week13

Lu Zhang

7/28/2019

Model Diagnostics

Model Assumptions

Checking Assumptions

use data simulated from three models

1 Fitted versus Residuals Plot

```
sim_1 = function(sample_size = 500) {
  x = runif(n = sample_size) * 5
  y = 3 + 5 * x + rnorm(n = sample_size, mean = 0, sd = 1)
  data.frame(x, y)
}

sim_2 = function(sample_size = 500) {
  x = runif(n = sample_size) * 5
  y = 3 + 5 * x + rnorm(n = sample_size, mean = 0, sd = x)
  data.frame(x, y)
}

sim_3 = function(sample_size = 500) {
  x = runif(n = sample_size) * 5
  y = 3 + 5 * x ^ 2 + rnorm(n = sample_size, mean = 0, sd = 5)
  data.frame(x, y)
}
```

Fitted versus Residuals Plot

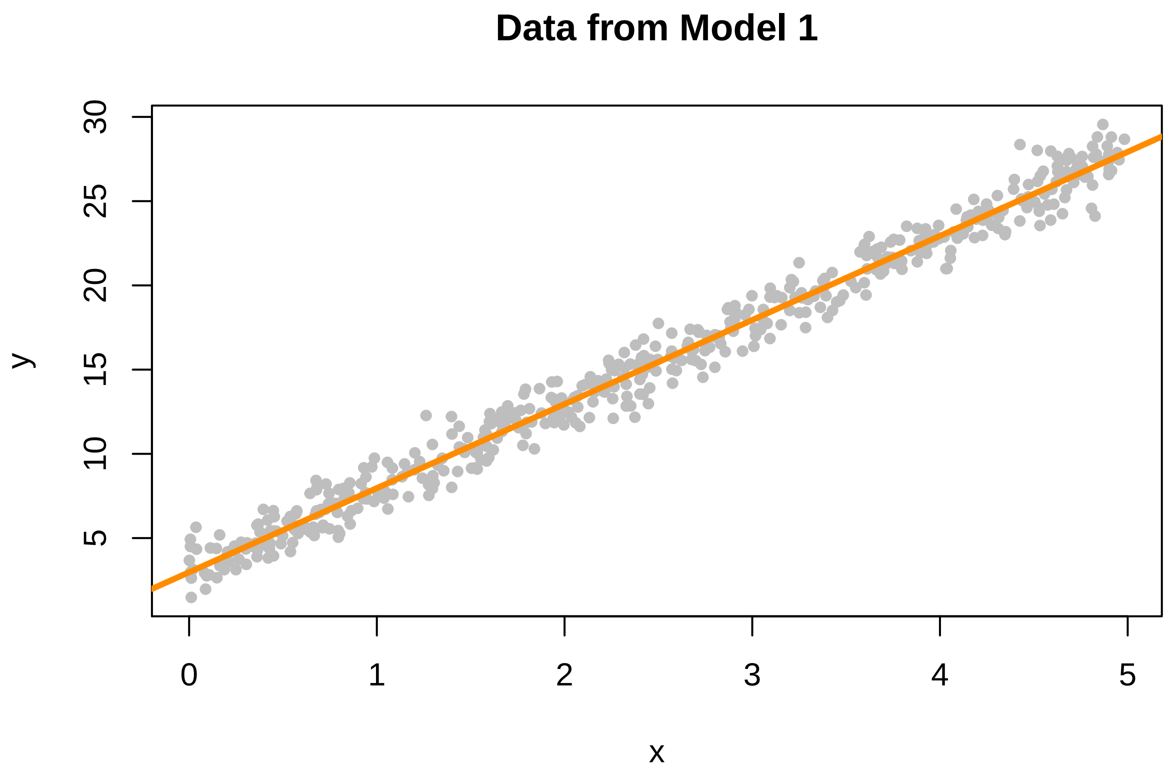
useful for checking both the linearity and constant variance assumptions

```
set.seed(42)
sim_data_1 = sim_1()
head(sim_data_1)
```

```
##          x         y
## 1 4.574030 24.773995
## 2 4.685377 26.475936
## 3 1.430698  8.954993
## 4 4.152238 23.951210
## 5 3.208728 20.341344
## 6 2.595480 14.943525
```

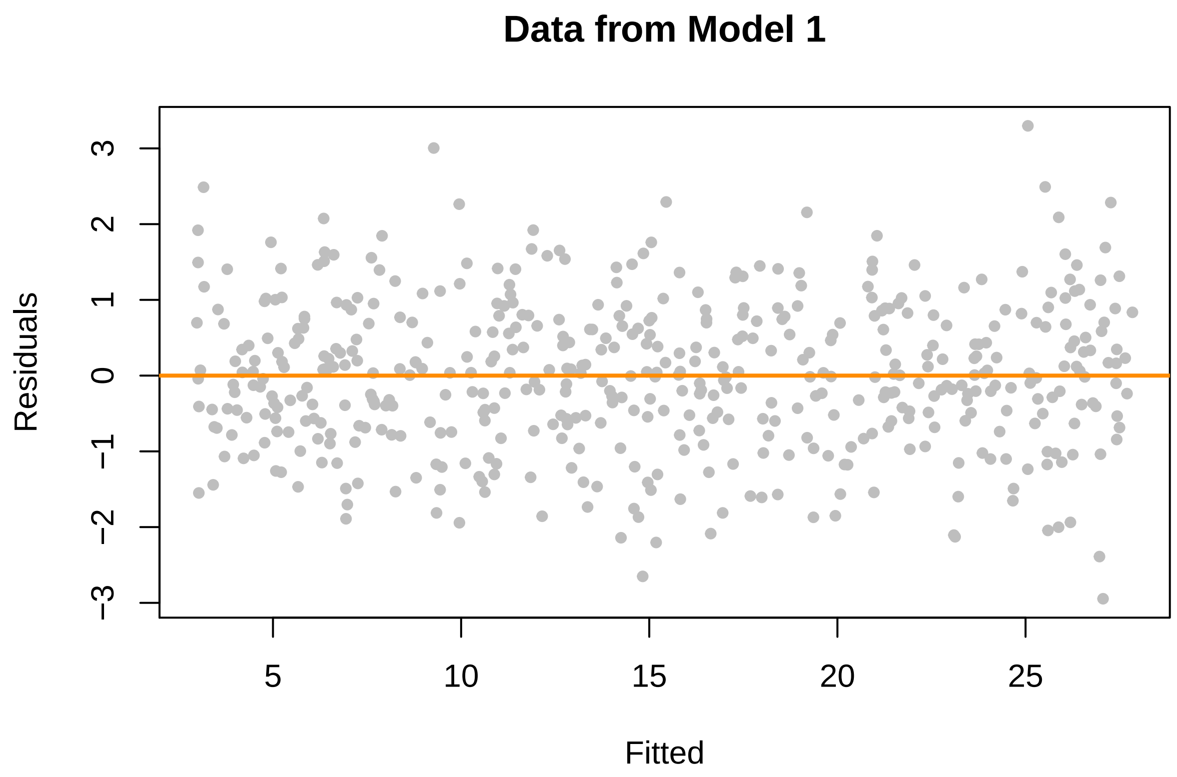
fit the model and add the fitted line to a scatterplot

```r
plot(y ~ x,
     data = sim_data_1,
     col = "grey",
     pch = 20,
     main = "Data from Model 1")
fit_1 = lm(y ~ x,
           data = sim_data_1)
abline(fit_1,
       col = "darkorange",
       lwd = 3)
```



