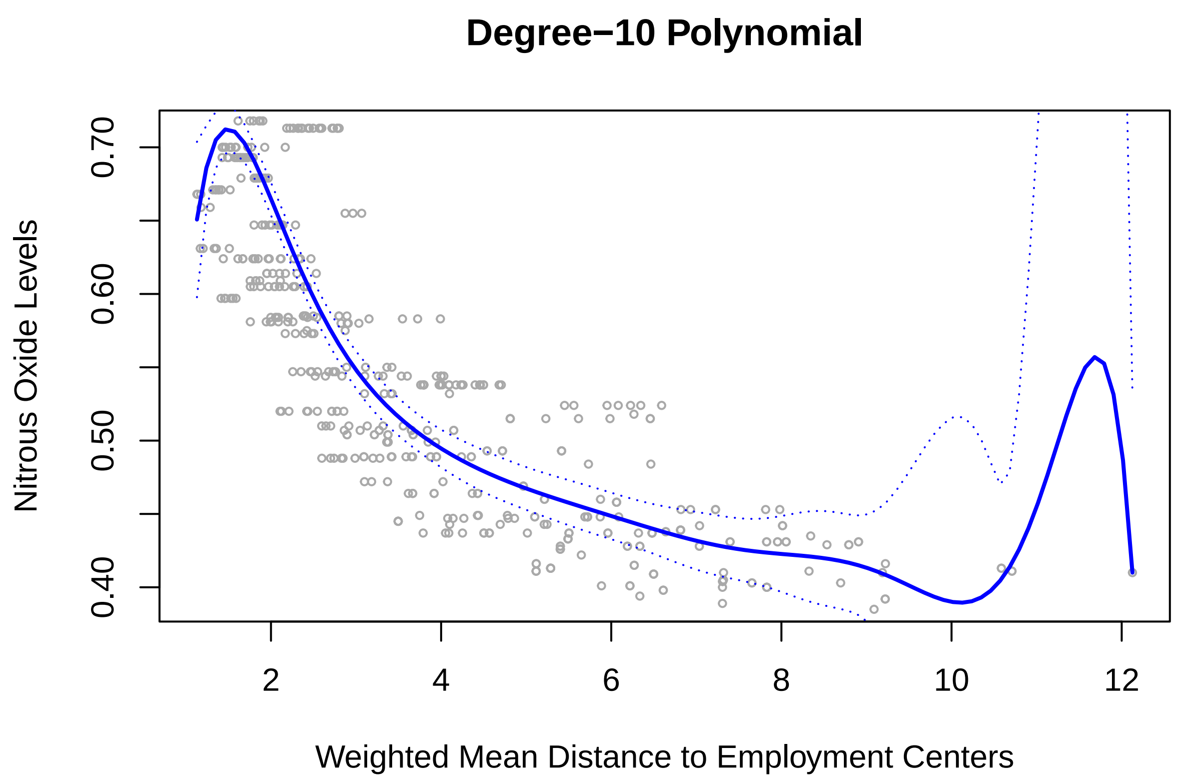
Solution HW 3

2023-11-1

Problem 4:

Part 1:

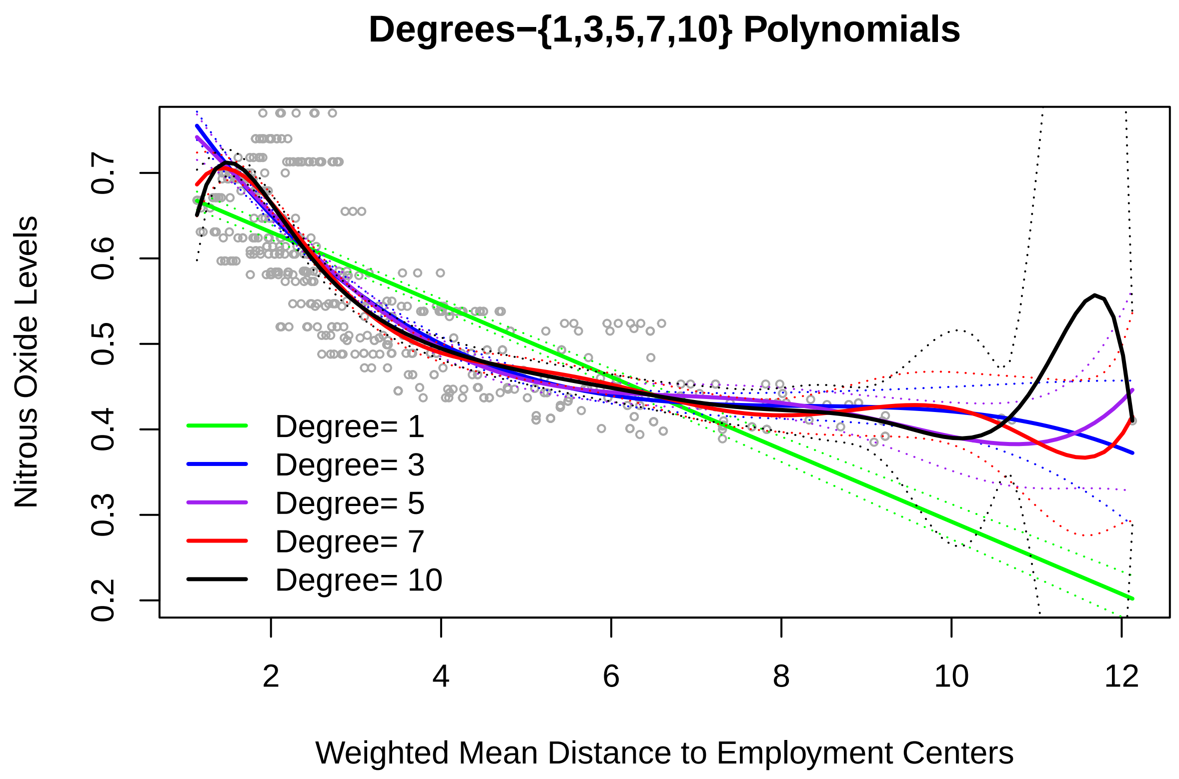
```r
rm(list = ls())
library(MASS)
data("Boston")
mod1 <- glm(nox ~ poly(dis, 10), data=Boston)

dislims <- range(Boston$dis)
dis.grid <- seq(from = dislims[1], to = dislims[2], length.out = 100)

preds <- predict(mod1, newdata = list(dis=dis.grid), se=TRUE)
# obtain confidence interval for fitted value
se.bands <- cbind(preds$fit+2*preds$se.fit,preds$fit-2*preds$se.fit)

plot(Boston$dis, Boston$nox, cex=.5, col="darkgrey",
     xlab="Weighted Mean Distance to Employment Centers",
     ylab="Nitrous Oxide Levels", ylim = range(preds$fit))
title(main="Degree-10 Polynomial")
lines(dis.grid, preds$fit, lwd=2, col="blue")
matlines(dis.grid, se.bands, lwd=1, col="blue", lty=3)
```



The width of confidence intervals gest wild at the tails where much fewer points are available for fitting the model.

Part 2:

```
dim_set <- c(1,3,5,7,10)
preds_list <- list(dim=dim_set,
                   color=c("green", "blue", "purple", "red", "black"),
                   preds=matrix(NA, nrow=length(dim_set), ncol=length(dis.grid)),
                   se.bands=lapply(1:length(dim_set), matrix,
                                   data=NA, nrow=length(dis.grid), ncol=2),
                   rss = numeric(length(dim_set)))
rownames(preds_list$preds) <- dim_set
names(preds_list$se.bands) <- dim_set

for(i in 1:length(dim_set)){
  cur_dim <- dim_set[i]
  mod <- glm(nox ~ poly(dis, cur_dim), data=Boston)
  preds <- predict(mod, newdata = list(dis=dis.grid), se=TRUE)
  preds2 <- predict(mod, newdata = Boston)
  se.bands <- cbind(preds$fit+2*preds$se.fit,preds$fit-2*preds$se.fit)
  preds_list$preds[i,] <- preds$fit
  preds_list$se.bands[[i]] <- se.bands
  preds_list$rss[i] <- sum((preds2-Boston$nox)^2)
}


plot(Boston$dis, Boston$nox, cex=.5, col="darkgrey",
     xlab="Weighted Mean Distance to Employment Centers",
     ylab="Nitrous Oxide Levels", ylim = range(preds_list$preds))
title(main="Degrees-{1,3,5,7,10} Polynomials")
for(i in 1:length(dim_set)){
  lines(dis.grid, preds_list$preds[i,], lwd=2, col=preds_list$color[i])
  matlines(dis.grid, preds_list$se.bands[[i]], lwd=1,
           col=preds_list$color[i], lty=3)
}
legend("bottomleft", legend = paste("Degree=", dim_set), lwd = 2, lty = 1,
       col = preds_list$color, bty = "n")
```



```
knitr::kable(cbind(preds_list$dim, preds_list$rss),
             col.names = c("Polynomial Degree", "RSS"),
             digits=3)
```

Polynomial Degree	RSS
1	2.769
3	1.934
5	1.915
7	1.849
10	1.832

The RSS decreases as the degree gets larger. This is as expected because higher degree means using additional features in the polynomial regression, resulting smaller training MSEs.

Part 3:

```r
library(boot)
# set.seed(20231101)
cv_error <- numeric(length(dim_set))

for(i in 1:length(dim_set)){
  cur_dim <- dim_set[i]
  mod <- glm(nox ~ poly(dis, cur_dim), data=Boston)
  cv <- cv.glm(Boston, mod, K=10)
  cv_error[i] <- cv$delta[1]
}

cv_error
```

```
## [1] 0.005529399 0.003882156 0.004097241 0.008113967 0.010557779
```

The degree-3 polynomial seems to have the smallest 10-CV error.

