Linear regression exercise

Linear modeling

Q0: The dataset

A dataset called `trees` comes packaged with R and is always available in your workspace. This dataset contains measurements of felled timber from 31 cherry trees:

- `Height` (in feet)
- `Volume` (in cubic ft)
- `Girth` (actually it’s the tree diameter, measured 4’6" above the ground (in inches)

Check out the dataset to get a feel for what it looks like.

```
# inspect the dataset
head(trees)
```

```
##   Girth Height Volume
## 1   8.3     70   10.3
## 2   8.6     65   10.3
## 3   8.8     63   10.2
## 4  10.5     72   16.4
## 5  10.7     81   18.8
## 6  10.8     83   19.7
```

```
str(trees)
```

```
## 'data.frame':    31 obs. of  3 variables:
##  $ Girth : num  8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
##  $ Height: num  70 65 63 72 81 83 66 75 80 75 ...
##  $ Volume: num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
```

Q1: Create linear models of Volume vs. predictor variables

We are interested in predicting the tree Volume from the measured height and diameter. First, let’s create the simplest possible linear models, using one or both predictive variables.

Note: Remember that there are two equivalent expressions for specifying the formula: you can either refer to the columns of the columns directly using the `$` notation, or you can specify the data to be used with `data=` and just use the column names in the formula. The latter is a bit easier to type and to read.

```
## lm ######################################################################
# create linear models & check summaries

# Height
# tree_lm = lm(trees$Volume ~ trees$Height) # equivalent, harder to read
tree_lm1 = lm(Volume ~ Height, data = trees)
summary(tree_lm1)
```

```
##
## Call:
```

```
## lm(formula = Volume ~ Height, data = trees)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -21.274  -9.894  -2.894  12.068  29.852 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -87.1236    29.2731  -2.976 0.005835 ** 
## Height        1.5433     0.3839   4.021 0.000378 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 13.4 on 29 degrees of freedom
## Multiple R-squared:  0.3579, Adjusted R-squared:  0.3358 
## F-statistic: 16.16 on 1 and 29 DF,  p-value: 0.0003784
```

```r
# Girth
tree_lm2 = lm(Volume ~ Girth, data = trees)
summary(tree_lm2)
```

```
## 
## Call:
## lm(formula = Volume ~ Girth, data = trees)
## 
## Residuals:
##    Min     1Q Median     3Q    Max 
## -8.065 -3.107  0.152  3.495  9.587 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -36.9435     3.3651  -10.98 7.62e-12 ***
## Girth         5.0659     0.2474   20.48  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 4.252 on 29 degrees of freedom
## Multiple R-squared:  0.9353, Adjusted R-squared:  0.9331 
## F-statistic: 419.4 on 1 and 29 DF,  p-value: < 2.2e-16
```

```r
# Height + Girth
tree_lm3 = lm(Volume ~ Height + Girth, data = trees)
summary(tree_lm3)
```

```
## 
## Call:
## lm(formula = Volume ~ Height + Girth, data = trees)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -6.4065 -2.6493 -0.2876  2.2003  8.4847 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -57.9877     8.6382  -6.713 2.75e-07 ***
## Height        0.3393     0.1302   2.607   0.0145 *  
## Girth         4.7082     0.2643  17.816  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.882 on 28 degrees of freedom
## Multiple R-squared:  0.948,  Adjusted R-squared:  0.9442
## F-statistic:   255 on 2 and 28 DF,  p-value: < 2.2e-16
```

What do the models tell you about how much of the variation in Volume are explained by Height and Girth? You should consider the following in your evaluation:

- Residuals – How big are they?
- Coefficients –
- R-squared – How much of the variation in the data are explained by the Explanatory variable? Does this look encouraging?

Which model seems best at first glance, and why? Is there anything troubling about these models? Why or why not?

Based on the statistics from the linear regression, how would you evaluate the goodness-of-fit? Do the data violate any of the basic assumptions for linear modeling?

Q2: Simple plots of Response vs. Explanatory variables

Before deciding upon a model, we need to check whether any of the basic assumptions of linear models are violated. What are these assumptions?

Let's draw some simple plots to get an intial feel for how well the models fit the assumptions of linear models.

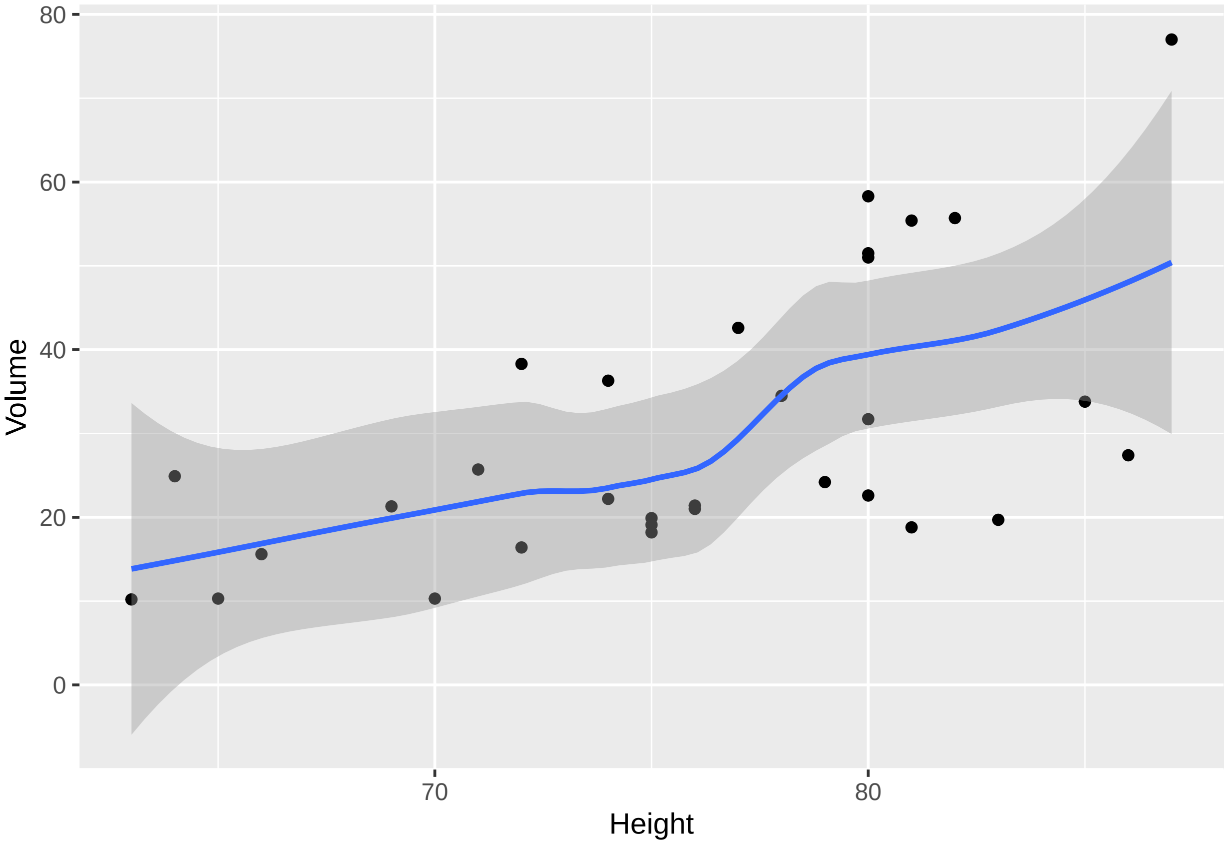
First, explore the data by superimposing a smoothened line over the datapoints using `geom_smooth()`.