fitted versus residuals plot

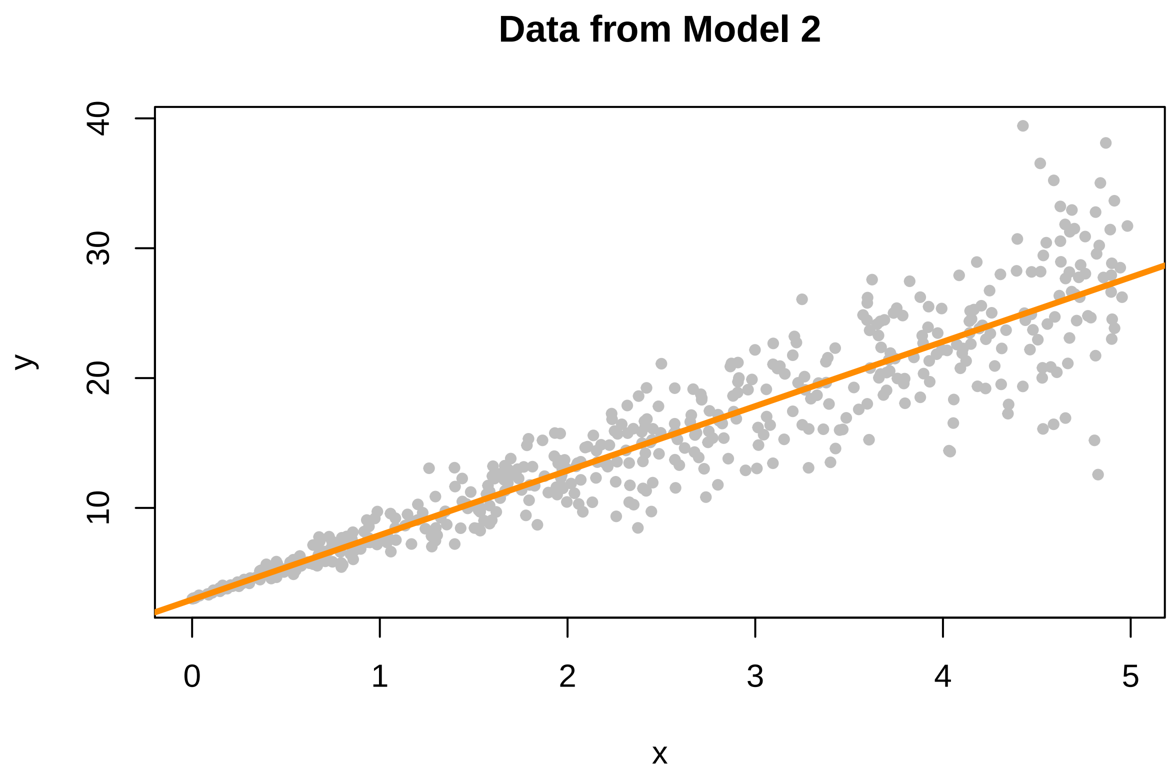
```r
plot(fitted(fit_1),
     resid(fit_1),
     col = "grey",
     pch = 20,
     xlab = "Fitted",
     ylab = "Residuals",
     main = "Data from Model 1")
abline(h = 0,
       col = "darkorange",
       lwd = 2)
```



Model 2 non-constant variance

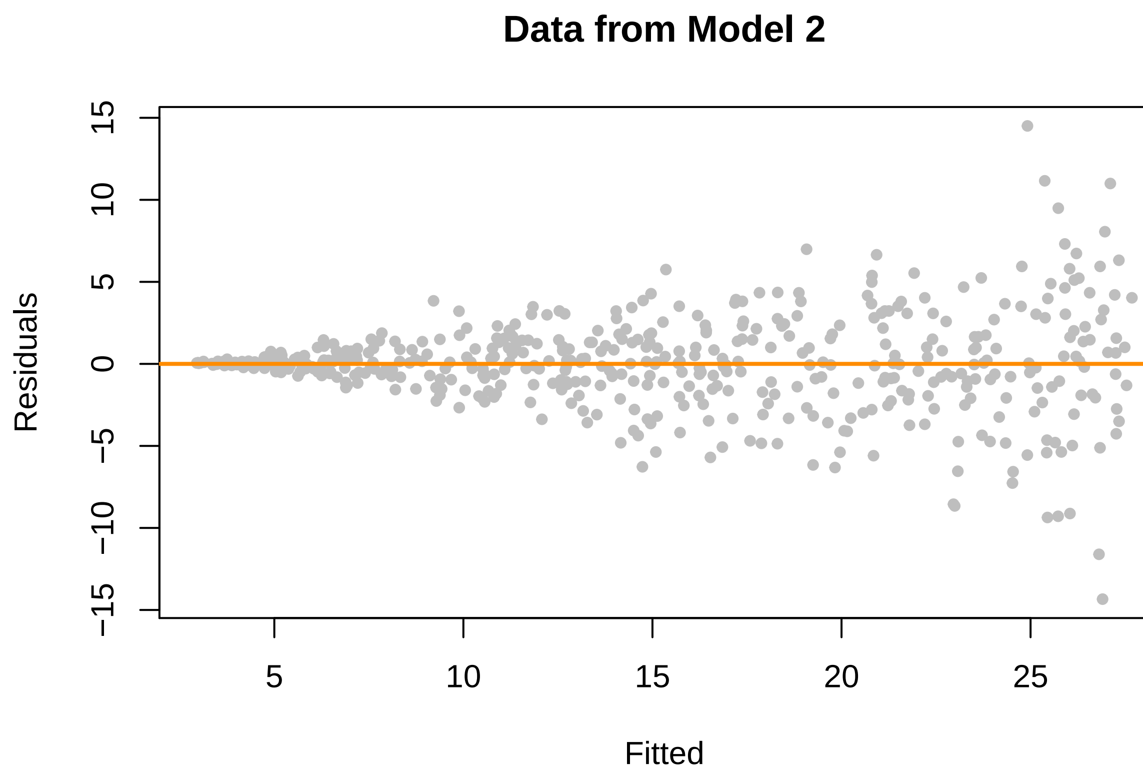
(the variance is larger for larger values of the predictor variable x)

```r
set.seed(42)
sim_data_2 = sim_2()
fit_2 = lm(y ~ x, data = sim_data_2)
plot(y ~ x, data = sim_data_2, col = "grey", pch = 20,
     main = "Data from Model 2")
abline(fit_2, col = "darkorange", lwd = 3)
```