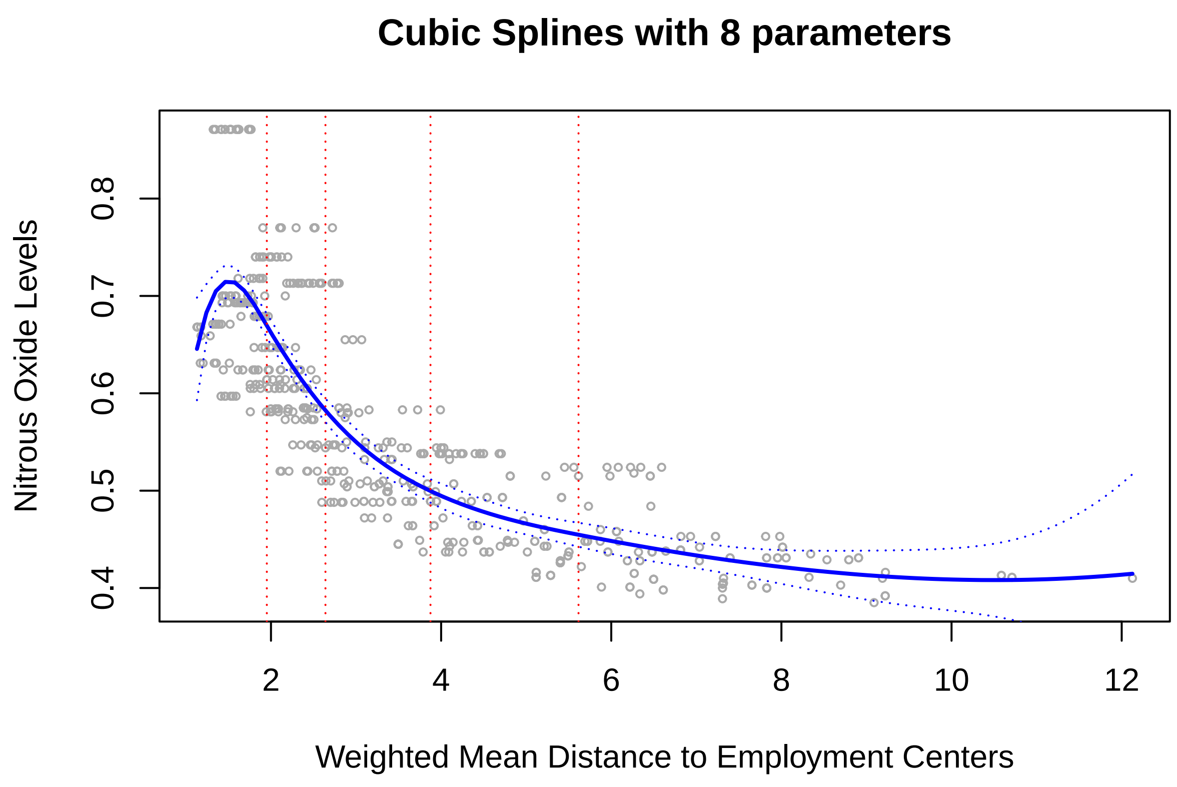
Part 4:

```r
library(splines)

mod_bs1 <- glm(nox ~ bs(dis, df=7), data=Boston)
knots_bs <- attr(bs(Boston$dis, df=7), "knots")
knots_bs
```

```
## [1] 1.9512 2.6403 3.8750 5.6150
```

We are using the default knot choices in `bs()` so the knots are chosen based on the quantiles of the original feature. Since `bs()` excludes the intercept term, we will have $7 - 3 = 4$ knots.

```r
pred_bs1 <- predict(mod_bs1, newdata = list(dis=dis.grid), se = TRUE)

se.bands_bs1 <- cbind(pred_bs1$fit+2*pred_bs1$se.fit,
                      pred_bs1$fit-2*pred_bs1$se.fit)
plot(Boston$dis, Boston$nox, xlim=dislims, cex=.5, col="darkgrey",
     xlab="Weighted Mean Distance to Employment Centers",
     ylab="Nitrous Oxide Levels")
title(main="Cubic Splines with 8 parameters")
lines(dis.grid, pred_bs1$fit, lwd=2, col = "blue")
for (k in 1:length(knots_bs))
  abline(v = knots_bs[k], col = "red", lty = 3)
matlines(dis.grid, se.bands_bs1, lwd=1, col="blue", lty=3)
```