Note: The default method for smoothing is called `LOESS`, which stands for "locally estimated scatterplot smoothing". This method uses locally weighted least-squares across a sliding window to compute the best fit for each datapoint. The "smoothing parameter", a.k.a. the "bandwidth", determines how much of the data is used at each step.

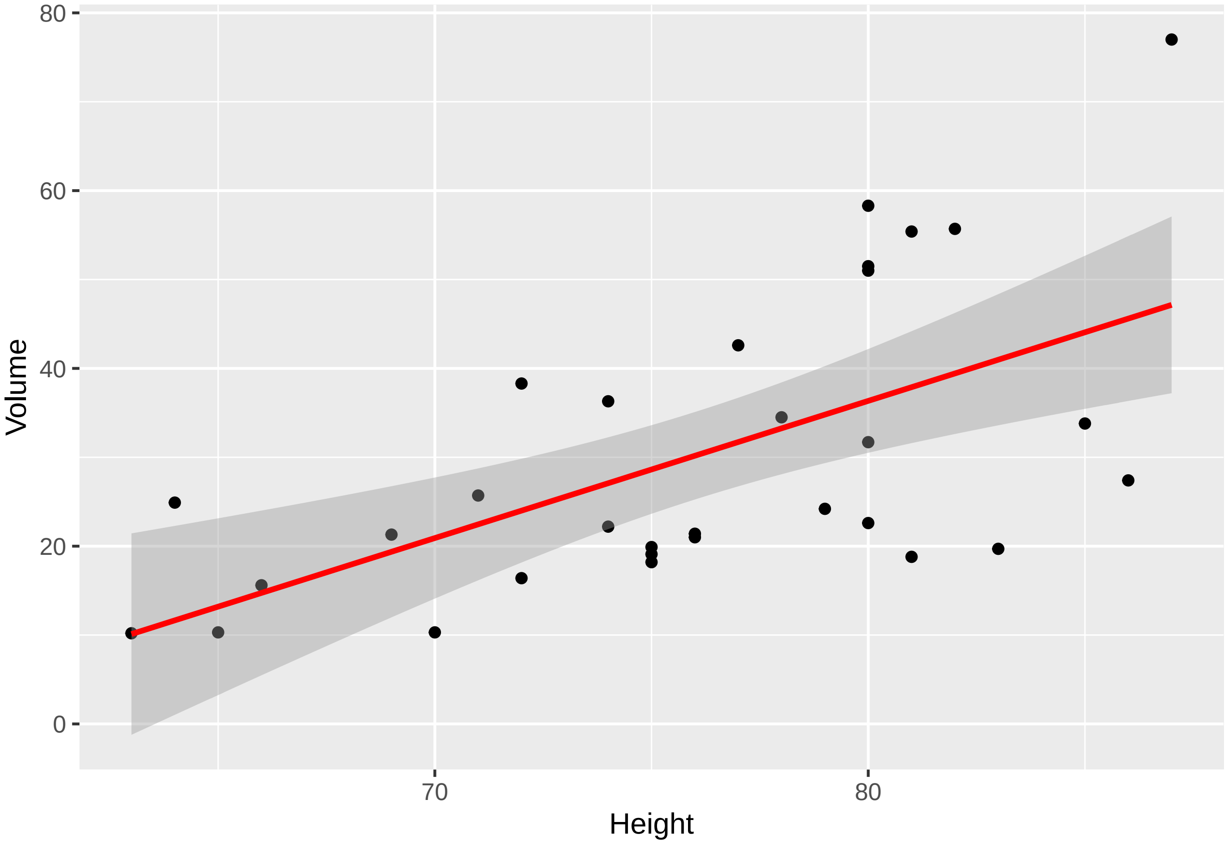
Next, remake the plots to show the best-fit regression line using `stat_smooth()` with the "lm" method. This will automatically show the 95% CI around the regression line.

```
# load the ggplot library
library(ggplot2)

################################################################################
## 1) create a plot of Volume ~ Height using ggplot

# a) simple plot -- show the data points + a smoothened line
ggplot(trees, aes(x=Height, y=Volume))+
    geom_point() +
    geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

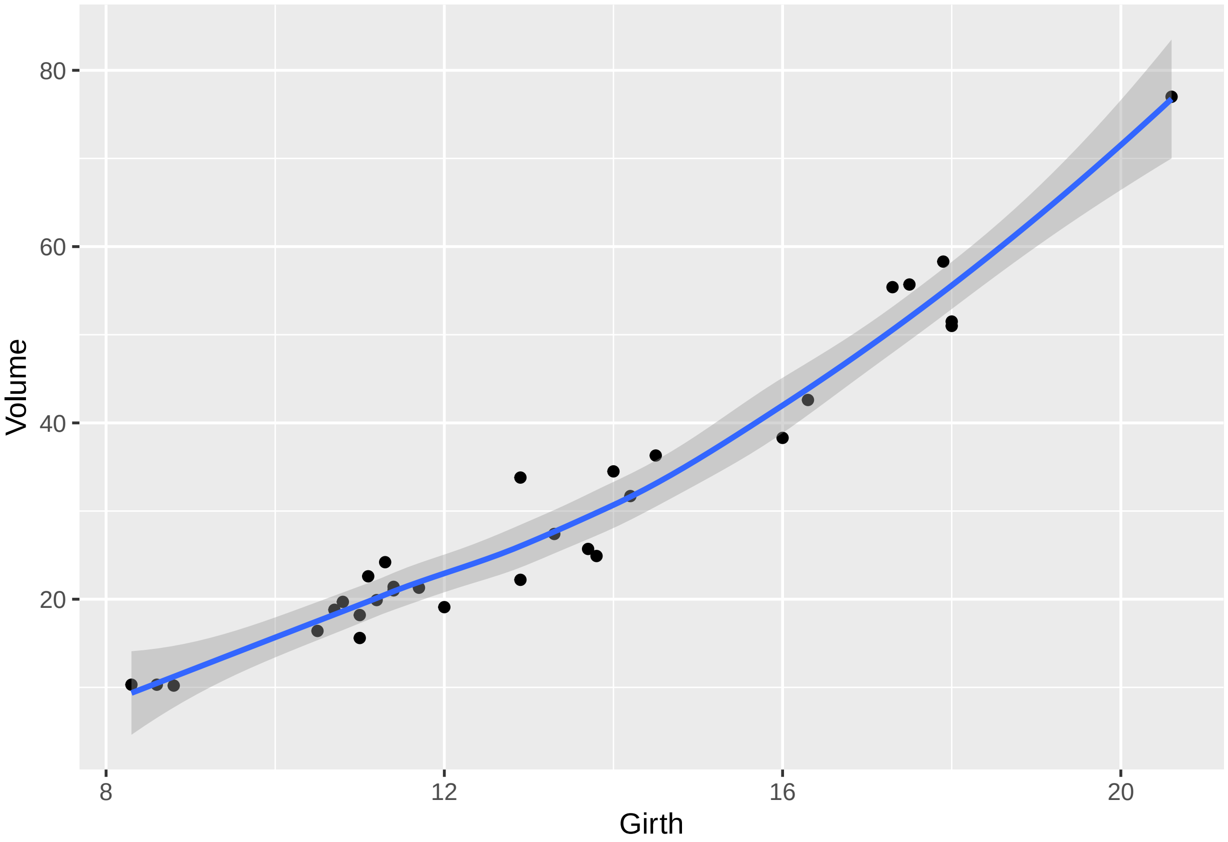