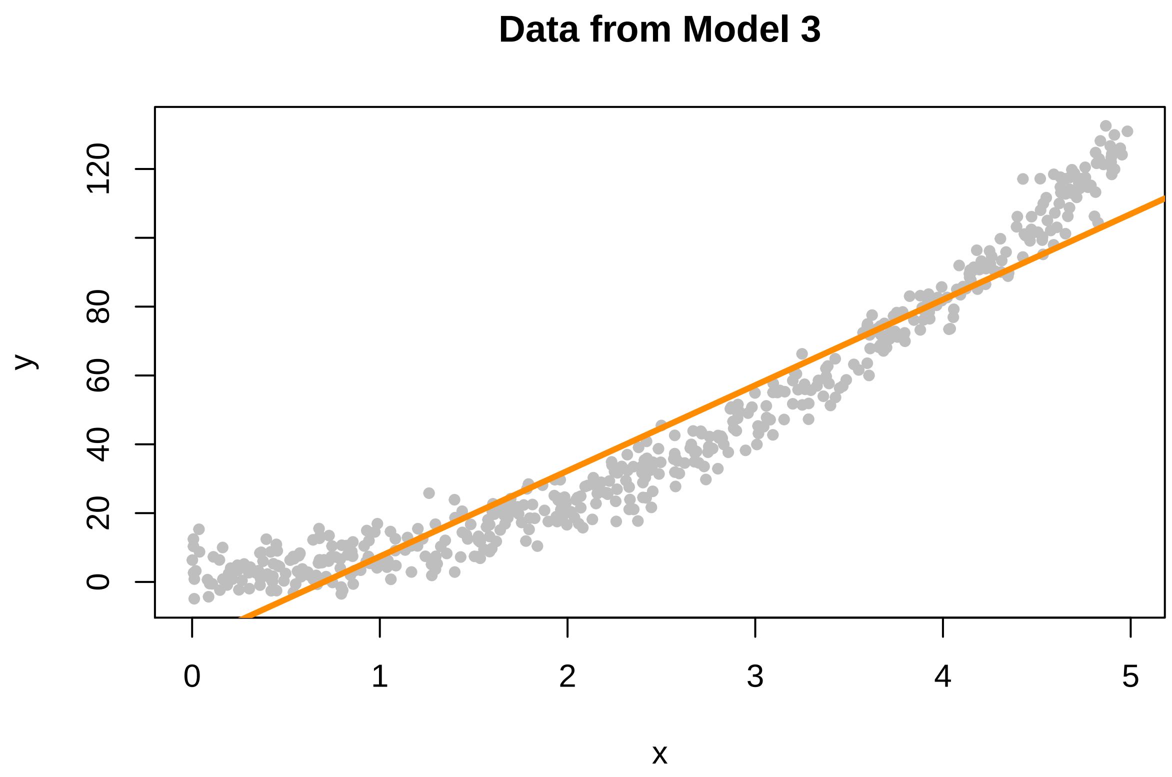
fitted versus residuals plot

```r
plot(fitted(fit_2),
     resid(fit_2),
     col = "grey",
     pch = 20,
     xlab = "Fitted",
     ylab = "Residuals",
     main = "Data from Model 2")
abline(h = 0,
       col = "darkorange",
       lwd = 2)
```

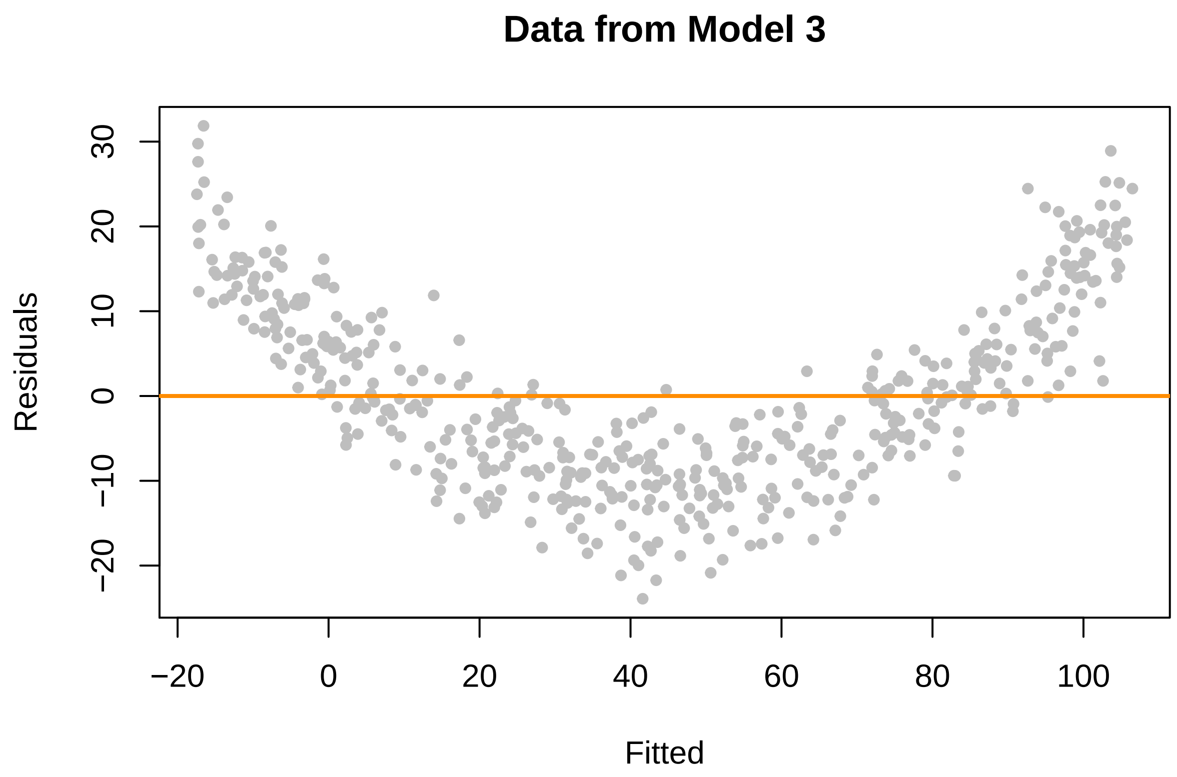


Model 3 Y is not a linear combination of the predictors. Y ~ X^2

```
set.seed(42)
sim_data_3 = sim_3()
fit_3 = lm(y ~ x,
           data = sim_data_3)
plot(y ~ x,
     data = sim_data_3,
     col = "grey",
     pch = 20,
     main = "Data from Model 3")
abline(fit_3,
       col = "darkorange",
       lwd = 3)
```



```
plot(fitted(fit_3),
     resid(fit_3),
     col = "grey",
     pch = 20,
     xlab = "Fitted",
     ylab = "Residuals",
     main = "Data from Model 3")
abline(h = 0,
       col = "darkorange",
       lwd = 2)
```



2 Breusch-Pagan Test

```
#install.packages("lmtest")
library(lmtest)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
bptest(fit_1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  fit_1
## BP = 1.0234, df = 1, p-value = 0.3117
```

```r
bptest(fit_2)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  fit_2
## BP = 76.693, df = 1, p-value < 2.2e-16
```