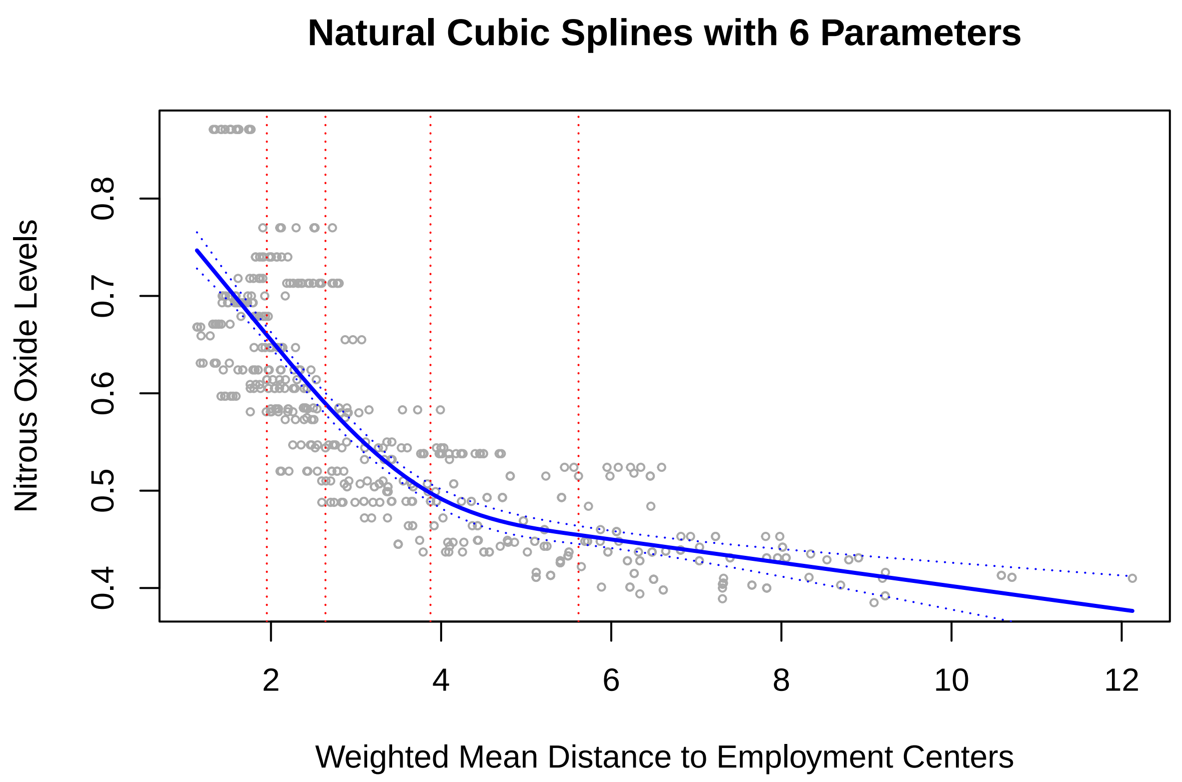
Part 5:

```
mod_ns1 <- glm(nox ~ ns(dis, knots = c(knots_bs[2:3]),
                        Boundary.knots = range(knots_bs)), data = Boston)
knots_ns <- attr(ns(Boston$dis, knots = c(knots_bs[2:3]),
                    Boundary.knots = range(knots_bs)), "knots")
knots_ns
```

```
## [1] 2.6403 3.8750
```

We are using `ns()` to specify the knots. To match with the knots in the `bs()` as specified above, we set the interior knots as the 2nd and 3rd knots from `bs()` and set the smallest and largest knots from `bs()` as the boundary knots.

```
pred_ns1 <- predict(mod_ns1, newdata = list(dis=dis.grid), se = TRUE)

se.bands_ns1 <- cbind(pred_ns1$fit+2*pred_ns1$se.fit,
                      pred_ns1$fit-2*pred_ns1$se.fit)
plot(Boston$dis, Boston$nox, xlim=dislims, cex=.5, col="darkgrey",
     xlab="Weighted Mean Distance to Employment Centers",
     ylab="Nitrous Oxide Levels")
title(main="Natural Cubic Splines with 6 Parameters")
lines(dis.grid, pred_ns1$fit, lwd=2, col = "blue")
for (k in 1:length(knots_bs))
  abline(v = knots_bs[k], col = "red", lty = 3)
matlines(dis.grid, se.bands_ns1, lwd=1, col="blue", lty=3)
```

