Selected R functions

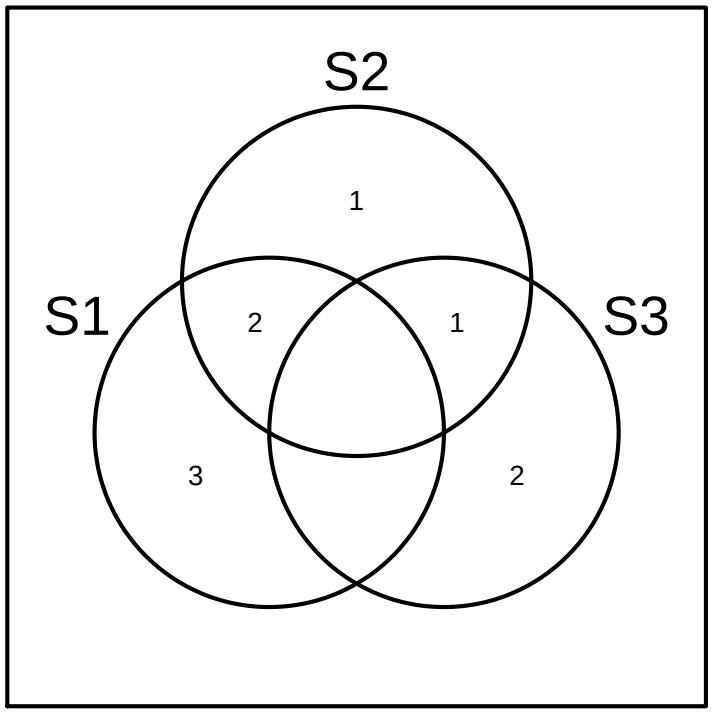
Marcin Kierczak

18 Nov 2016

Set operations

In R, working with sets is very natural. Let us begin by defining three sets and plotting Venn diagram of these.

```
library(venn)
S1 = c(1:5); S2=(4:7); S3=c(7:9)
venn(list(S1=S1, S2=S2, S3=S3))
```



Set operations cted.

$S_1 = 1, 2, 3, 4, 5; S_2 = 4, 5, 6, 7; S_3 = 7, 8, 9$

```
union(S1, S2) # Union
```

```
## [1] 1 2 3 4 5 6 7
```

```
intersect(S2, S3) # Intersection
```

```
## [1] 7
```

```
setdiff(S3, S2) # Difference
```

```
## [1] 8 9
```

Set operations cted.

$S_1 = 1, 2, 3, 4, 5; S_2 = 4, 5, 6, 7; S_3 = 7, 8, 9$

```
setequal(S1, S3) # Equality rel.
```

```
## [1] FALSE
```

```
is.element(2, S3) # Is element rel.
```

```
## [1] FALSE
```

Selected math functions – polynomials

To be able to work with polynomials, such as $x^3 + x^2 + 2x + 7$, we need to install and load the `polynom` package. Once this is done, let us define two polynomials, the one above and $3x^2 + 5x - 3$.

```
library(polynom)
poly1 <- polynomial(c(7,2,1,1))
poly1
```

```
## 7 + 2*x + x^2 + x^3
```

```
poly2 <- polynomial(c(-3, 5,3))
poly2
```

```
## -3 + 5*x + 3*x^2
```

Polynomials – defining, alternative 2

Polynomials can be also defined in terms of their zeros, i.e. the points where their value equals zero. Instead of zeros, one can provide coords of the points the polynomial has to contain.

