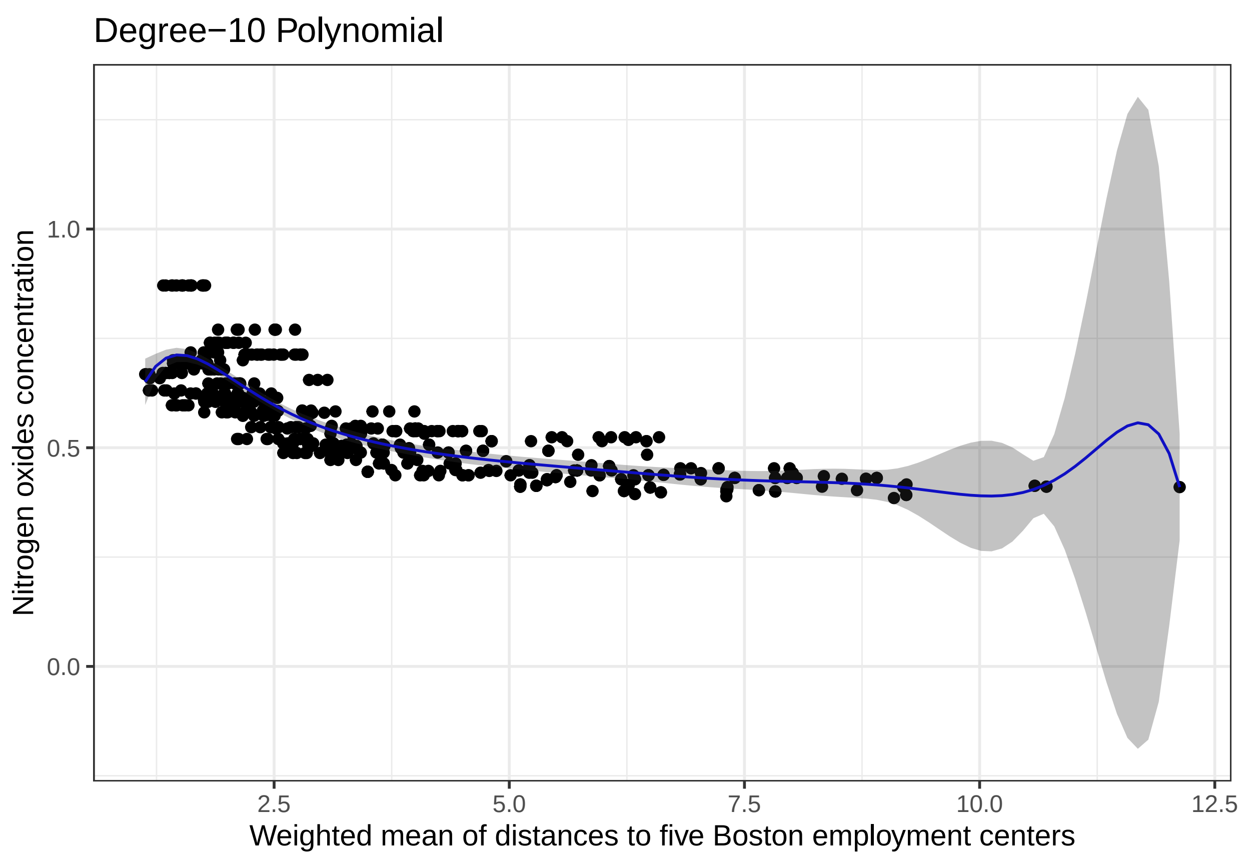
Polynomial Regression and Step Functions on the Boston Housing Data Set

Alexandre H.

```r
library(MASS)
library(tidyverse)
library(patchwork)
library(boot)
library(splines)

set.seed(0)


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

Part 1: Polynomial Regression and Step Functions

```r
fit.p10 <- lm(nox~poly(dis, 10), data=Boston)
preds.p10 <- predict(fit.p10, data.frame(dis=dis.grid), se=TRUE)
se_bands.p10 <- cbind("upper" = preds.p10$fit+2*preds.p10$se.fit,
"lower" = preds.p10$fit-2*preds.p10$se.fit)

plot.p10 <- ggplot() +
geom_point(data=Boston, aes(x = dis, y = nox)) +
geom_line(aes(x = dis.grid, y = preds.p10$fit), color = "#0000FF") +
geom_ribbon(aes(x = dis.grid,
ymin = se_bands.p10[,"lower"],
ymax = se_bands.p10[,"upper"]),
alpha = 0.3) +
xlim(dislims) +
labs(title = "Degree-10 Polynomial") +
theme_bw() +
xlab("Weighted mean of distances to five Boston employment centers") +
ylab("Nitrogen oxides concentration")

plot.p10
```

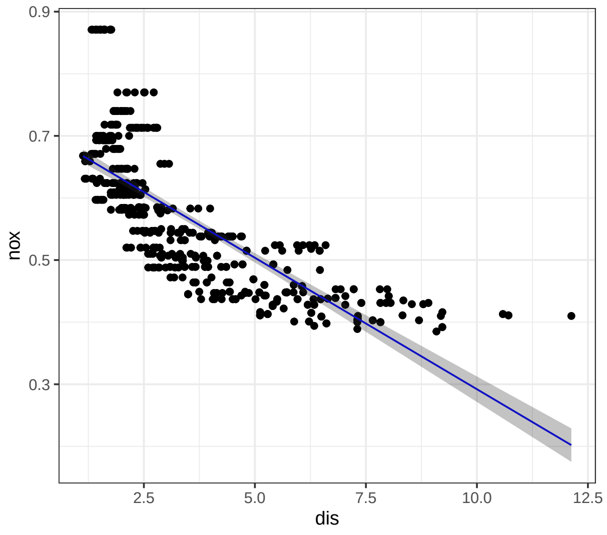


The confidence interval is very narrow where the data is dense, and very wide where the data is sparse. This is because the confidence interval is based on the standard error of the fit, which is based on the variance of the residuals. The variance of the residuals is small where the data is dense, and large where the data is sparse. We can observe from the plot that the bulk of the data is between a value for dis of 0 to approximately 7.5. For values greater than this, the confidence band widens significantly.

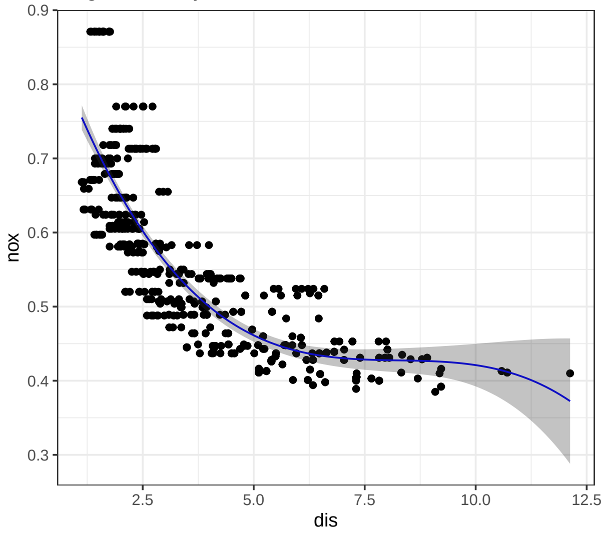
Part 2: Step Functions