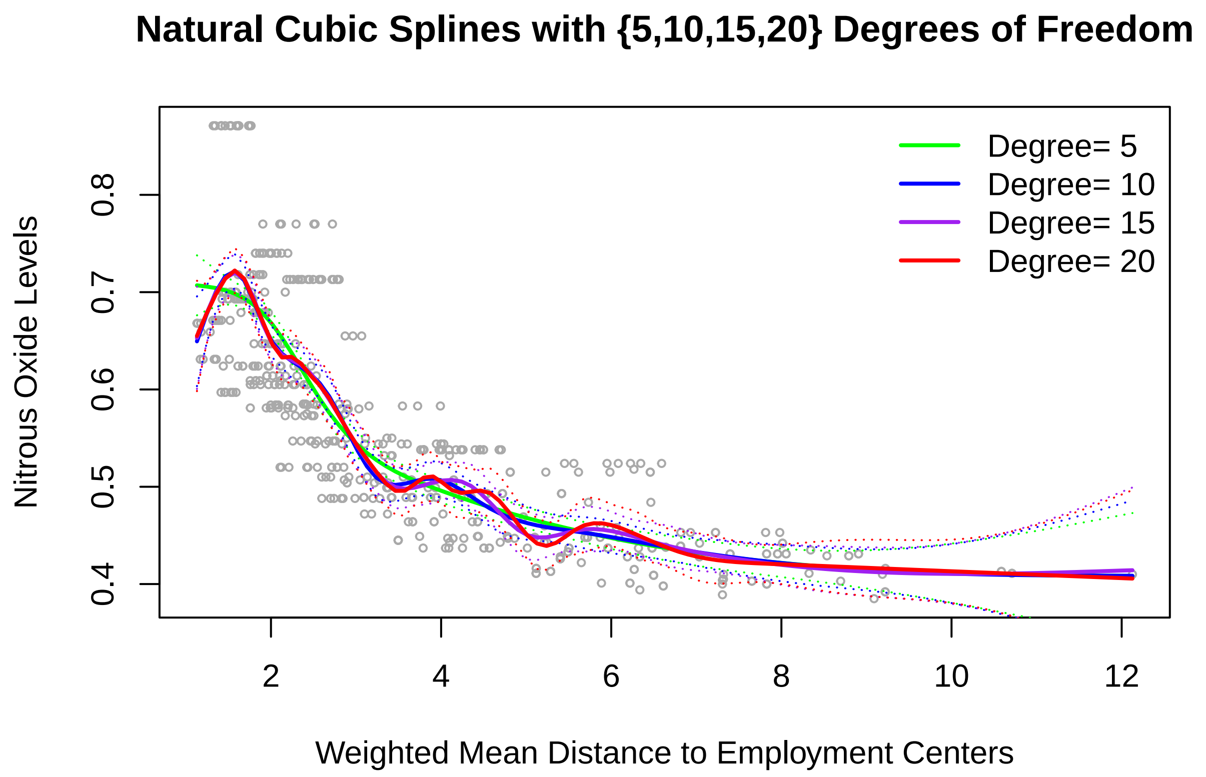


As expected, the fitted lines are linear beyond the boundary knots. Moreover, the confidence band gets narrow near the tails comparing to the cubic splines.

Part 6:

```
df_set <- c(5, 10, 15, 20)
preds_list <- list(df=df_set,
                   color=c("green", "blue", "purple", "red"),
                   preds=matrix(NA, nrow=length(df_set), ncol=length(dis.grid)),
                   se.bands=lapply(1:length(df_set), matrix,
                                   data=NA, nrow=length(dis.grid), ncol=2),
                   rss = numeric(length(df_set)))
rownames(preds_list$preds) <- df_set
names(preds_list$se.bands) <- df_set


for(i in 1:length(df_set)){
  cur_df <- df_set[i]
  mod <- glm(nox ~ ns(dis, df=cur_df), data=Boston)
  preds <- predict(mod, newdata = list(dis=dis.grid), se=TRUE)
  preds2 <- predict(mod, newdata = Boston)
  se.bands <- cbind(preds$fit+2*preds$se.fit,preds$fit-2*preds$se.fit)
  preds_list$preds[i,] <- preds$fit
  preds_list$se.bands[[i]] <- se.bands
  preds_list$rss[i] <- sum((preds2-Boston$nox)^2)
}

plot(Boston$dis, Boston$nox, cex=.5, col="darkgrey",
```

```r
     xlab="Weighted Mean Distance to Employment Centers",
     ylab="Nitrous Oxide Levels")
title(main="Natural Cubic Splines with {5,10,15,20} Degrees of Freedom")
for(i in 1:length(df_set)){
  lines(dis.grid, preds_list$preds[i,], lwd=2, col=preds_list$color[i])
  matlines(dis.grid, preds_list$se.bands[[i]], lwd=1,
           col=preds_list$color[i], lty=3)
}
legend("topright", legend = paste("Degree=", df_set), lwd = 2, lty = 1,
       col = preds_list$color, bty = "n")
```



```r
knitr::kable(cbind(preds_list$df, preds_list$rss),
             col.names = c("Specified Degrees of Natural Cubic Splines", "RSS"),
             digits=3)
```

Specified Degrees of Natural Cubic Splines	RSS
5	1.860
10	1.789
15	1.780
20	1.771

Unsurprisingly, the model with the most knots has the smallest training RSS. This comes at the cost of potentially overfitting the data.