```r
# b) plot datapoints with linear regression line, in RED
ggplot(trees, aes(x=Height, y=Volume))+
    geom_point() +
    stat_smooth(method="lm", col="red")
```

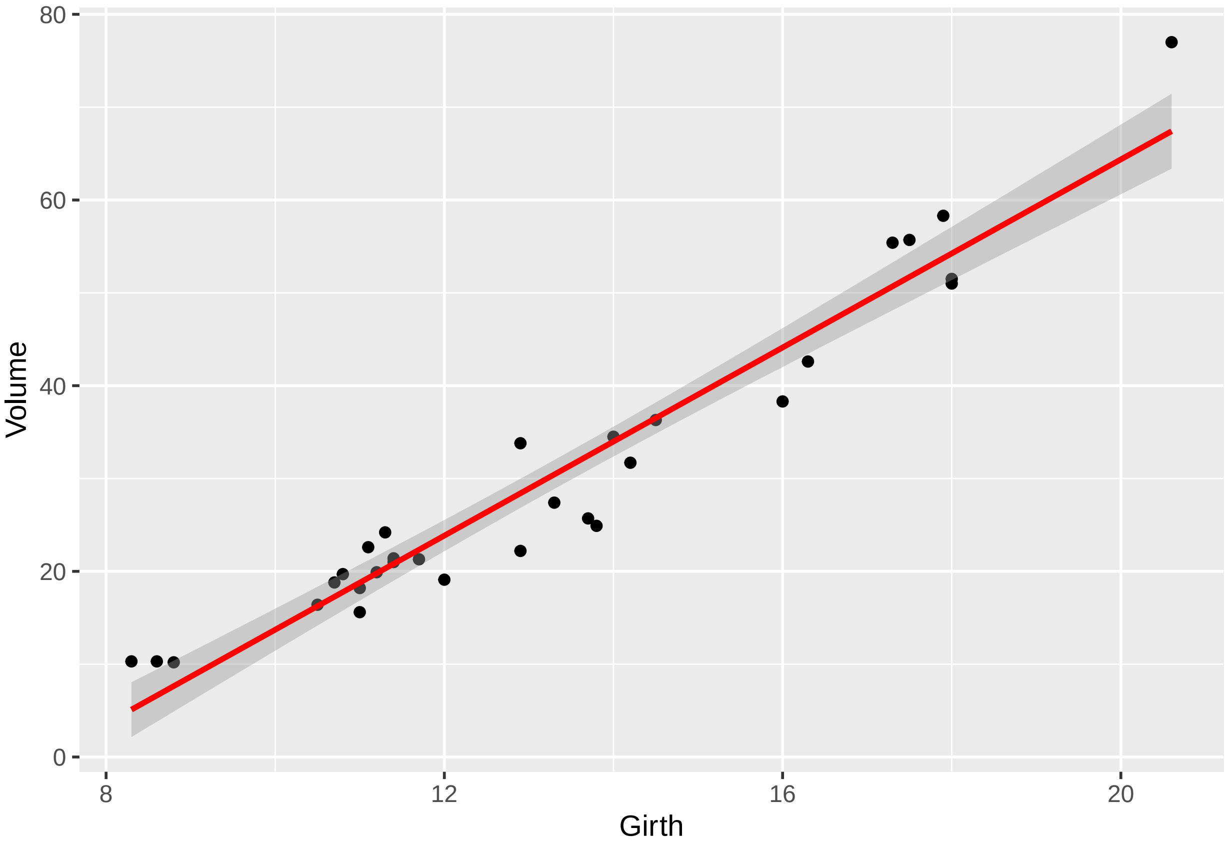


