123

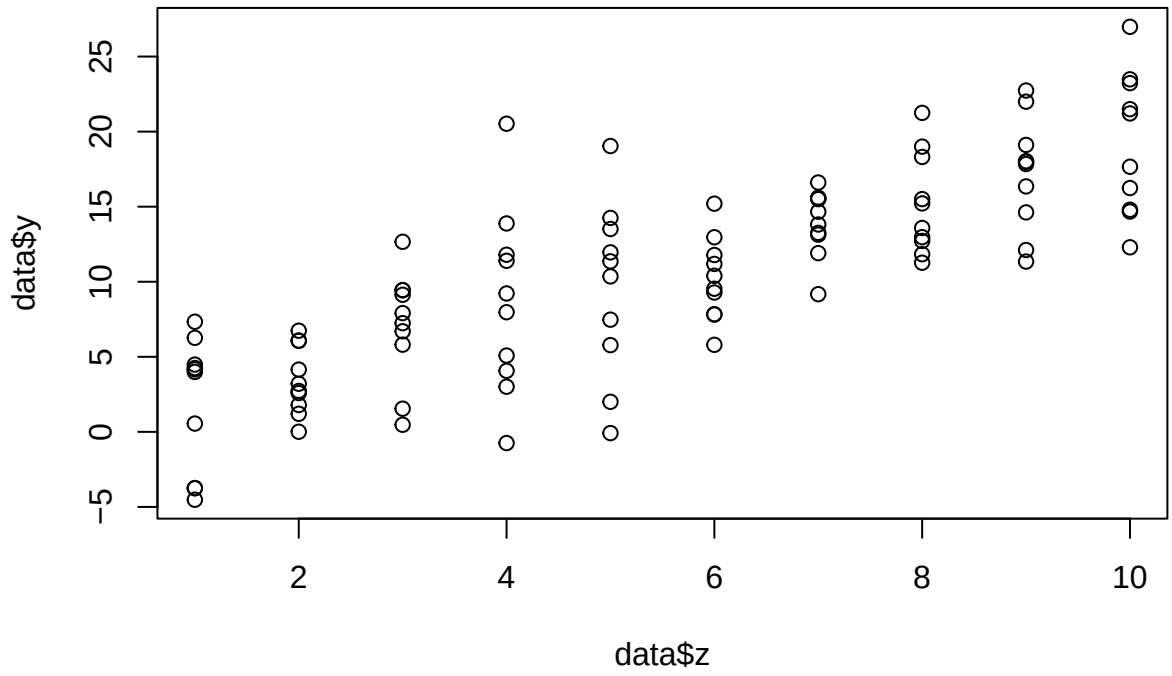
Qinqing Li

2024-08-18

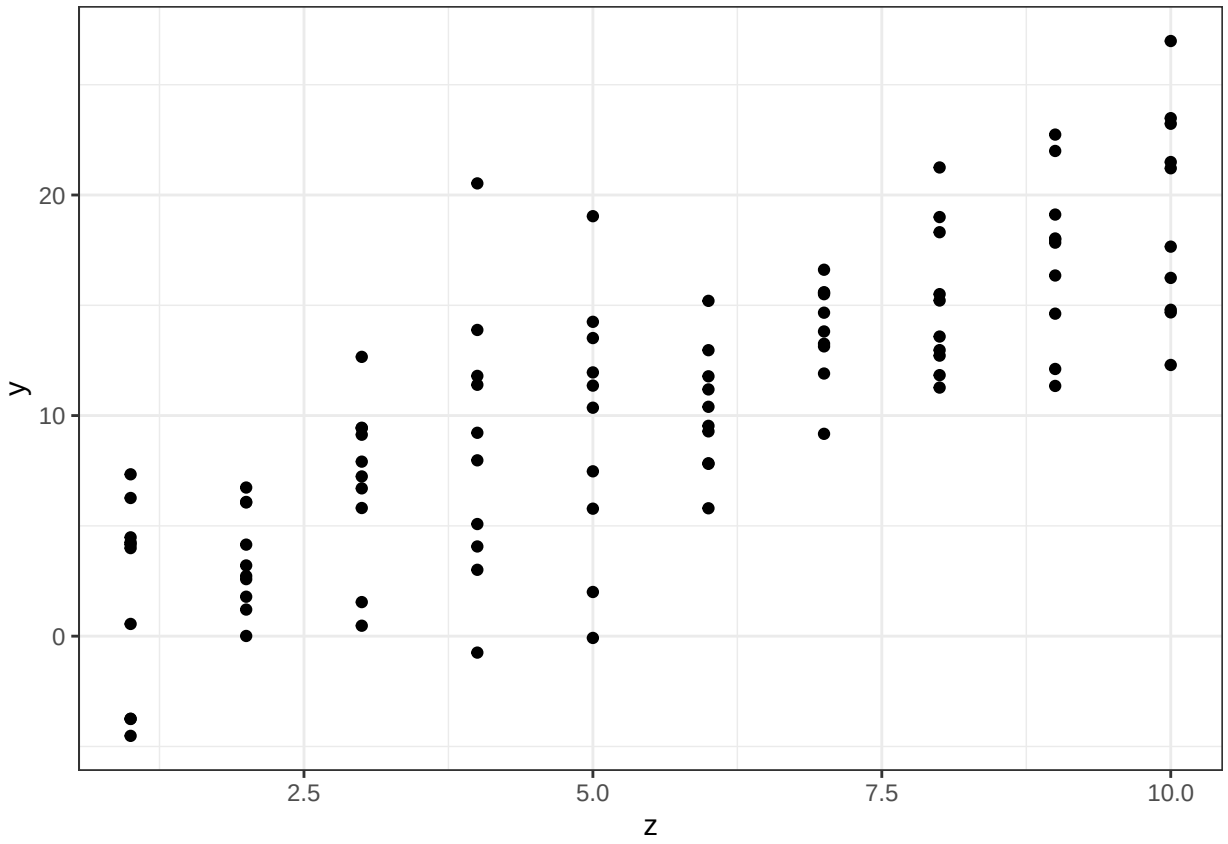
linear model estimation

```
z <- rep(1:10, times = 10)
data <- data.frame(z = z, y = 2 * z + rnorm(length(z), sd = 4))
```

```
plot(data$z, data$y)
```