Part 7:

```r
library(boot)
cv_error <- numeric(length(df_set))

for(i in 1:length(df_set)){
  cur_df <- df_set[i]
  mod <- glm(nox ~ ns(dis, df=cur_df), data=Boston)
  cv <- cv.glm(Boston, mod, K=10)
  cv_error[i] <- cv$delta[1]
}
cv_error
```

```
## [1] 0.003744090 0.003652044 0.003746249 0.003794834
```

The best model is the natural cubic spline corresponding to d.f. equal to 10 in `ns()`.

Problem 5

The following function unifies the implementation of the three procedures, specified by the argument {`method`}.

```r
rm(list = ls())

# The following function implements the three procedures, specified by the argument
# method in {"knn", "wknn", "wlm"}
local_reg <- function(newx, x, y, k, method) {
  dist_vec <- abs(x - newx)
  ind_nn <- order(dist_vec)[1:k]
  # cat(ind_nn)
  x_vec <- x[ind_nn]
  max_dist <- dist_vec[ind_nn[k]]

  if (method == "knn")
    return(mean(y[ind_nn]))
  else {
    weight_vec <- (1 - (dist_vec[ind_nn] / max_dist) ** 3) ** 3
    weight_vec <- weight_vec / sum(weight_vec)

    if (method == "wknn")
      return(sum(weight_vec * y[ind_nn]))
    else # perform weighted linear regression
      lm_md <- lm(y[ind_nn] ~ x_vec, weights = weight_vec)
      newx_vec <- data.frame(x_vec = newx)
      return(predict(lm_md, newx_vec))
  }
}
```

Now let's generate the data and set the grid of x to draw fitted lines.

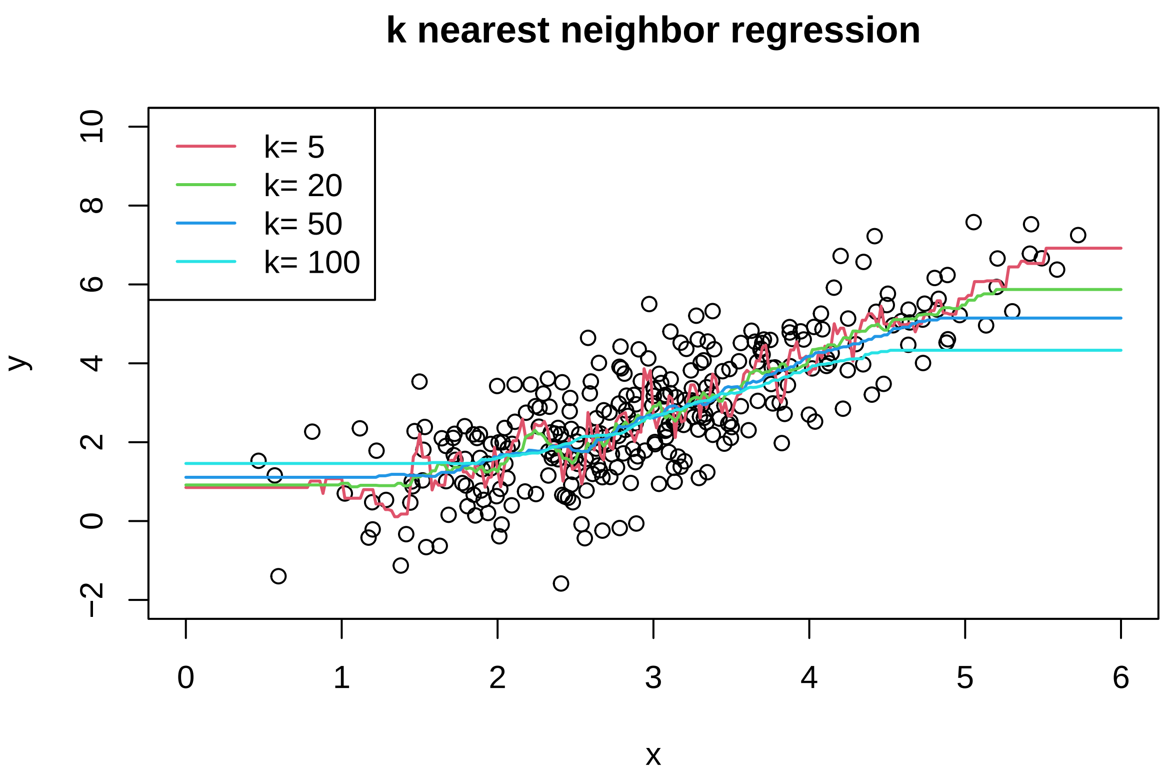
```r
# Simulate the data

n <- 300
set.seed(20231101)
```

```r
x_train <- rnorm(n, 3, 1)
y_train <- 0.5 + 0.1 *  x_train +  0.2 * x_train ^ 2 + rnorm(n)

x_test <- rnorm(n, 3, 1)
y_test <- 0.5 + 0.1 * x_test +  0.2 * x_test ^ 2 + rnorm(n)


x_grid <- seq(0, 6, by = 0.02)
```