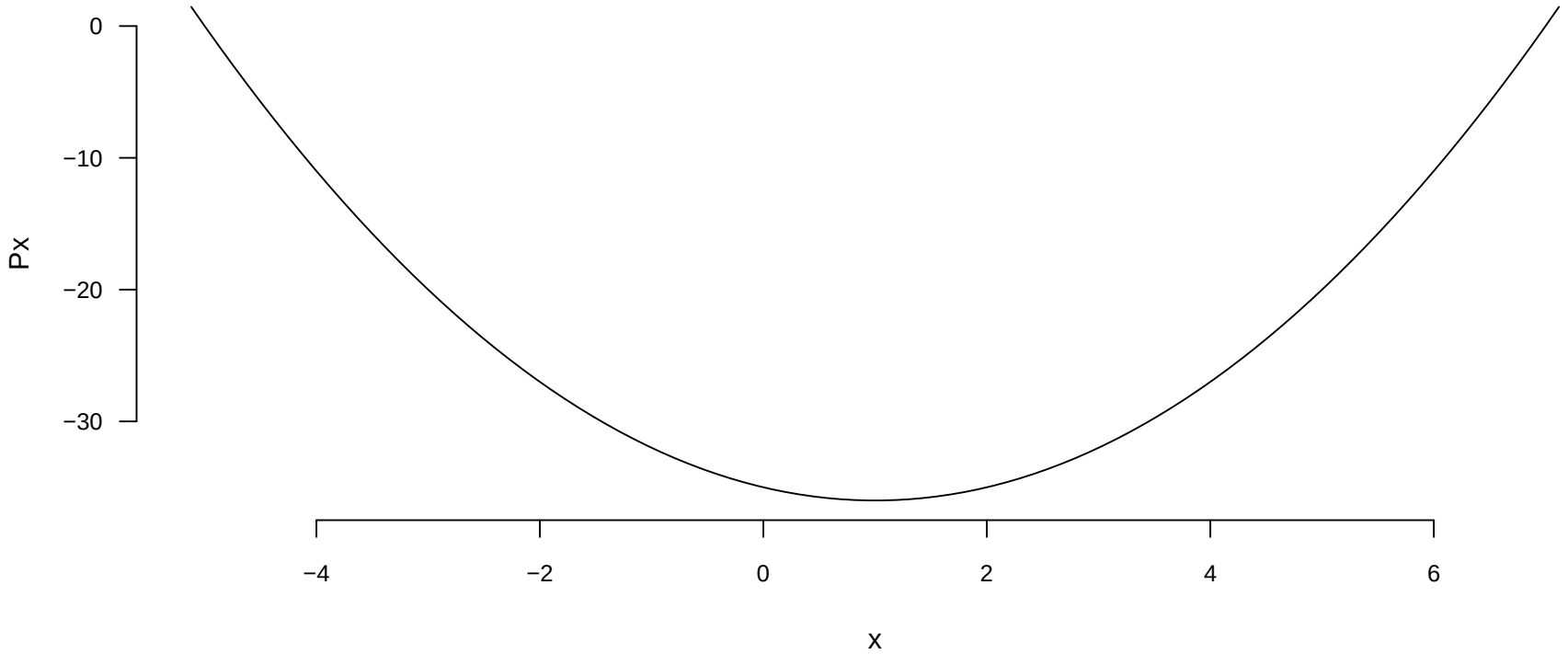
```
poly3 <- poly.calc(c(-5,7)) # zeros

# points A(1,-1); B(4,4); C(9,5)
poly.calc(c(1,4,9), c(-1,4,5))
```

```
## -3.4 + 2.583333*x - 0.1833333*x^2
```

Polynomials – plot

Let us visualize one of the defined polynomials and see whether its zeros are where they should be...

```
poly3 <- poly.calc(c(-5,7)) # zeros
p <- plot(poly3, bty='n', las=1, cex.axis=.8)
```



Polynomials – basic operations

Polynomials can be added, subtracted, multiplied and divided using standard operators:

```
poly1 + poly2
```

```
## 4 + 7*x + 4*x^2 + x^3
```

```
poly1 / poly2
```

```
## -0.2222222 + 0.3333333*x
```

Polynomials – calculus

We can also compute derivatives and integrals of polynomials:

```
# integrate on the [-2,2] interval
integral(poly1, c(-2,2))
```

```
## [1] 33.33333
```

```
deriv(poly2)
```

```
## 5 + 6*x
```

Polynomials – GCD and LCM

It is also easy to find the Greatest Common Divisor and the Least Common Multiple of two polynomials:

```
# integrate on the [-2,2] interval
GCD(poly1, poly3)
```

```
## 1
```

```
LCM(poly1, poly2)
```

```
## -21 + 29*x + 28*x^2 + 8*x^3 + 8*x^4 + 3*x^5
```

Math functions

R can also be used to examine functions: find their extrema and zeros. Say, we want to examine the $y = (x - 9)^2 - 5x - 2$ function. Let's define it first:

```
f <- function(x) {
  y <- (x - 9)^2 - 5*x - 2
  return(y)
}
f(2) # compute its value at x=2
```

```
## [1] 37
```

Math functions - zeros

To find zeros of the $y = (x-9)^2 - 5x - 2$ function we us the `uniroot` function. Note, we have to define the interval on which we are searching.

