of `qnorm(0.25, 0, 2)`. As you can see in the R output, the solutions are identical.

```
> ## Reproduce the same solution with R
> qnorm(0.25, 0, 2)
[1] -1.34898
> bw <- 4*abs(qnorm(0.25,0,2))
> fit <- ksmooth(year, traffic_jam, kernel="normal", bandwidth=bw, x.points=2004:2013)
> plot(year, traffic_jam, xlab="Year", ylab="Days", pch=20, ylim=c(70,170))
> title("Days with traffic jams - Gaussian kernel smoother (with R)", cex.main = 0.75)
> lines(fit$x, fit$y, col="blue")
```



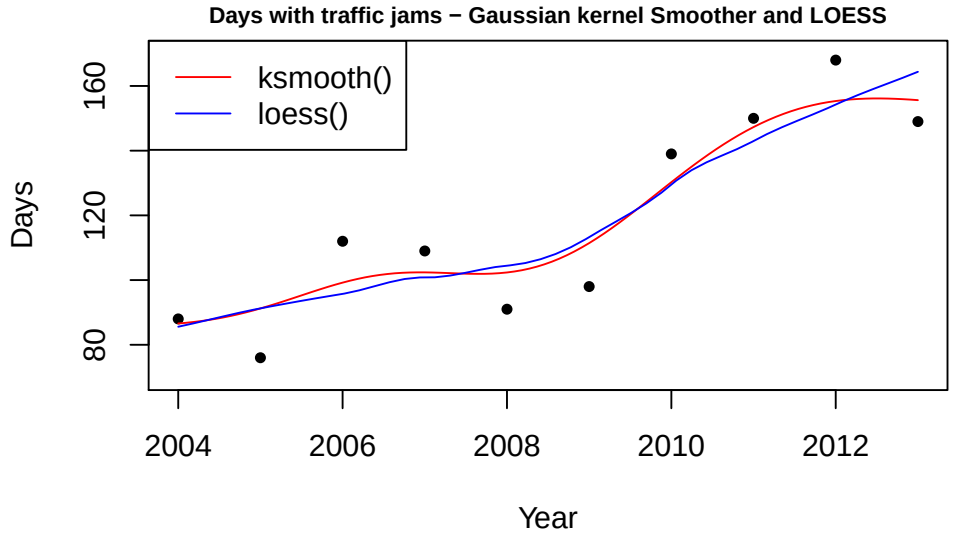
The smoothing parameter should be chosen such that the systematic part of the relation is represented while the random fluctuations should not be visible. How to achieve this separation is somewhat subjective. Here, we obtain a good result with `bandwidth=3`. Please note, however, that there is no "correct" solution in this case – there are only more suitable resp. less suitable values.

```
> ## Experiment with other bandwidths
> plot(year, traffic_jam, xlab="Year", ylab="Days", pch=20, ylim=c(70,170))
> fit <- ksmooth(year, traffic_jam, kernel="normal", bandwidth=3)
> title("Days with traffic jams - Gaussian kernel smoother (with R, bandwidth = 3)", cex.main
> lines(fit$x, fit$y, col="red")
```