```
################################################################################
## 2) plot Volume ~ Girth

# a) simple plot -- datapoints and smoothened line
ggplot(trees, aes(x=Girth, y=Volume))+
    geom_point() +
    geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



```
# b) plot datapoints with linear regression line, in RED
ggplot(trees, aes(x=Girth, y=Volume))+
    geom_point() +
    stat_smooth(method="lm", col="red")
```



How do these plots look to you? Do you notice anything funny about the LOESS curve for Volume vs. Girth?

Q3: Diagnostic residual plots

You can assess the quality of any linear model using a variety of plots that show different properties of the **residuals** (the distance from each data point to the fitted line).

Types of Diagnostic Plots

Residuals vs. Fitted

This plot looks at the distance of the data points to the predicted line. There are several things to look for on this plot:

- **Small residuals?** Ideally, the points should be close to the dotted line (where the residual is 0). Small residuals reflect a good fit to the data.
- **Uniform residuals?** The magnitude of the residuals should be about the same across the range of measured x-values (equal variance).
- **Outliers?** Outliers are numbered accoring to their index in the data frame to call attention to them.
- **Linear relationship?** If the relationship is linear, then the fitted line should be relatively straight. If this is not the case, then applying some transformations to the predictor and/or response variables may help make the data more linear.

Normal QQ plot of residuals

This plot is a quantile-quantile plot that shows whether the Residuals are normally distributed. Again, the best case scenario is that the points fall on the dashed line.

Scale-location plot

This plot shows the **variance** in the residuals to the regression line. Ideally, the line should be horizontal, indicating that the variance is evenly distributed across all the datapoints. **Homoscedasticity**, or equal variance, is an assumption of certain statistical tests.

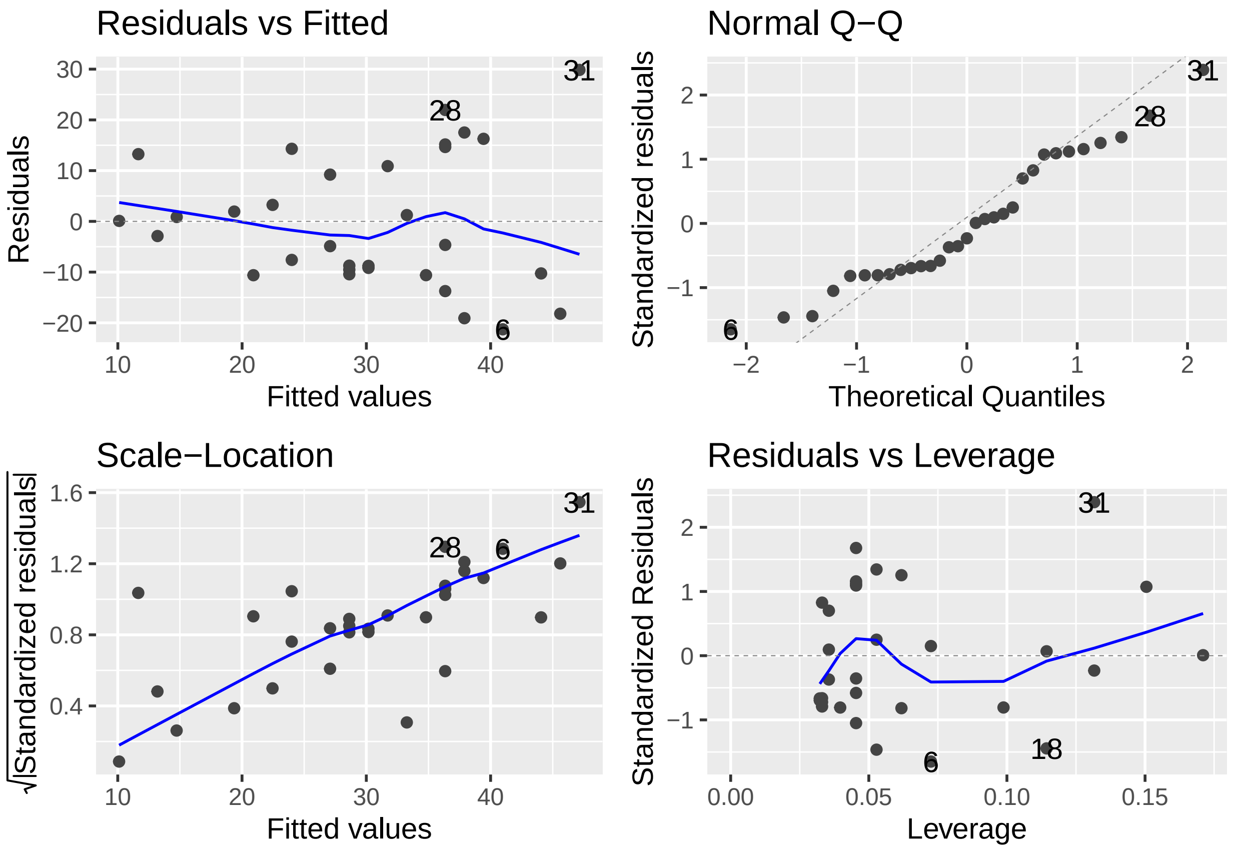
Residuals vs. Leverage

Cook's distance is calculated by removing a specific data point and checking how much the model has changed. The greater the Cook's distance, the more influential that point is on the model. The dashed horizontal line corresponds to the ideal situation in which each datapoint has little effect on the overall model. Outliers can be very disruptive, so this is a good way to determine if there are indeed outliers that have a strong negative impact on the linear model.