Part 1

Here we draw fitted lines of k-nn for $k \in \{5, 20, 50, 100\}$.

```r
k_seq <- c(5, 20, 50, 100)

for (i in 1:length(k_seq)) {
  ki <- k_seq[i]
  y_fit_i <- sapply(x_grid, local_reg, x = x_train, y = y_train, k = ki,
                    method = "knn")
  if (i == 1)
    plot(x_train, y_train, xlim = c(0, 6), ylim = c(-2, 10),
         main = paste("k nearest neighbor regression"), ylab = "y", xlab = "x")

  lines(x_grid, y_fit_i, type = "l", lwd = 1.5, col = i+1)
}
legend("topleft", legend = paste("k=", k_seq), col = 2:(length(k_seq)+1), lwd = 1.5)
```

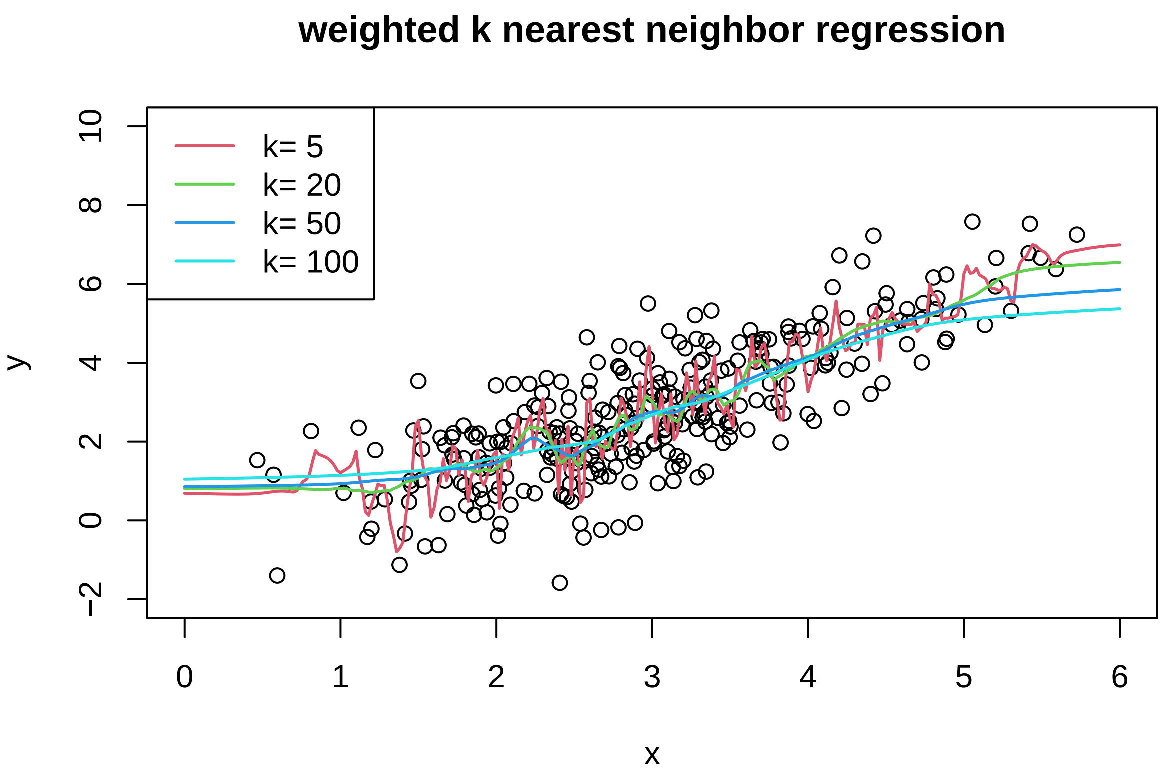


The larger k is, the more smoothing the fitted line is.

Part 2

Here we draw fitted lines of weighted k-nn for $k \in \{5, 20, 50, 100\}$.

```r
for (i in 1:length(k_seq)) {
  ki <- k_seq[i]
  y_fit_i <- sapply(x_grid, local_reg, x = x_train, y = y_train, k = ki,
                    method = "wknn")
  if (i == 1)
    plot(x_train, y_train, xlim = c(0, 6), ylim = c(-2, 10),
         main = paste("weighted k nearest neighbor regression"), ylab = "y", xlab = "x")

  lines(x_grid, y_fit_i, type = "l", lwd = 1.5, col = i+1)
}
legend("topleft", legend = paste("k=", k_seq), col = 2:(length(k_seq)+1), lwd = 1.5)
```

