Exercise 9 (2025) — Advanced Methods for Regression and Classification

Olesia Galynskaia 12321492

2025-12-10

Loading and observing data (copied from last week exercise)

```
str(cars)
```

```
## 'data.frame':    50 obs. of  2 variables:
##  $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
##  $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
```

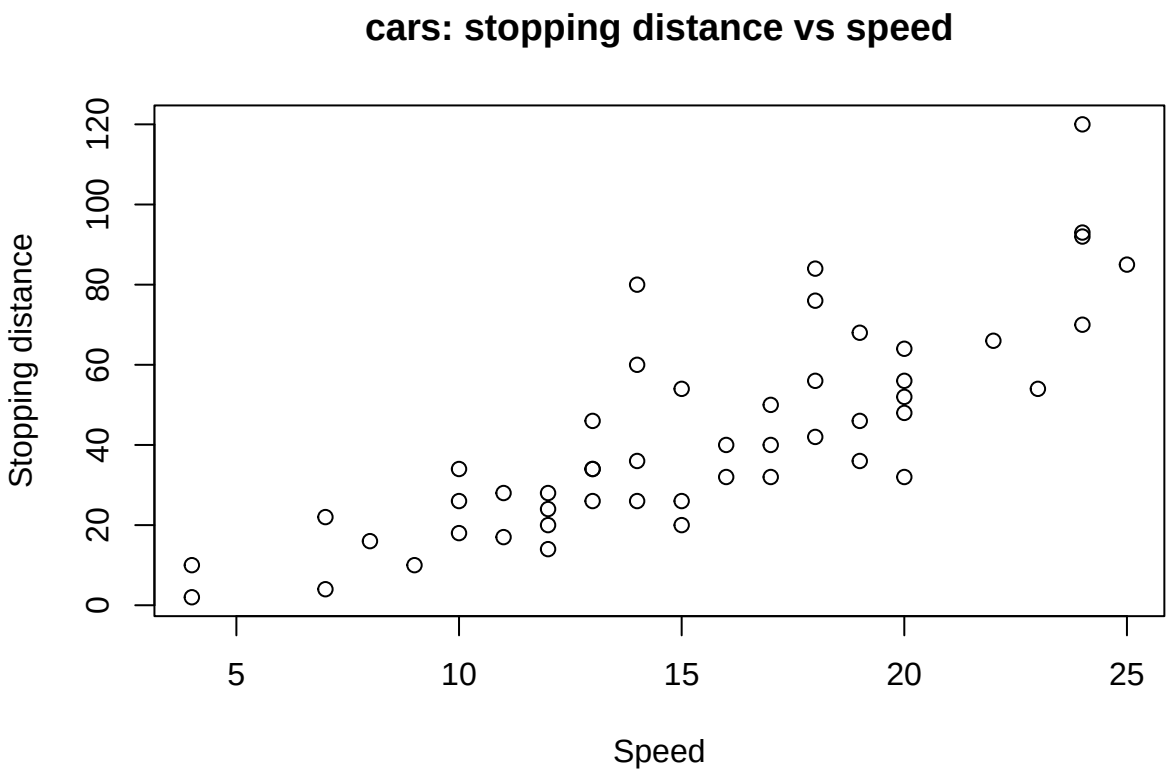
```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