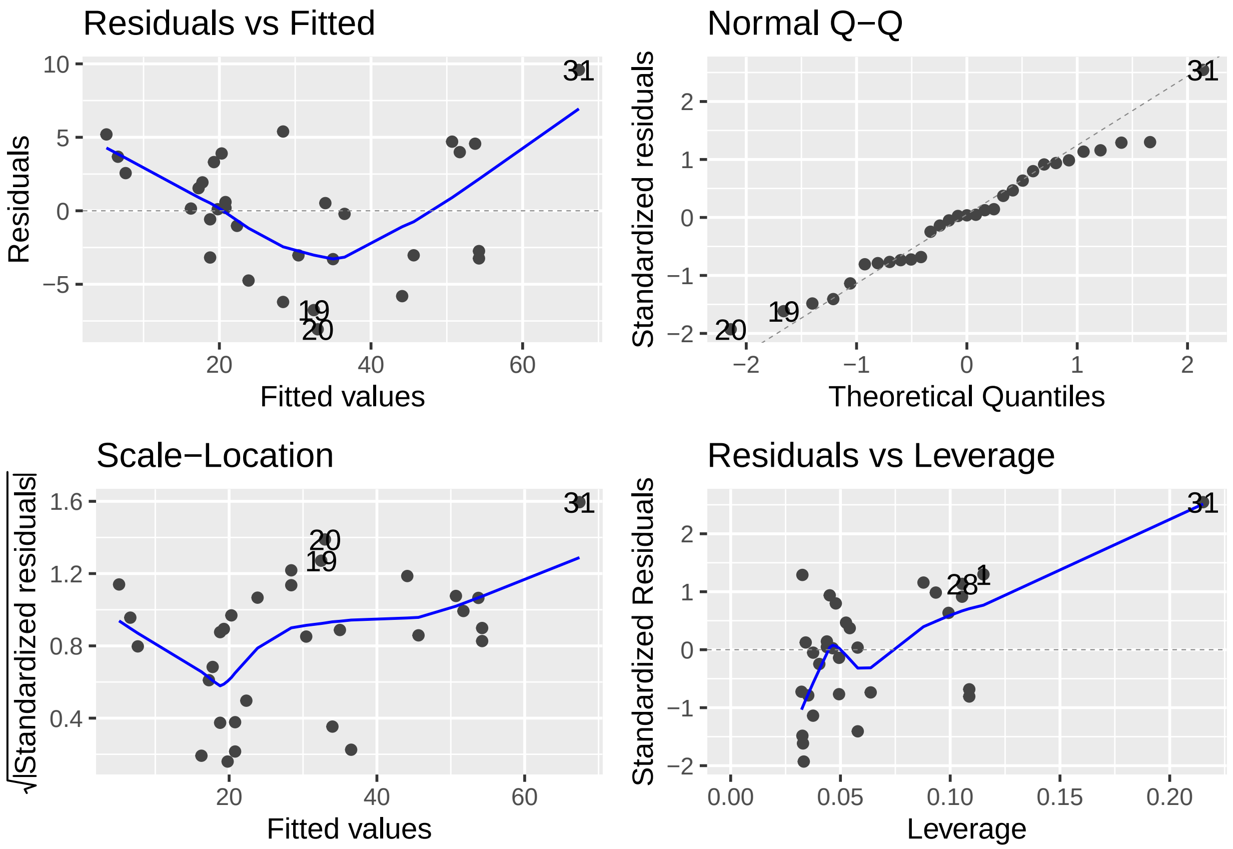
Examine the residuals plots

Here we will use a `ggplot2` companion package called `ggfortify`, which takes the normal base R graphics and converts them into ggplot graphs. This package defines the function `autoplot()`, which automatically creates one or more standard plots that are tailored to specific object classes. Here we will apply `autoplot` to our alternative linear models.