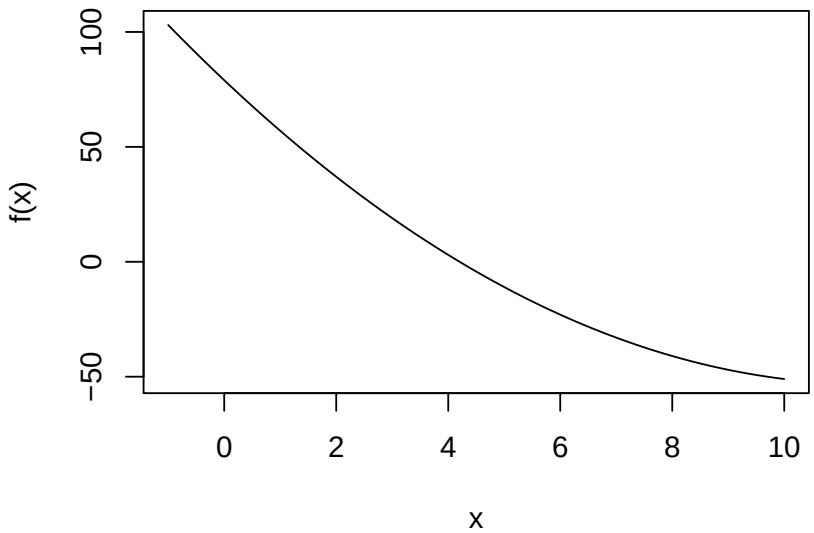
```
uniroot(f, lower=-1, upper=10)
```

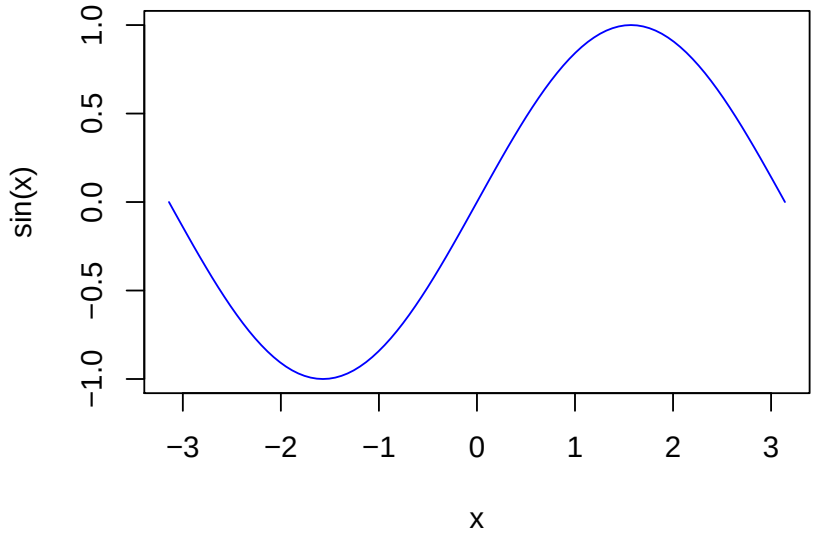
```
## $root
## [1] 4.20274
##
## $f.root
## [1] -1.518598e-06
##
## $iter
## [1] 6
##
## $init.it
## [1] NA
##
```

Math functions - plotting

Sure, one can plot a function, such as: $y = (x-9)^2 - 5x - 2$ or $sin(x)$.

```
par(mfrow=c(1,2))
curve(f, from = -1, to=10)
curve(sin, from=-pi, to=pi, col='blue')
```





Math formulas in R

In R, one can use the so-called *formulas*. A formula is a symbolic representation of a relationship between variables. Typically, on the left side of a formula, we have one variable, the dependent variable and on the right side, we have one or more explanatory variables that are forming an *expression*. For instance, we can write **y** $\sim$ **x** which means that y depends on x. The former is the *dependent variable* while the latter is the *independent* or *explanatory variable*.