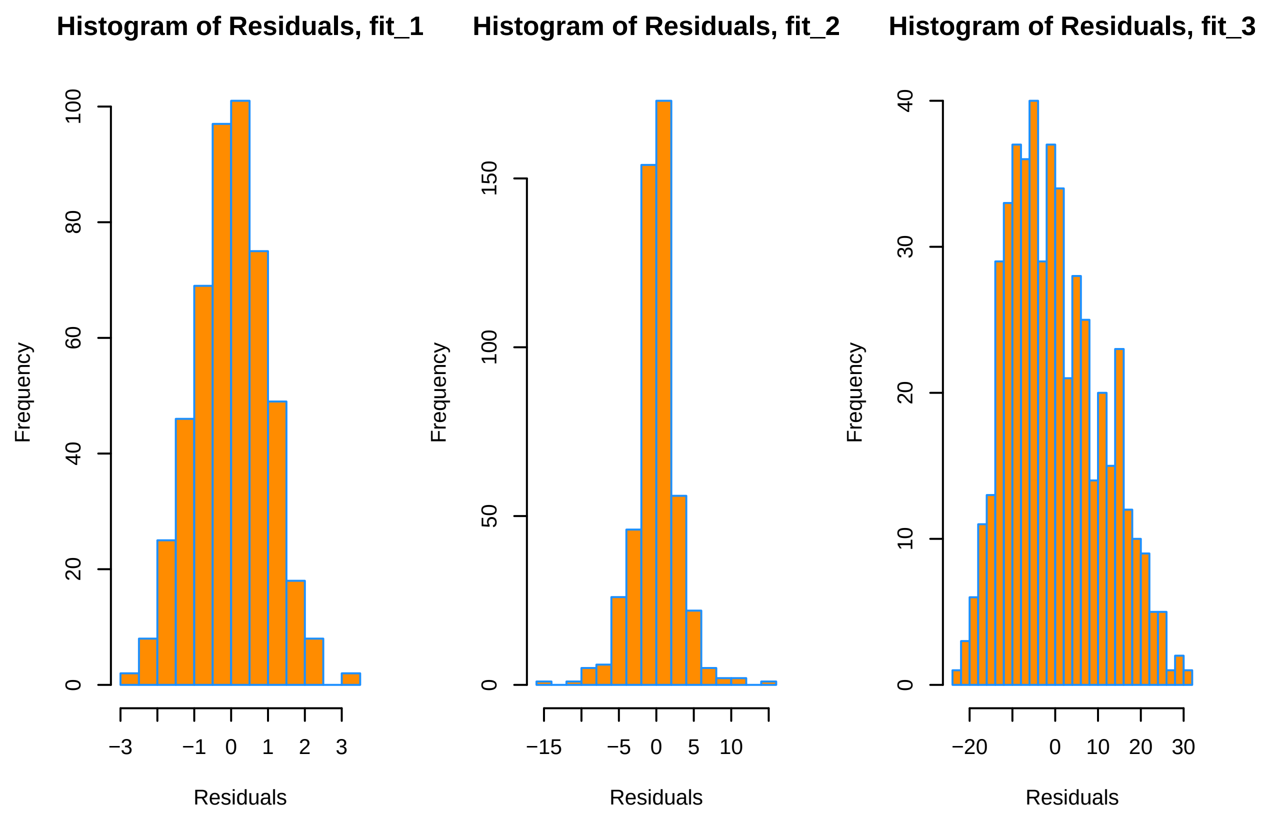
```r
bptest(fit_3)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  fit_3
## BP = 0.33466, df = 1, p-value = 0.5629
```

3 Histograms

make a histogram of the residuals

```r
par(mfrow = c(1, 3))

hist(resid(fit_1),
     xlab   = "Residuals",
     main   = "Histogram of Residuals, fit_1",
     col    = "darkorange",
     border = "dodgerblue",
     breaks = 20)

hist(resid(fit_2),
     xlab   = "Residuals",
     main   = "Histogram of Residuals, fit_2",
     col    = "darkorange",
     border = "dodgerblue",
     breaks = 20)

hist(resid(fit_3),
     xlab   = "Residuals",
     main   = "Histogram of Residuals, fit_3",
     col    = "darkorange",
     border = "dodgerblue",
     breaks = 20)
```

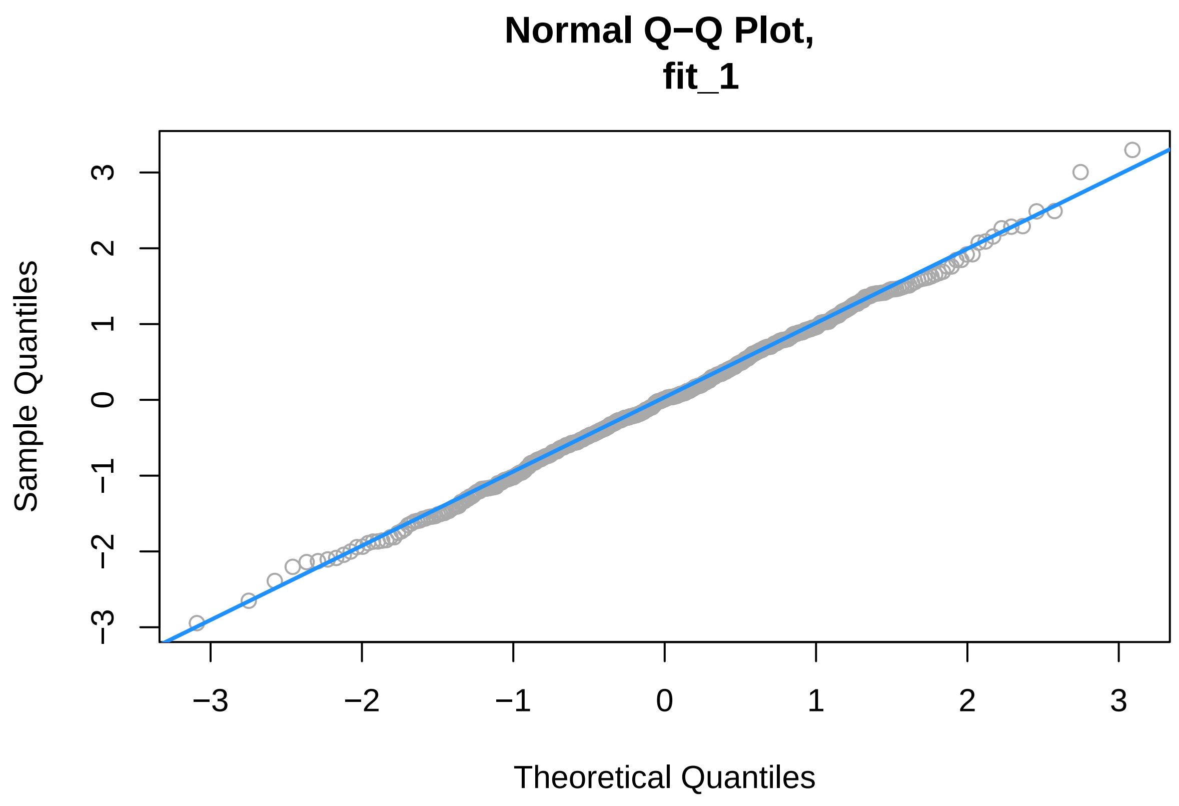


4 Q-Q Plots

quantile-quantile plot

qqnorm() function plots the points qqline() function adds the necessary line

```r
qqnorm(resid(fit_1),
       main = "Normal Q-Q Plot,
       fit_1",
       col = "darkgrey")

qqline(resid(fit_1),
       col = "dodgerblue",
       lwd = 2)
```

