Eyeballing your Models: Using visual diagnostics to assess model fit

EPsy Computing Club - Yadira Peralta

Eyeballing your Models

- Plots to assess model fit
- Plots generated by base R
- Using fortify() and ggplot()
- Customize your graphs

Linear regression assumptions and plots to diagnose them

Assumption	Plot
Independence of residuals	It has to do with study design or data collection. No plot for cross-sectional data
Homoscedasticity (constant variance of the residuals)	Residuals or standardized residuals vs fitted values
Residuals are normally distributed	Q-Q plot of residuals
Linear relationship between dependent and independent variables	Plot residuals versus individual independent variables and/or residuals vs fitted values

Example 1 - well-behaved data

Generating the data

```
set.seed(13)
x = runif(n = 100, min = 0, max = 10)
y = 2 + 0.8*x + rnorm(n = 100, mean = 0, sd = 1)
mydata = data.frame(x,y)
```

Fitting a linear regression

```
mylinreg = lm(y ~ x, data = mydata)
```

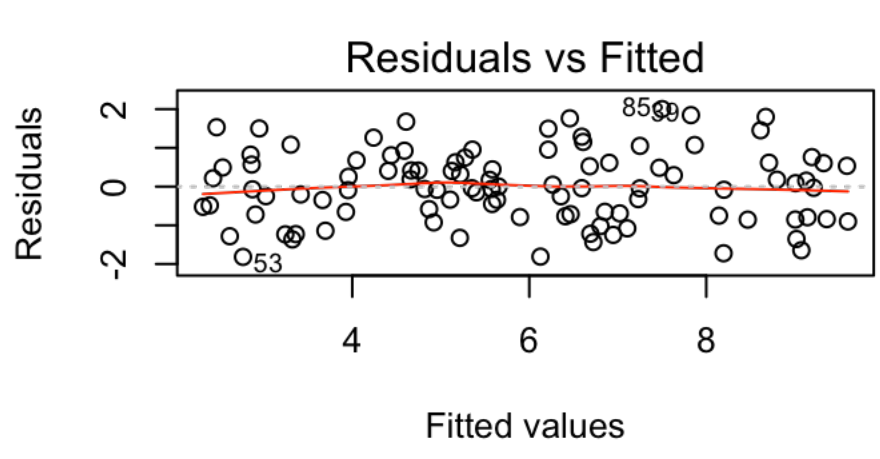
Example 1: well-behaved data

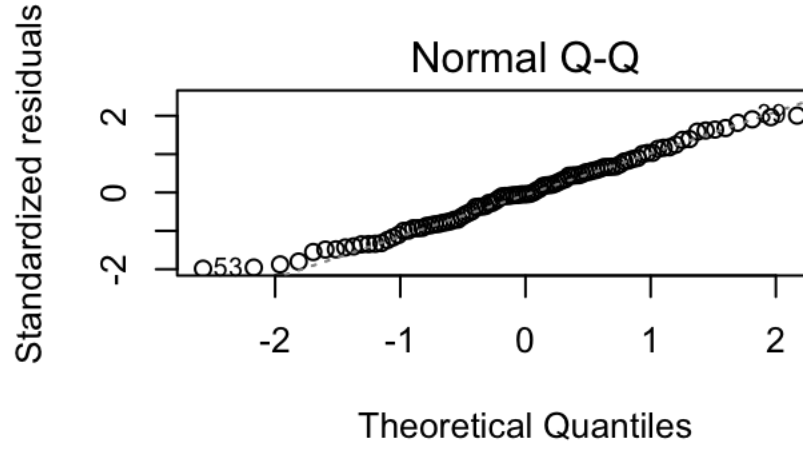
```
summary(mylinreg)
```

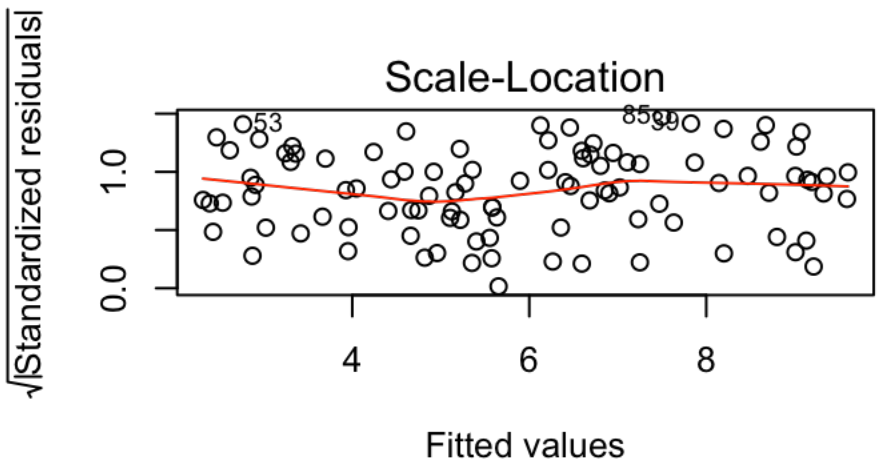
```
##
## Call:
## lm(formula = y ~ x, data = mydata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.81621 -0.72905 -0.04243  0.61503  2.00172
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.23274    0.18724   11.93   <2e-16 ***
## x            0.73731    0.03286   22.44   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9292 on 98 degrees of freedom
## Multiple R-squared:  0.8371, Adjusted R-squared:  0.8354
## F-statistic: 503.5 on 1 and 98 DF,  p-value: < 2.2e-16
```

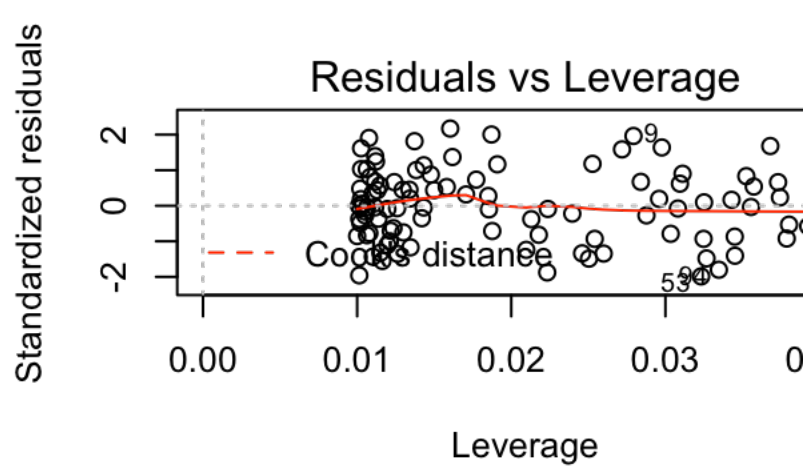
Plots generated by base R

```
par(mfrow = c(2, 2))
plot(mylinreg)
```