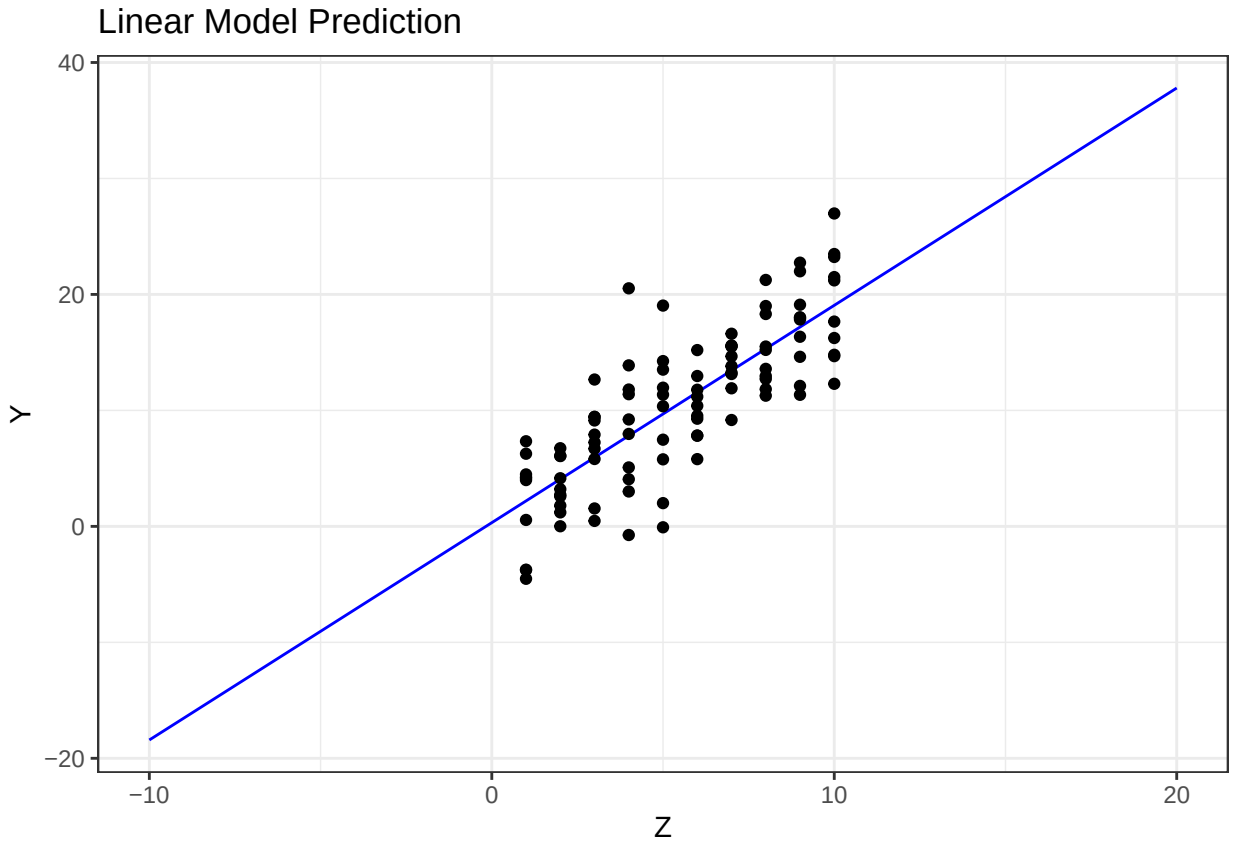



```
ggplot(data) + geom_point(aes(z, y))
```