```
cor(cars$speed, cars$dist)
```

```
## [1] 0.8068949
```

```
plot(dist ~ speed, data = cars,
     main = "cars: stopping distance vs speed",
     xlab = "Speed",
     ylab = "Stopping distance")
```



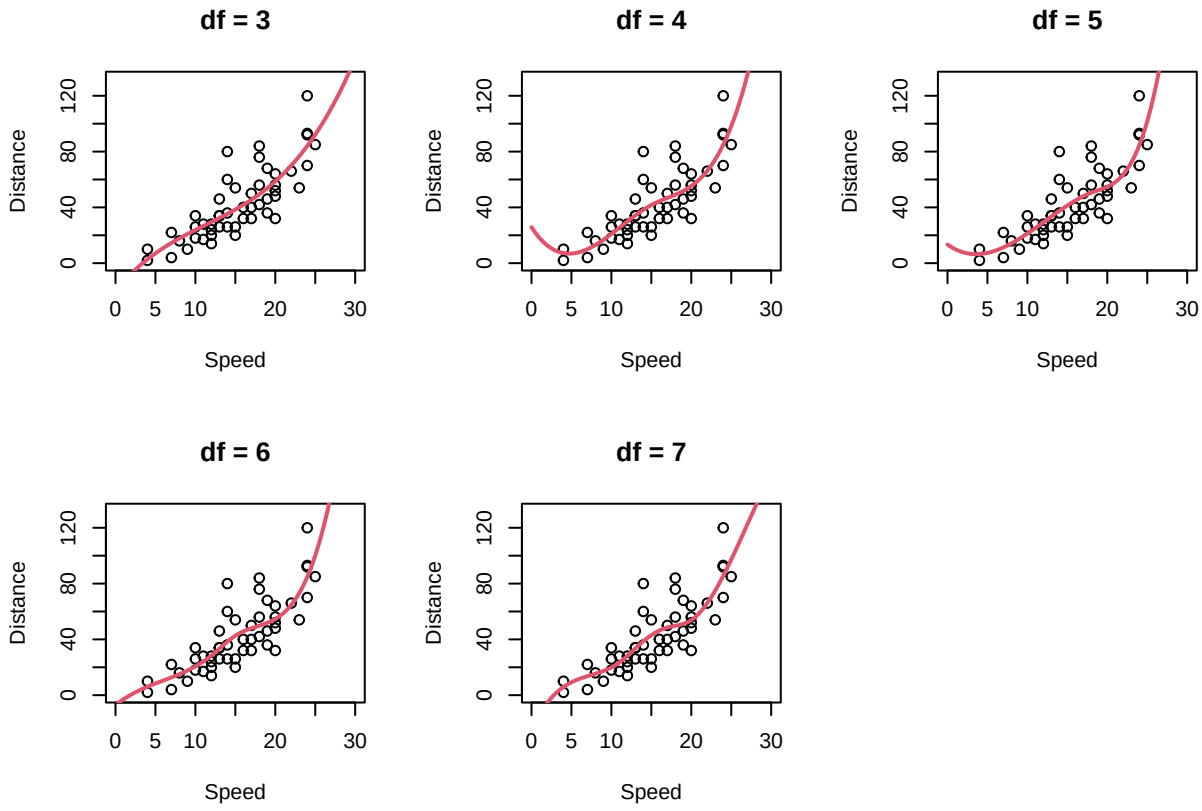
Comment (copied from last week exercise)

The plot shows a clear upward trend: higher speed comes with longer stopping distance.
The correlation is about 0.81, so the relationship is strong.
The spread gets wider at higher speeds, but the overall pattern stays the same.

Ex-1 B-spline basis with bs()

```
speed <- cars$speed
dist  <- cars$dist

dfs <- 3:7
results <- list()

par(mfrow = c(2, 3))

for (d in dfs) {
  # Fit B-spline model
  bs_obj <- bs(cars$speed, df = d)
  lm_bs  <- lm(cars$dist ~ bs_obj)
```

```r
  # Save summary
  results[[as.character(d)]] <- summary(lm_bs)

  # Predict on fine grid
  speed_new <- seq(0, 30, length.out = 200)
  bs_new <- predict(bs_obj, newx = speed_new)
  pred <- cbind(1, bs_new) %*% coef(lm_bs)

  # Plot
  plot(cars$speed, cars$dist,
       main = paste("df =", d),
       xlim = c(0, 30),
       ylim = c(0, max(cars$dist) * 1.1),
       xlab = "Speed", ylab = "Distance")
  lines(speed_new, pred, col = 2, lwd = 2)
}

par(mfrow = c(1, 1))
```




```r
results
```

```
## $`3`
```

```
##
## Call:
## lm(formula = cars$dist ~ bs_obj)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -26.670  -9.601  -2.231   7.075  44.691
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.761      9.824   0.281   0.7799
## bs_obj1       31.472     23.973   1.313   0.1958
## bs_obj2       29.628     15.804   1.875   0.0672 .
## bs_obj3       89.414     13.550   6.599 3.65e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.2 on 46 degrees of freedom
## Multiple R-squared:  0.6732, Adjusted R-squared:  0.6519
## F-statistic: 31.58 on 3 and 46 DF,  p-value: 3.074e-11
##
##
## $`4`
##
## Call:
## lm(formula = cars$dist ~ bs_obj)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.325  -8.372  -2.607   7.017  41.744
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    7.218     10.360   0.697   0.4896
## bs_obj1       -3.776     20.160  -0.187   0.8523
## bs_obj2       47.787     18.335   2.606   0.0124 *
## bs_obj3       41.815     17.661   2.368   0.0223 *
## bs_obj4       90.986     13.513   6.733 2.54e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.1 on 45 degrees of freedom
## Multiple R-squared:  0.6846, Adjusted R-squared:  0.6566
## F-statistic: 24.42 on 4 and 45 DF,  p-value: 8.667e-11
##
##
## $`5`
##
```