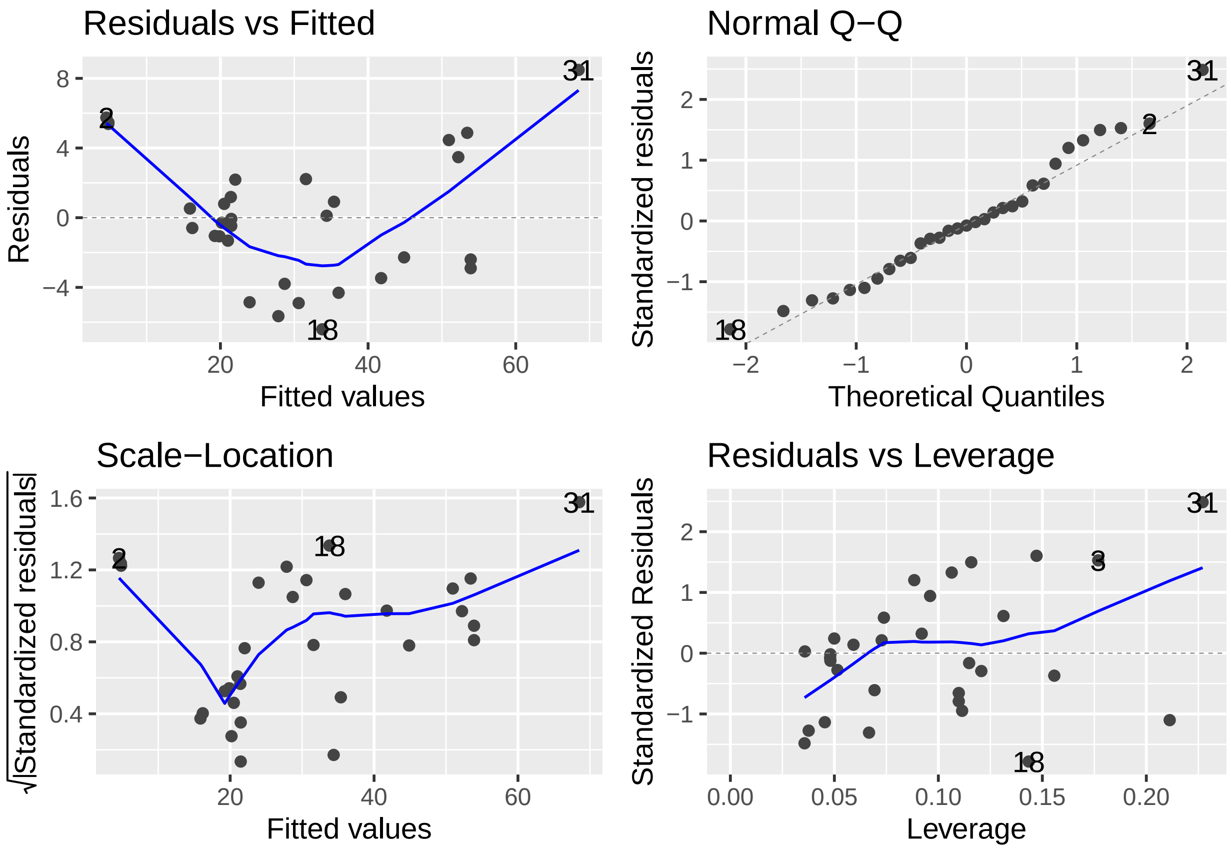
```r
# load the ggfortigy library
# install.packages("ggfortify") -- uncomment if you haven't already loaded the package
library(ggfortify)

# use autoplot to check on the different models
autoplot(tree_lm1)
```



```r
autoplot(tree_lm2)
```



```r
autoplot(tree_lm3)
```



Examine the plots for the three models and discuss the following:

Model 1: Volume ~ Height

Comment specifically on the uniformity of variation as revealed by the Residuals vs. Fitted and Scale-Location plots. What do the patterns reveal?

Models 2 & 3: Volume ~ Girth and Volume ~ Height + Girth

Comment specifically on the Residuals vs. Fitted plot. What shape is the LOESS line? Can you think of any functions that resemble this pattern?

Also consider the Scale-Location and Residuals vs. Leverage plots. What do you learn from these?

Finally, discuss what you think the main dependencies in this dataset are, i.e. how influential each predictor is. The way you think about this will determine the downstream analyses that you will perform.

Q4: Data transformation

The above plots reveal that linear models of the raw data might not provide the best picture of the relationship of Height and Girth with Volume.

Reflecting upon the exercises above, and *your own knowledge of volumetric measurements*, can you think of a way to transform the data that might improve the model? Explain your reasoning. If you have a particular mathematical formula in mind, please include it in your answer.

Hint: You may want to play around with a few transformations on your own to see what might work best (do not include your exploratory analysis here). Since this is not a trick question, stick to common transformations like log / power functions and quadratics (square / square root). You can also just answer this question based on first principles.

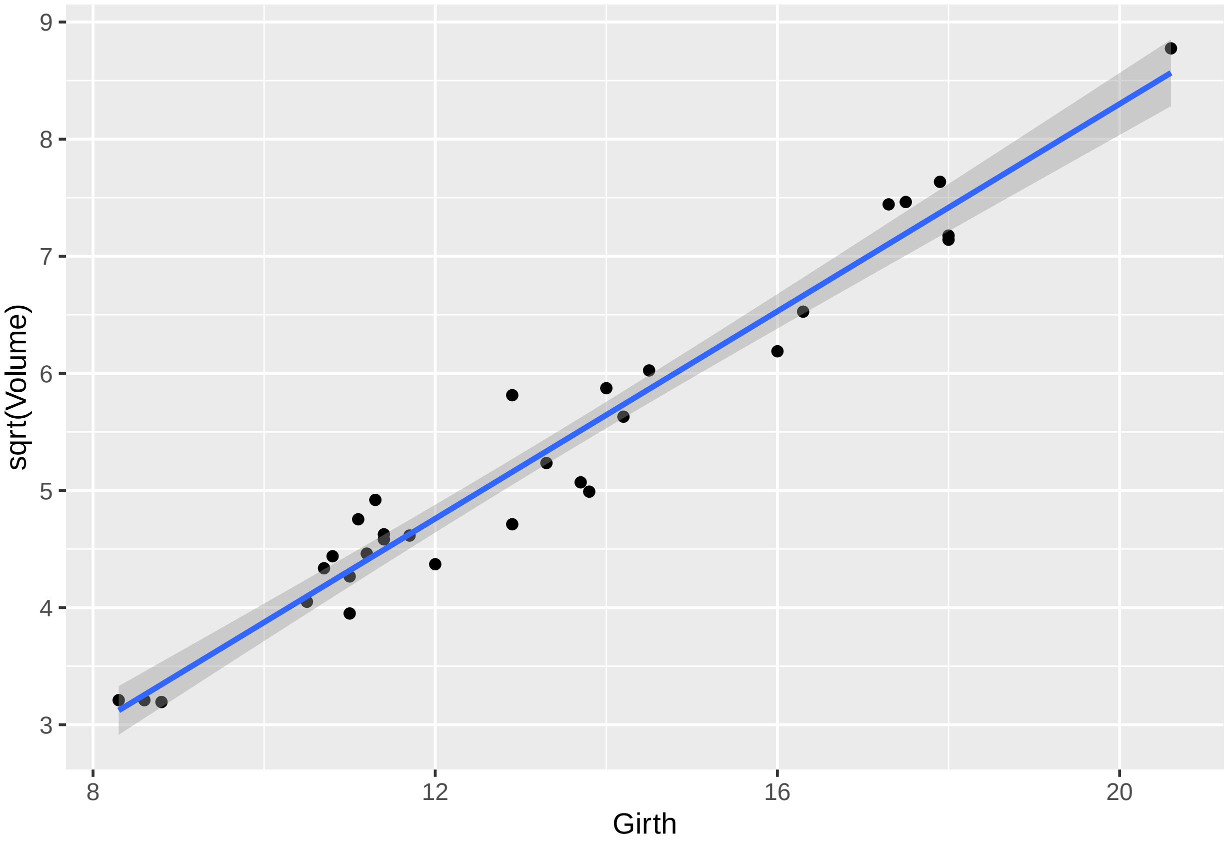
Below, create a linear model using the transformation you think works best. Since Height made little contribution to the model, just use Girth as the Predictor variable.