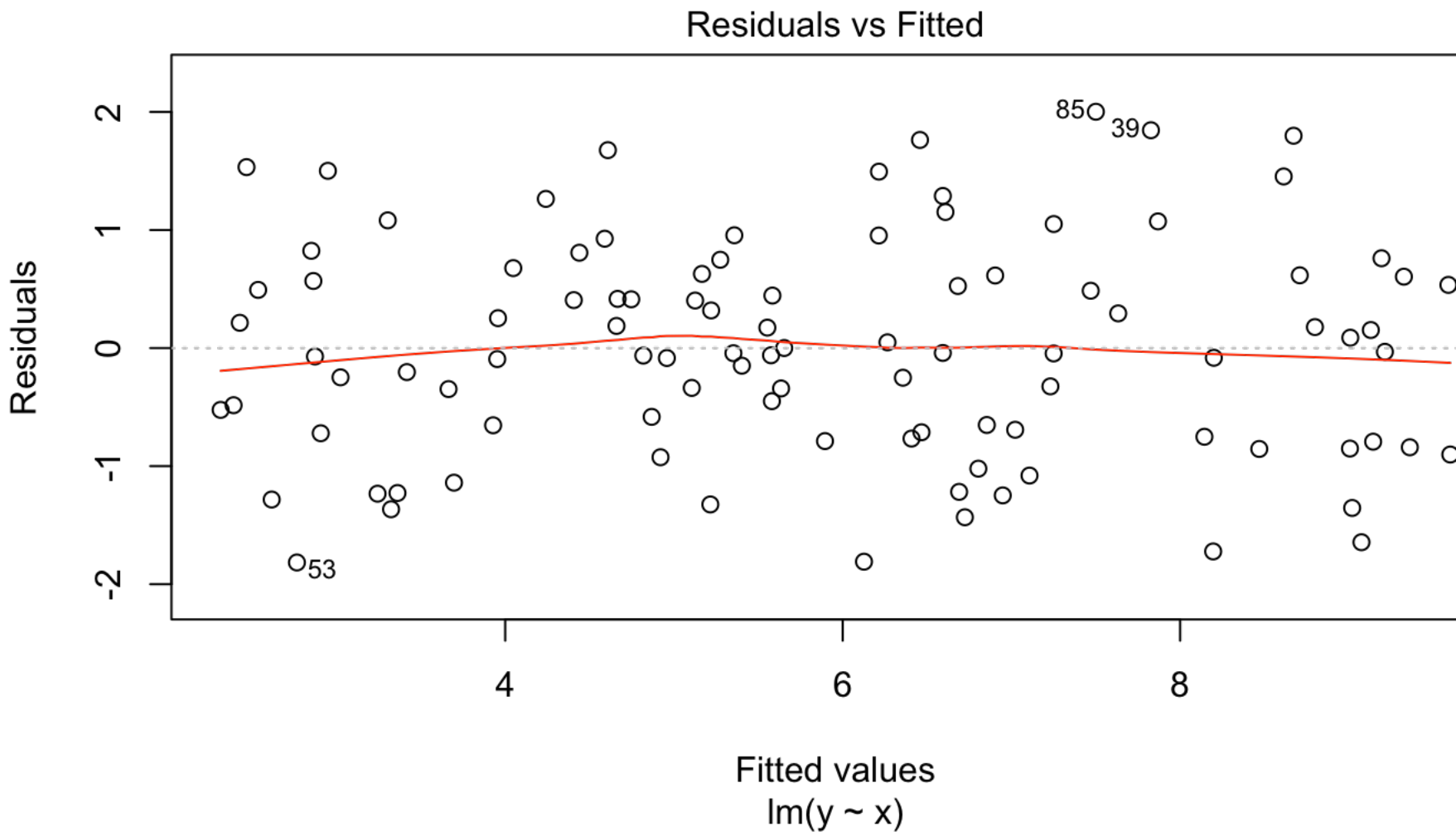





```
par(mfrow = c(1, 1))
```

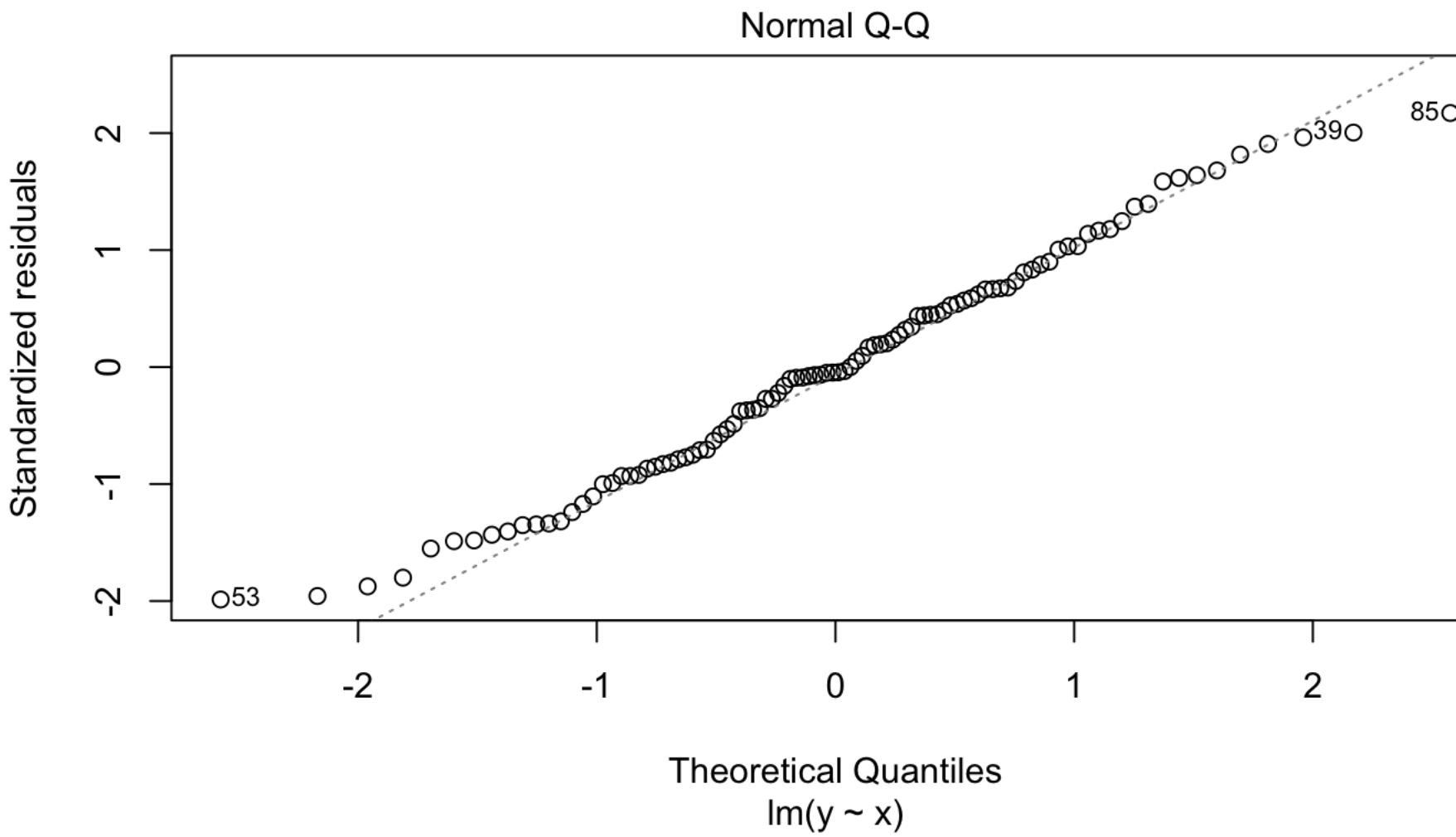
Plots generated by base R

Linearity and homoscedasticity assumptions



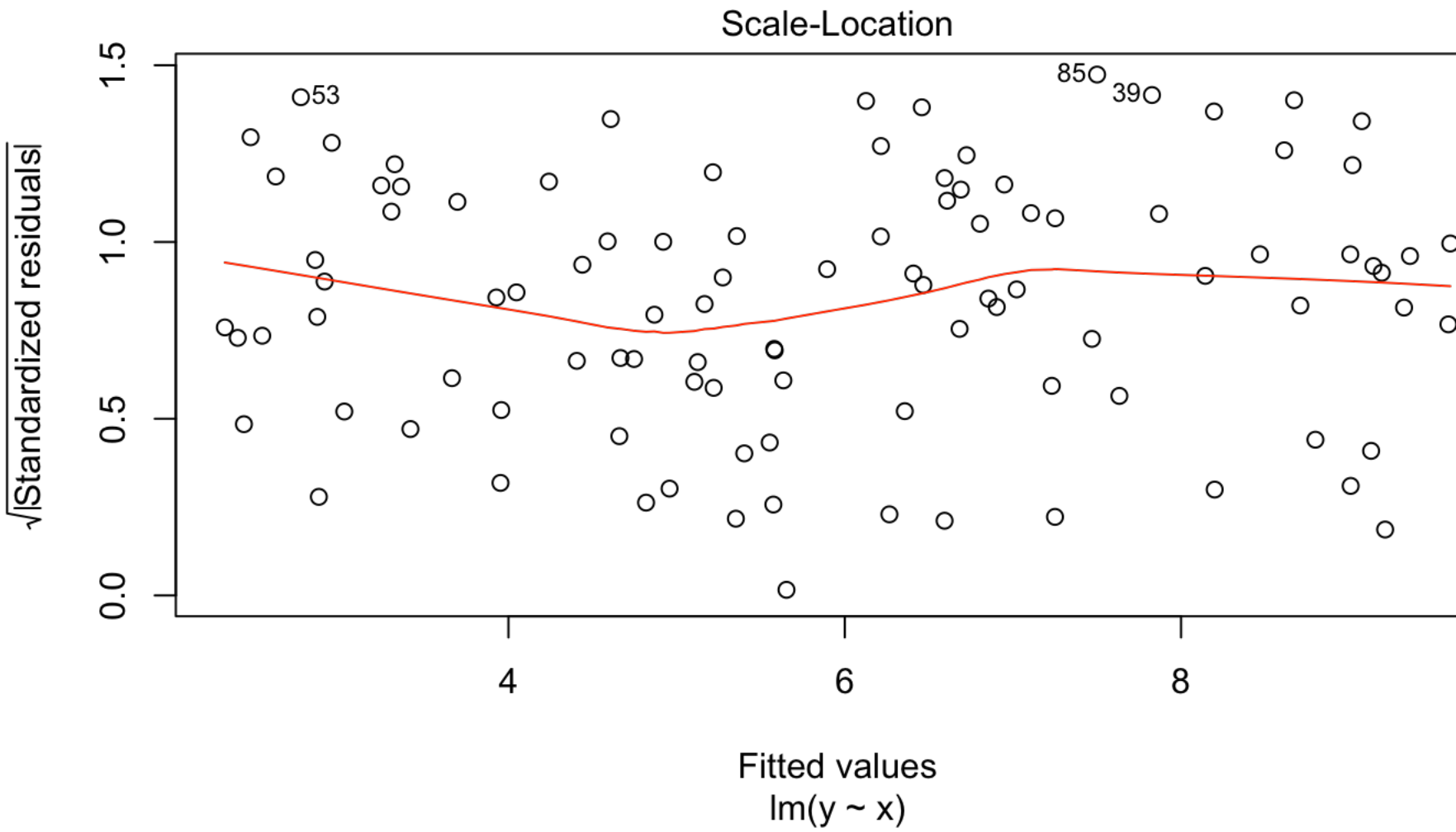
Plots generated by base R

Normality assumption



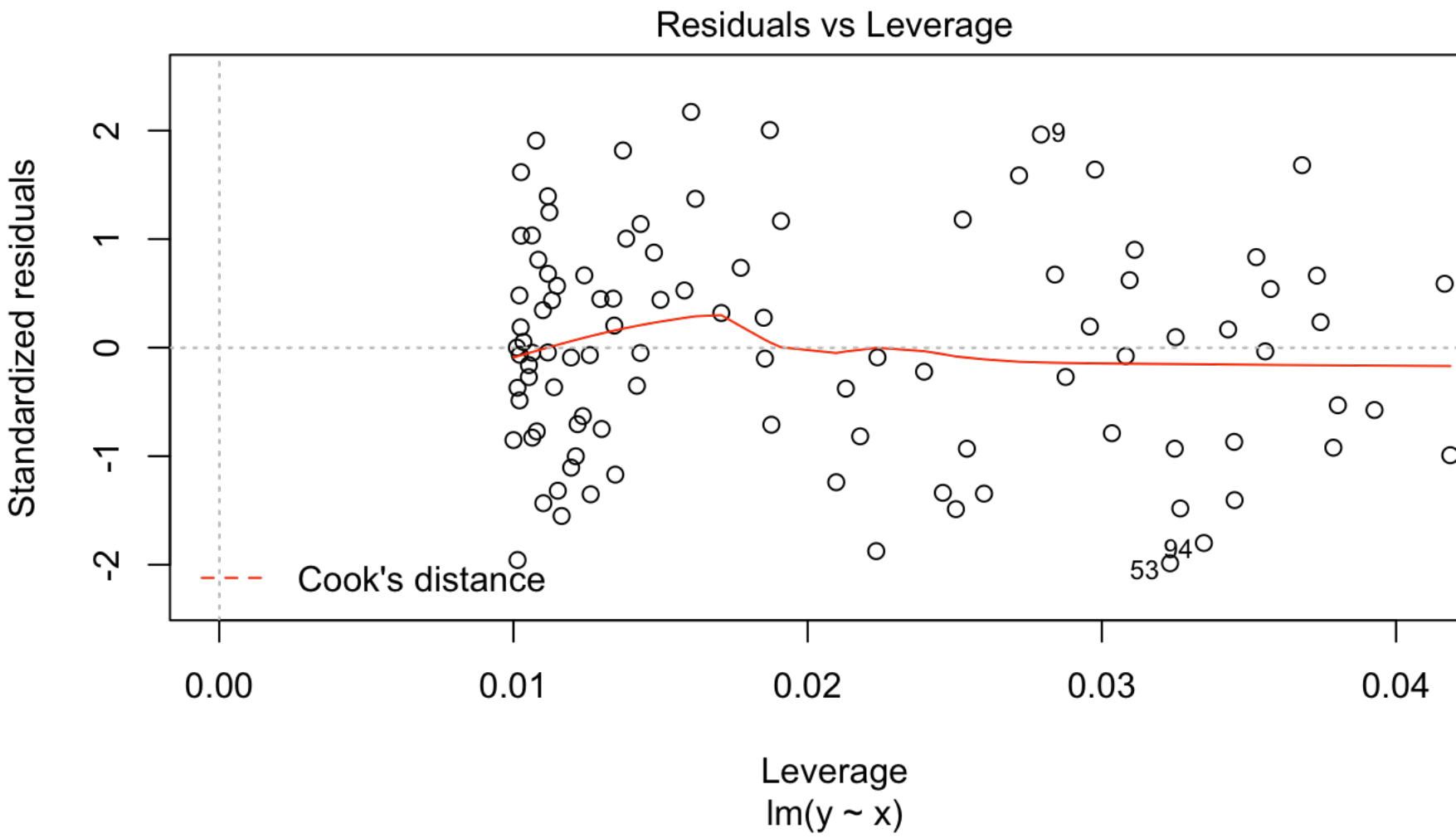
Plots generated by base R

Homoscedasticity assumption



Plots generated by base R

Detect extreme values



Using fortify() and ggplot()

- fortify(): Adds the variables listed below to the original dataset

Assumption	Plot
.hat	Measure of the leverage of a data point
.sigma	Estimate of residual standard deviation when corresponding observation is dropped from the model
.cooksd	Cook's distances
.fitted	Fitted values
.resid	Residuals
.stdresid	Standardized residuals

Using fortify() and ggplot()

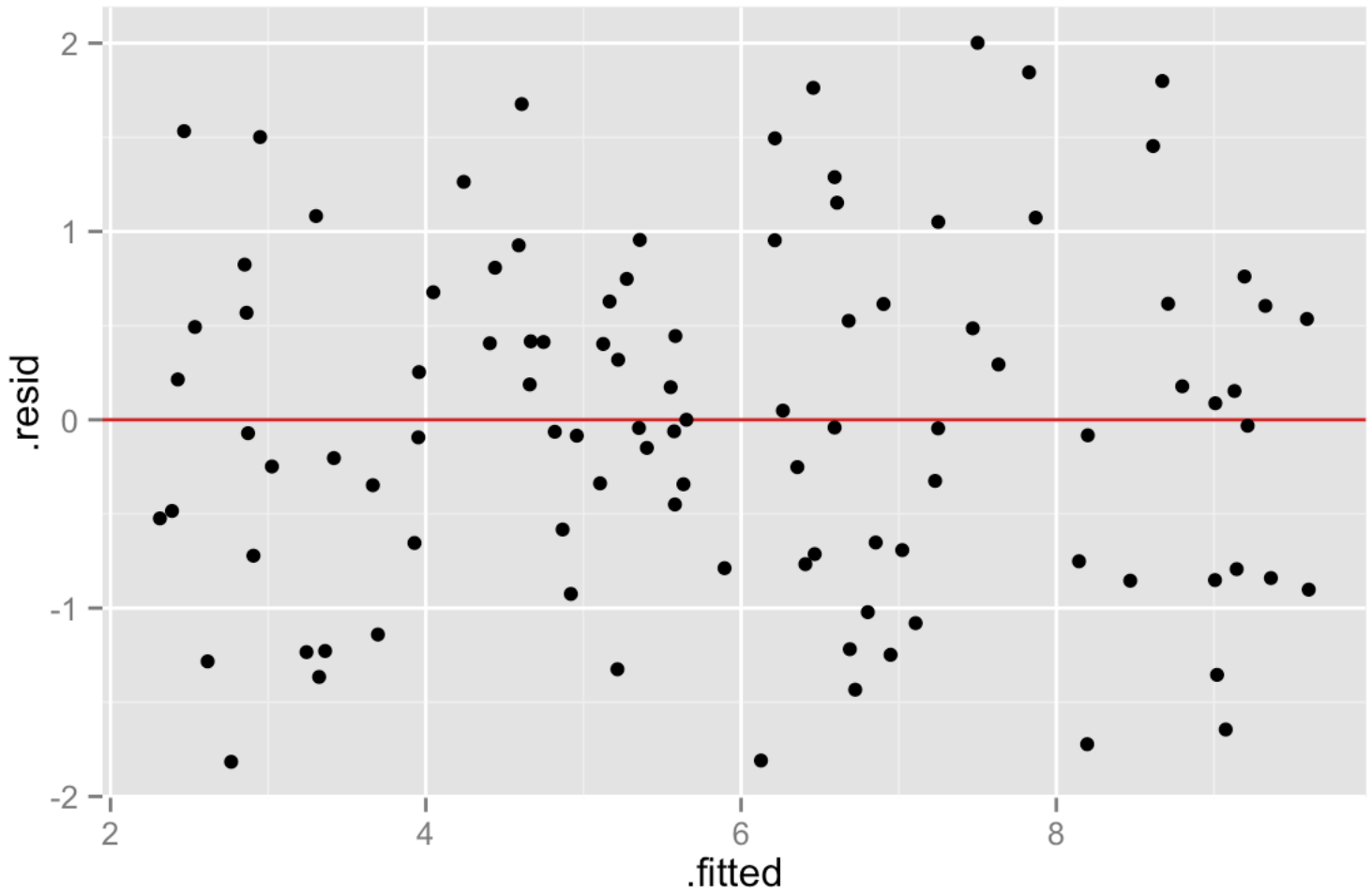
With fortified data we can recreate the plots given by plot() and we can even modify them

```
library(ggplot2)
myfortdata = fortify(mylinreg)
head(myfortdata)
```

```
##          y         x       .hat    .sigma      .cooksd  .fitted     .re
## 1 7.955709 7.1032245 0.01581233 0.9326433 0.0022297853 7.470050  0.485
## 2 4.724623 2.4613730 0.01772706 0.9313875 0.0048776791 4.047544  0.677
## 3 4.768057 3.8963444 0.01138095 0.9333307 0.0007681778 5.105570 -0.337
## 4 2.184879 0.9138367 0.03034237 0.9309981 0.0097323900 2.906522 -0.721
## 5 9.930878 9.6206454 0.03731024 0.9318682 0.0085249246 9.326183  0.604
## 6 1.789638 0.1093333 0.03926667 0.9323899 0.0067571505 2.313350 -0.523
##    .stdresid
## 1  0.5268507
## 2  0.7352235
## 3 -0.3653181
## 4 -0.7886952
## 5  0.6632686
## 6 -0.5750243
```

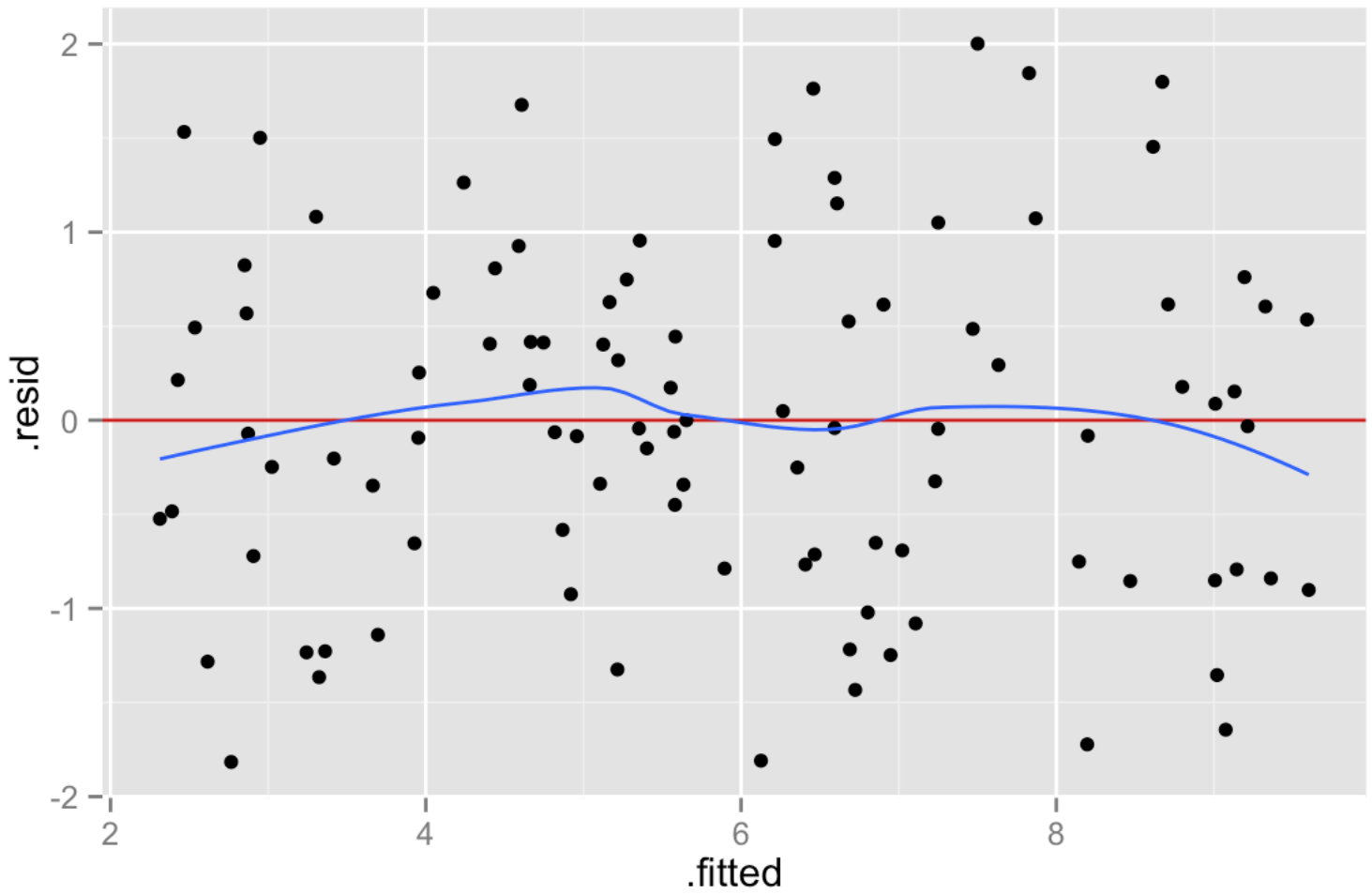
Homoscedasticity assumption

```
ggplot(data = myfortdata, aes(x = .fitted, y = .resid)) +
  geom_hline(yintercept = 0, colour = "firebrick3") +
  geom_point()
```