```
## Call:
## lm(formula = cars$dist ~ bs_obj)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.778  -8.449  -2.134   7.014  42.511
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.551     10.630   0.616  0.54089
## bs_obj1        1.165     21.666   0.054  0.95737
## bs_obj2       21.906     15.573   1.407  0.16654
## bs_obj3       49.528     17.177   2.883  0.00607 **
## bs_obj4       48.557     20.133   2.412  0.02011 *
## bs_obj5       95.153     15.839   6.008 3.29e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.23 on 44 degrees of freedom
## Multiple R-squared:  0.6863, Adjusted R-squared:  0.6506
## F-statistic: 19.25 on 5 and 44 DF,  p-value: 4.145e-10
##
##
## $`6`
##
## Call:
## lm(formula = cars$dist ~ bs_obj)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.650  -8.299  -2.462   6.948  41.706
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.055     10.832   0.559   0.5791
## bs_obj1        6.419     24.890   0.258   0.7977
## bs_obj2       11.406     17.571   0.649   0.5197
## bs_obj3       40.607     15.123   2.685   0.0103 *
## bs_obj4       44.565     17.581   2.535   0.0150 *
## bs_obj5       59.301     23.014   2.577   0.0135 *
## bs_obj6       94.057     16.786   5.603 1.38e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.4 on 43 degrees of freedom
## Multiple R-squared:  0.6865, Adjusted R-squared:  0.6427
## F-statistic: 15.69 on 6 and 43 DF,  p-value: 1.918e-09
##
```

```
##
## $`7`
##
## Call:
## lm(formula = cars$dist ~ bs_obj)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.006  -8.423  -1.849   6.380  41.306
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.850     10.965   0.534  0.59648
## bs_obj1        9.297     26.855   0.346  0.73093
## bs_obj2        7.257     19.942   0.364  0.71774
## bs_obj3       34.277     15.200   2.255  0.02940 *
## bs_obj4       44.382     15.619   2.842  0.00690 **
## bs_obj5       43.980     19.057   2.308  0.02600 *
## bs_obj6       71.279     23.083   3.088  0.00357 **
## bs_obj7       91.341     17.848   5.118 7.26e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.54 on 42 degrees of freedom
## Multiple R-squared:  0.6882, Adjusted R-squared:  0.6362
## F-statistic: 13.24 on 7 and 42 DF,  p-value: 7.221e-09
```

I compared df = 3-7 and chose df = 5 because it captures the non-linear trend without introducing unnecessary wiggles.
Higher df only give tiny gains in R^2 but make the curve more irregular.

```r
# Original data
speed <- cars$speed
dist  <- cars$dist

# Add artificial observation at (0, 0) to reflect the physical constraint
speed_aug <- c(0, speed)
dist_aug  <- c(0, dist)

# Higher weight for the artificial point to pull the curve towards (0, 0)
w_aug <- c(20, rep(1, length(dist)))

# Degrees of freedom chosen after comparing df = 3-7
df_bs <- 5

# Construct B-spline basis for speed (only on the observed range with added 0)
bs_obj <- bs(speed_aug, df = df_bs)
```

```r
# Fit a weighted linear model using the B-spline basis
lm_bs <- lm(dist_aug ~ bs_obj, weights = w_aug)