```r
polynomial_degrees <- c(1,3,5,7,10)
fit.p <- map(polynomial_degrees, ~lm(nox~poly(dis, .x), data=Boston))

preds.p <- predict(fit.p10, data.frame(dis=dis.grid), se=TRUE)
fit.pred.p <- map(fit.p, ~predict(.x, data.frame(dis=dis.grid), se=TRUE))
fit.rss.p <- map_dbl(fit.p, ~sum(.x$residuals^2))

poly.plots <- map2(fit.pred.p, polynomial_degrees, ~ggplot() +
geom_point(data=Boston, aes(x = dis, y = nox)) +
geom_line(aes(x = dis.grid, y = .x$fit), color = "#0000FF") +
geom_ribbon(aes(x = dis.grid, ymin = .x$fit-2*.x$se.fit, ymax = .x$fit+2*.x$se.fit),
            alpha = 0.3) +
xlim(dislims) +
labs(title = paste0("Degree-", .y, " Polynomial")) +
theme_bw() +
xlab("dis") +
ylab("nox")) %>%
wrap_plots(ncol = 2)

colors <- c("Degree 1" = "#0000FF", "Degree 3" = "#FF0000",
            "Degree 5" = "#00FF00", "Degree 7" = "#000000",
            "Degree 10" = "#FF00FF")
poly.plot.stacked <- ggplot() +
geom_point(data=Boston, aes(x = dis, y = nox)) +
geom_line(aes(x = dis.grid, y = fit.pred.p[[1]]$fit, color = "Degree 1")) +
geom_line(aes(x = dis.grid, y = fit.pred.p[[2]]$fit, color = "Degree 3")) +
geom_line(aes(x = dis.grid, y = fit.pred.p[[3]]$fit, color = "Degree 5")) +
geom_line(aes(x = dis.grid, y = fit.pred.p[[4]]$fit, color = "Degree 7")) +
geom_line(aes(x = dis.grid, y = fit.pred.p[[5]]$fit, color = "Degree 10")) +
xlim(dislims) +
labs(title = "Polynomial Fits", color = "Polynomial Degree") +
theme_bw() +
xlab("dis") +
ylab("nox") +
scale_color_manual(values = colors) +
  theme(legend.position = c(0.9, 0.9), legend.justification = c("right", "top"),
        legend.text = element_text(size = 10),
  legend.key.size = unit(0.2, "cm"))

rss.plot <- ggplot() +
geom_point(data = tibble(degree = polynomial_degrees, rss = fit.rss.p),
aes(x = degree, y = rss)) +
geom_line(data = tibble(degree = polynomial_degrees, rss = fit.rss.p),
aes(x = degree, y = rss)) +
labs(title = "RSS vs. Polynomial Degree") +
theme_bw() +
scale_x_continuous(breaks = polynomial_degrees) +
xlab("Polynomial Degree") +
ylab("Residual Sum of Squares")