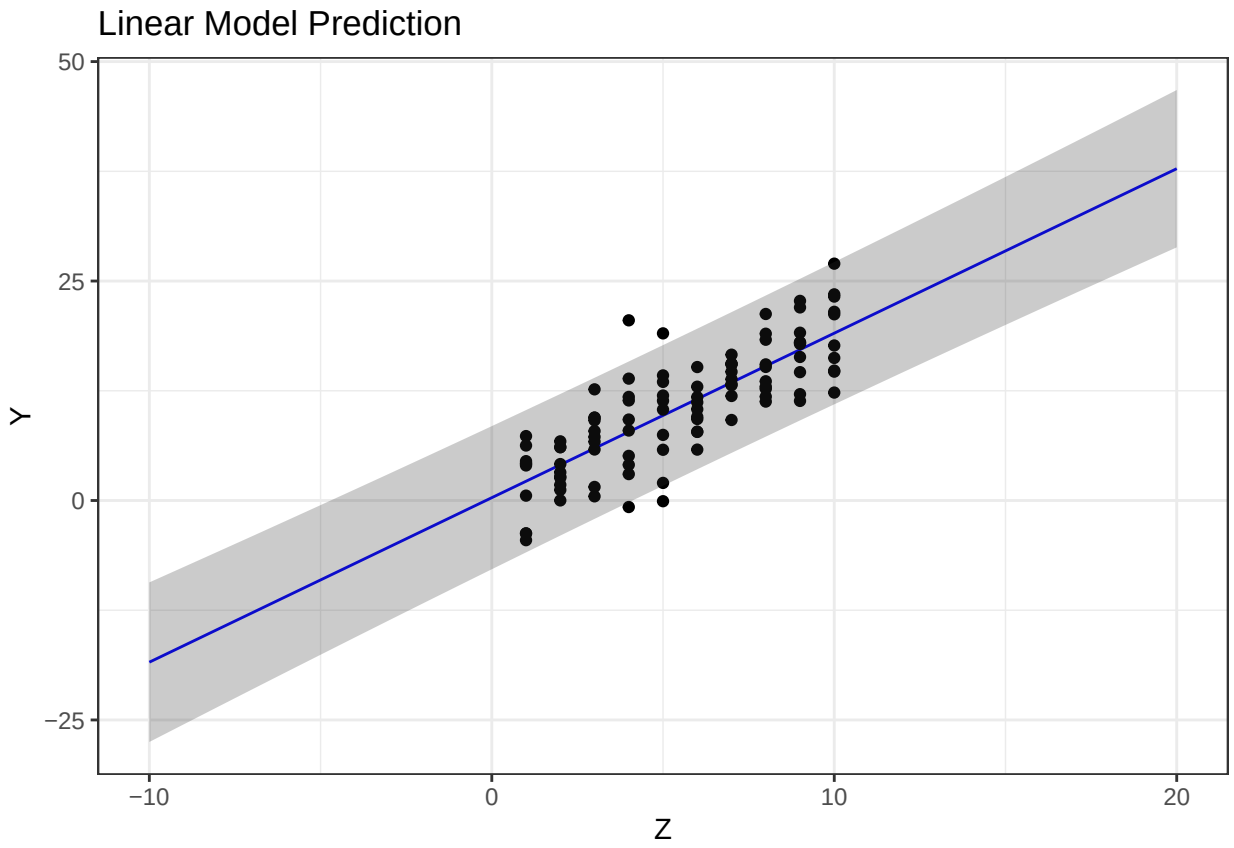


plot linear regression model

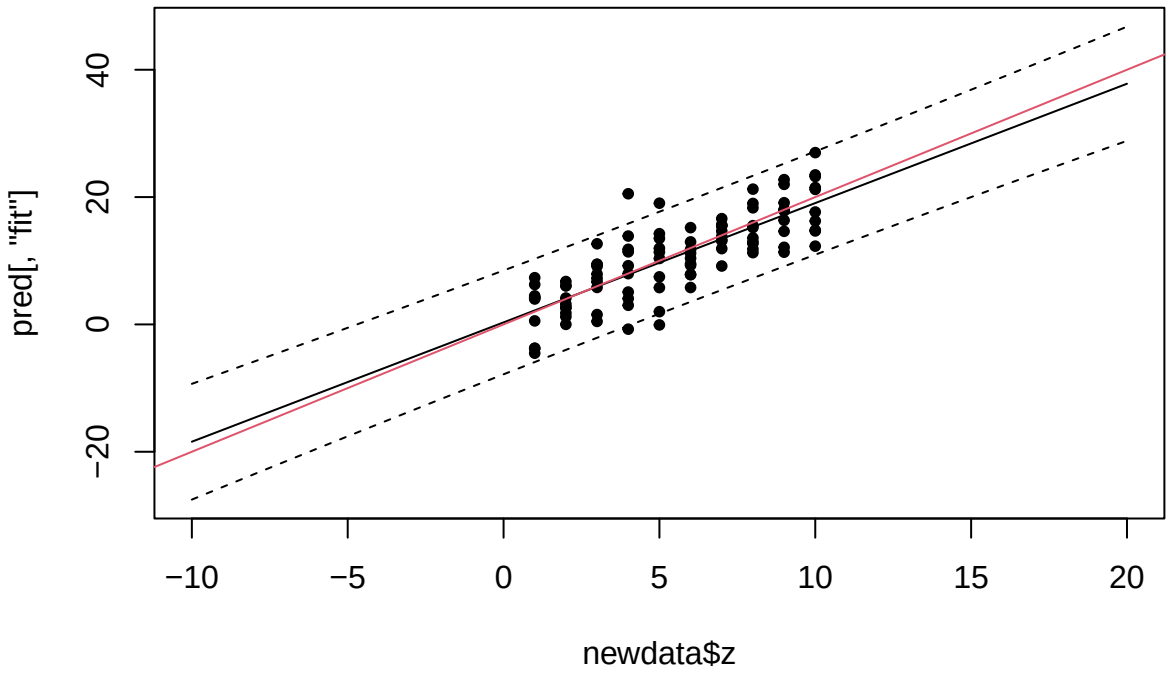
```
mod <- lm(y ~ z, data)
newdata <- data.frame(z = -10:20)
predicted_model <- predict(mod, newdata)

newdata$predicted_y <- predicted_model

# Now use ggplot2 to visualize
ggplot(newdata, aes(z)) +
  geom_line(aes(y = predicted_y), color = "blue") + # Plot the predicted line
  geom_point(data = data, aes(z, y)) +
  labs(title = "Linear Model Prediction",
       x = "Z",
       y = "Y")
```




```r
predicted_model2 <- predict(mod, newdata, interval = "prediction")

# Convert predicted_model2 (matrix) to a data frame and combine it with newdata
pred <- cbind(newdata, predicted_model2)

# Now use ggplot2 to visualize
ggplot(newdata, aes(z)) +
  geom_line(aes(y = predicted_y), color = "blue") +
  geom_point(data = data, aes(z, y)) +
  labs(title = "Linear Model Prediction",
       x = "Z",
       y = "Y")+
  geom_ribbon(data = pred, aes(x = z, ymin = lwr, ymax = upr), alpha = 0.25)
```




```
plot(newdata$z, pred[, "fit"], type = "l", ylim = range(pred))
lines(newdata$z, pred[, "lwr"], lty = 2)
lines(newdata$z, pred[, "upr"], lty = 2)

points(data$z, data$y, pch = 20)
abline(0, 2, col = 2)
```