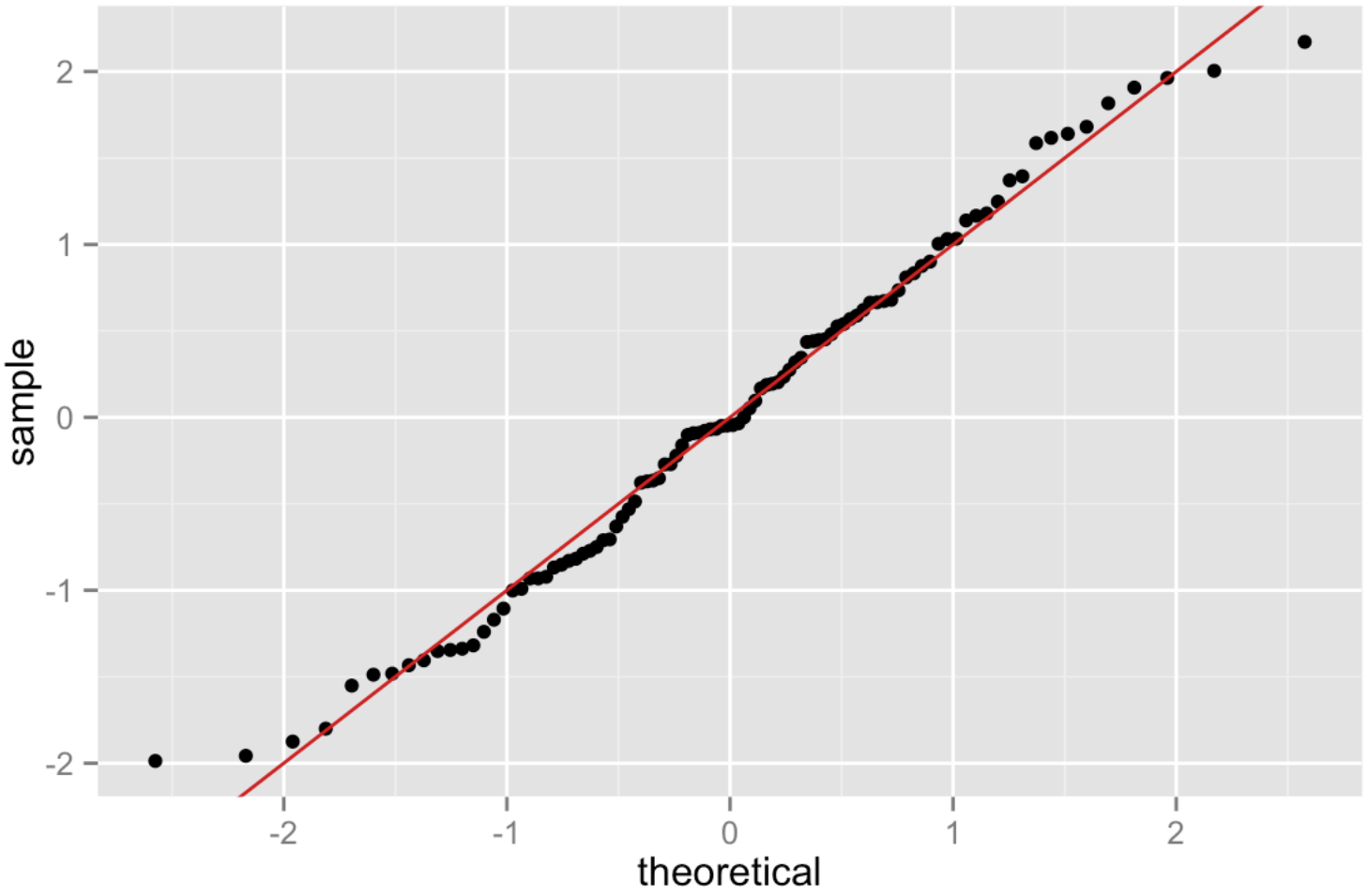
Linearity assumption

```
ggplot(data = myfortdata, aes(x = .fitted, y = .resid)) +
  geom_hline(yintercept = 0, colour = "firebrick3") +
  geom_point() +
  geom_smooth(se = FALSE)
```



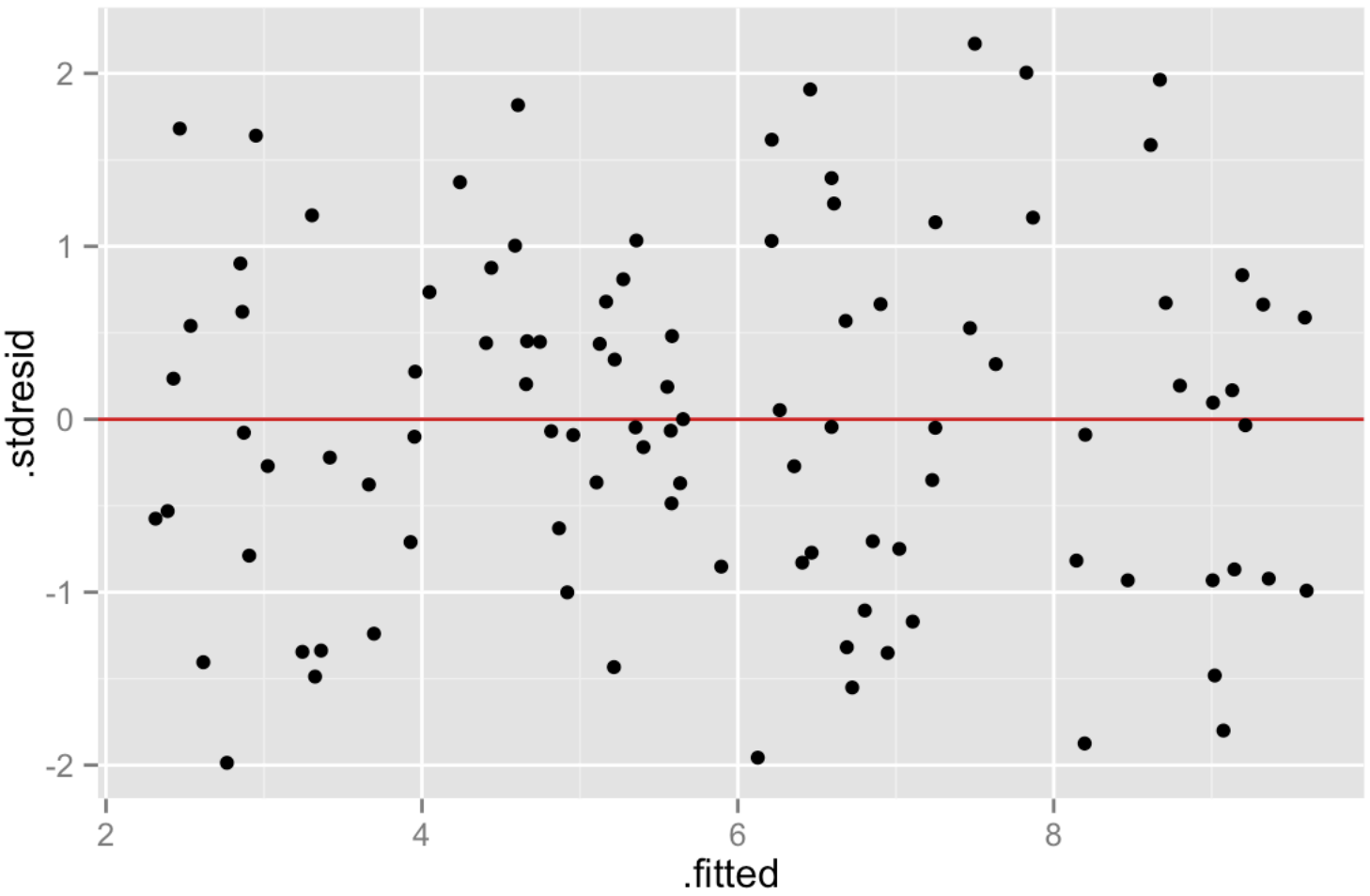
Normality assumption

```
ggplot(data = myfortdata, aes(sample = .stdresid)) +
  stat_qq() +
  geom_abline(colour = "firebrick3")
```



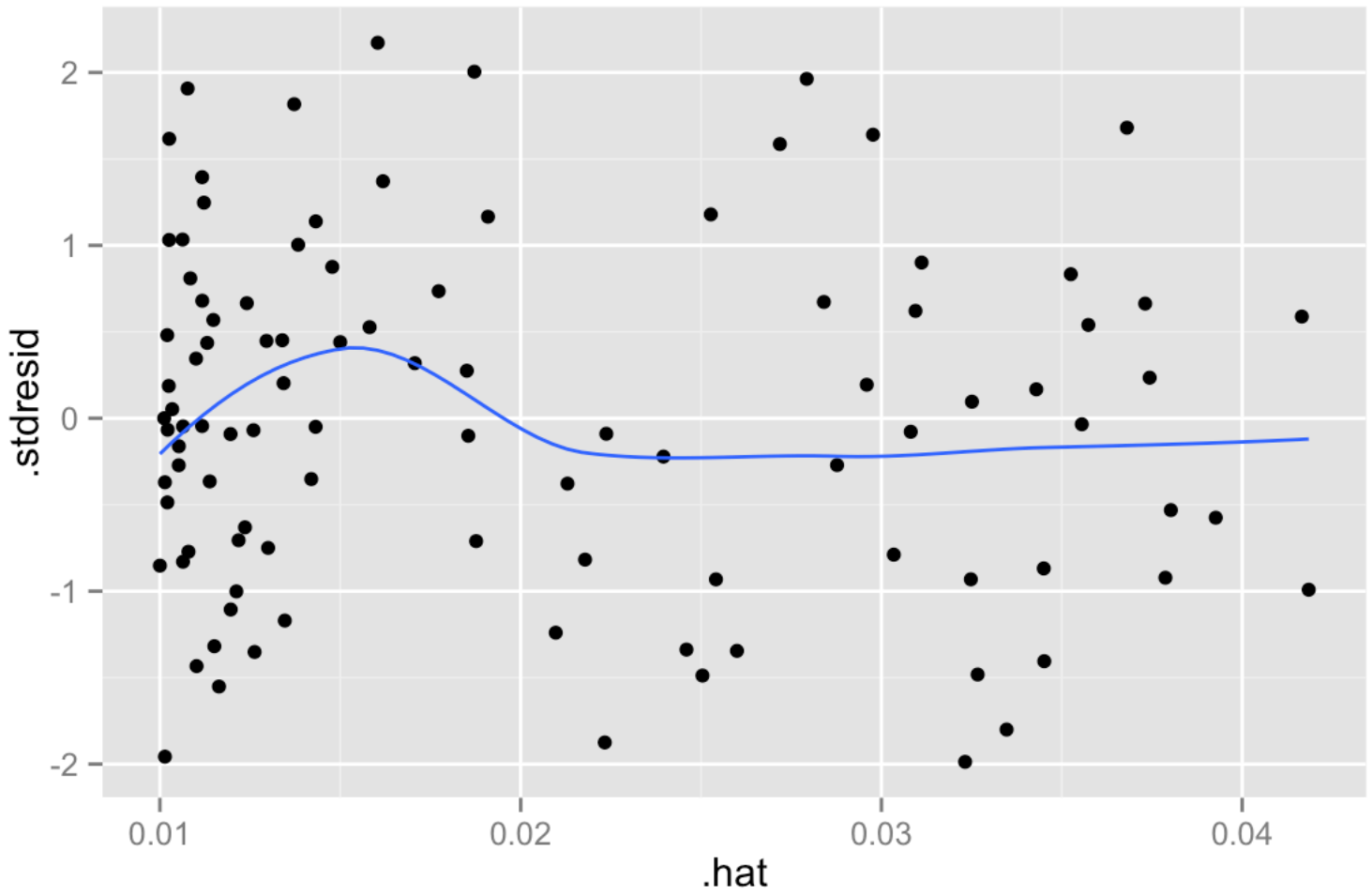
Standardized residuals vs fitted values

```
ggplot(data = myfortdata, aes(x = .fitted, y = .stdresid)) +
  geom_hline(yintercept = 0, colour = "firebrick3") +
  geom_point()
```



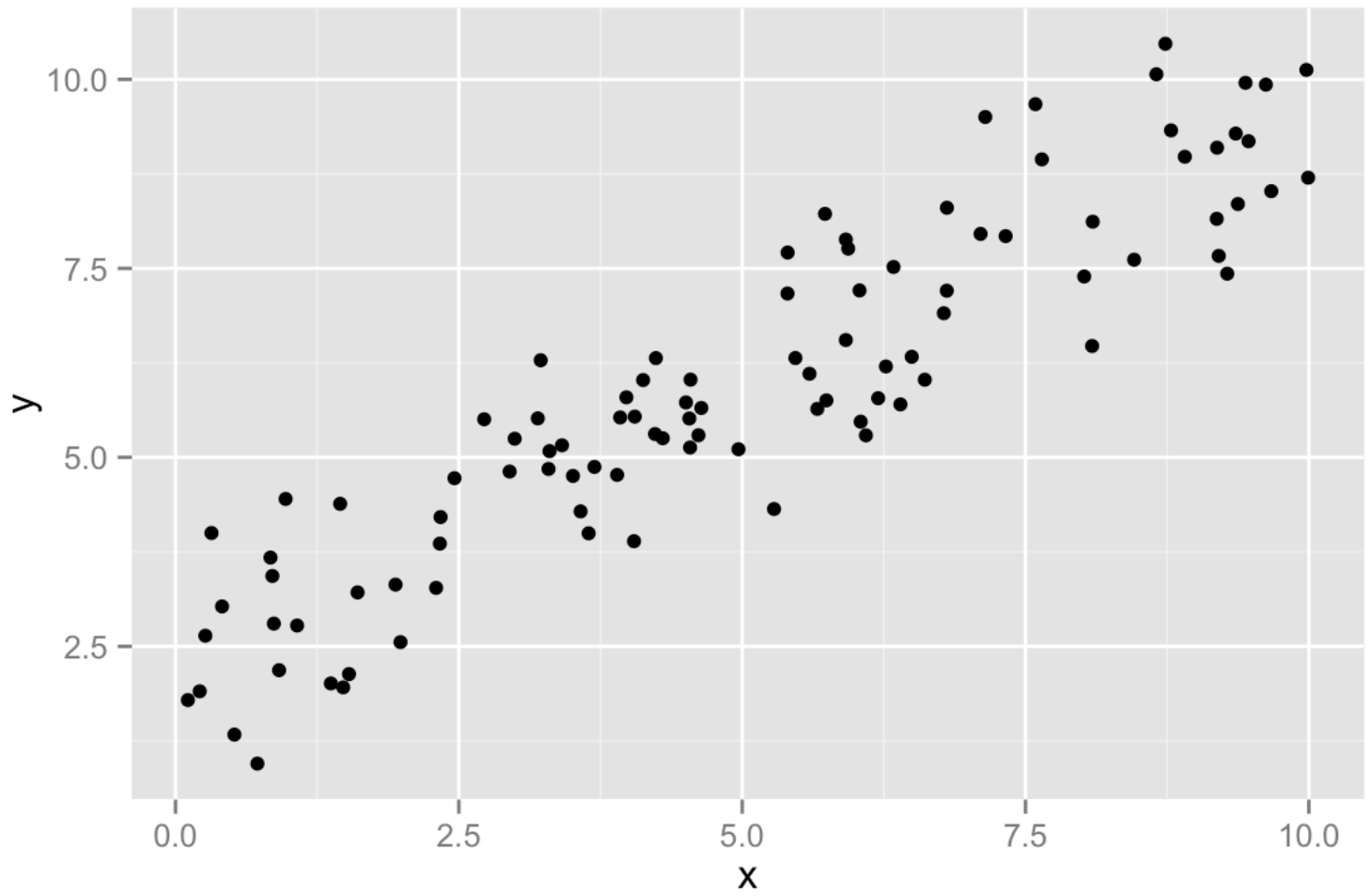
Residuals vs. leverages

```
ggplot(data = myfortdata, aes(x = .hat, y = .stdresid)) +
  geom_point() +
  geom_smooth(se = FALSE)
```