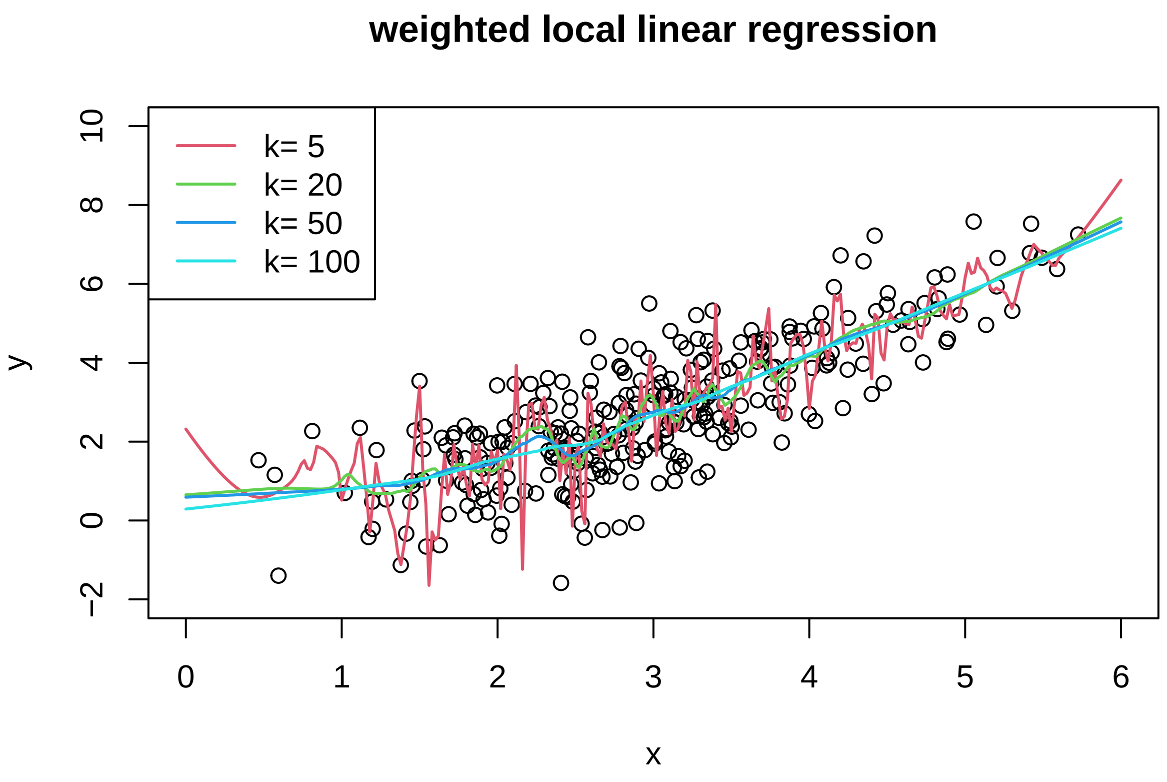


The smoothness of fitted lines gets improved comparing to knn. Only $k = 5$ appears to be very non-smooth.

Part 3

Here we draw fitted lines of local linear regressions for $k \in \{5, 20, 50, 100\}$.

```r
for (i in 1:length(k_seq)) {
  ki <- k_seq[i]
  y_fit_i <- sapply(x_grid, local_reg, x = x_train, y = y_train, k = ki,
                    method = "wlm")
  if (i == 1)
    plot(x_train, y_train, xlim = c(0, 6), ylim = c(-2, 10),
         main = paste("weighted local linear regression"), ylab = "y", xlab = "x")
  lines(x_grid, y_fit_i, type = "l", lwd = 1.5, col = i+1)
}
legend("topleft", legend = paste("k=", k_seq), col = 2:(length(k_seq)+1), lwd = 1.5)
```



The fitted line of $k = 5$ seems very unstable but the ones of other k are rather similar and smooth.

Part 4

We evaluate different procedures on the test data and compute their test MSEs.

```
mse_mat <- c()

for (i in 1:length(k_seq)) {
  mse_knn <- mean((y_test - sapply(x_test, local_reg, x = x_train, y = y_train,
                                   k = k_seq[i], method = "knn")) ** 2)
  mse_wknn <- mean((y_test - sapply(x_test, local_reg, x = x_train, y = y_train,
                                    k = k_seq[i], method = "wknn")) ** 2)
  mse_wlm <- mean((y_test - sapply(x_test, local_reg, x = x_train, y = y_train,
                                   k = k_seq[i], method = "wlm")) ** 2)
  mse_mat <- rbind(mse_mat, c("knn" = mse_knn, "wknn" = mse_wknn, "wlm" = mse_wlm))
}

rownames(mse_mat) <- paste("k=", k_seq)
round(mse_mat, 3)
```

```
##           knn  wknn   wlm
## k= 5    1.289 1.508 2.110
## k= 20   1.097 1.130 1.098
## k= 50  1.121 1.088 1.030
## k= 100 1.220 1.099 1.013
```

The local linear regression with $k = 100$ has the smallest test MSEs.