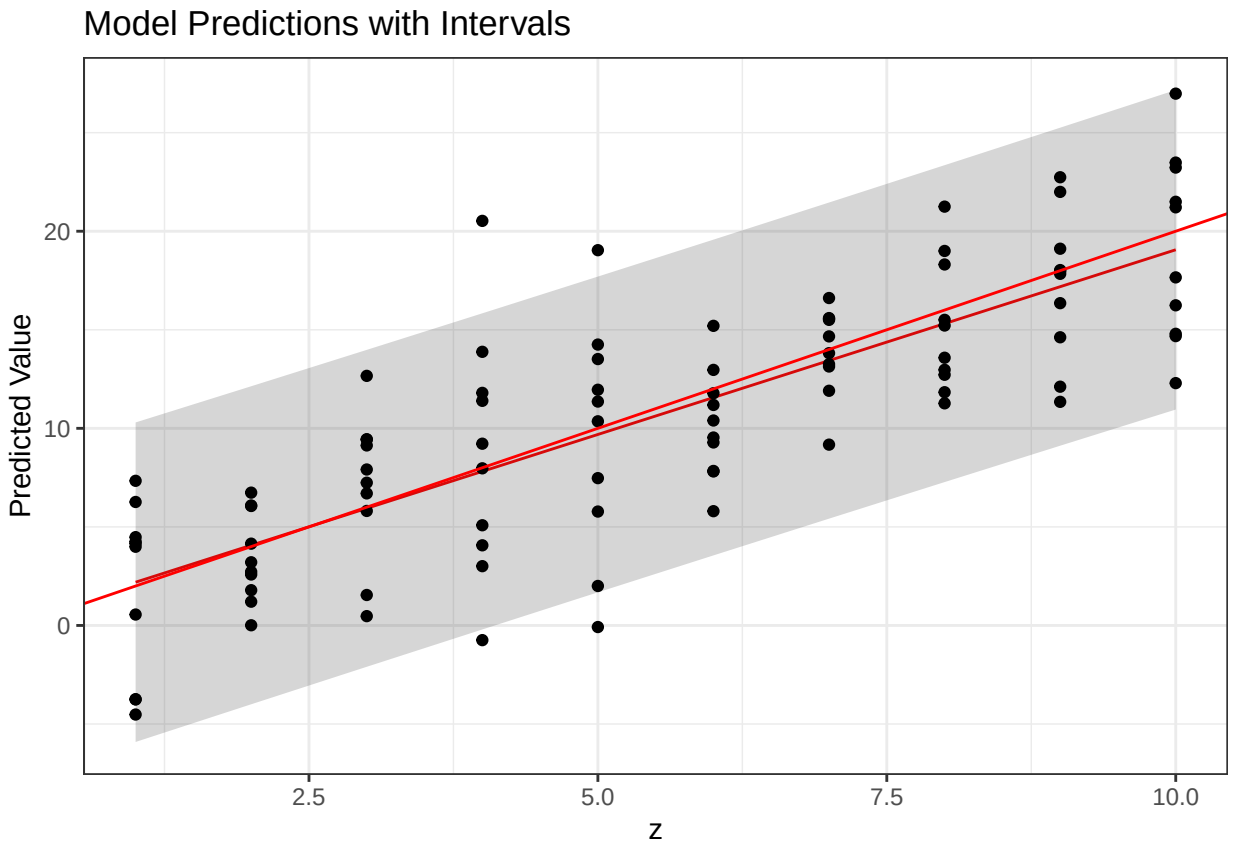


create a similar function to plot the graph:

```
plot_predictions <- function(model, ata, xname) {

  pred <- predict(model, data, interval = "prediction")

  newdata_with_preds <- cbind(data, pred)

  ggplot(newdata_with_preds, aes_string(x = xname)) +
    geom_line(aes(y = fit), color = "red") +                    # Predicted line
    geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.2) +     # Prediction interval
    labs(title = "Model Predictions with Intervals",
         x = xname,
         y = "Predicted Value")
}
```

```
plot_predictions(mod, newdata, xname = "z") +
  geom_point(data = data, aes(z, y)) +
  geom_abline(intercept = 0, slope = 2, col = "red")
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation ideoms with `aes()`
```

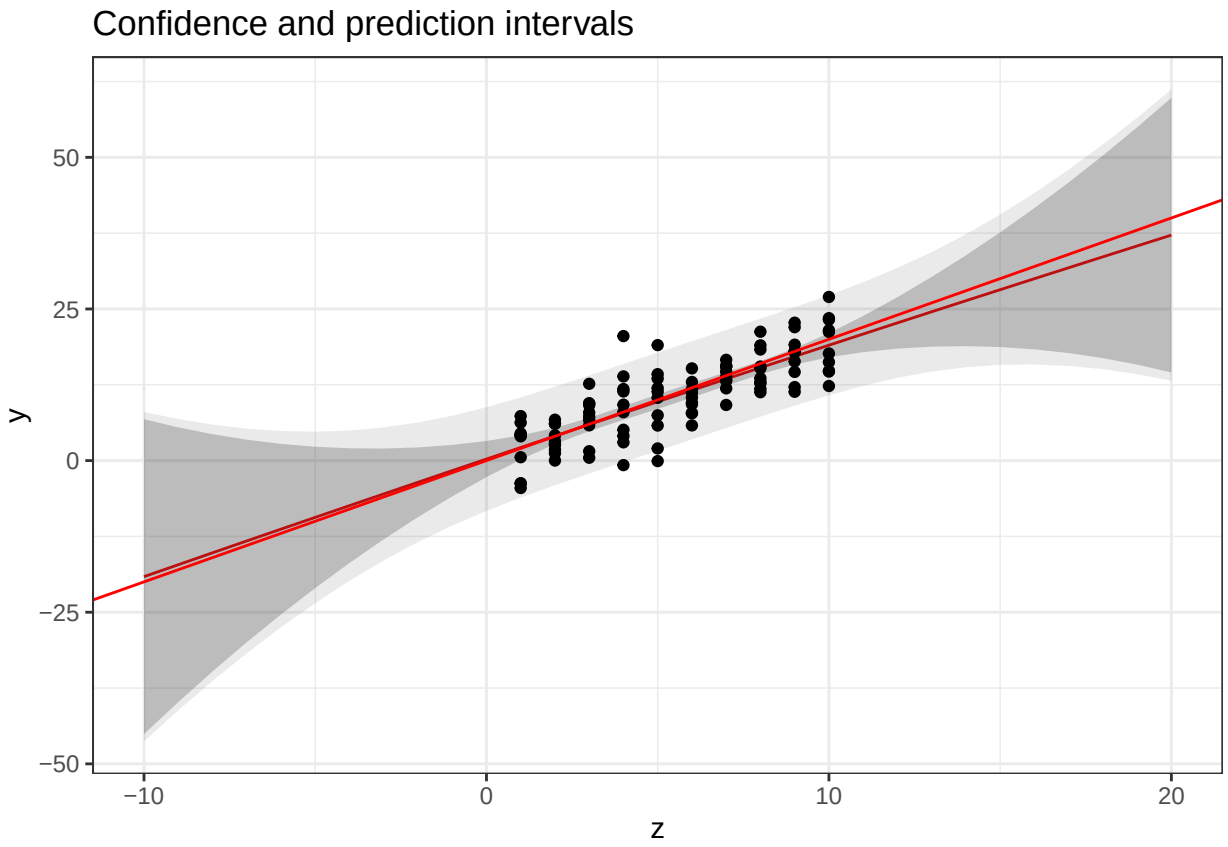


use my new function (with confidence interval) to plot the quadratic model:

```
plot_predictions2 <- function(model, newdata, xname, ylab = "Predicted Value") {
  # Generate predictions with intervals
  pred <- predict(model, newdata, interval = "prediction")

  # Combine predictions with newdata
  newdata_with_preds <- cbind(newdata, pred)

  # Create the plot
  pl <- ggplot(newdata_with_preds, aes_string(x = xname)) +
    geom_line(aes(y = fit), color = "red") +                    # Predicted line
    geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.1) +     # Prediction interval
    labs(title = "Model Predictions with Intervals",
         x = xname,
         y = ylab)
   # Also add the confidence intervals for the predictor curve
  conf <- cbind(newdata,
                predict(model, newdata, interval = "confidence"))
  pl <- pl +
    geom_ribbon(data = conf,
                aes_string(xname, ymin = "lwr", ymax = "upr"), alpha = 0.25)
}
```