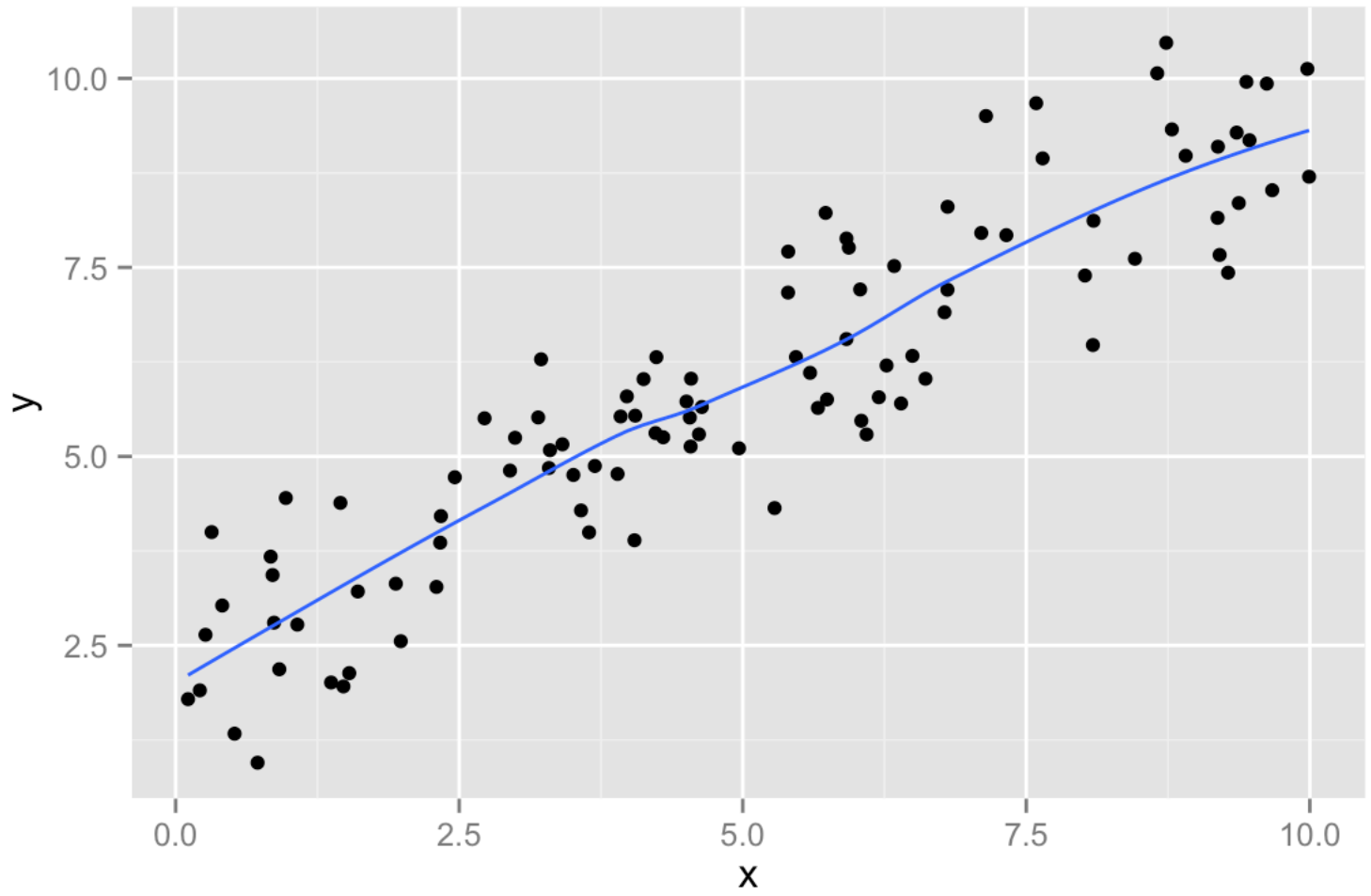
Customized plots

```
# Start with a scatter plot of your data
ggplot(data = myfortdata, aes(x = x, y = y)) +
  geom_point()
```



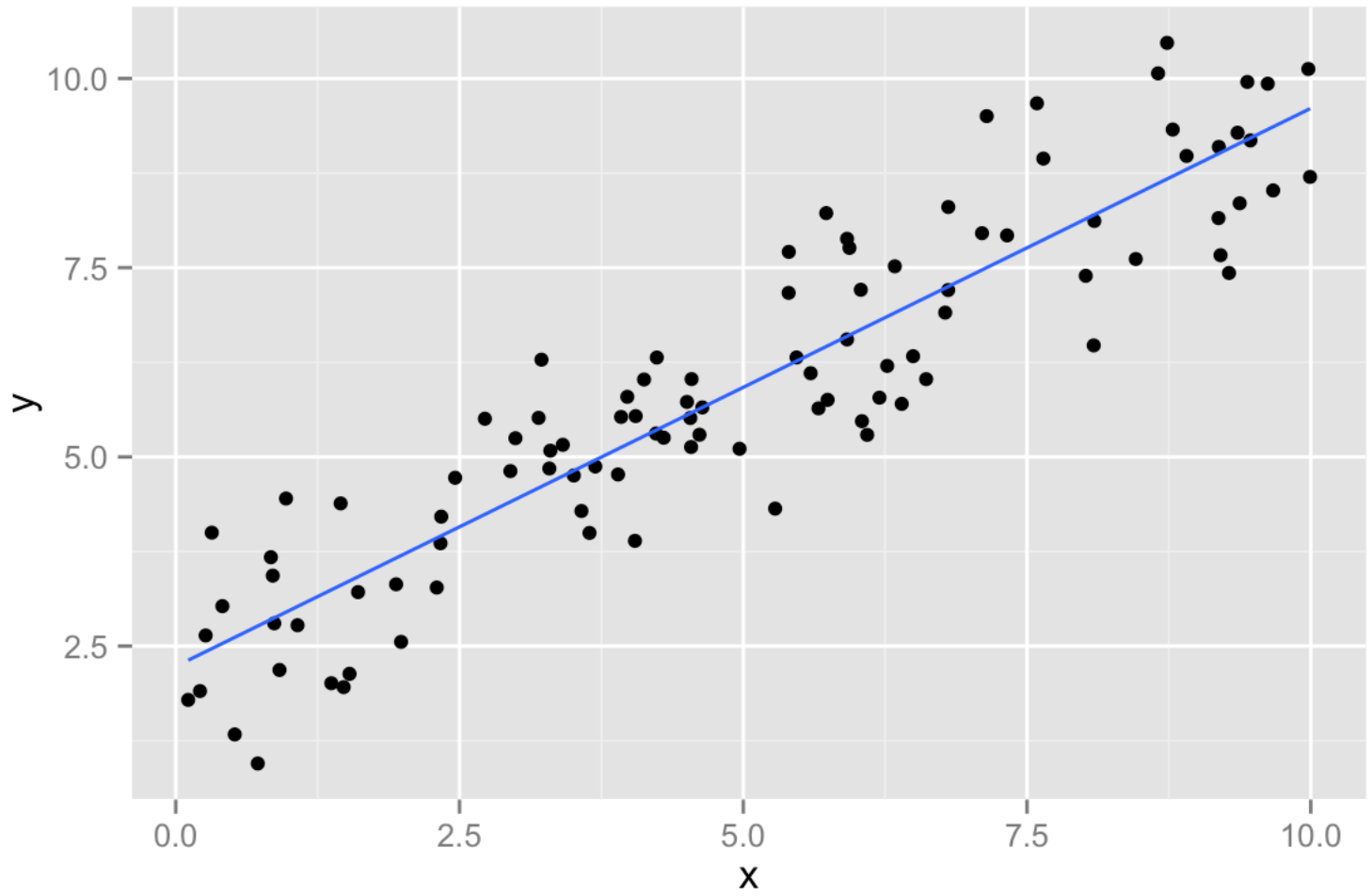
Customized plots

```
# Explore the relationship using "loess"
ggplot(data = myfortdata, aes(x = x, y = y)) +
  geom_point() +
  stat_smooth(method = "loess", se = FALSE)
```



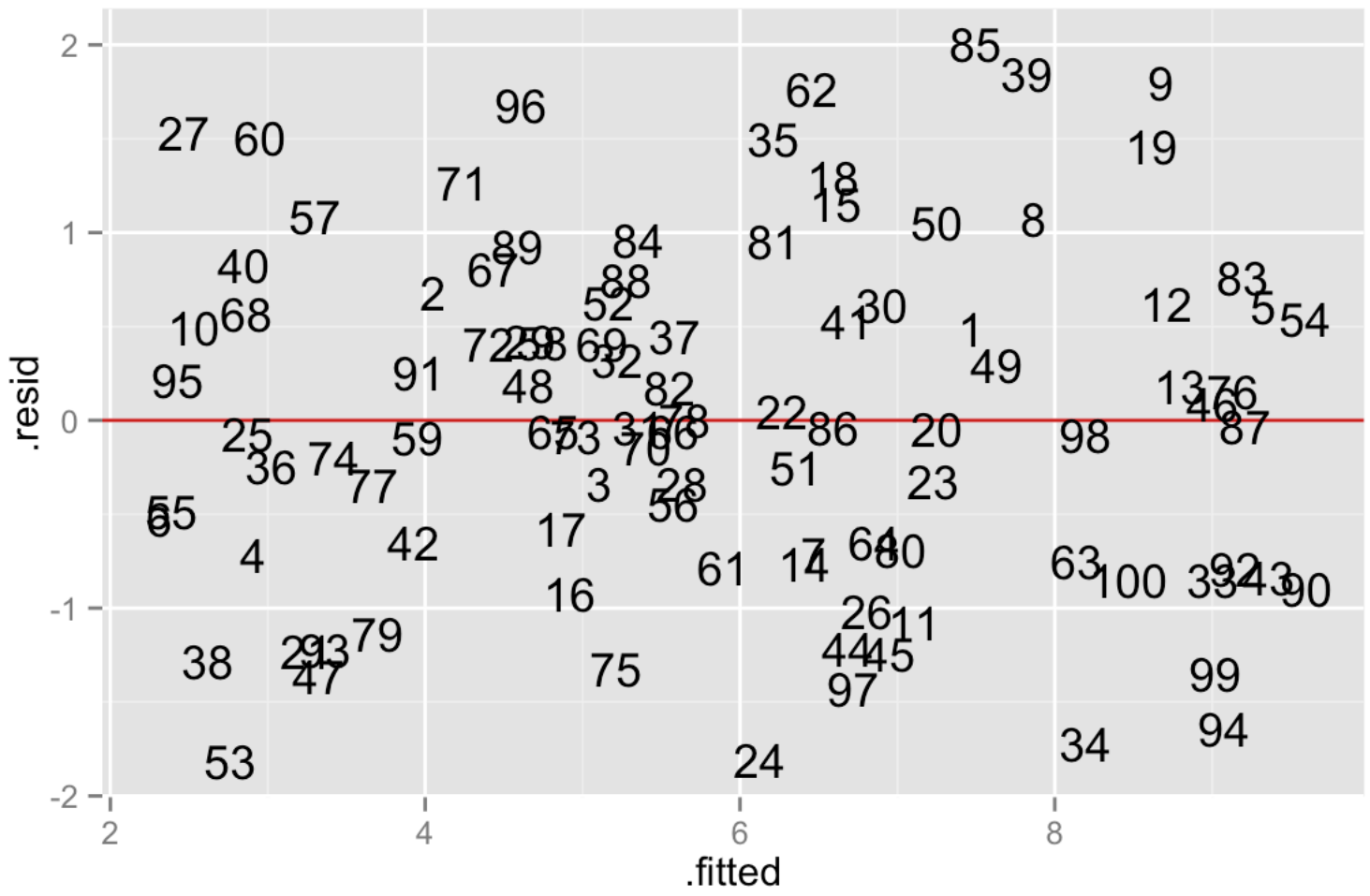
Customized plots

```
# Add a regression line
ggplot(data = myfortdata, aes(x = x, y = y)) +
  geom_point() +
  stat_smooth(method = "lm", formula = y ~ x, se = FALSE)
```



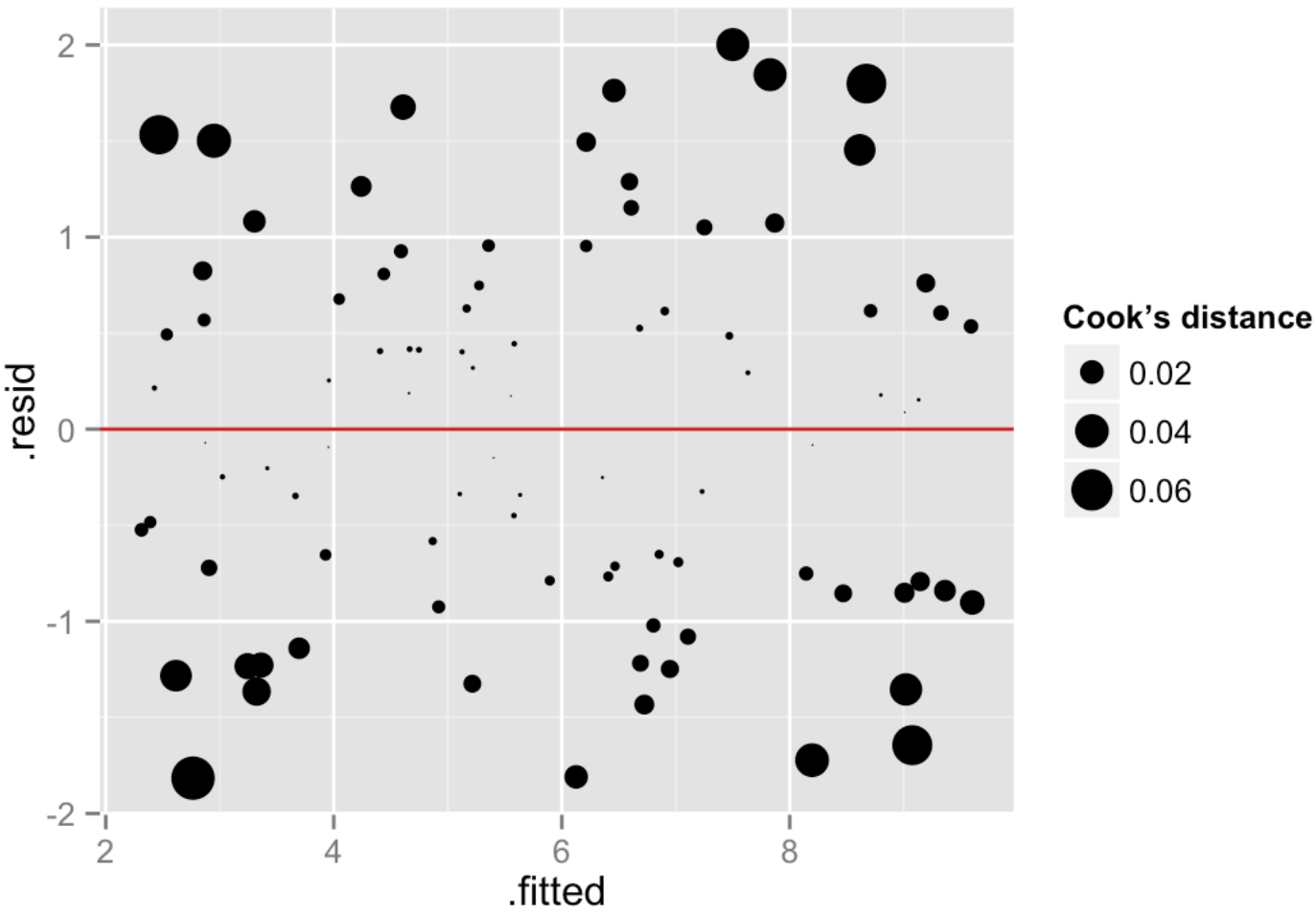
Customized plots

```
# Adding observation number
ggplot(data = myfortdata, aes(x = .fitted, y = .resid)) +
  geom_hline(yintercept = 0, colour = "firebrick3") +
  geom_text(label = rownames(myfortdata))
```



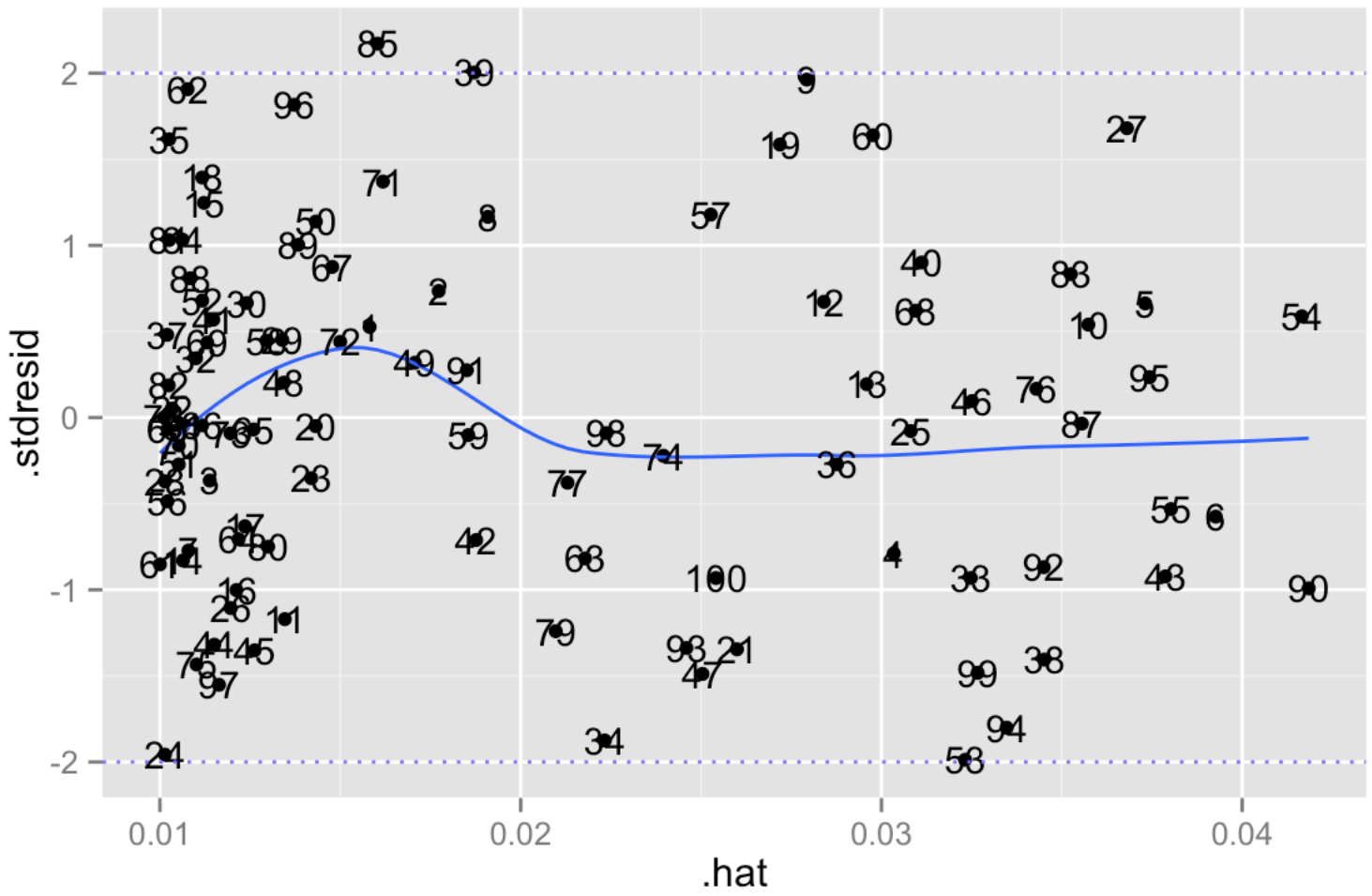
Customized plots

```
# Points size reflecting Cook's distance
ggplot(data = myfortdata, aes(x = .fitted, y = .resid, size = .cooksd)) +
  geom_hline(yintercept = 0, colour = "firebrick3") +
  geom_point() +
  scale_size_area("Cook's distance")
```