- We can have more independent variables forming an expression, e.g.: **y** $\sim$ **x^2 + 2z + 3b**
- We also can have interactions between variables: **y** $\sim$ **a + b + a:b** which can also be written as: **y** $\sim$ **a*b**

Math formulas 2

- Writing: **y ~ (a + b) %in% c** is equivalent to: **y ~ a:c + b:c**,
- Similarly, the division sign **y ~ a/b** yields **y ~ a + a:b**,
- The minus sign excludes the variable: **y ~ a * b - a** equals to: **y ~ b + a:b**,
- To add the constant term, one can write +1, e.g.: **y ~ a + b + 1**,
- To remove the constant term, one can use either +0 or -1: **y ~ +0 + a + b**.

The formulas are very useful when defining *linear models* and in *calculus*.

Math – calculus

The symbolic calculus is implemented in package *stats* and pre-loaded automaticaly when you start R session. Derivation is implemented in two functions, *D()* and *deriv()* that differ in the form in which they take arguments: the former takes an expression as argument, the latter a formula with left side undifined. Let us do some derivation on one of the polynomials we have defined:

```
poly1
```

```
## 7 + 2*x + x^2 + x^3
```

```
deriv(poly1, 'x')
```

```
## 2 + 2*x + 3*x^2
```

Math – calculus cted.

To do the same using the *D()* function:

```
poly1
```

```
## 7 + 2*x + x^2 + x^3
```

```
p1 <- expression(x^3 +x^2 + 2*x)
deriv.call <- deriv(p1, 'x')
deriv.call
```

```
## expression({
##     .expr2 <- x^2
##     .expr4 <- 2 * x
##     .value <- x^3 + .expr2 + .expr4
##     .grad <- array(0, c(length(.value), 1L), list(NULL,
##     .grad[, "x"] <- 3 * .expr2 + .expr4 + 2
##     attr(.value, "gradient") <- .grad
##     .value
```

Math – calculus cted.

Let us now evaluate this derivative in points $x = \{1, \ldots, 5\}$:

```
x <- 1:5
eval(deriv.call)
```

```
## [1]   4  16  42  88 160
## attr(,"gradient")
##       x
## [1,]  7
## [2,] 18
## [3,] 35
## [4,] 58
## [5,] 87
```

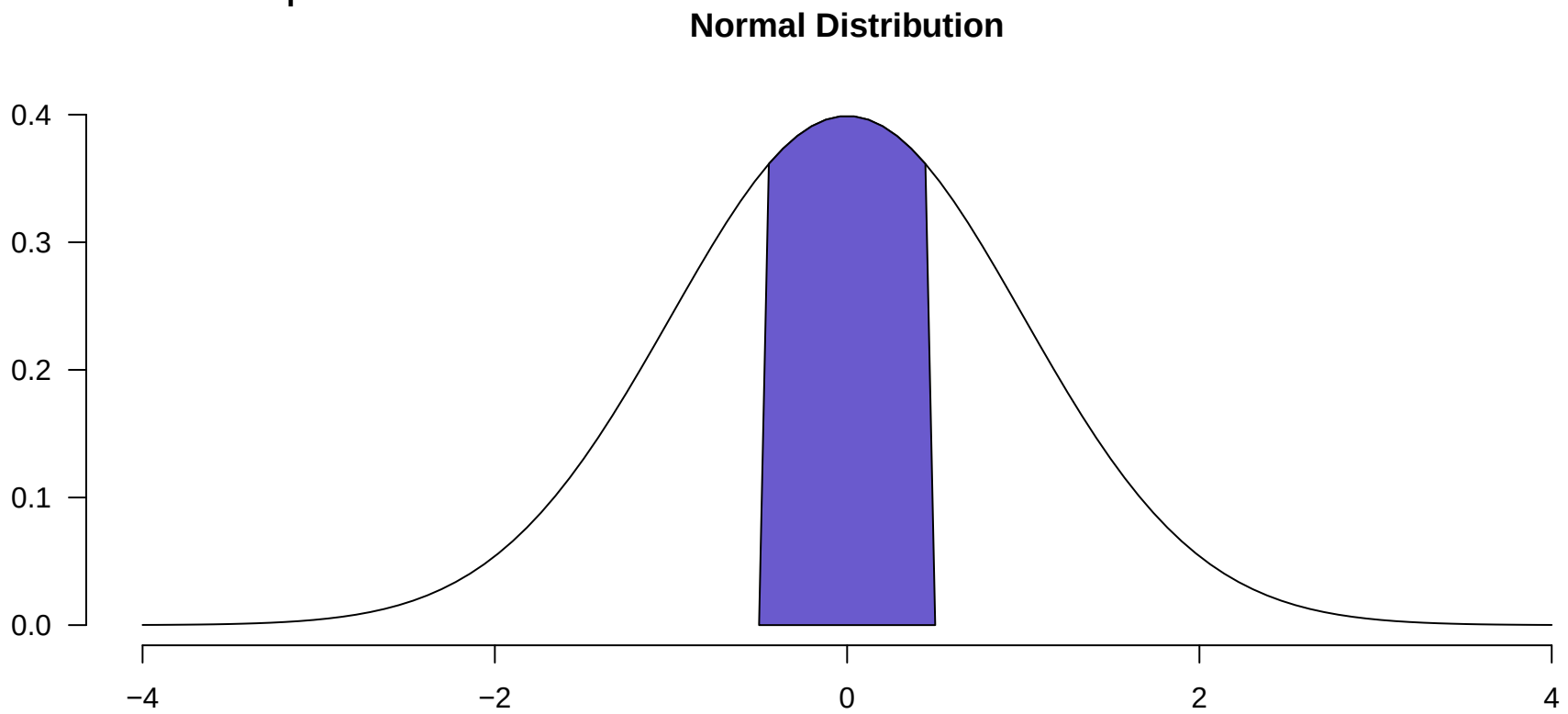
Math – calculus, integration.