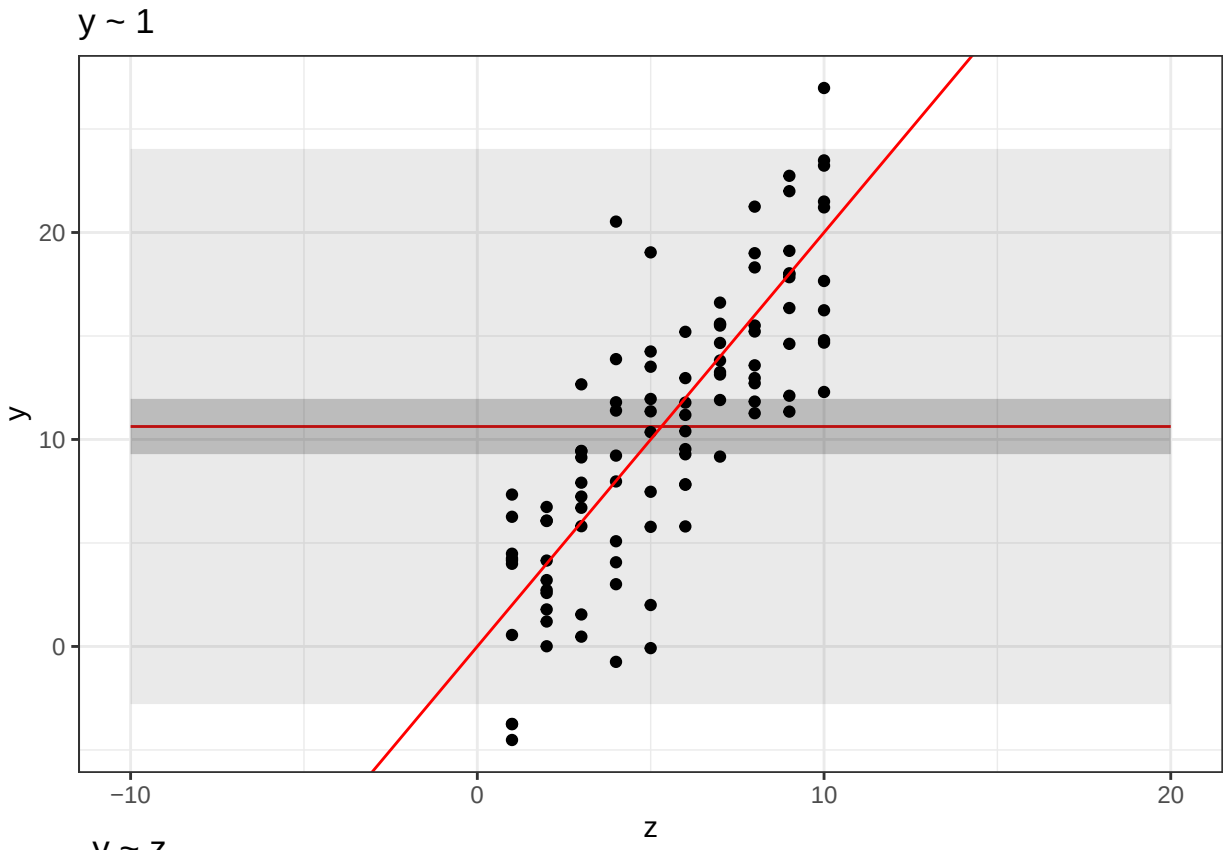
```
mod2 <- lm(y ~ 1 + z + I(z ^ 2), data)

plot_predictions2(mod2, newdata, xname = "z", ylab = "y") +
  geom_point(data = data, aes(z, y)) +
```