Customized plots

```
# Residuals vs. leverages with observation number
ggplot(data = myfortdata, aes(x = .hat, y = .stdresid)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  geom_text(label = rownames(myfortdata), size = 4) +
  geom_hline(yintercept = 2, lty = "dotted", colour = "slateblue1") +
  geom_hline(yintercept = -2, lty = "dotted", colour = "slateblue1")
```



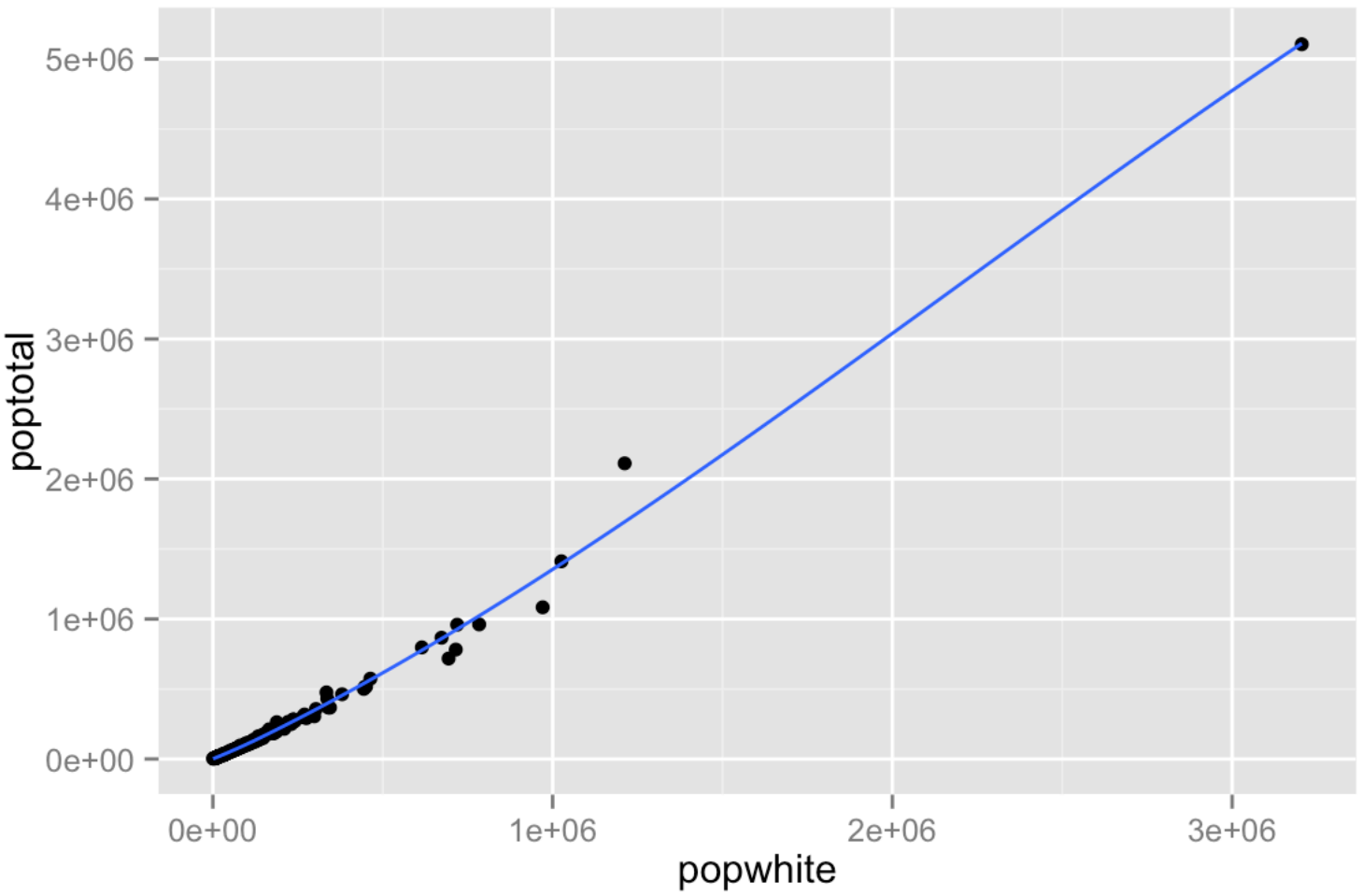
Example 2: midwest data

```
mylinreg2 = lm(poptotal ~ popwhite + popblack, data = midwest)
summary(mylinreg2)
```

```
##
## Call:
## lm(formula = poptotal ~ popwhite + popblack, data = midwest)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -167803    439    1855    2547  166183
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.413e+03  7.710e+02  -4.427 1.21e-05 ***
## popwhite     1.057e+00  7.212e-03 146.630  < 2e-16 ***
## popblack     1.179e+00  1.829e-02  64.484  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13520 on 434 degrees of freedom
## Multiple R-squared:  0.998,  Adjusted R-squared:  0.9979
## F-statistic: 1.059e+05 on 2 and 434 DF,  p-value: < 2.2e-16
```

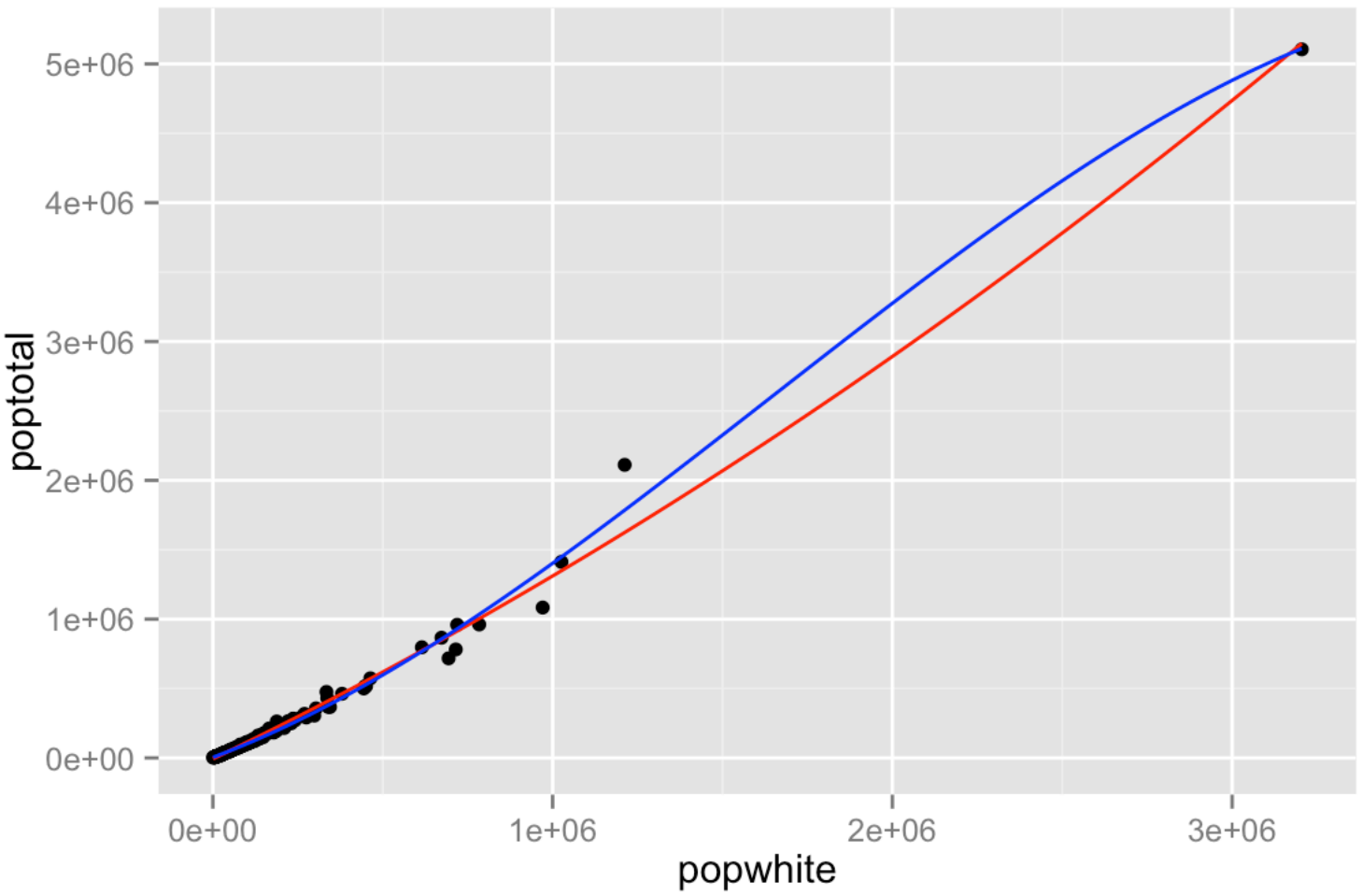
Explore your data

```
myfortdata2 = fortify(mylinreg2)
ggplot(data = myfortdata2, aes(x = popwhite, y = poptotal)) +
  geom_point() +
  stat_smooth(method = "loess", se = FALSE)
```



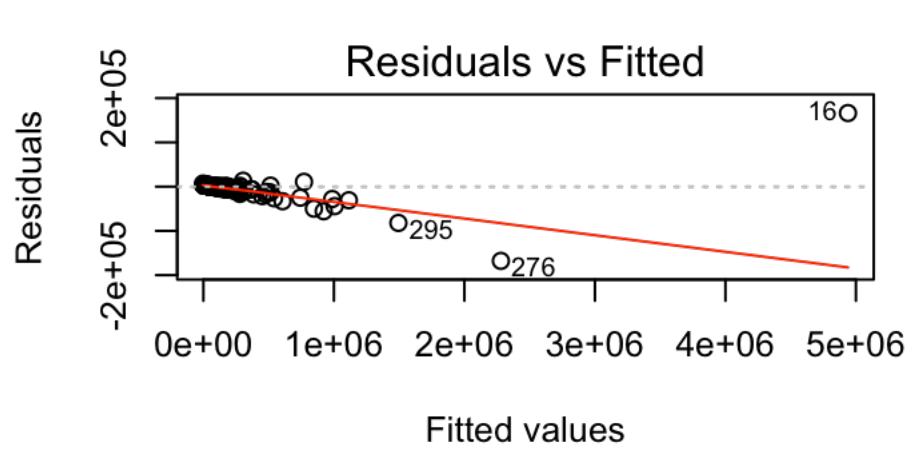
Explore your data

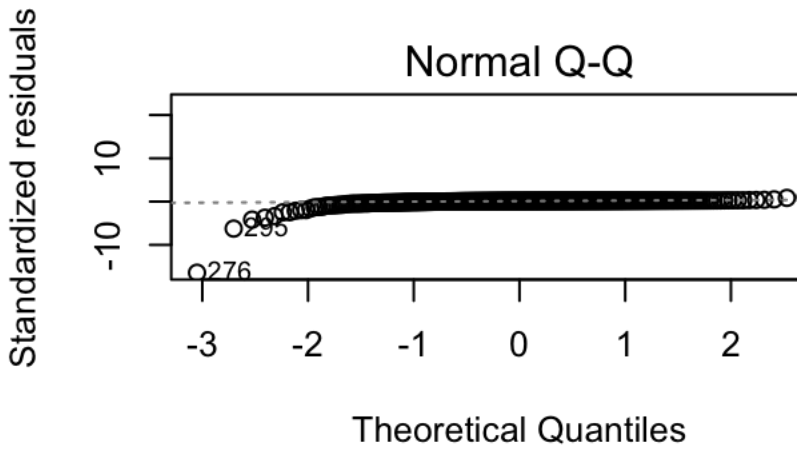
```
ggplot(data = myfortdata2, aes(x = popwhite, y = poptotal)) +
  geom_point() +
  stat_smooth(method = "lm", formula = y ~ x + I(x^2), colour = "red", se
  stat_smooth(method = "lm", formula = y ~ x + I(x^2) + I(x^3), colour = "
```

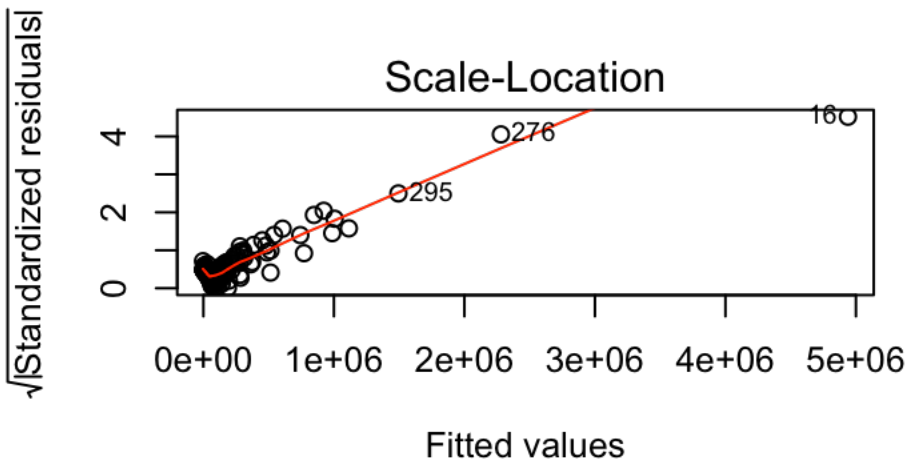


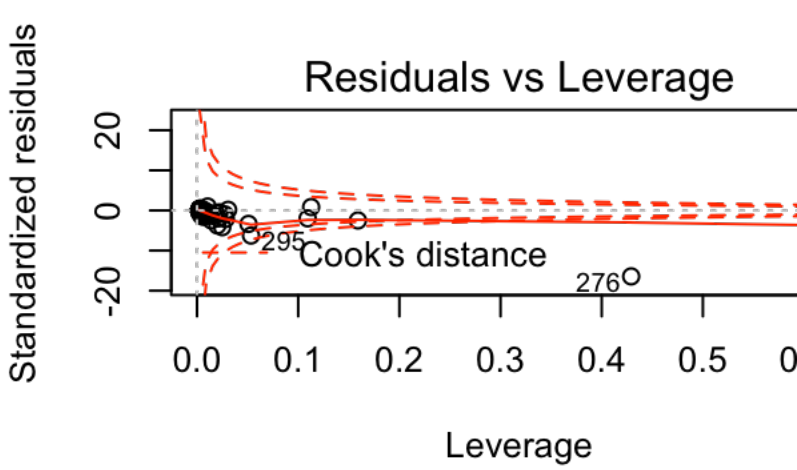
Plots generated by base R

```
par(mfrow = c(2, 2))
plot(mylinreg2)
```