Symbolic integration is a bit more tricky, let's look at the numerical integration in R. We are, say, interested in the area under the $N(0, 1)$ on the $-.5, .5$ interval:

```
integrate(dnorm, lower = -.5, upper=.5, mean = 0, sd = 1)
```

```
## 0.3829249 with absolute error < 4.3e-15
```

And here is the graphical illustration of the area integrated in the above example:



Linear regression – Galton's Dataset

It was Sir Francis Galton, a British *man of Reinassance*, who introduced the term **regression**. In 1885, he was looking at the relationship between the height of parents and the height of children. To his surprise, he has noticed that, on average, children of higher parents are also higher than average, but shorter than the mean of their parents. This phenomenon he has termed *regression towards the mean*.

Linear regression – Galton's Dataset cted.

Package `UsingR` contains Galton's original data: *Data set from tabulated data set used by Galton in 1885 to study the relationship between a parent's height and their childrens. The midparent's height is an average of the fathers height and 1.08 times the mother's.*

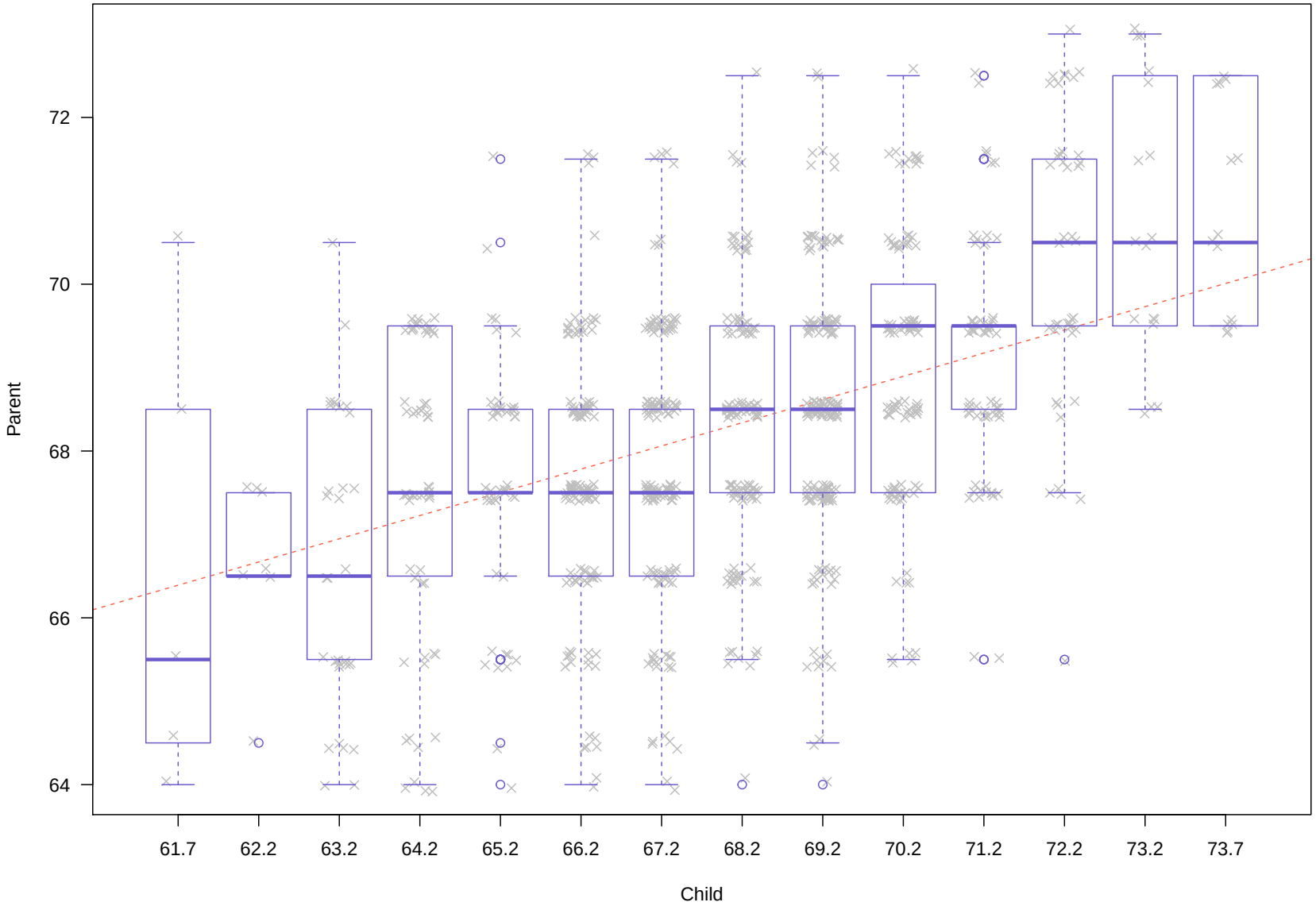
```
library(UsingR)
data(galton)
dim(galton)
```

```
## [1] 928   2
```

```
head(galton, n=2)
```

```
##   child parent
## 1  61.7   70.5
## 2  61.7   68.5
```

Galton's data – plot



Fitting linear model to Galton's data