poly.plots + poly.plot.stacked
```

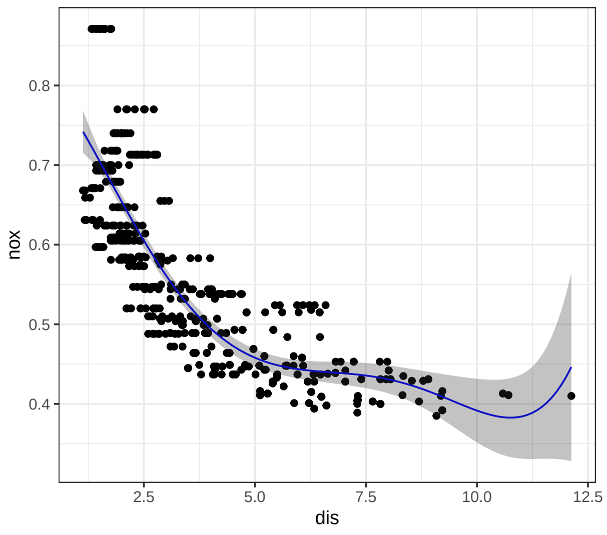
Degree−1 Polynomial



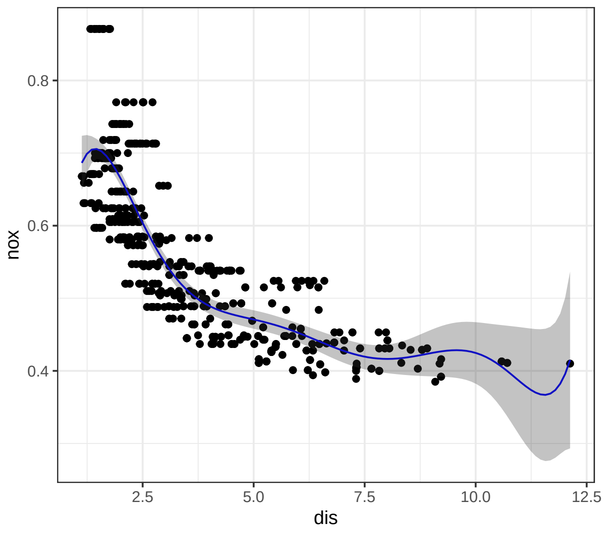
Degree−3 Polynomial



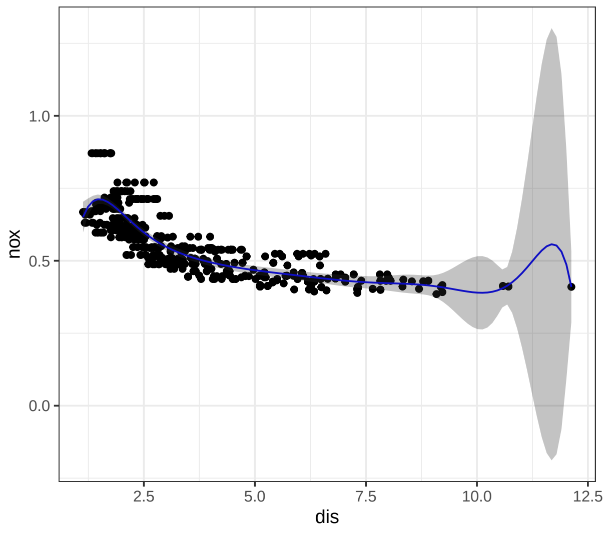
Degree−5 Polynomial



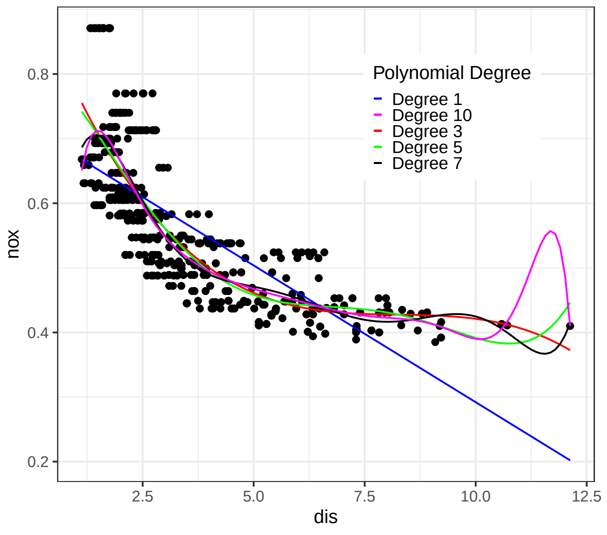
Degree−7 Polynomial



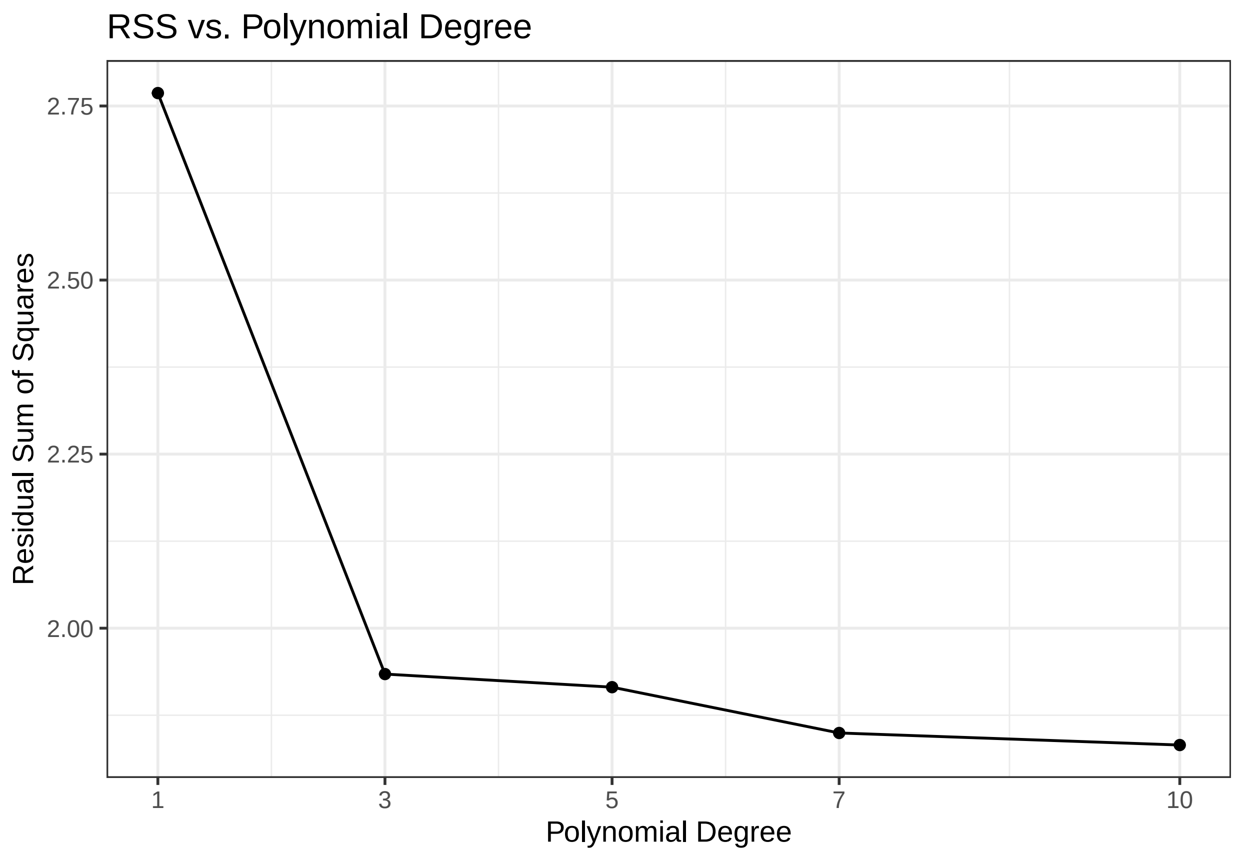
Degree−10 Polynomial



Polynomial Fits



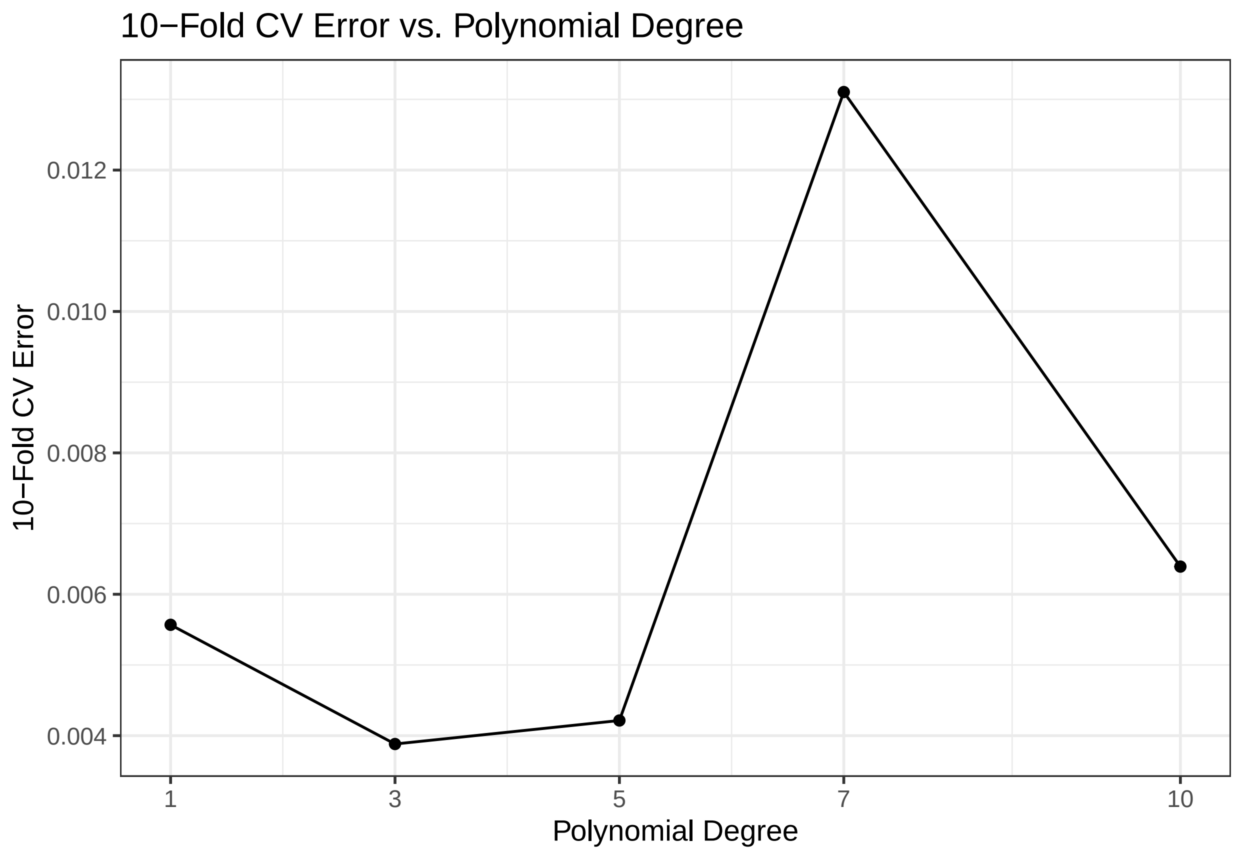
```
rss.plot
```



We observe that the RSS decreases as the polynomial degree increases. We have learned that the RSS always decreases as the number of predictors increases, so this is not surprising. However, we also observe that the RSS decreases more slowly as the polynomial degree increases. This can indicate two things. First, a linear model may not be appropriate. It will likely have high test MSE since the decrease in RSS from degree 1 to 3 is large. Second, a polynomial degree of 10 may be overfitting the data, and will likely have high test RSS since the decrease in RSS from degree 7 to 10 is small.

Part 3: Cross-Validation