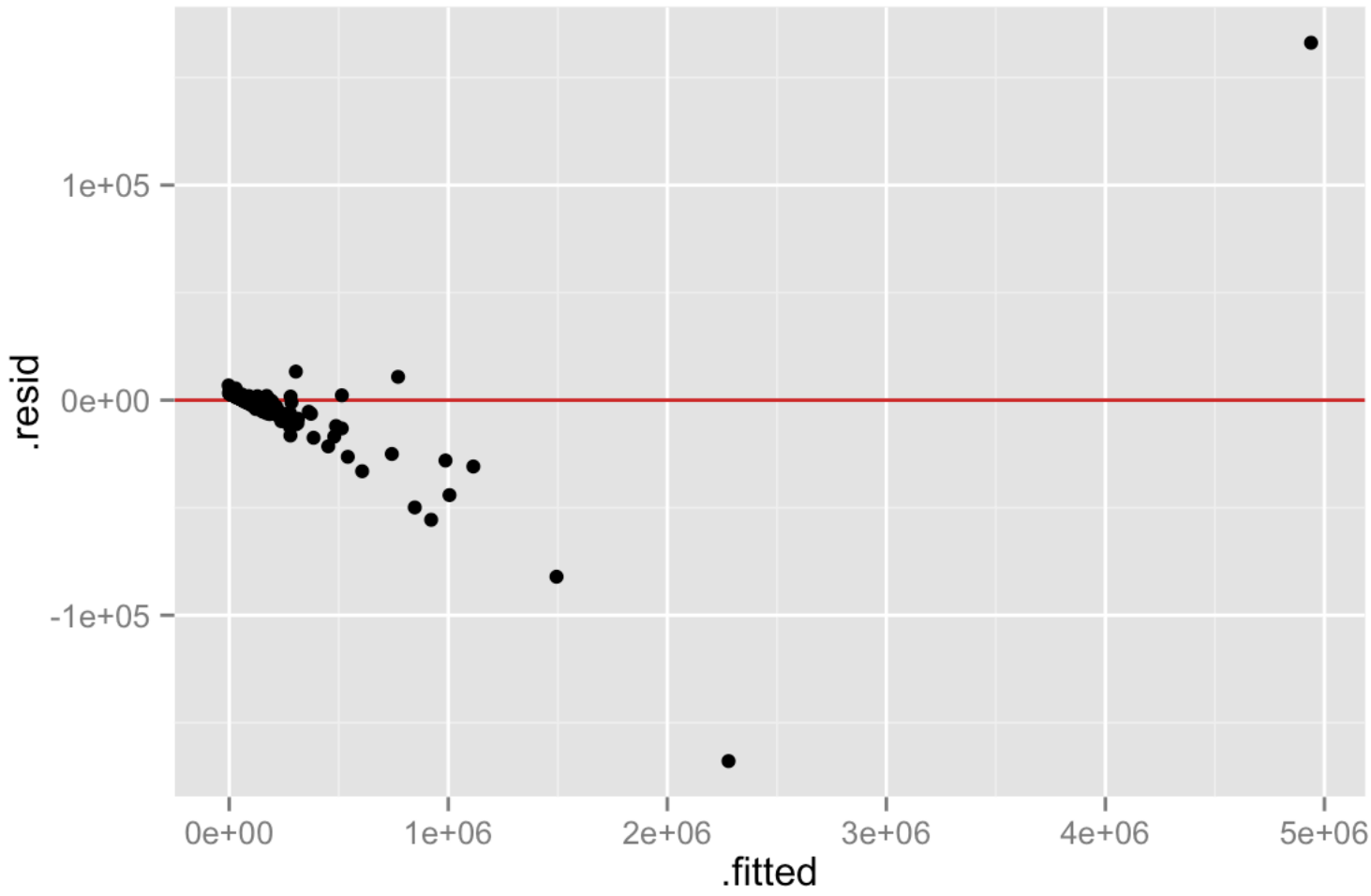





```
par(mfrow = c(1, 1))
```

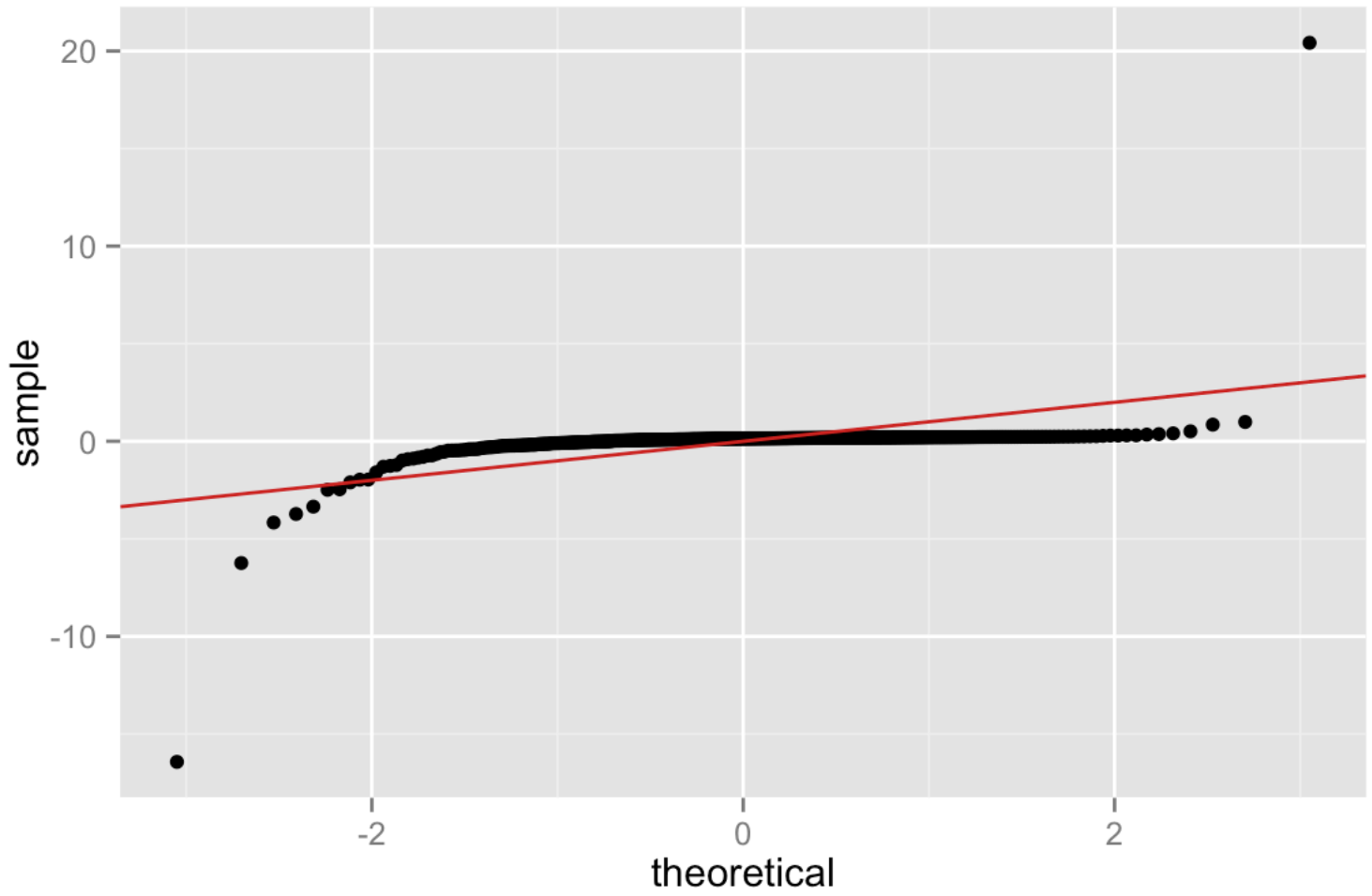
Homoscedasticity assumption

```
ggplot(data = myfortdata2, aes(x = .fitted, y = .resid)) +
  geom_hline(yintercept = 0, colour = "firebrick3") +
  geom_point()
```



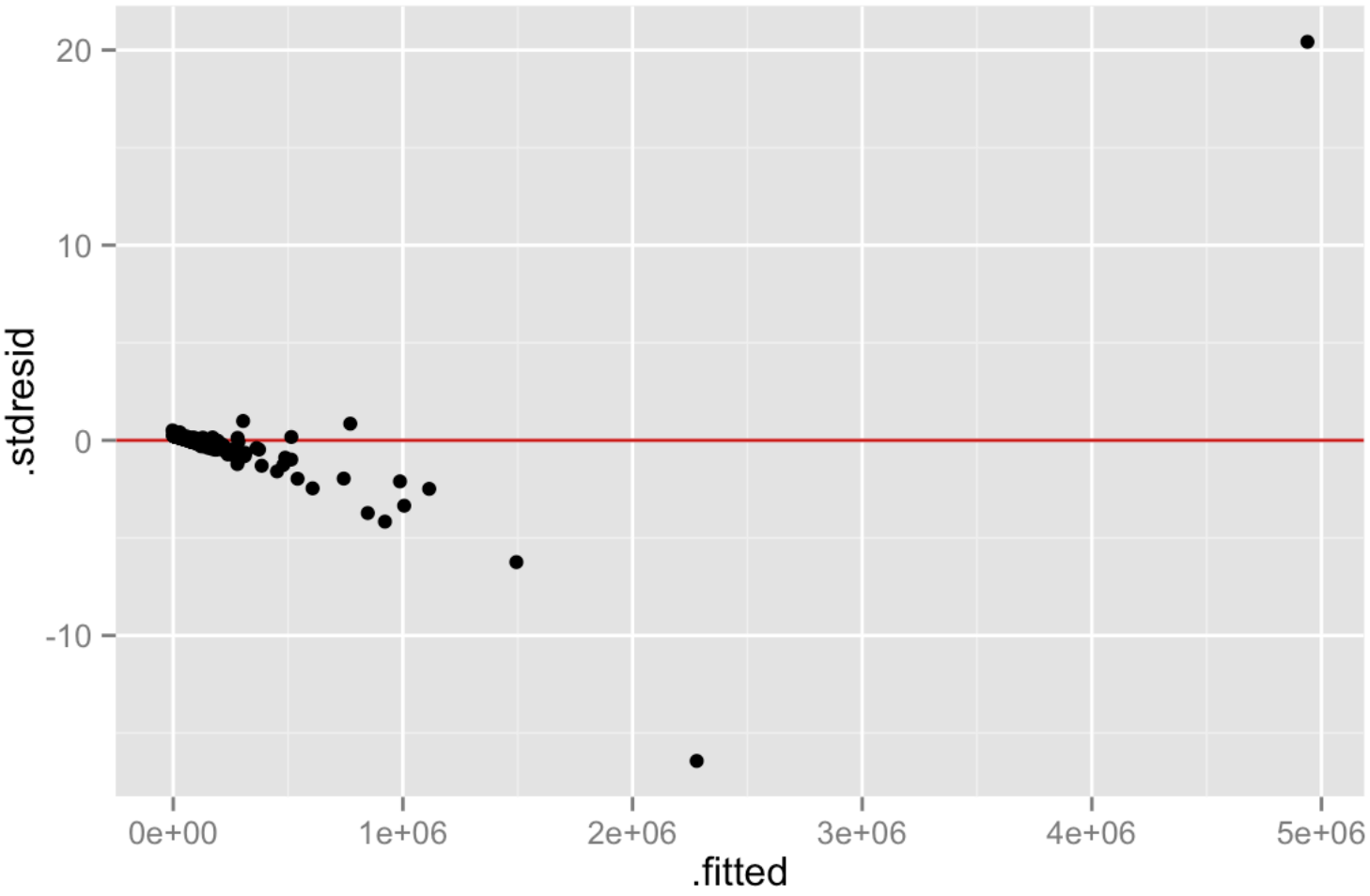
Normality assumption

```
ggplot(data = myfortdata2, aes(sample = .stdresid)) +
  stat_qq() +
  geom_abline(colour = "firebrick3")
```



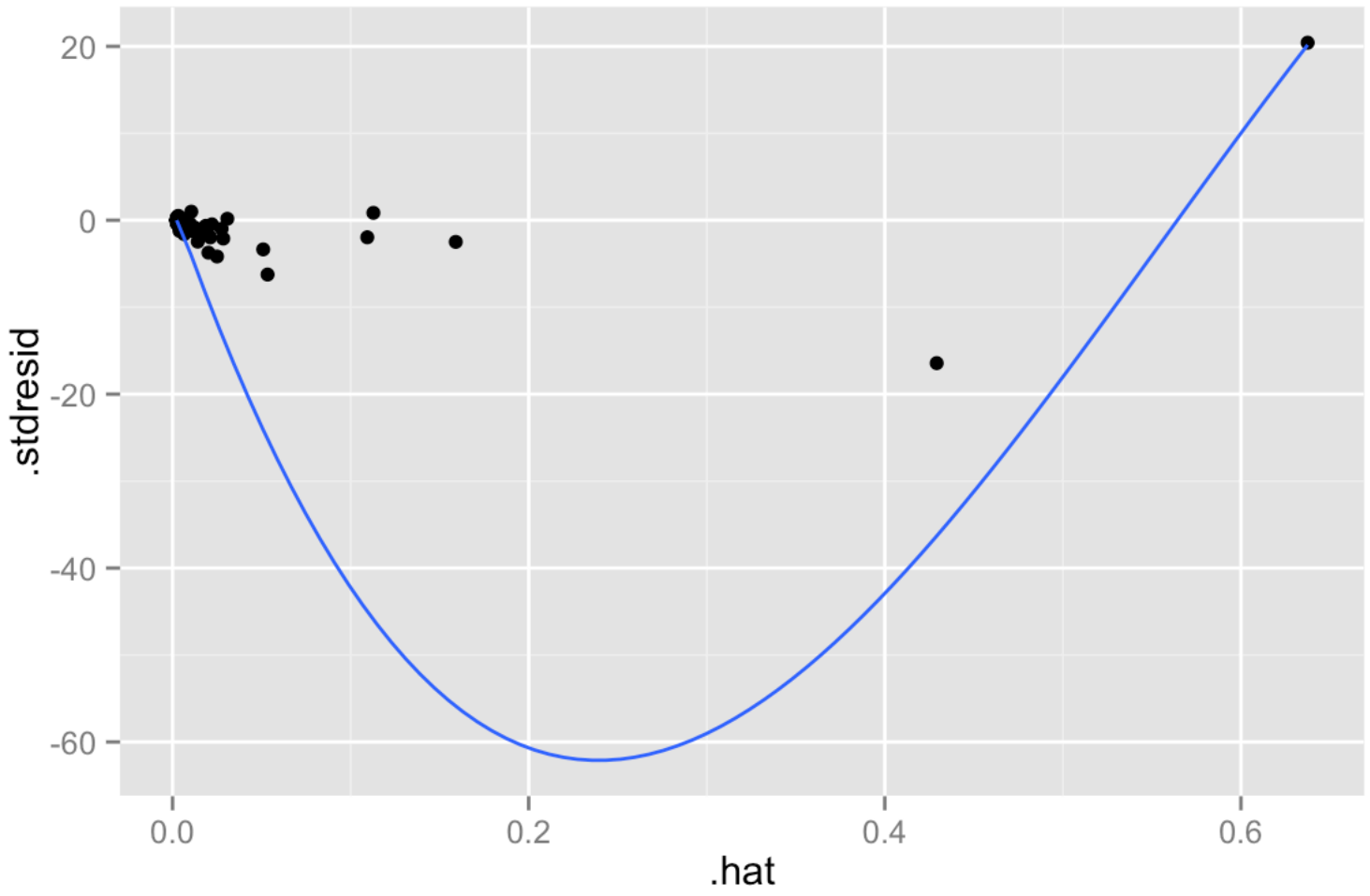
Standardized residuals vs fitted values

```
ggplot(data = myfortdata2, aes(x = .fitted, y = .stdresid)) +
  geom_hline(yintercept = 0, colour = "firebrick3") +
  geom_point()
```



Residuals vs. leverages

```
ggplot(data = myfortdata2, aes(x = .hat, y = .stdresid)) +
  geom_point() +
  geom_smooth(se = FALSE)
```



Next examples

In the previous examples we focused in plots to assess model assumptions once we have fitted the model. In the following two examples we will explore the data and assess linear relationships before fitting any model, specially for correlated data.