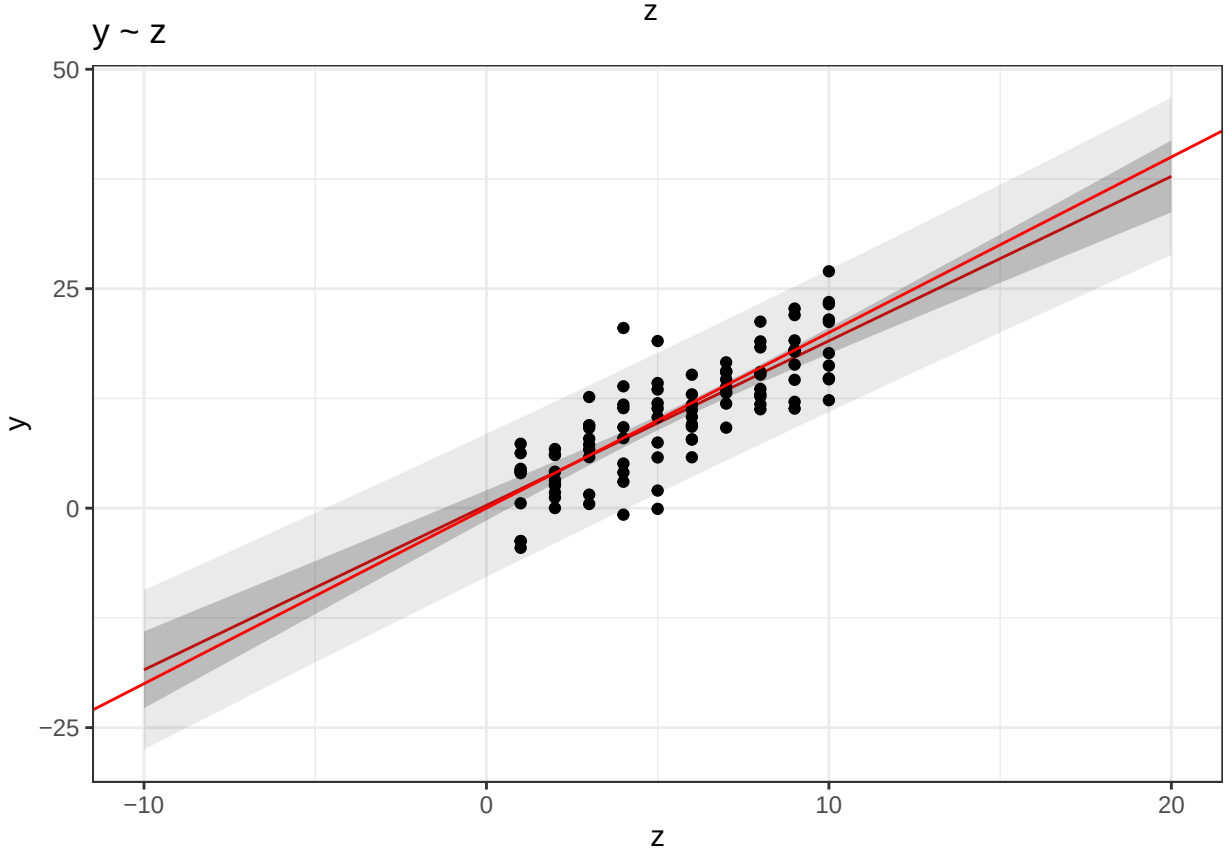
```
  geom_abline(intercept = 0, slope = 2, col = "red") +
  ggtitle("Confidence and prediction intervals")
```