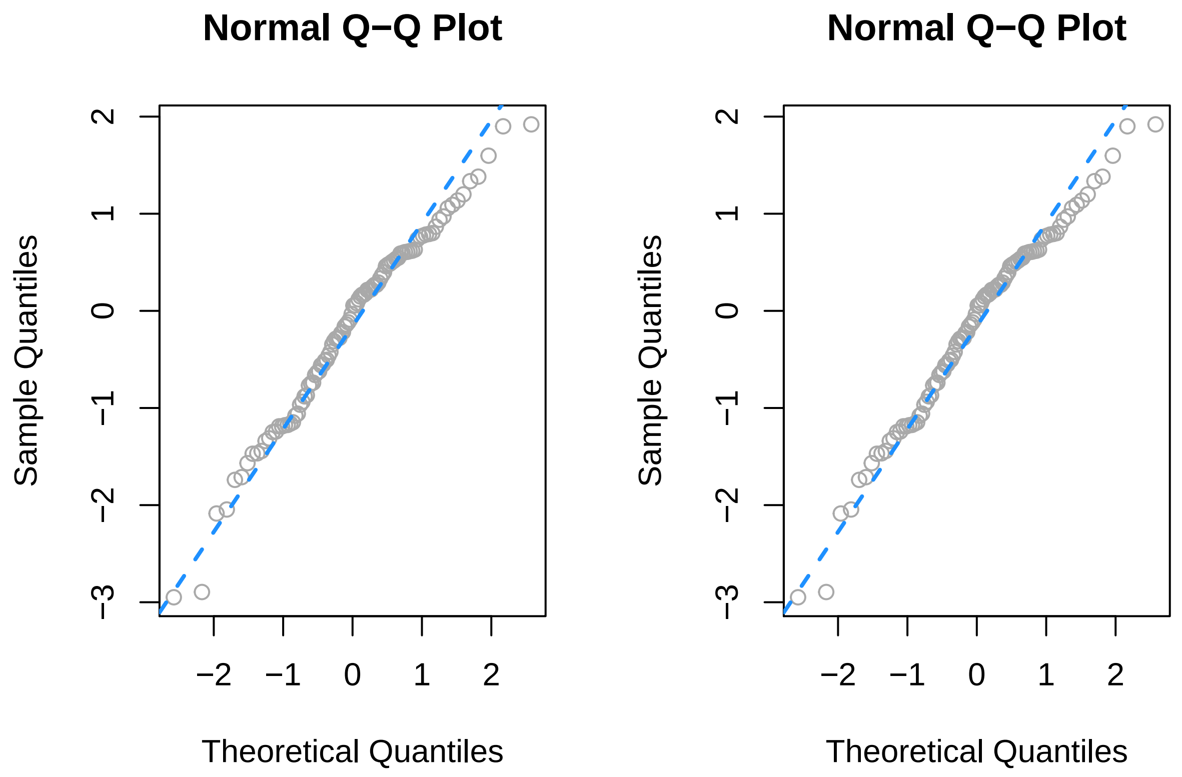


Q-Q plots

```
qq_plot = function(e) {

  n = length(e)
  normal_quantiles = qnorm(((1:n - 0.5) / n))
  # normal_quantiles = qnorm(((1:n) / (n + 1)))

  # plot theoretical verus observed quantiles
  plot(normal_quantiles, sort(e),
       xlab = c("Theoretical Quantiles"),
       ylab = c("Sample Quantiles"),
       col = "darkgrey")
  title("Normal Q-Q Plot")

  # calculate line through the first and third quartiles
  slope     = (quantile(e, 0.75) - quantile(e, 0.25)) / (qnorm(0.75) - qnorm(0.25))
  intercept = quantile(e, 0.25) - slope * qnorm(0.25)

  # add to existing plot
  abline(intercept, slope, lty = 2, lwd = 2, col = "dodgerblue")
}
```

qqnorm() and qqline()

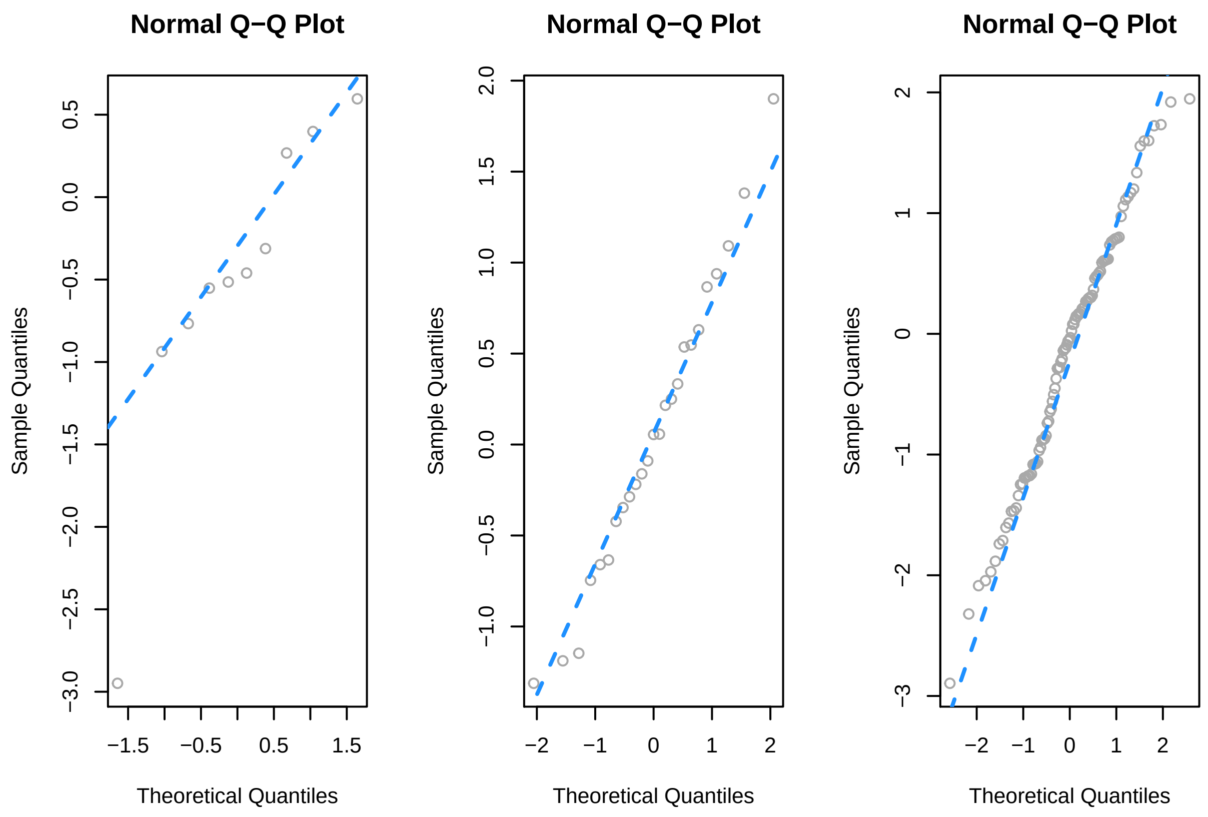
```
set.seed(420)
x = rnorm(100, mean = 0 , sd = 1)
par(mfrow = c(1, 2))
qqnorm(x, col = "darkgrey")
qqline(x, lty = 2, lwd = 2, col = "dodgerblue")
qq_plot(x)
```



```
# equivalent
```