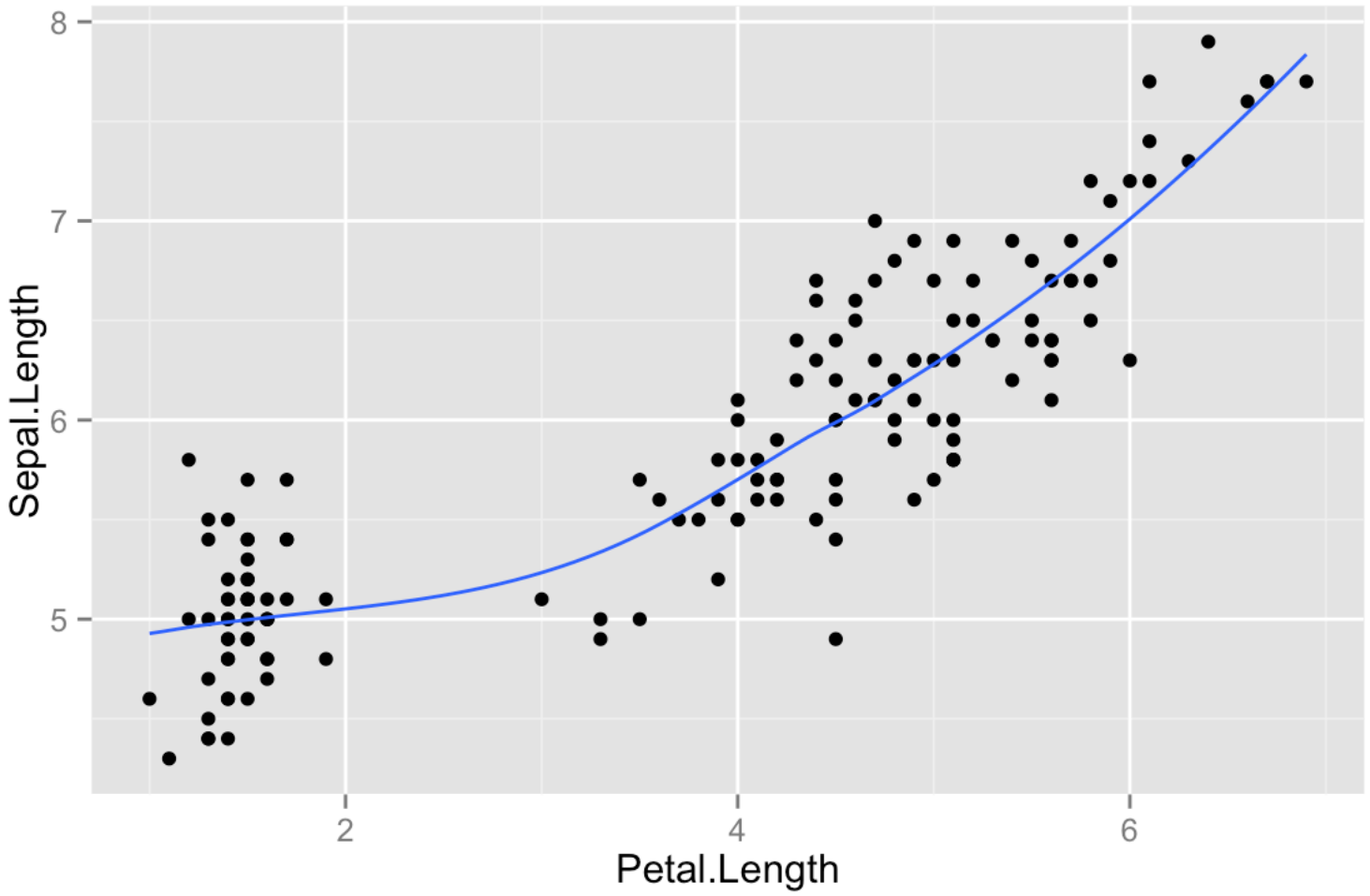
Example 3: iris data

```
data(iris)
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

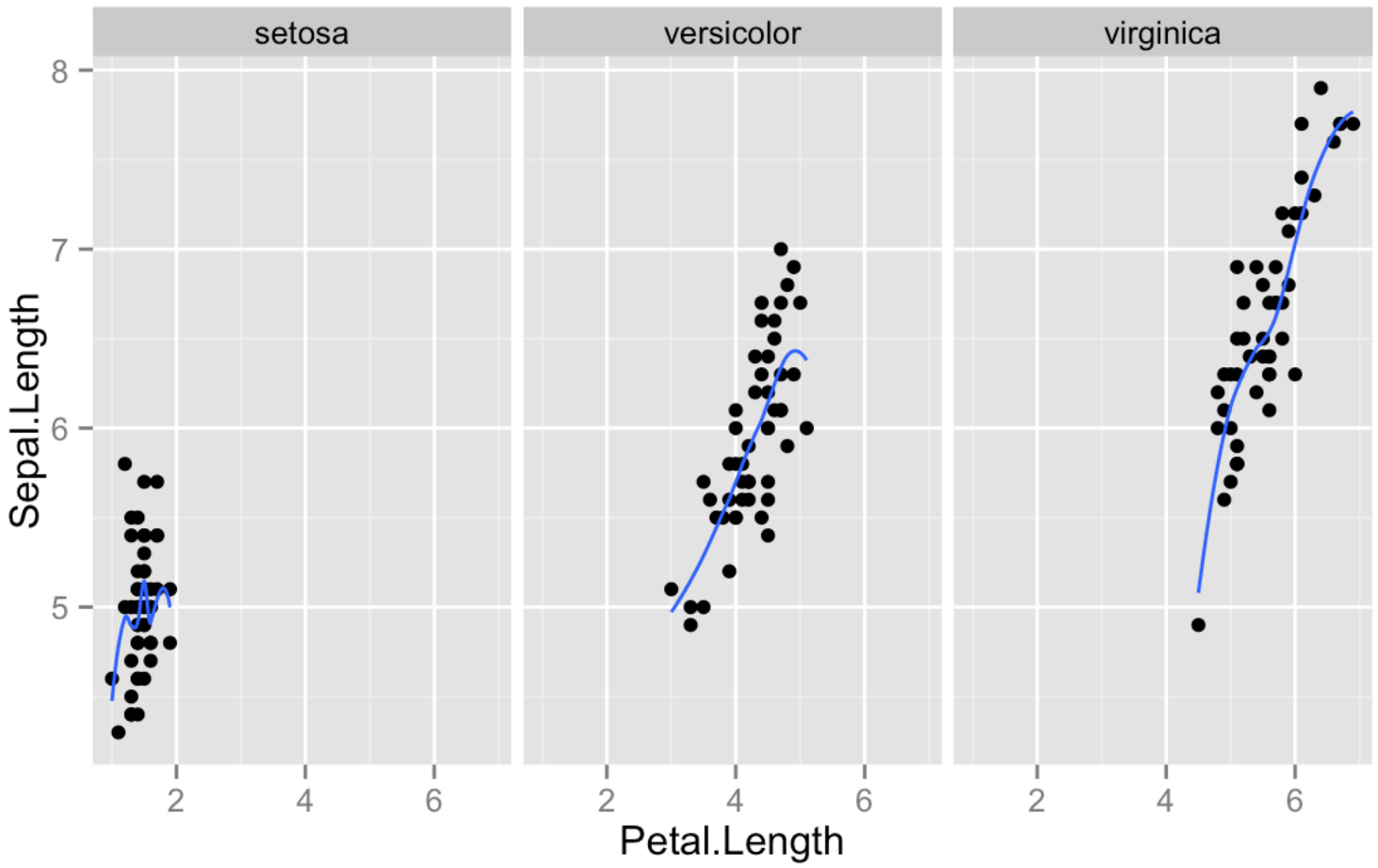
Explore your data

```
# Scatter plot and smoother
ggplot(data = iris, aes(x = Petal.Length, y = Sepal.Length)) +
  geom_point() +
  stat_smooth(method = "loess", se = FALSE)
```



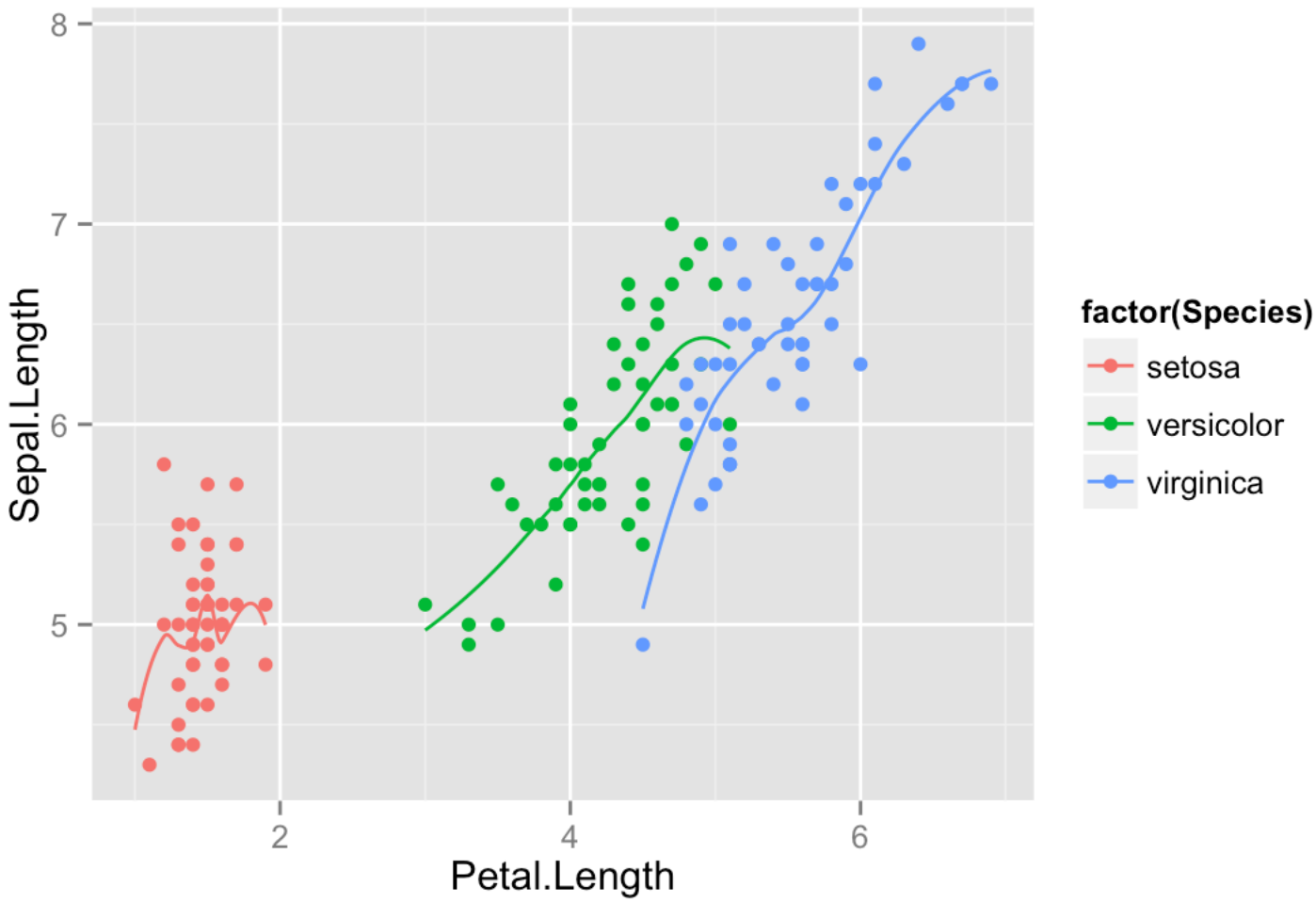
Explore your data

```
# Faceting by Species
ggplot(data = iris, aes(x = Petal.Length, y = Sepal.Length)) +
  geom_point() +
  stat_smooth(method = "loess", se = FALSE) +
  facet_wrap( ~ Species)
```



Explore your data

```
# Pattern within each group in the same graph
ggplot(data = iris, aes(x = Petal.Length, y = Sepal.Length, colour = facto
  geom_point() +
  stat_smooth(method = "loess", se = FALSE)
```



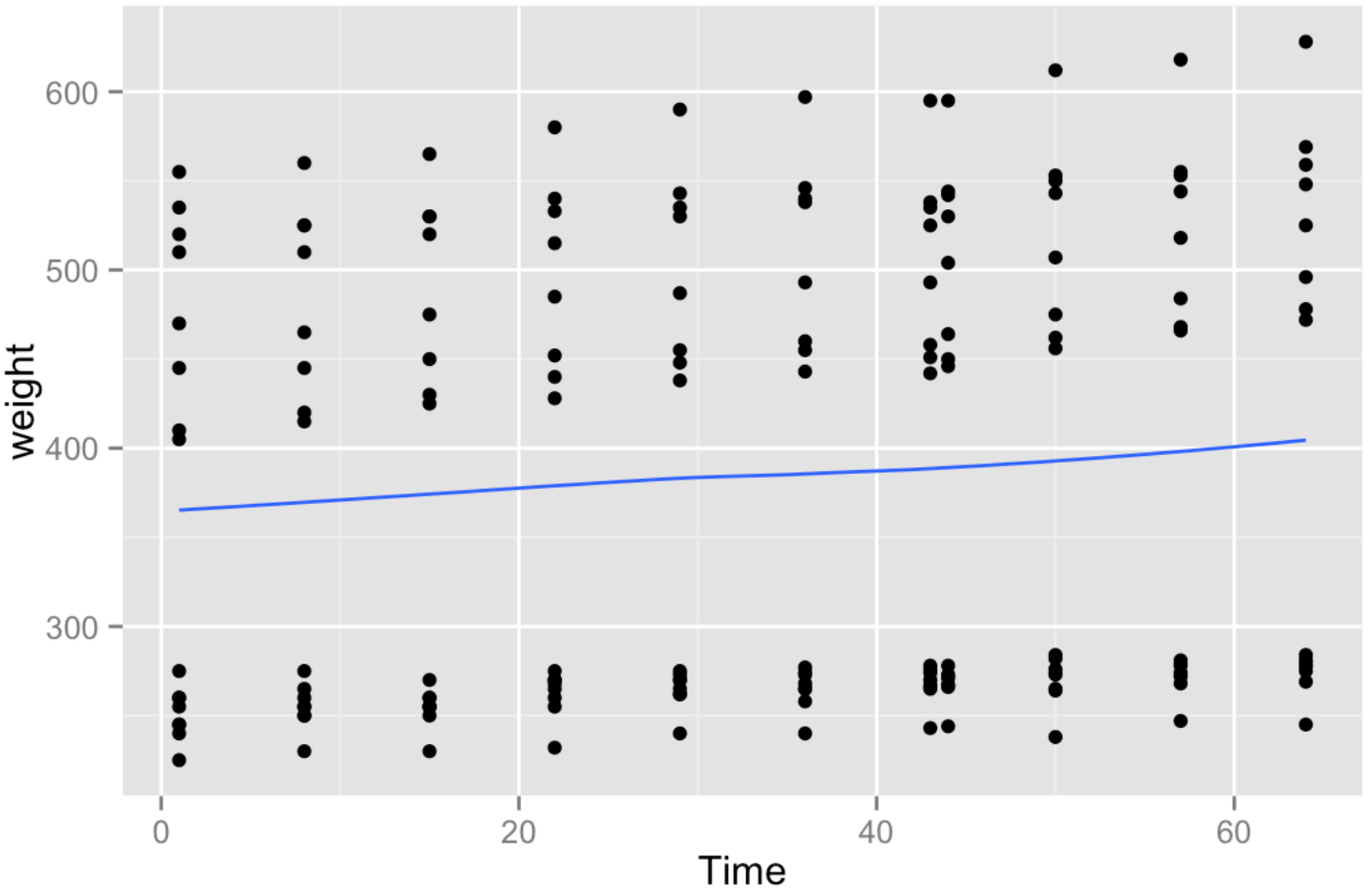
Example 4: BodyWeight data