```
p.cv.error.10 <- rep(NA, length(polynomial_degrees))
for (i in seq_along(polynomial_degrees)){
glm.fit <- glm(nox~poly(dis, polynomial_degrees[i]),data=Boston)
p.cv.error.10[i] <- cv.glm(Boston,glm.fit,K=10)$delta[1] #does 10-fold CV by setting K=10
}

cv.plot <- ggplot() +
geom_point(data = tibble(degree = polynomial_degrees, cv.error = p.cv.error.10),
aes(x = degree, y = cv.error)) +
geom_line(data = tibble(degree = polynomial_degrees, cv.error = p.cv.error.10),
aes(x = degree, y = cv.error)) +
labs(title = "10-Fold CV Error vs. Polynomial Degree") +
theme_bw() +
scale_x_continuous(breaks = polynomial_degrees) +
xlab("Polynomial Degree") +
ylab("10-Fold CV Error")

cv.plot
```



The polynomial degree with the lowest 10-fold CV error is 3

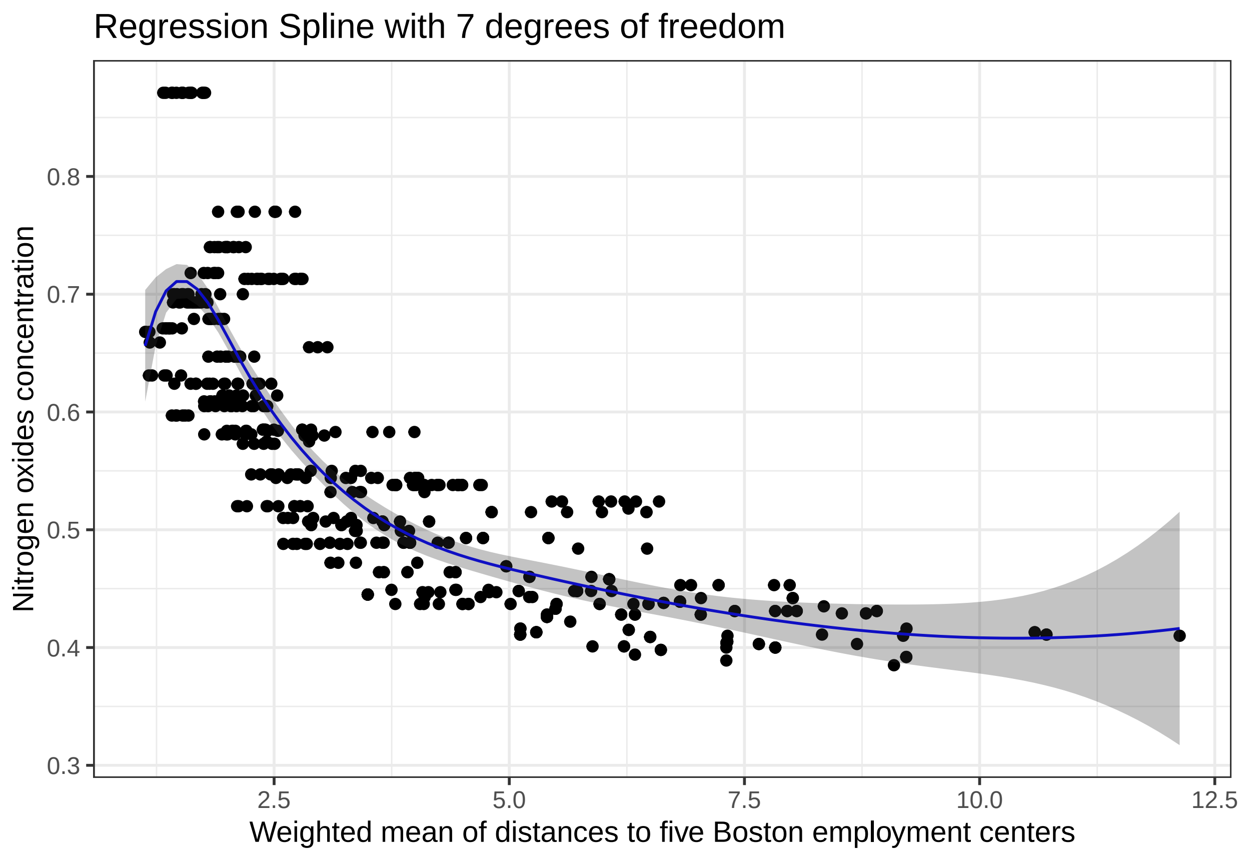
Part 4: Regression Splines