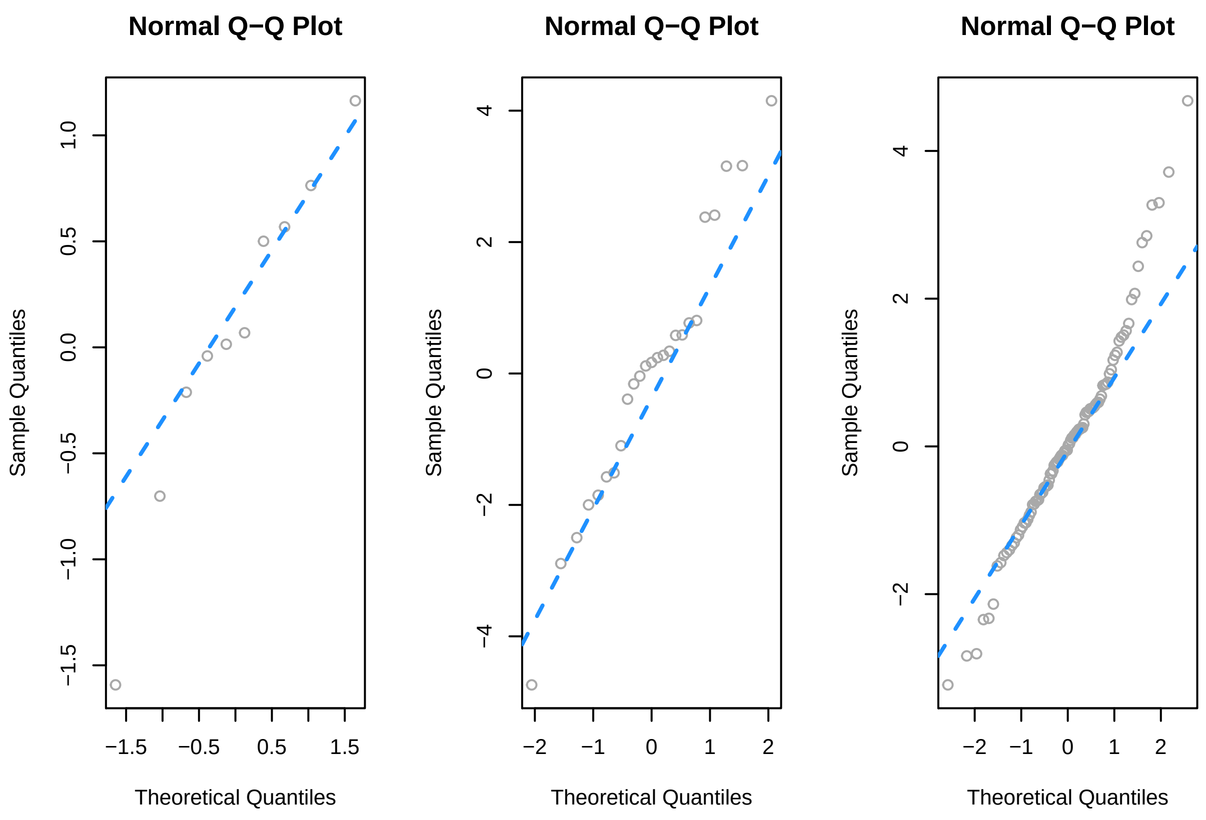
perform a number of simulations “close to the line”

```
# different sample sizes

par(mfrow = c(1, 3))
set.seed(420)
qq_plot(rnorm(10))
qq_plot(rnorm(25))
qq_plot(rnorm(100))
```



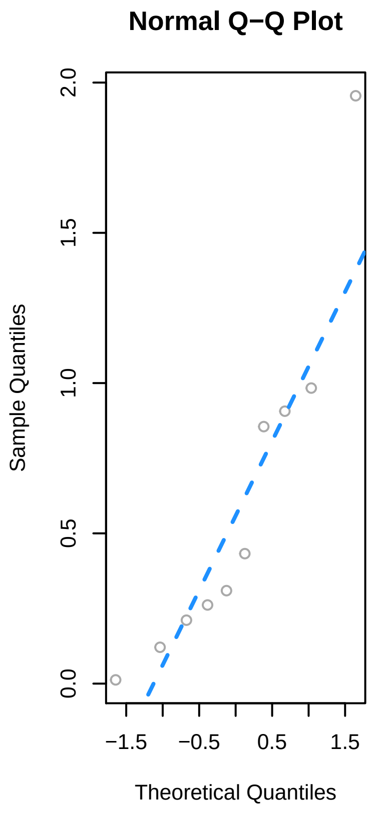
simulate data from a t distribution with a small degrees of freedom, for different sample sizes.

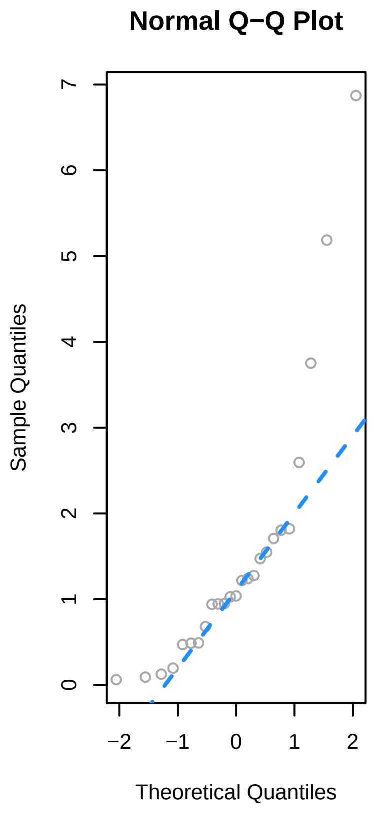
```r
par(mfrow = c(1, 3))
set.seed(420)
qq_plot(rt(10, df = 4))
qq_plot(rt(25, df = 4))
qq_plot(rt(100, df = 4))
```

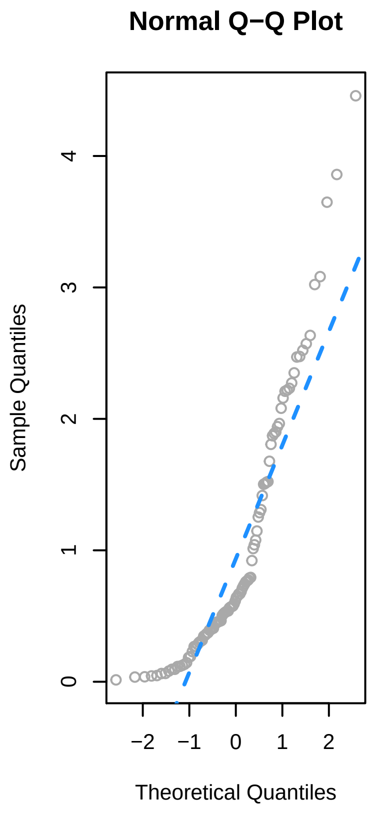


simulate data from an exponential distribution

```
par(mfrow = c(1, 3))
set.seed(420)
qq_plot(rexp(10))
qq_plot(rexp(25))
qq_plot(rexp(100))
```