Below, we will try to fit linear model to Galton's data and do some model diagnostics in order to see how good it is.

```
lm.galton <- lm(formula=child~parent, data=galton)
```

We used standard R `lm` function. We want to know how child's height depends on parents' height: `child parent` and we use the `lm` function.

Summary of the model

```
summary(lm.galton)
```

```
Call:
lm(formula = child ~ parent, data = galton)

Residuals:
   Min     1Q Median     3Q    Max
-7.805 -1.366  0.049  1.634  5.926

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  23.9415     2.8109    8.52   <2e-16 ***
parent        0.6463     0.0411   15.71   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.24 on 926 degrees of freedom
Multiple R-squared:  0.21,  Adjusted R-squared:  0.21
F-statistic:  247 on 1 and 926 DF,  p-value: <2e-16
```

Summary explained

- the distribution of residuals: minimum, maximum, median, the 1^{st} and the 3^{rd} quantile,
- the estimation of β parameters. in our case it is $\hat{\beta}_0 = 23.94$ and $\hat{\beta}_1 = 0.65$,
- both estimates, we have very high significance ($p < 2 \times 10^{-16}$) also indicated by three asterisks.
- *goodnes of fit* measured by R^2 and the value of *F-statistic* and the corresponding p-value. The R^2 tells what percentage of the variability of y the model explains,
- F-statistic is basically a measure of significance of all coefficients together **excluding** the effect of the mean $\hat{\beta}_0$ a.k.a $\hat{\mu}$.