```
fit_spline <- lm(nox ~ bs(dis, df=6), data=Boston)


preds.fit_spline <- predict(fit_spline, data.frame(dis=dis.grid), se=TRUE)
se_bands.fit_spline <- cbind("upper" = preds.fit_spline$fit+2*preds.fit_spline$se.fit,
"lower" = preds.fit_spline$fit-2*preds.fit_spline$se.fit)

plot.fit_spline <- ggplot() +
geom_point(data=Boston, aes(x = dis, y = nox)) +
geom_line(aes(x = dis.grid, y = preds.fit_spline$fit), color = "#0000FF") +
geom_ribbon(aes(x = dis.grid,
ymin = se_bands.fit_spline[,"lower"],
ymax = se_bands.fit_spline[,"upper"]),
alpha = 0.3) +
xlim(dislims) +
labs(title = "Regression Spline with 7 degrees of freedom") +
theme_bw() +
xlab("Weighted mean of distances to five Boston employment centers") +
ylab("Nitrogen oxides concentration")


knots <- attr(bs(Boston$dis, df=6),"knots")
print(knots)
```

```
## [1] 2.100175 3.207450 5.188425
```

We fitted cubic spline with three knots that has seven degrees of freedom (since a cubic spline has K+4 degrees of freedom); these degrees of freedom are used up by an intercept, plus six basis functions. In this case R chooses knots at values 2.100175, 3.20745, and 5.188425 which correspond to the 25th, 50th, and 75th percentiles of dis.

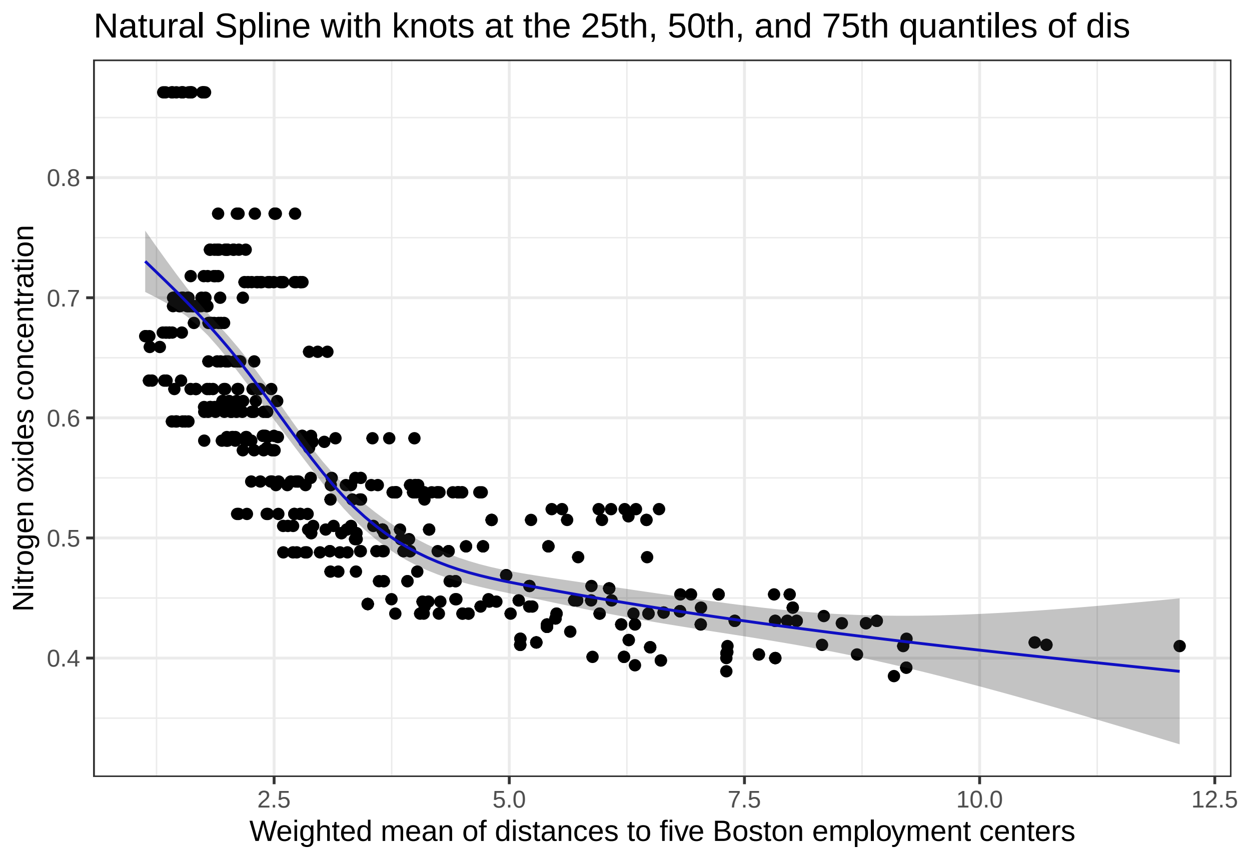
```
plot.fit_spline
```



The confidence band is narrow where the data is dense, which we can observe from the plot. Past the 75th percentile of dis, the confidence band widens significantly, as we see that the data is more sparse in this region. This is because the confidence interval is based on the standard error of the fit, which is based on the variance of the residuals.

Part 5: Smoothing Splines

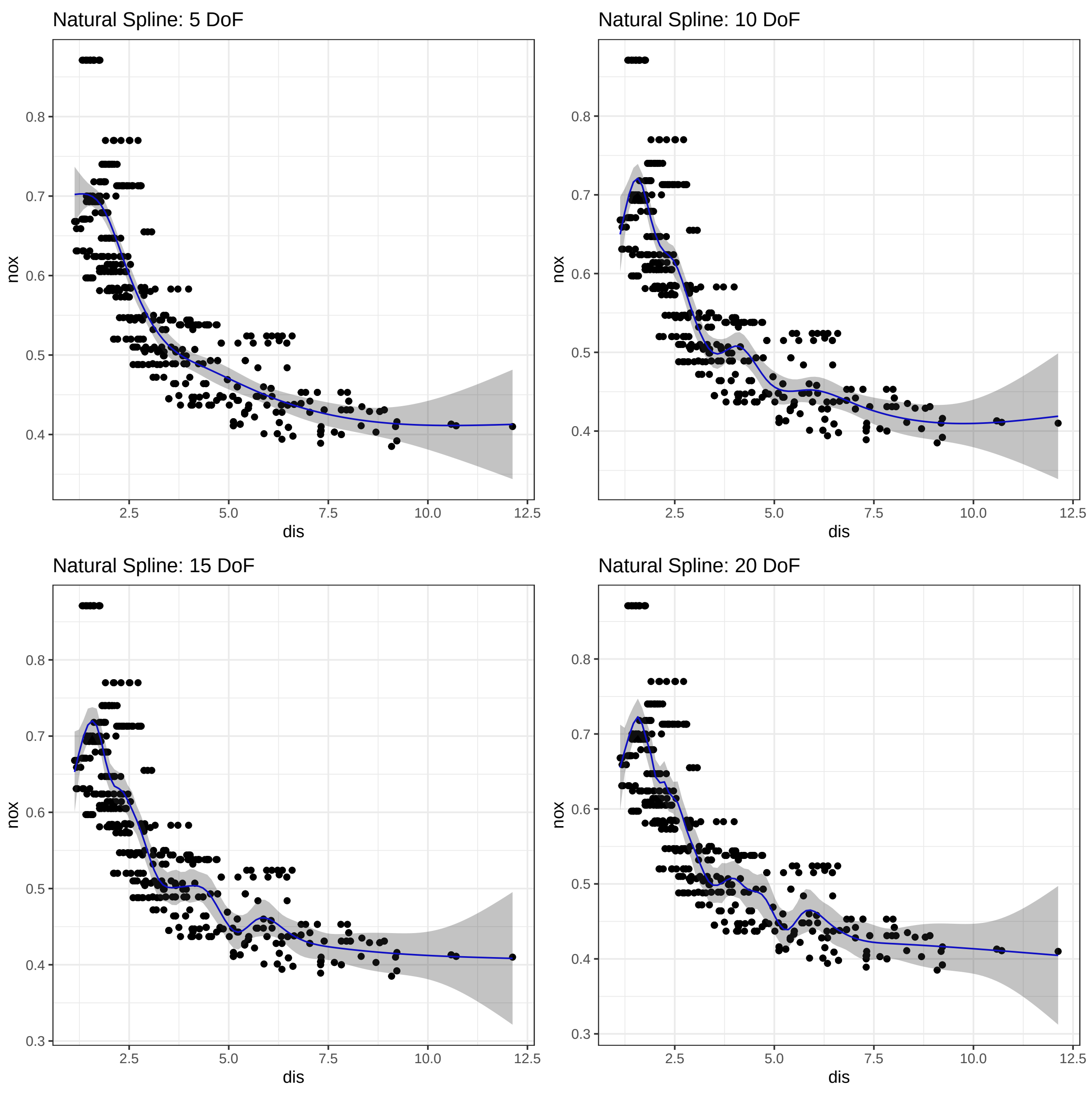
```
fit_natural_spline <- lm(nox ~ ns(dis, knots=knots), data=Boston)