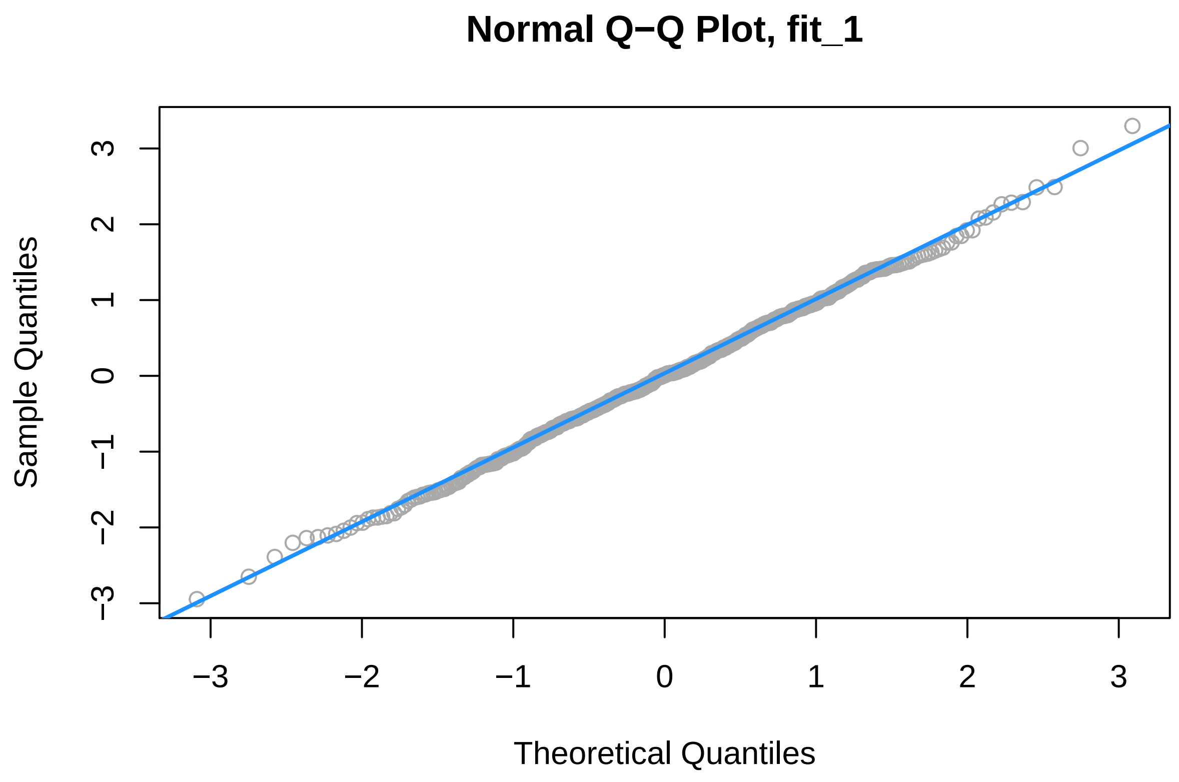






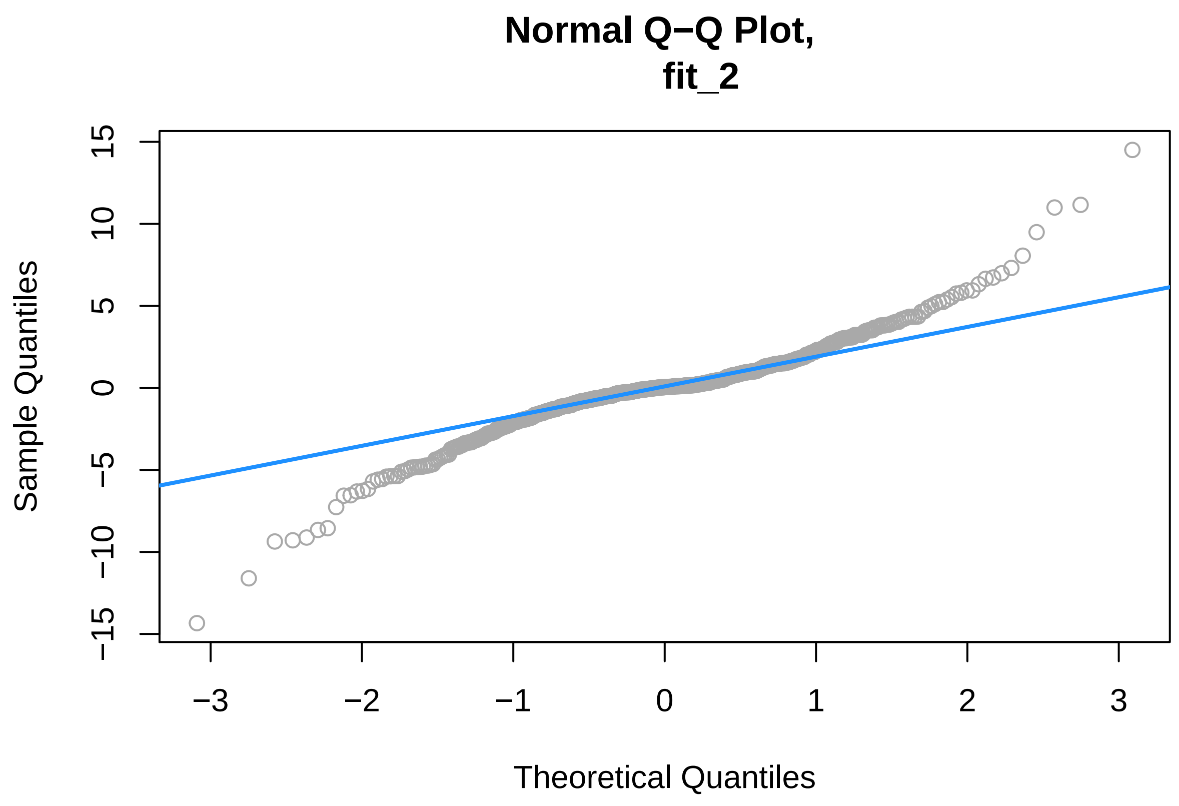
create a Q-Q plot for each to assess normality of errors

```r
qqnorm(resid(fit_1), main = "Normal Q-Q Plot, fit_1", col = "darkgrey")
qqline(resid(fit_1), col = "dodgerblue", lwd = 2)
```