Extracting coefficients

We can easily extract model coefficients and use them to plot the fitted line on top of the original data:

```
plot(jitter(galton$child), jitter(galton$parent),
     col="slateblue", axes=T, xlab="Child",
      ylab="Parents", pch=19, cex=.5)

abline(a=lm.galton$coefficients[1],
       b=lm.galton$coefficients[2],
       col="tomato", lty=1, lwd=2, las=1)
```

Extracting coefficients – plot

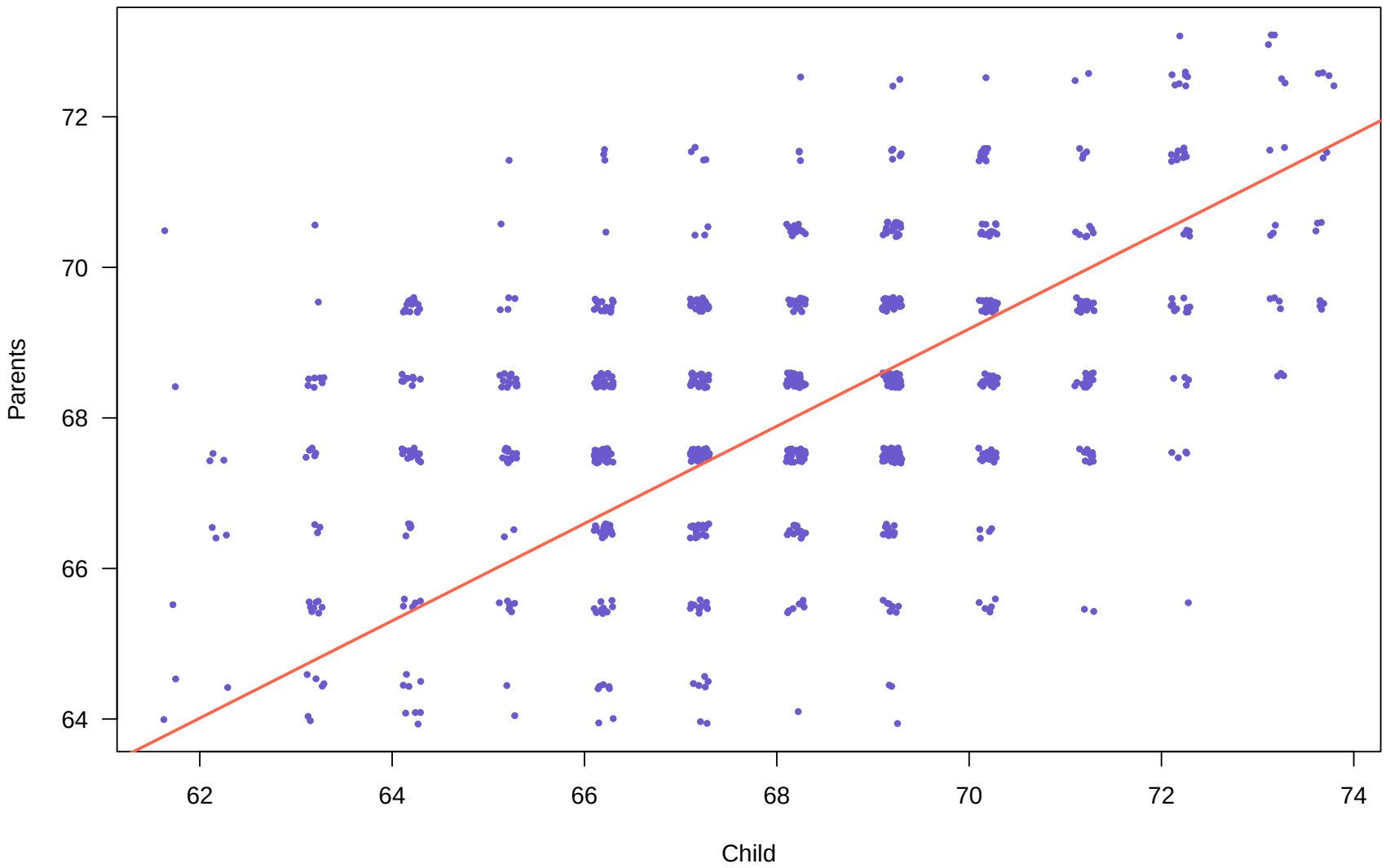


Figure 1: Original (jittered) data points from Galton's dataset together with the line fitted using linear model.