Note: Please transform the Response variable rather than the Predictor variable(s) in the linear model. This will allow you to predict Volume from any set of new measured values for Girth, which you will be asked to do below. Making predictions on new data, based on models generated from training data, is a common task in data analysis.

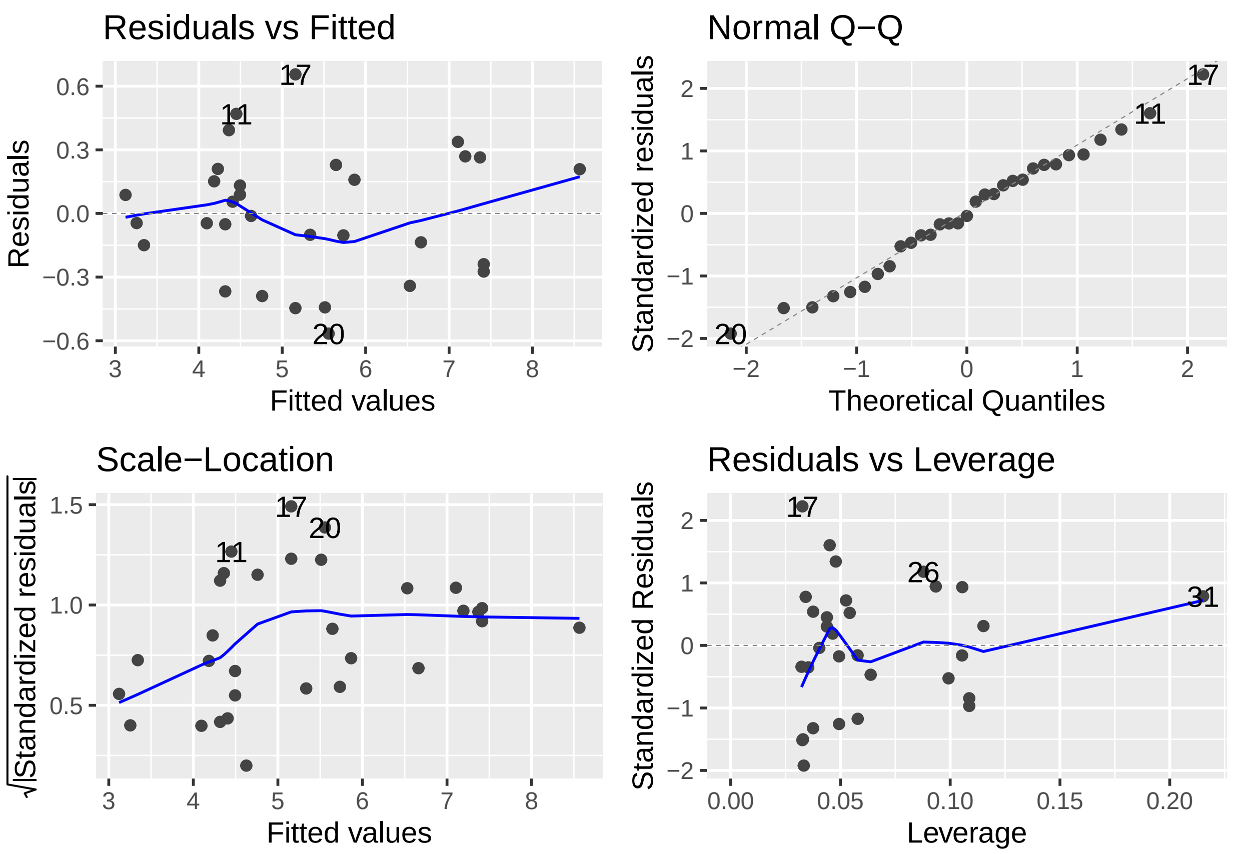
```
# make a linear model and look at the summary
model = lm(sqrt(Volume) ~ Girth, data = trees)
summary(model)
```

```
##
## Call:
## lm(formula = sqrt(Volume) ~ Girth, data = trees)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.56640 -0.19429 -0.01169  0.20934  0.65575
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.55183    0.23719  -2.327   0.0272 *
## Girth        0.44262    0.01744  25.385   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2997 on 29 degrees of freedom
## Multiple R-squared:  0.9569, Adjusted R-squared:  0.9555
## F-statistic: 644.4 on 1 and 29 DF,  p-value: < 2.2e-16
```

```
# use ggplot to make a plot of the transformed data
# using `stat_smooth` to show the regression line
ggplot(trees, aes(x=Girth, y=sqrt(Volume))) +
    geom_point() +
    stat_smooth(method="lm")
```



```
# check out the diagnostic plots for the residuals
autoplot(model)
```



How have your stats and diagnostic plots changed? What features of the particular transformation you have chosen do you think correct the problems you saw before?

Q5: Prediction and Confidence Intervals

First, apply the `predict.lm()` function to your best linear model to predict the Volume across a range of values for Girth. Read the documentation for this function, and include lower and upper *Prediction Intervals* in the model.