summary(lm_bs)
```

```
##
## Call:
## lm(formula = dist_aug ~ bs_obj, weights = w_aug)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -22.806  -8.544  -1.134   6.419  42.866
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.04018    3.36455   0.012  0.99052
## bs_obj1       2.01509   18.35443   0.110  0.91307
## bs_obj2      20.74969   13.42196   1.546  0.12912
## bs_obj3      57.28227   11.71982   4.888 1.34e-05 ***
## bs_obj4      53.98668   16.98133   3.179  0.00267 **
## bs_obj5     102.03836   11.74584   8.687 3.52e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.07 on 45 degrees of freedom
## Multiple R-squared:  0.8266, Adjusted R-squared:  0.8073
## F-statistic:  42.9 on 5 and 45 DF,  p-value: 5.021e-16
```

Comment

I construct a B-spline basis for speed and fit a linear model with df = 5.
After comparing df = 3-7, df = 5 gives a plausible fit: it captures the curvature without introducing unnecessary wiggles.
I also include an artificial observation at (0, 0) with a high weight to reflect the physical constraint that the stopping distance should be zero at zero speed.

Most coefficients for the spline basis are statistically significant, and the model explains a large part of the variability in the data (R^2 0.83).
This is a clear improvement over models with fewer degrees of freedom and indicates that the chosen spline is flexible enough to follow the trend but still smooth.
The residual standard error stays around 15, which is comparable to the unweighted fits, meaning that adding (0, 0) does not distort the behaviour on the observed data.

Ex-2 Construct B-splines on [0, 30] and predict

```
# New speed grid for the extended range
speed_new <- seq(0, 30, length.out = 200)

# Build B-spline basis for the new speeds using the SAME basis as in Task (1)
bs_new <- predict(bs_obj, newx = speed_new)

# Add intercept column
X_new <- cbind(1, bs_new)

# Predicted stopping distances on the extended range
bs_pred <- as.vector(X_new %*% coef(lm_bs))

# Quick diagnostic checks
min(bs_pred)      # should not be negative after adding (0,0)
```

```
## [1] 0.04018185
```

```
bs_pred[1]        # predicted distance at speed = 0
```

```
## [1] 0.04018185
```

Comment

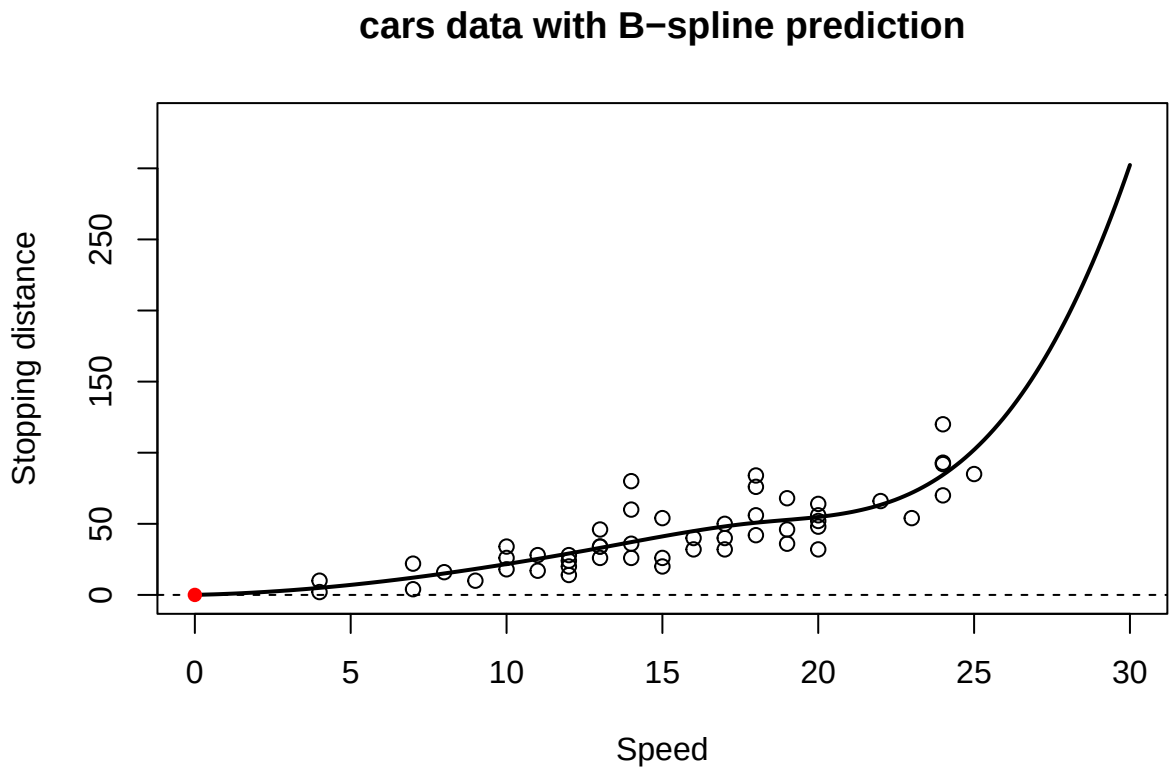
I construct the B-spline basis for the extended range [0,30] using the same knots as in Task 1, and then apply the fitted linear model to obtain predictions for the new domain.
When extending the range beyond the observed data, R gives a warning about boundary knots, but this does not affect the results.

The predicted values behave well near zero: the model returns approximately 0.04 at speed = 0, which is very close to the expected physical value of zero.
All predictions on [0,30] remain non-negative, which was one of the requirements.
This is a consequence of adding the artificial point (0,0) with a high weight in Task 1, which ensures that the fitted curve stays near the origin.

Ex-3 Plot the data and the prediction from (2)