preds.fit_natural_spline <- predict(fit_natural_spline, data.frame(dis=dis.grid), se=TRUE)
se_bands.fit_natural_spline <- cbind("upper" =
                                       preds.fit_natural_spline$fit+2*preds.fit_natural_spline$se.fit,
                                     "lower" =
                                       preds.fit_natural_spline$fit-2*preds.fit_natural_spline$se.fit)


plot.fit_natural_spline <- ggplot() +
geom_point(data=Boston, aes(x = dis, y = nox)) +
geom_line(aes(x = dis.grid, y = preds.fit_natural_spline$fit), color = "#0000FF") +
geom_ribbon(aes(x = dis.grid,
ymin = se_bands.fit_natural_spline[,"lower"],
ymax = se_bands.fit_natural_spline[,"upper"]),
alpha = 0.3) +
xlim(dislims) +
labs(title = "Natural Spline with knots at the 25th, 50th, and 75th quantiles of dis") +
theme_bw() +
xlab("Weighted mean of distances to five Boston employment centers") +
ylab("Nitrogen oxides concentration")

plot.fit_natural_spline
```



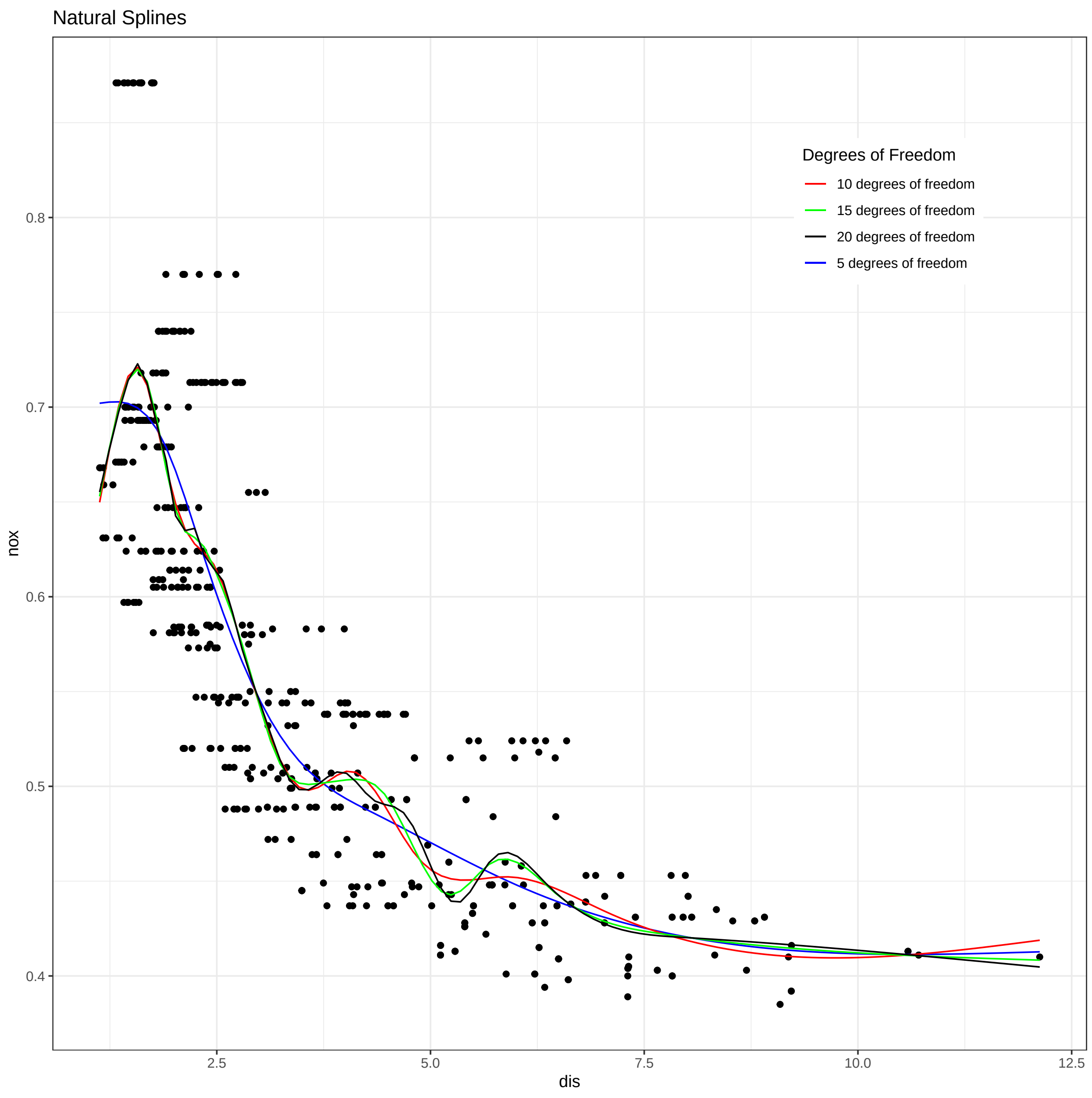
Comparing the natural spline to the previous, we can first the natural spline is smoother. This is because the natural has a constraint of 0 on its second derivative at the boundary knots. This ensures that the natural splines is linear at the extremities of the range of dis. Comparing to the previous plot, we see that the regression spline is not linear at the extremities, and is more wiggly. This is because the regression spline does not have the constraint of 0 on its second derivative at the boundary knots. The confidence band is narrow where the data is dense, which we can observe from the plot. Past the 75th percentile of dis, the confidence band widens significantly, just like the previous plot.

Part 6: Choosing the Number of Knots

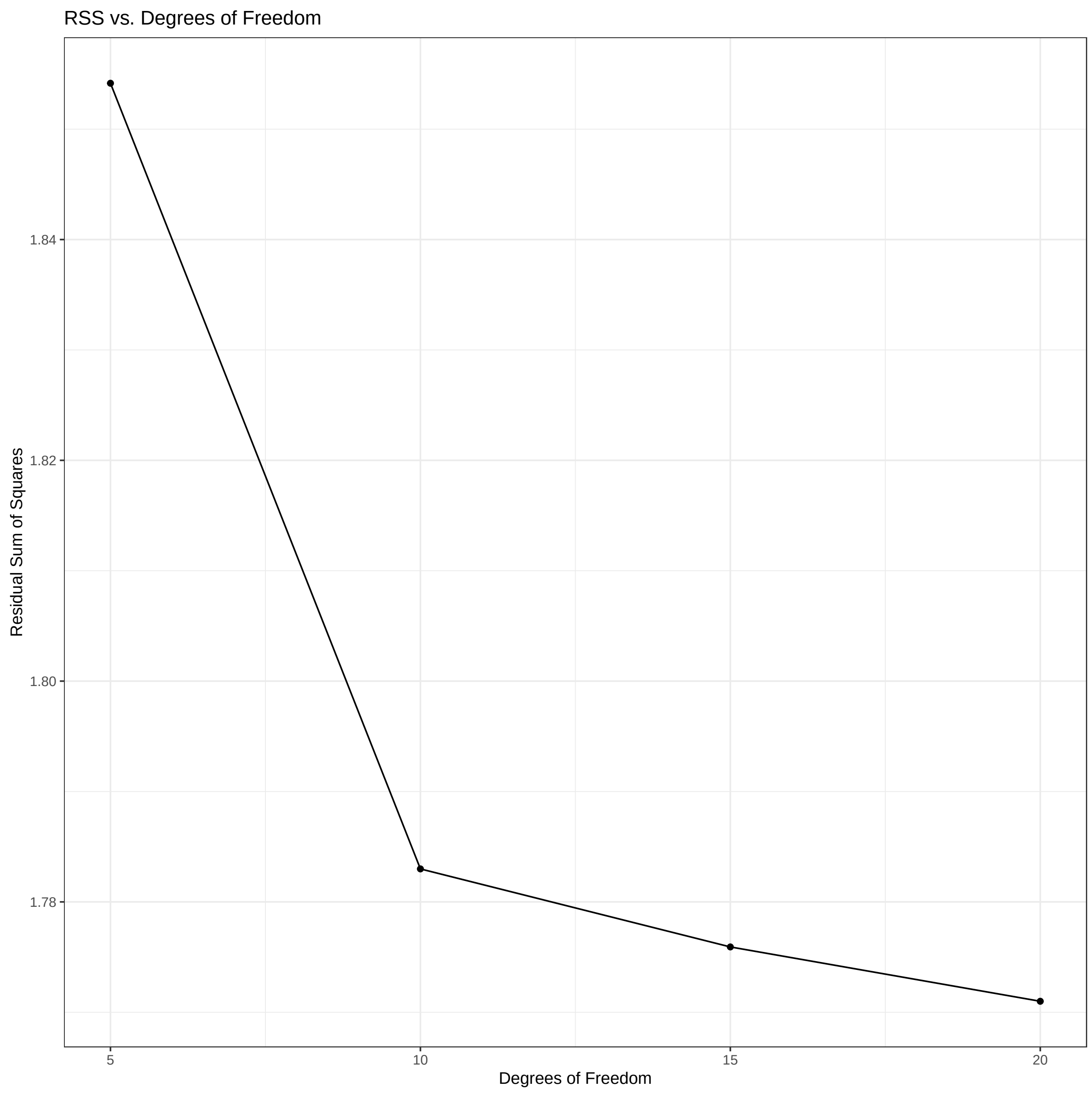
```r
degrees_of_freedom <- c(5, 10, 15, 20)

fit_natural_splines <- map(degrees_of_freedom, ~lm(nox ~ ns(dis, df=.x+1), data=Boston))
preds.fit_natural_splines <- map(fit_natural_splines,
                                 ~predict(.x, data.frame(dis=dis.grid), se=TRUE))
rss.fit_natural_splines <- map_dbl(fit_natural_splines, ~sum(.x$residuals^2))

natural_spline.plots <- map2(preds.fit_natural_splines, degrees_of_freedom, ~ggplot() +
geom_point(data=Boston, aes(x = dis, y = nox)) +
geom_line(aes(x = dis.grid, y = .x$fit), color = "#0000FF") +
geom_ribbon(aes(x = dis.grid, ymin = .x$fit-2*.x$se.fit,
                ymax = .x$fit+2*.x$se.fit), alpha = 0.3) +
xlim(dislims) +
labs(title = paste0("Natural Spline: ", .y, " DoF")) +
theme_bw() +
xlab("dis") +
ylab("nox")) %>%
wrap_plots(ncol = 2)

colors <- c("5 degrees of freedom" = "#0000FF", "10 degrees of freedom" = "#FF0000",
            "15 degrees of freedom" = "#00FF00", "20 degrees of freedom" = "#000000")