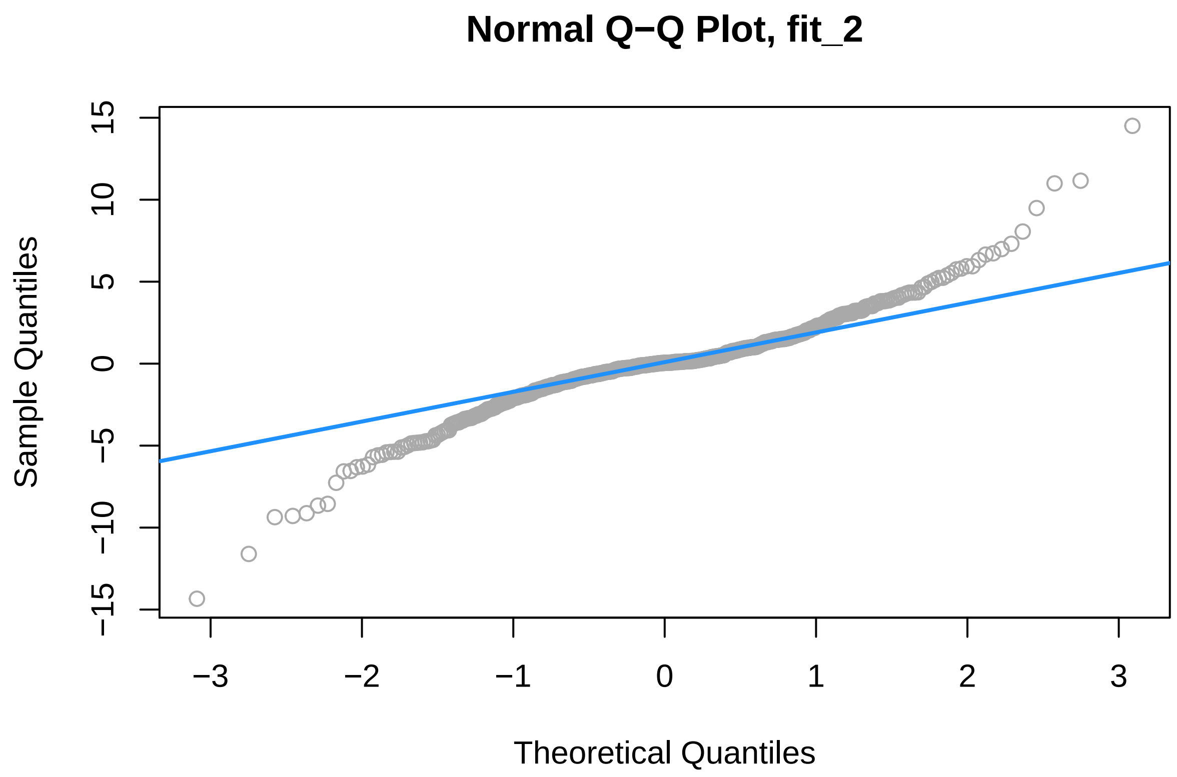
fit_1, perfect Q-Q plot, errors follow a normal distribution

```r
qqnorm(resid(fit_2),
       main = "Normal Q-Q Plot,
       fit_2",
       col = "darkgrey")
qqline(resid(fit_2),
       col = "dodgerblue",
       lwd = 2)
```



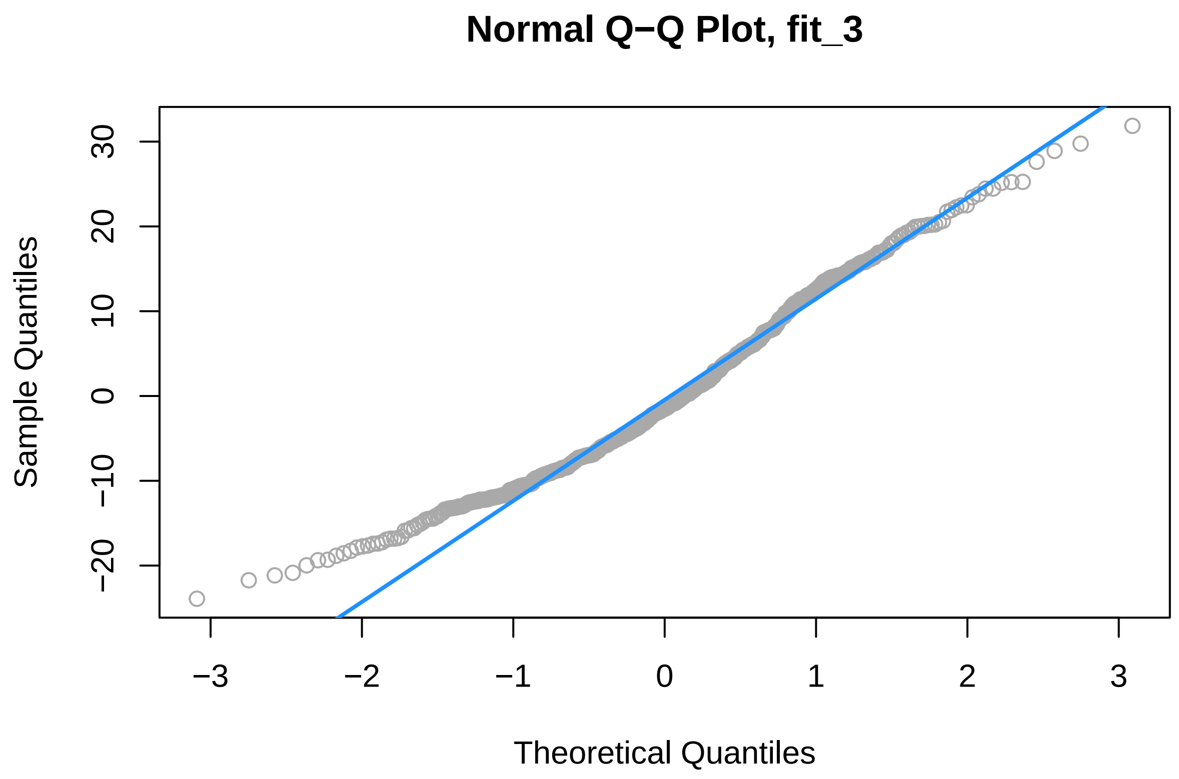
fit_2, suspect Q-Q plot

```
qqnorm(resid(fit_2), main = "Normal Q-Q Plot, fit_2", col = "darkgrey")
qqline(resid(fit_2), col = "dodgerblue", lwd = 2)
```



fit_3, suspect Q-Q plot

```
qqnorm(resid(fit_3), main = "Normal Q-Q Plot, fit_3", col = "darkgrey")
qqline(resid(fit_3), col = "dodgerblue", lwd = 2)
```



5 Shapiro-Wilk Test

formal testing shapiro.test()

```
set.seed(42)
shapiro.test(rnorm(25))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  rnorm(25)
## W = 0.9499, p-value = 0.2495
```

```
## W = 0.71164, p-value = 1.05e-05

shapiro.test(rexp(25))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  rexp(25)
## W = 0.71164, p-value = 1.05e-05
## W = 0.99858, p-value = 0.9622

shapiro.test(resid(fit_3))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(fit_3)
## W = 0.97643, p-value = 3.231e-07
```

```
## W = 0.97643, p-value = 3.231e-07
```

Outliers

resid() for residual rstandard() for Standardized residual