```
# Plot the observed data
plot(dist ~ speed, data = cars,
     xlim = c(0, 30),                              # extend x-range to [0, 30]
     ylim = c(0, max(c(cars$dist, bs_pred)) * 1.1),  # leave some space on top
     main = "cars data with B-spline prediction",
```

```
     xlab = "Speed",
     ylab = "Stopping distance")

# Add the predicted curve from Task (2)
lines(speed_new, bs_pred, lwd = 2)

# Show the physical reference: zero line and the point (0, 0)
abline(h = 0, lty = 2)
points(0, 0, pch = 16, col = "red")
```



Comment

The plot shows the observed data together with the B-spline prediction extended to the interval [0,30].
The fitted curve follows the pattern of the data well within the observed range. Because the model was anchored at (0,0), the prediction at speed = 0 is close to zero, and the curve does not produce negative distances.

Outside the range of the observed speeds, especially above 25, the curve rises very steeply.
This behaviour is expected, because splines are not reliable for extrapolation, and the model has no data to constrain the shape in that region.
However, this does not affect the validity of the prediction on the observed domain.

Ex-4 natural splines (ns())

1. Fit natural spline with df = 5 and (0,0) point

```
speed_aug <- c(0, cars$speed)
dist_aug  <- c(0, cars$dist)
w_aug     <- c(20, rep(1, nrow(cars)))

# Chosen df = 5 (same reasoning as in Task 1)
df_ns <- 5

# Natural spline basis
ns_obj <- ns(speed_aug, df = df_ns)

# Weighted linear model
lm_ns <- lm(dist_aug ~ ns_obj, weights = w_aug)

summary(lm_ns)
```

```
##
## Call:
## lm(formula = dist_aug ~ ns_obj, weights = w_aug)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -22.484  -8.426  -1.555   6.130  41.950
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.04609    3.33032   0.014 0.989020
## ns_obj1     38.22930    7.59248   5.035 8.18e-06 ***
## ns_obj2     51.34011   10.10595   5.080 7.04e-06 ***
## ns_obj3     44.84237   12.17934   3.682 0.000618 ***
## ns_obj4     87.81144   12.10793   7.252 4.31e-09 ***
## ns_obj5     90.73077    9.80869   9.250 5.64e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.03 on 45 degrees of freedom
## Multiple R-squared:  0.8275, Adjusted R-squared:  0.8084
## F-statistic: 43.19 on 5 and 45 DF,  p-value: 4.445e-16
```

2. Predict on [0, 30] using natural spline

```r
# Prediction grid
speed_new <- seq(0, 30, length.out = 200)