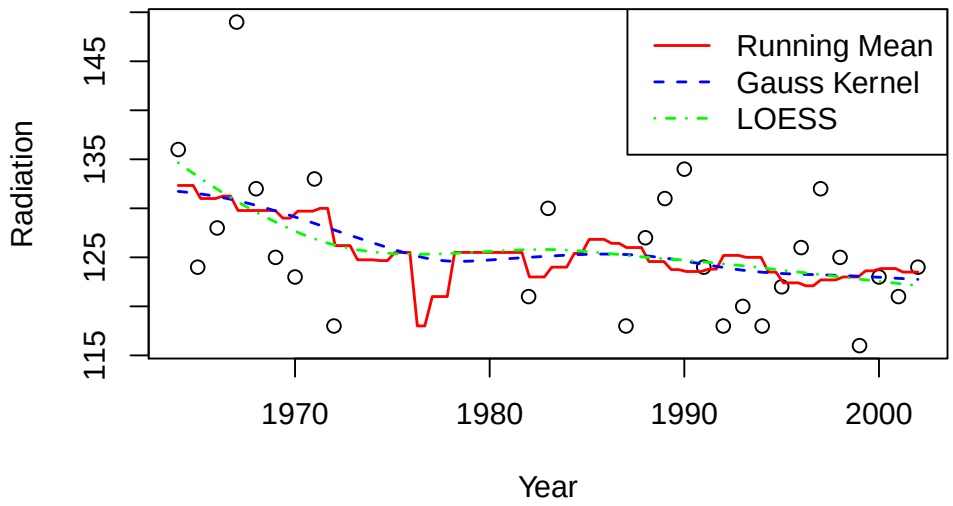
e) When we set `degree=2`, the local parameters are adjusted. As you can see in the plot, this can yield a visible effect at the corners: the blue smoother shows a steep increase in the year 2009/2010 – whether this increase is a random effect or a systematic change is difficult to tell from the data. "More conservative" in this respect is a LOESS smoother that only uses linear functions locally, i.e. `degree=1`.

```
> plot(year, traffic_jam, xlab="Year", ylab="Days", pch=20, ylim=c(70,170))
> # Gaussian smoother
> lines(fit$x, fit$y, col="red")
> ## Loess smoother
> lines(loess.smooth(year, traffic_jam, degree=1, span=0.75), col="blue")
> ## Add legend
> legend("topleft", lty=1, col=c("red", "blue"), legend=c("ksmooth()", "loess()"))
> ## Add title
> title("Days with traffic jams - Gaussian kernel Smoother and LOESS", cex.main = 0.75)
```



2. **a)** First of all, we note that the data set contains the value 99999. This is not an observation but 99999 encodes the fact that the corresponding observation is missing. Therefore, we replace it with `NA` (not available). Subsequently, we fit different smoothers. As noted above, the running mean smoother does not yield a smooth function since the points fall out of the respective window quite abruptly. The Gaussian kernel smoother and the LOESS smoother do not show the same issue and yield very similar results.

```
> # Load data
> load("solar.radiation.rda")
> # Ignore corrupted data points
> sol.rad[sol.rad==99999] <- NA
> sol.rad <- na.omit(sol.rad)
> # Scatter plot
> plot(sol.rad,xlab="Year", ylab="Radiation")
> # Running Mean
> lines(ksmooth(sol.rad$jahr, sol.rad$rad, kernel="box", bandwidth=10), lwd=1.5,
        col="red")
> # Gaussian Kernel Smoother
> lines(ksmooth(sol.rad$jahr, sol.rad$rad, kernel="normal", bandwidth=10), lty=2,
        lwd=1.5, col="blue")
> # LOESS
> fit <- loess(rad~jahr, sol.rad)
> x <- seq(1964, 2002, length.out=100)
> y <- predict(fit, newdata=data.frame(jahr=x))
> lines(x, y, lty=4, lwd=1.5, col="green")
> # Add legend
> legend("topright", lwd=1.5, lty=c(1,2,4), col=c("red", "blue", "green"),
         legend=c("Running Mean", "Gauss Kernel", "LOESS"))
```



b) Visually, there are two clusters that divide the observations: the first group contains the observations of the 60/70s, the second groups contains the observations after 1980. There seems to be a slight decrease in both of these clusters. However, we can only make a qualitative statement since is not possible to give quantitative evidence to this claim just by using non-parametric smoothing. Parametric methods such as linear regression would allow us to verify this claim formally.

3. **a)** False. The Gaussian kernel smoother is particularly sensitive to outliers whereas the LOESS smoother (using the R function with the default settings) is not as much influenced by outliers.

b) False. When using a smoother, the functional relation remains unknown and it is only possible to make predictions in the range of the observed values. When using a linear model, a formula is computed which can in principle be used for arbitrary values of the input. However, this is usually not a good idea, i.e. the results for values outside of the range of the observations are often questionable.

c) False. A smoother can always be chosen such that the residual sum of squares is arbitrarily small. Therefore, this criterion is only suitable when the functional form is predetermined (as for instance in a linear model).

d) True. The model is linear in the parameters $\beta_1, \ldots, \beta_4$.

e) False. The model is not linear in the parameters β_1, β_2. When taking partial derivatives $\frac{\partial}{\partial \beta_j}$, β_2 does not vanish from the resulting expression.