```
library(nlme)
data(BodyWeight)
summary(BodyWeight)
```

```
##      weight           Time           Rat       Diet
##  Min.   :225.0   Min.   : 1.00   2      : 11   1:88
##  1st Qu.:267.0   1st Qu.:15.00   3      : 11   2:44
##  Median :344.5   Median :36.00   4      : 11   3:44
##  Mean   :384.5   Mean   :33.55   1      : 11
##  3rd Qu.:511.2   3rd Qu.:50.00   8      : 11
##  Max.   :628.0   Max.   :64.00   5      : 11
##                                  (Other):110
```

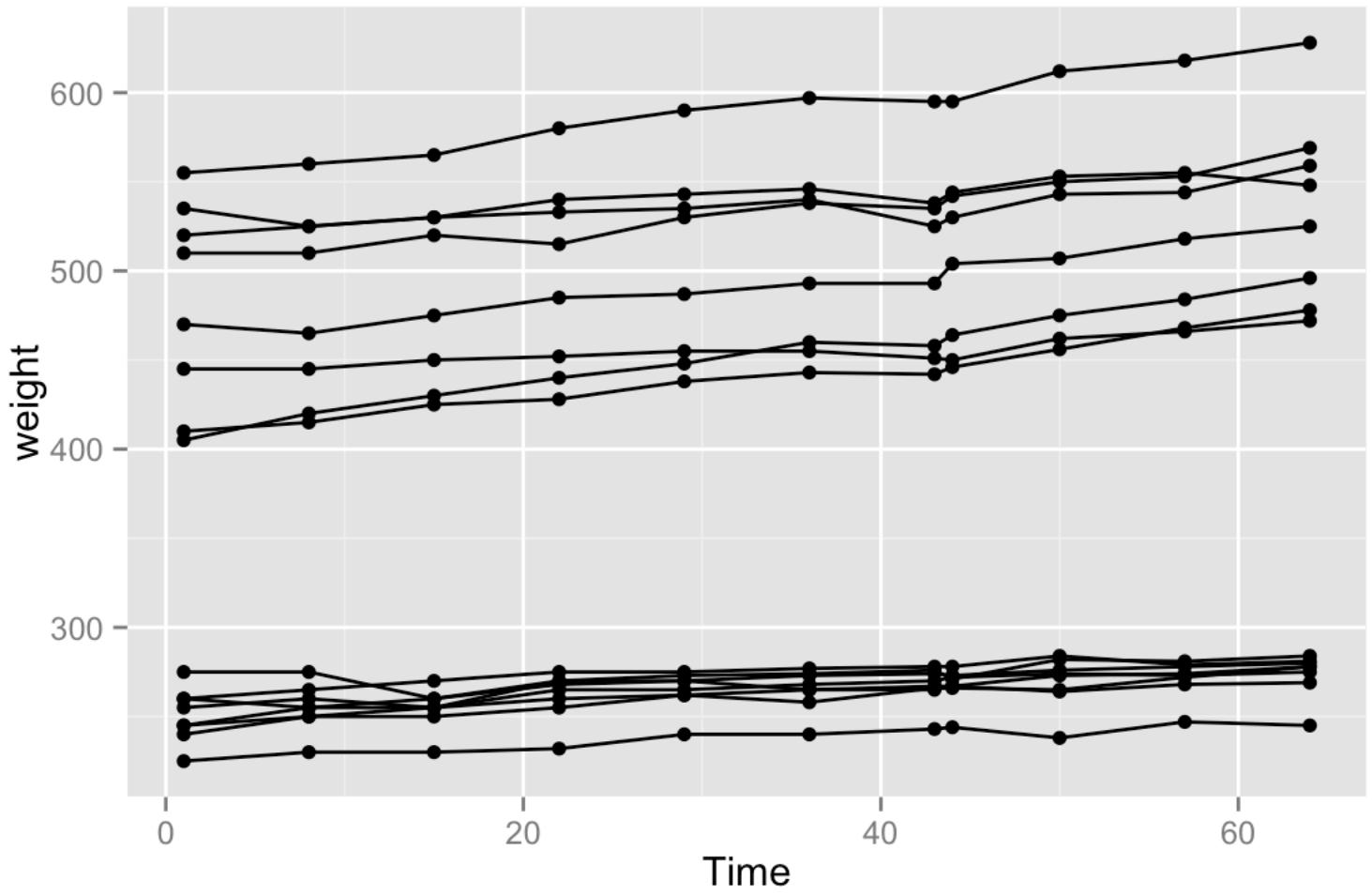
Explore your data

```
# Scatter plot and smoother
ggplot(data = BodyWeight, aes(x = Time, y = weight)) +
  geom_point() +
  stat_smooth(method = "loess", se = FALSE)
```



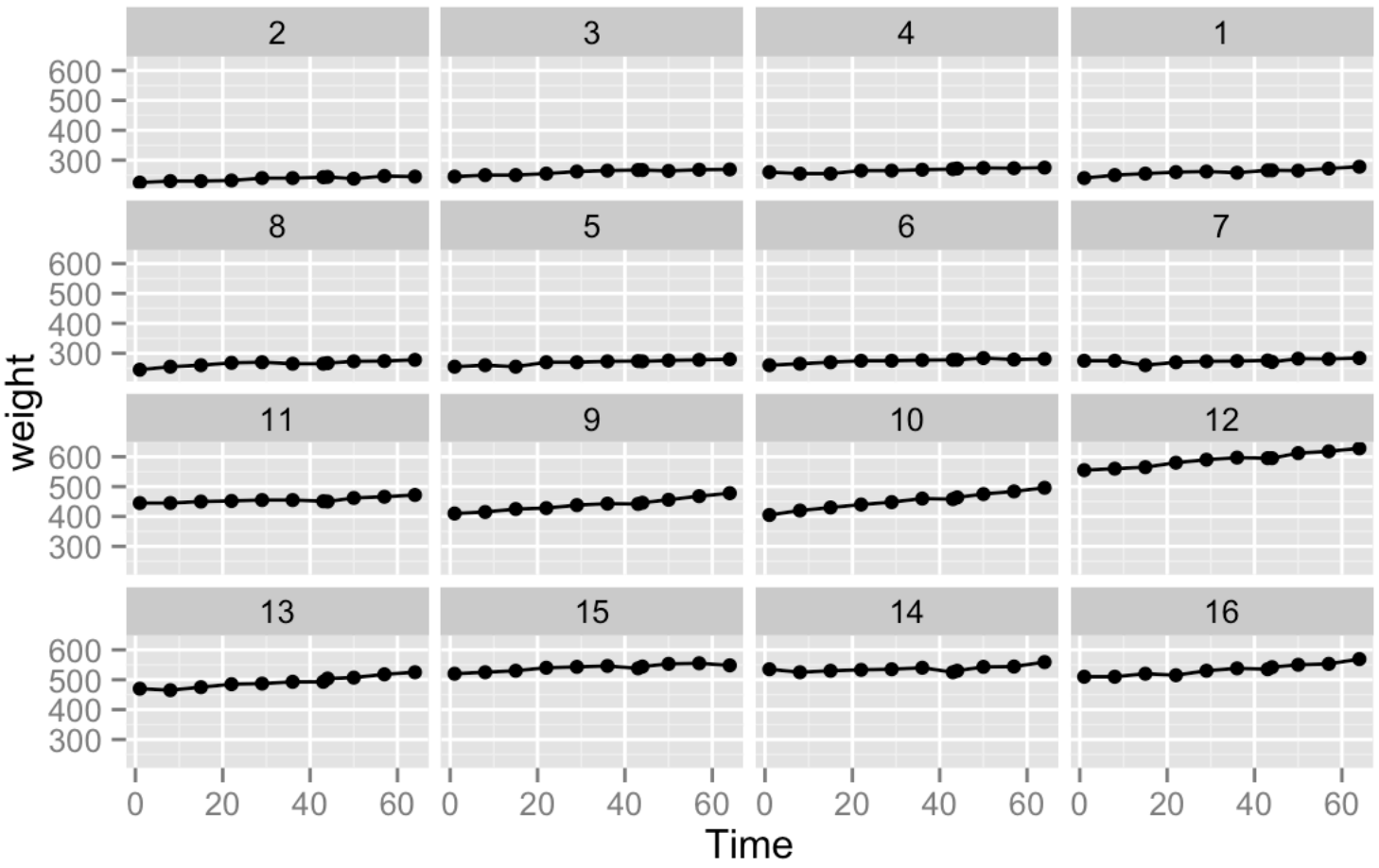
Explore your data

```
# Individual trajectories over time
ggplot(data = BodyWeight, aes(x = Time, y = weight, group = Rat)) +
  geom_point() +
  geom_line()
```



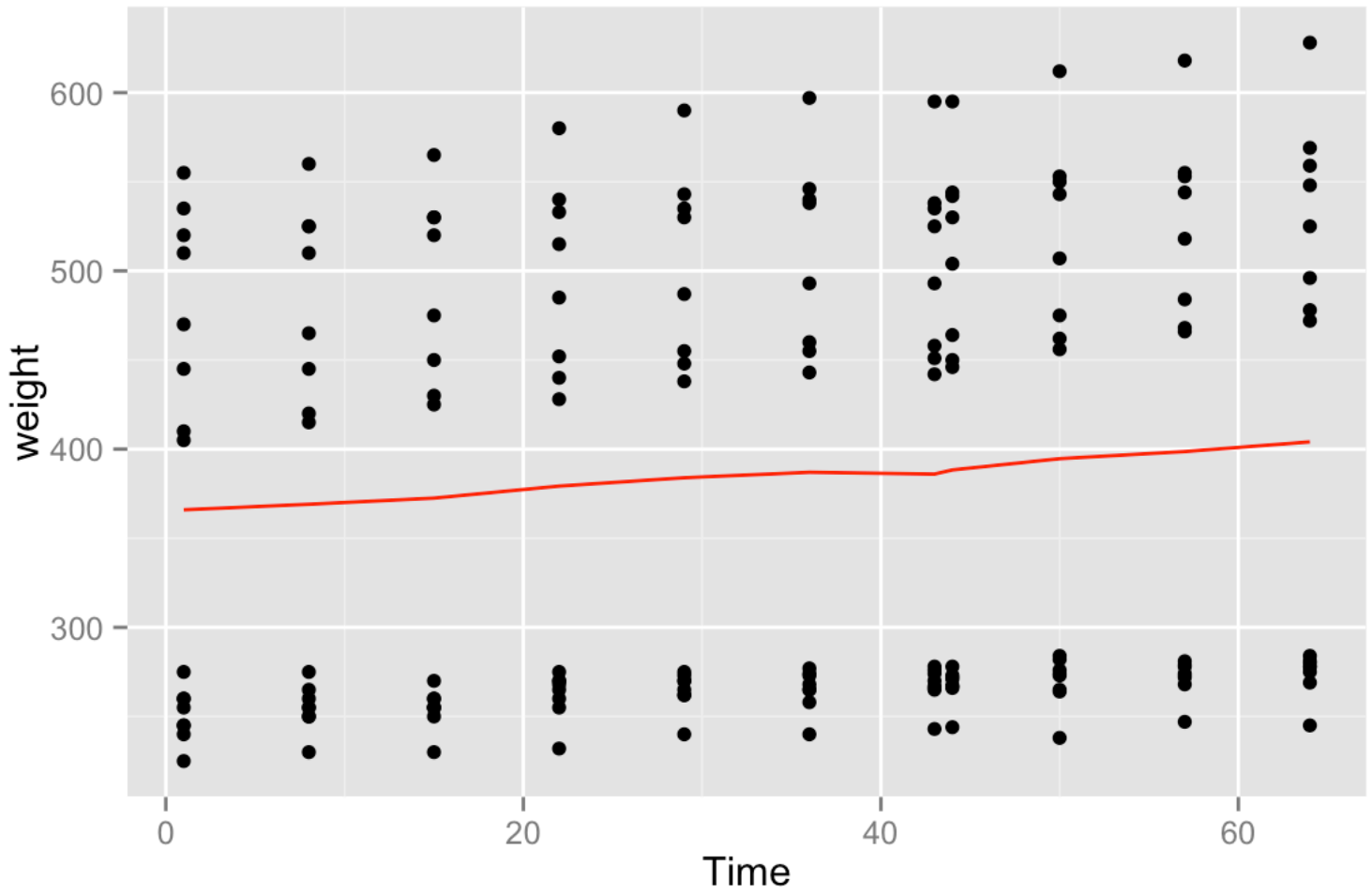
Explore your data

```
# Faceting by Rat
ggplot(data = BodyWeight, aes(x = Time, y = weight, group = Rat)) +
  geom_point() +
  geom_line() +
  facet_wrap( ~ Rat, ncol = 4)
```



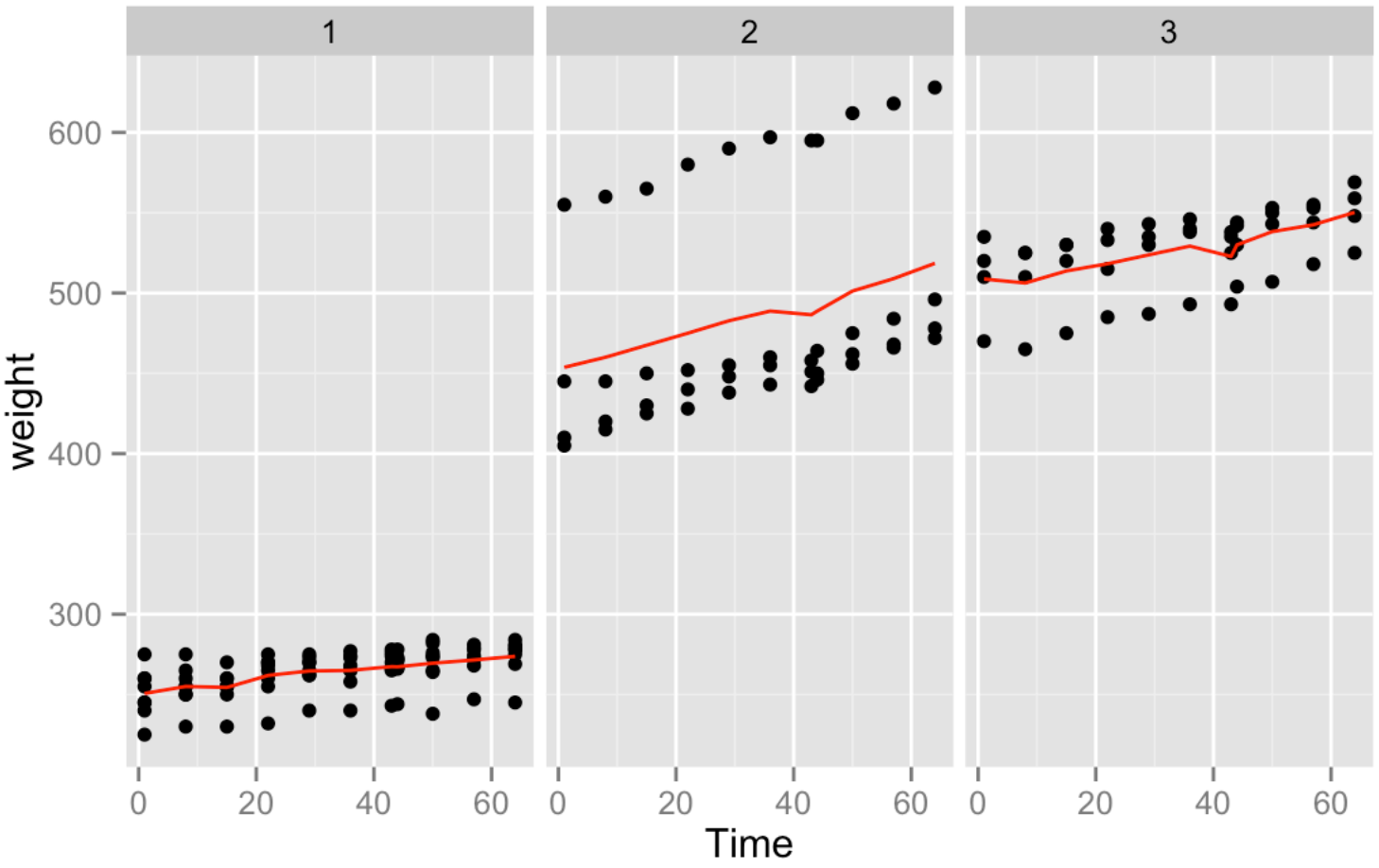
Explore your data

```
# Observed mean over time for all the data
ggplot(data = BodyWeight, aes(x = Time, y = weight)) +
  geom_point() +
  stat_summary(fun.y = mean, geom="line", colour = "red")
```



Explore your data

```
# Observed mean over time by Diet
ggplot(data = BodyWeight, aes(x = Time, y = weight)) +
  geom_point() +
  stat_summary(fun.y = mean, geom="line", colour = "red") +
  facet_wrap( ~ Diet)
```



Additional Resources

- Color options in ggplot
- Smooth options in ggplot
- Practical Regression and Anova using R by Julian J. Faraway