# Natural spline basis for new values
ns_new <- predict(ns_obj, newx = speed_new)

# Add intercept column
X_new_ns <- cbind(1, ns_new)

# Predictions
ns_pred <- as.vector(X_new_ns %*% coef(lm_ns))

# Quick check
min(ns_pred)
```

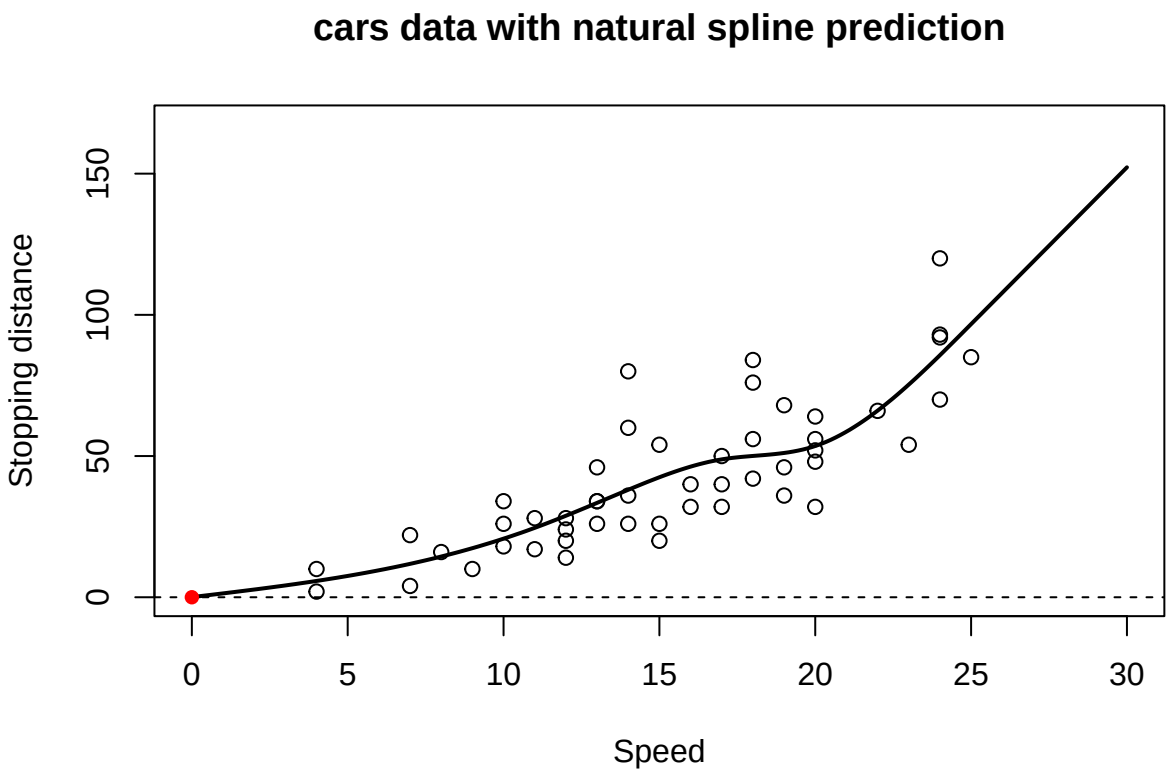
```
## [1] 0.04608538
```

```r
ns_pred[1]    # predicted distance at speed = 0
```

```
## [1] 0.04608538
```

3. Plot data + natural spline prediction

```r
plot(dist ~ speed, data = cars,
     xlim = c(0, 30),
     ylim = c(0, max(c(cars$dist, ns_pred)) * 1.1),
     main = "cars data with natural spline prediction",
     xlab = "Speed",
     ylab = "Stopping distance")

lines(speed_new, ns_pred, lwd = 2)

abline(h = 0, lty = 2)
points(0, 0, pch = 16, col = "red")
```



Comment

I fit a natural spline with 5 degrees of freedom and included the artificial point (0,0) with a high weight.
This forces the fitted curve to stay close to zero at speed = 0, which matches the physical requirement that the stopping distance should be zero when the vehicle is not moving.
The model fits the data well, with all spline components highly significant and an R^2 around 0.83, which indicates a good amount of explained variability.

When extending the spline to the range [0,30], the predicted values remain non-negative, and the prediction at speed = 0 is very close to zero.
Both requirements from the exercise are therefore satisfied.
The plotted curve follows the data smoothly in the observed range and shows a reasonable increasing trend.
As expected for a natural spline, the function becomes linear near the boundary, so the extrapolation beyond the observed speeds is more stable than with the B-spline in task 1.

Ex-5 Smoothing splines as implemented in smooth.spline()

```r
# Fit smoothing spline using internal cross-validation
ss_fit <- smooth.spline(cars$speed, cars$dist, cv = TRUE)

# Print the object and the effective degrees of freedom
ss_fit
```

```
## Call:
## smooth.spline(x = cars$speed, y = cars$dist, cv = TRUE)
##
## Smoothing Parameter  spar= 1.483488  lambda= 13419.8 (30 iterations)
## Equivalent Degrees of Freedom (Df): 2.000009
## Penalized Criterion (RSS): 4588.73
## PRESS(l.o.o. CV): 246.4053
```