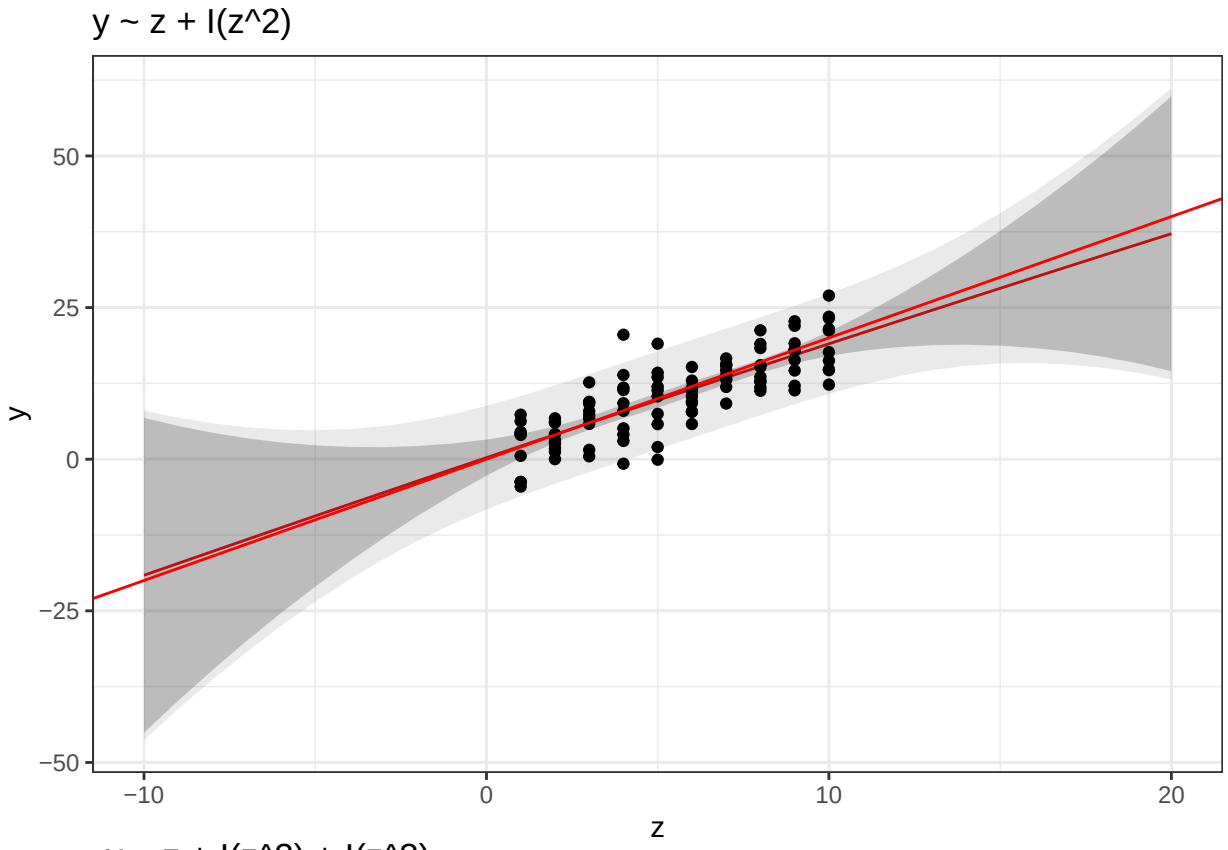


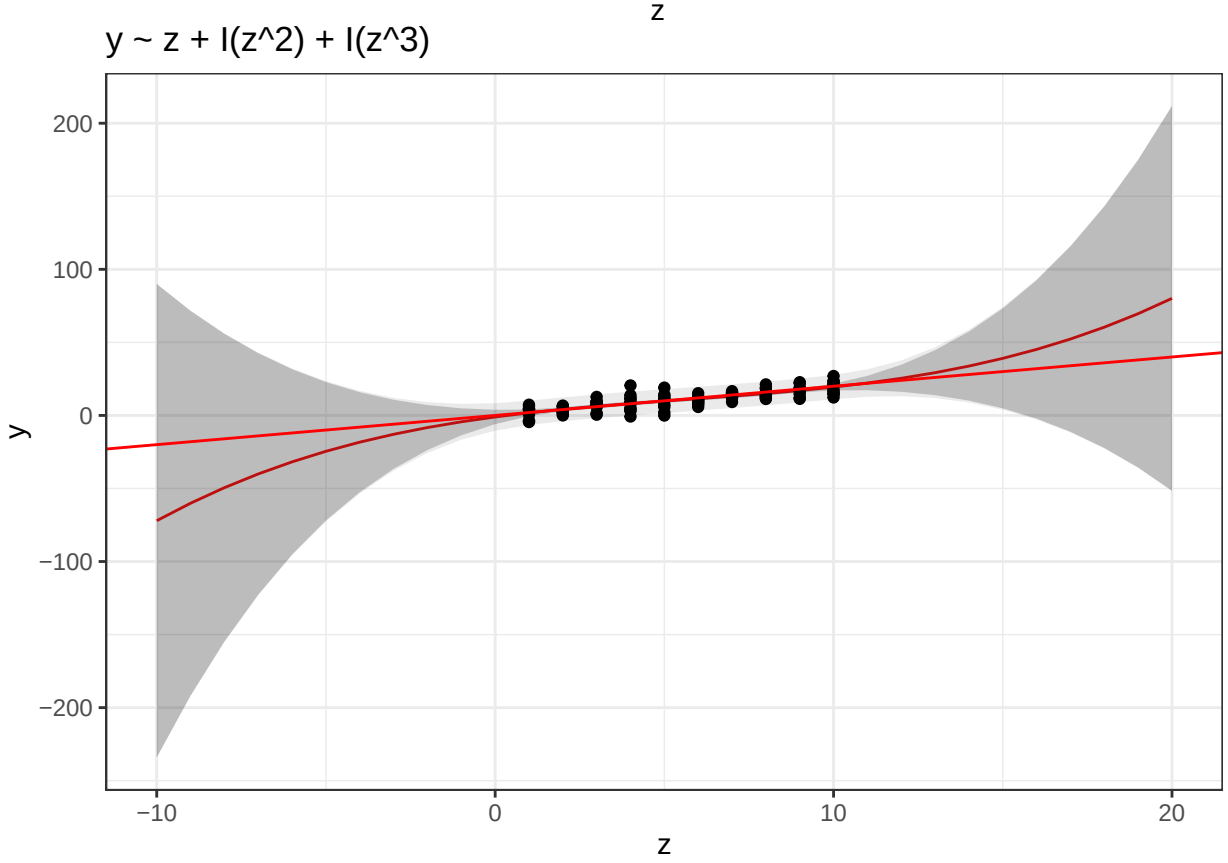

```
formulas <- c(y ~ 1,
              y ~ z,
              y ~ z + I(z^2),
              y ~ z + I(z^2) + I(z^3))
mods <- lapply(formulas, function(x) lm(x, data))
```

```
for (k in seq_along(formulas)) {
  pl <-
    plot_predictions2(mods[[k]], newdata, xname = "z", ylab = "y") +
      geom_point(data = data, aes(z, y)) +
      geom_abline(intercept = 0, slope = 2, col = "red") +
      ggtitle(as.character(formulas[k]))
  print(pl)
}
```


y ~ 1
20
10
0
y
−10
0
10
20
z


y ~ z
50
25
0
−25
y
−10
0
10
20
z





linear fits best with smallest conf interval band and best fit overall.