natural_spline.plot.stacked <- ggplot() +
geom_point(data=Boston, aes(x = dis, y = nox)) +
geom_line(aes(x = dis.grid,
              y = preds.fit_natural_splines[[1]]$fit, color = "5 degrees of freedom")) +
geom_line(aes(x = dis.grid,
              y = preds.fit_natural_splines[[2]]$fit, color = "10 degrees of freedom")) +
geom_line(aes(x = dis.grid,
              y = preds.fit_natural_splines[[3]]$fit, color = "15 degrees of freedom")) +
geom_line(aes(x = dis.grid,
              y = preds.fit_natural_splines[[4]]$fit, color = "20 degrees of freedom")) +
xlim(dislims) +
labs(title = "Natural Splines", color = "Degrees of Freedom") +
theme_bw() +
xlab("dis") +
ylab("nox") +
scale_color_manual(values = colors) +
theme(legend.position = c(0.9, 0.9), legend.justification = c("right", "top"))

rss.plot <- ggplot() +
geom_point(data = tibble(degree = degrees_of_freedom, rss = rss.fit_natural_splines),
aes(x = degree, y = rss)) +
geom_line(data = tibble(degree = degrees_of_freedom, rss = rss.fit_natural_splines),
aes(x = degree, y = rss)) +
labs(title = "RSS vs. Degrees of Freedom") +
theme_bw() +
scale_x_continuous(breaks = degrees_of_freedom) +
xlab("Degrees of Freedom") +
ylab("Residual Sum of Squares")

natural_spline.plots
```



```
natural_spline.plot.stacked
```



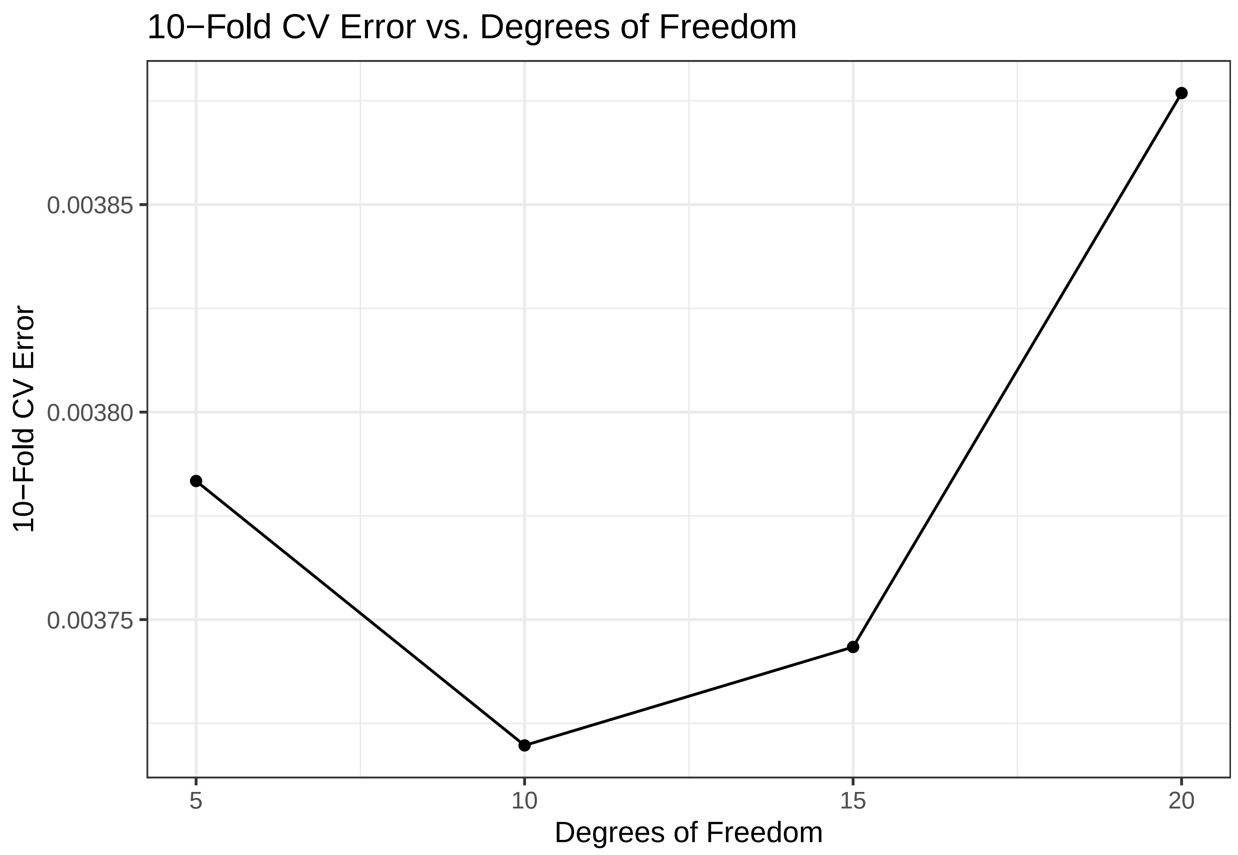
```
rss.plot
```



When comparing the natural splines with increasing degrees of freedom, we observe that the natural spline with 5 is the least wiggly, and the natural spline with 20 is the most wiggly. This is because the natural spline with 5 has the least degrees of freedom, and the natural spline with 20 has the most degrees of freedom. Additionally, in ares where the data is dense, the confidence band looks narrower for the natural spline with 5 degrees of freedom, and wider for the natural spline with 20 degrees of freedom. Also note that a natural cubic spline with k knots has k degrees of freedom. Additionally, k+1 basis functions produces k knots. The RSS decreases as the degrees of freedom increases. We have learned that the RSS always decreases as the number of predictors increases, so this is expected.

Part 7: Choosing the Number of Knots via Cross-Validation

```r
nspl.cv.error.10 <- rep(NA, length(degrees_of_freedom))
for (i in seq_along(degrees_of_freedom)){
glm.fit <- glm(nox~ns(dis, df=degrees_of_freedom[i]+1),data=Boston)
nspl.cv.error.10[i] <- cv.glm(Boston,glm.fit,K=10)$delta[1] #does 10-fold CV by setting K=10
}

cv.plot <- ggplot() +
geom_point(data = tibble(degree = degrees_of_freedom, cv.error = nspl.cv.error.10),
aes(x = degree, y = cv.error)) +
geom_line(data = tibble(degree = degrees_of_freedom, cv.error = nspl.cv.error.10),
aes(x = degree, y = cv.error)) +
labs(title = "10-Fold CV Error vs. Degrees of Freedom") +
theme_bw() +
scale_x_continuous(breaks = degrees_of_freedom) +
xlab("Degrees of Freedom") +
ylab("10-Fold CV Error")

cv.plot
```



The natural spline with the lowest 10-fold CV error is 10