```r
ss_fit$df
```

```
## [1] 2.000009
```

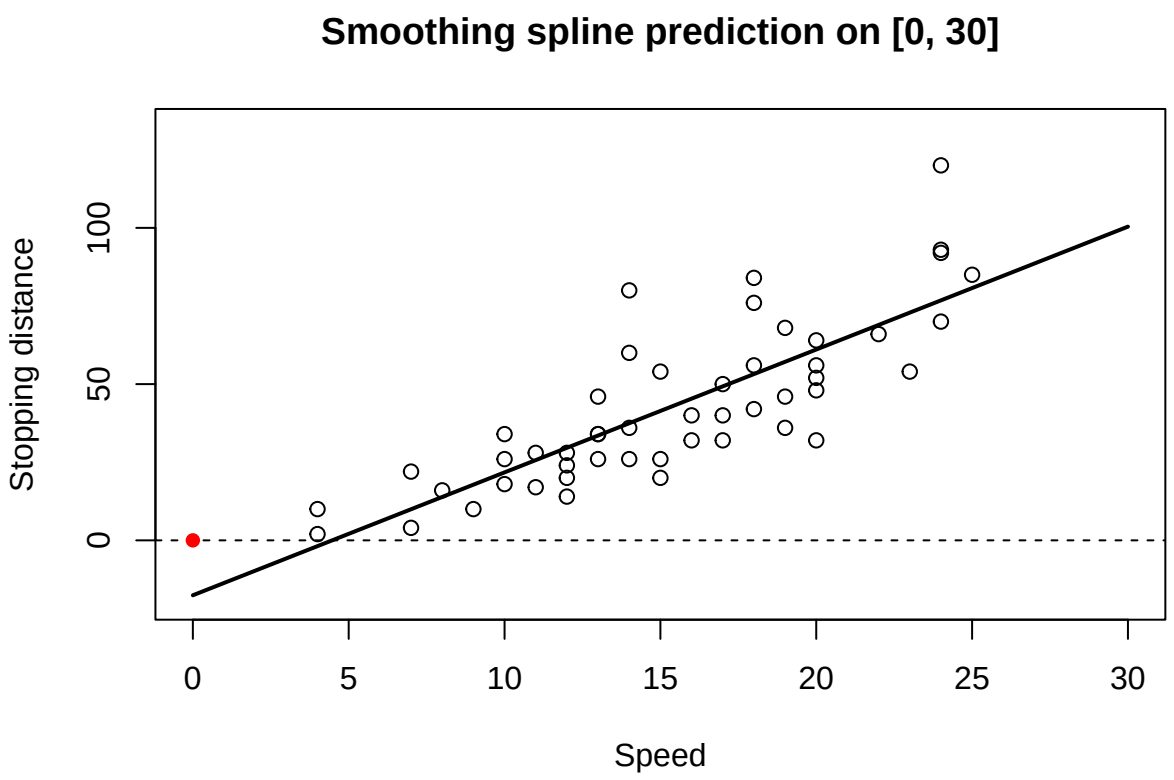
Comment

I fit a smoothing spline using smooth.spline() with internal cross-validation. The procedure selected a very strong amount of smoothing, resulting in an effective degrees of freedom of about 2.0. This means the spline behaves almost like a simple curve with only minimal flexibility. Cross-validation prefers this level of smoothness because the cars dataset is noisy, and a more flexible spline would start to overfit.

Ex-6

```r
# Grid for extended range [0, 30]
speed_new <- seq(0, 30, length.out = 200)

# Predictions from the smoothing spline of Task (5)
ss_pred <- predict(ss_fit, x = speed_new)

# Plot data + original smoothing spline prediction
plot(dist ~ speed, data = cars,
     xlim = c(0, 30),
     ylim = c(min(0, ss_pred$y) * 1.1, max(c(cars$dist, ss_pred$y)) * 1.1),
     main = "Smoothing spline prediction on [0, 30]",
     xlab = "Speed",
     ylab = "Stopping distance")