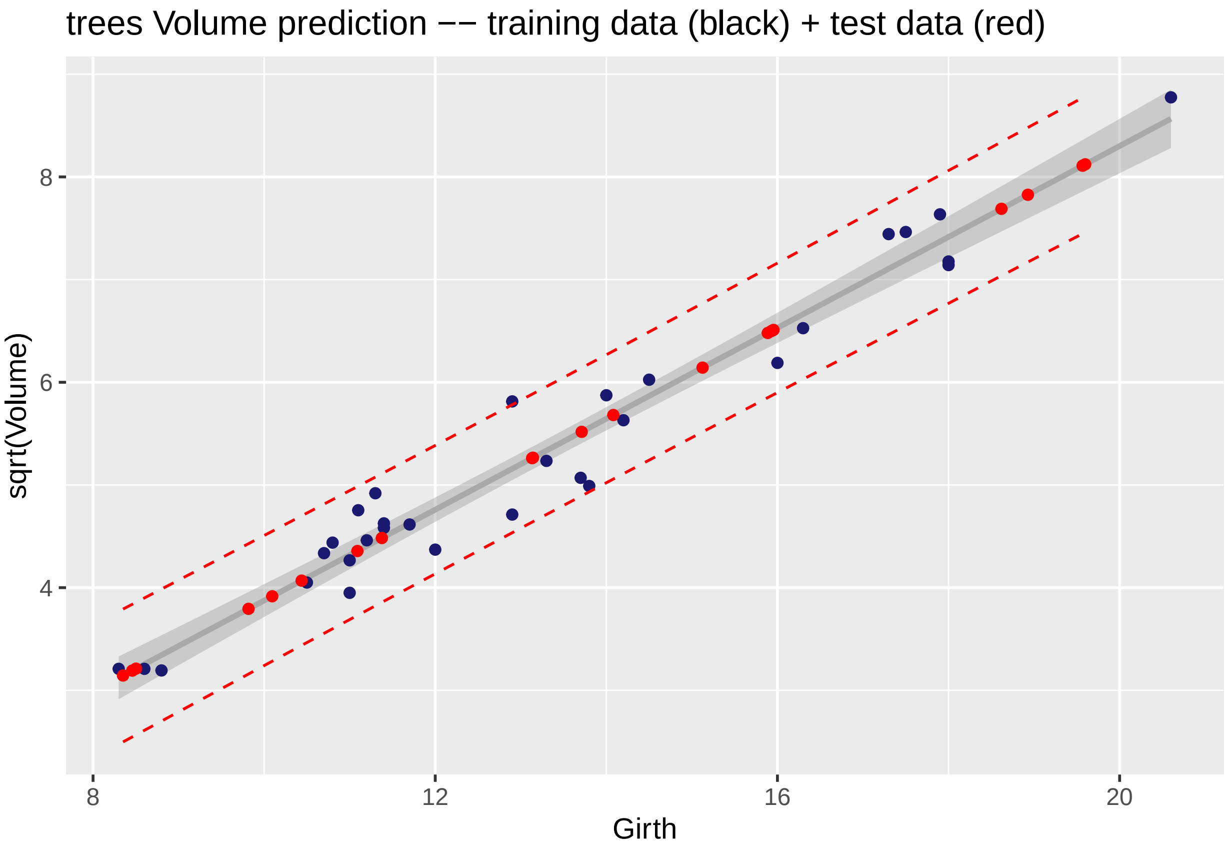
For the predictions, generate 20 random numbers chosen from a uniform distribution that spans the range of values in the original dataset.

```
test_set = data.frame(Girth = sort(runif(20, min=min(trees$Girth),
                                             max=max(trees$Girth))))
pred_data = data.frame(predict.lm(model, newdata = test_set, interval = "predict"))
test_set = cbind(test_set,
                 VolumeSqrt = pred_data$fit,
                 lwr = pred_data$lwr,
                 upr = pred_data$upr)
test_set
```

```
##        Girth VolumeSqrt      lwr      upr
## 1   8.350701   3.144397 2.497609 3.791185
## 2   8.458755   3.192224 2.546466 3.837983
## 3   8.502034   3.211381 2.566029 3.856732
## 4   9.817655   3.793707 3.159043 4.428370
## 5  10.094229   3.916125 3.283289 4.548961
## 6  10.438108   4.068334 3.437562 4.699106
## 7  11.090093   4.356919 3.729420 4.984417
## 8  11.375988   4.483463 3.857133 5.109792
## 9  13.132712   5.261032 4.638258 5.883806
## 10 13.142739   5.265470 4.642698 5.888241
## 11 13.711790   5.517346 4.894366 6.140325
## 12 14.081859   5.681147 5.057678 6.304616
## 13 15.124751   6.142757 5.516412 6.769101
## 14 15.887297   6.480278 5.850448 7.110108
## 15 15.928337   6.498444 5.868393 7.128494
## 16 15.953612   6.509631 5.879443 7.139819
## 17 18.618540   7.689193 7.037654 8.340733
## 18 18.927722   7.826045 7.171179 8.480910
## 19 19.566976   8.108994 7.446724 8.771265
## 20 19.597417   8.122468 7.459827 8.785109
```

Now, use `ggplot2` to plot the original data, the regression line including lower and upper confidence intervals, the prediction intervals for the model, and the predicted values for the test data.

```
ggplot() +
    stat_smooth(method="lm", data = trees, aes(x=Girth, y=sqrt(Volume)),
                col="darkgray") +
    geom_point(data = trees, aes(x=Girth, sqrt(Volume)),
               col="midnightblue") +

    geom_point(data = test_set, aes(Girth, VolumeSqrt), col="red") +
    geom_line(data = test_set, aes(Girth, lwr),
              color = "red", linetype = "dashed") +
    geom_line(data = test_set, aes(Girth, upr),
              color = "red", linetype = "dashed") +

    ggtitle("trees Volume prediction -- training data (black) + test data (red)") +
    xlab("Girth") + ylab("sqrt(Volume)")
```



And voilà! The End. ;-)