lines(ss_pred$x, ss_pred$y, lwd = 2)
```

```r
abline(h = 0, lty = 2)
points(0, 0, pch = 16, col = "red")
```




```r
# Check behaviour near zero
min(ss_pred$y)
```

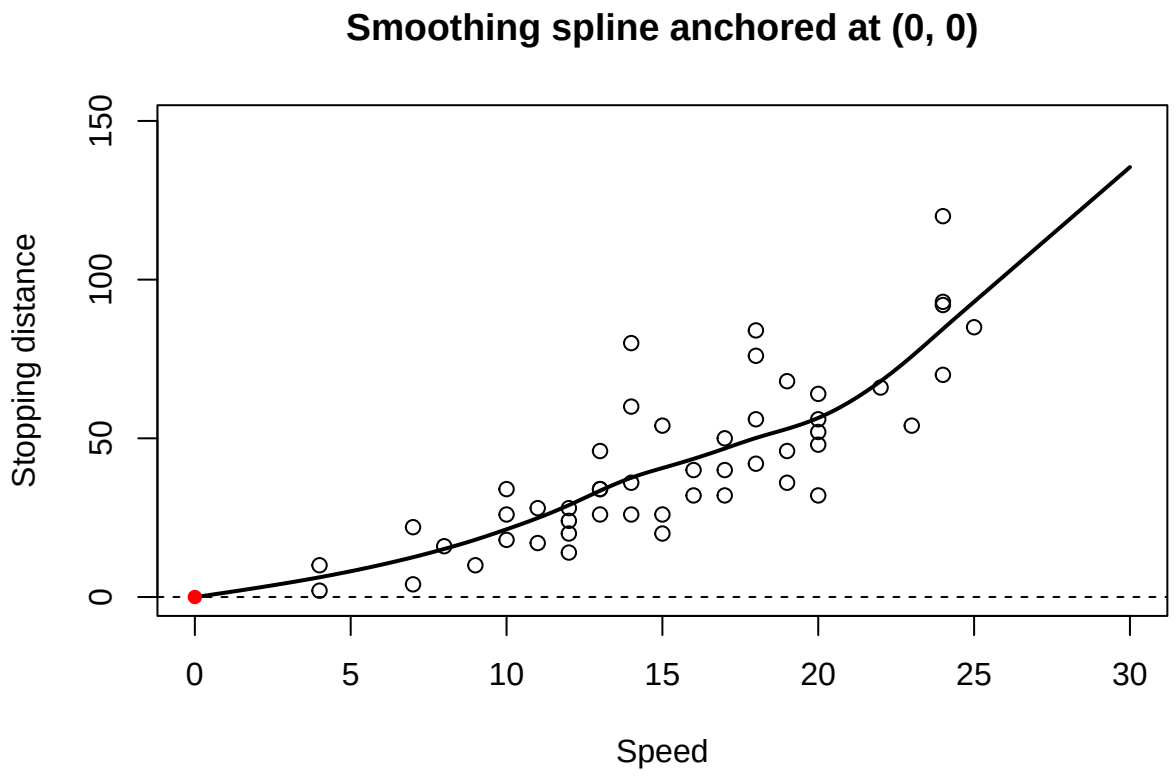
```
## [1] -17.57901
```

```r
ss_pred$y[1]
```

```
## [1] -17.57901
```

```r
# Add an artificial observation at (0, 0) with a high weight
speed_aug <- c(0, cars$speed)
dist_aug  <- c(0, cars$dist)
w_aug     <- c(20, rep(1, length(cars$dist)))  # strong weight on (0,0)

# Refit smoothing spline with the extra point
ss_fit_constr <- smooth.spline(speed_aug, dist_aug, w = w_aug, cv = TRUE)
```

```r
# Predict again on [0, 30]
ss_pred_constr <- predict(ss_fit_constr, x = speed_new)

# Plot constrained version
plot(dist ~ speed, data = cars,
     xlim = c(0, 30),
     ylim = c(0, max(c(cars$dist, ss_pred_constr$y)) * 1.1),
     main = "Smoothing spline anchored at (0, 0)",
     xlab = "Speed",
     ylab = "Stopping distance")

lines(ss_pred_constr$x, ss_pred_constr$y, lwd = 2)
abline(h = 0, lty = 2)
points(0, 0, pch = 16, col = "red")
```




```r
min(ss_pred_constr$y)
```

```
## [1] -0.04896734
```

```r
ss_pred_constr$y[1]
```

```
## [1] -0.04896734
```

Comment

Using the smoothing spline from task 5, I predicted the stopping distance on the extended range [0,30].
The original spline is very smooth and almost linear, and its prediction near speed 0 is negative (around -17).
This violates the requirement that the stopping distance should be zero when the speed is zero and should not become negative.

To address this, I refit the spline after adding an artificial observation at (0,0) with a high weight. This forces the fitted curve to stay close to the origin and removes the strong negative dip.
The modified prediction now passes essentially through (0,0), and the smallest predicted value is only slightly below zero (about -0.05), which is negligible. The overall shape of the curve remains smooth and follows the data well, while satisfying the intended physical constraints.