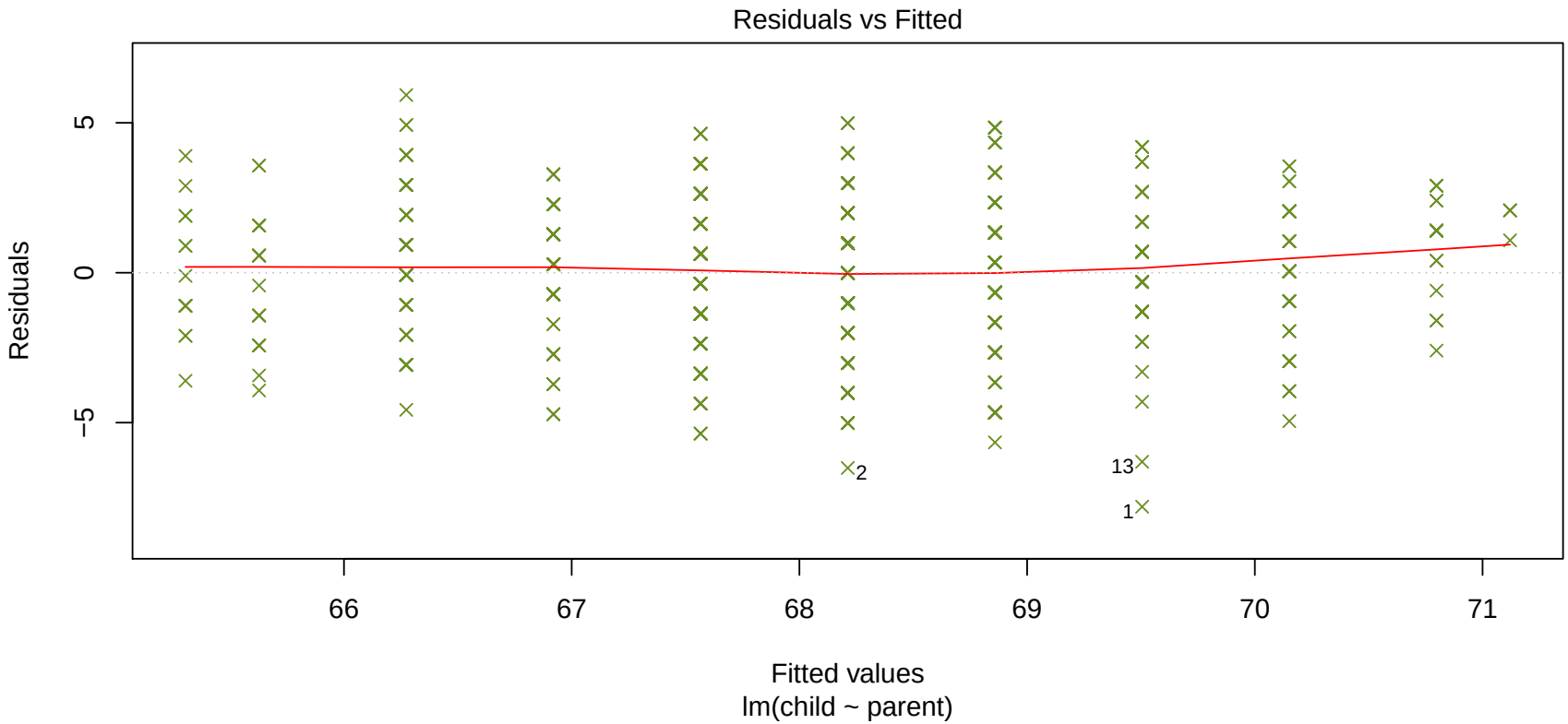
Linear model diagnostics

When fitting a linear model, we make some assumptions like the normality of distribution for residuals (error term). We need to check whether all these criteria are fulfilled. If they are, we can trust our model. R provides an overloaded `plot` function to display some diagnostic plots for a model.

Residuals vs. Fitted

The very first diagnostic plot shows residuals $\varepsilon_i = y_i - \hat{y}_i$ against fitted values $\hat{y}_i$. Intuitively (and it was our assumption), residuals should have their mean $\bar{\varepsilon} = 0$.

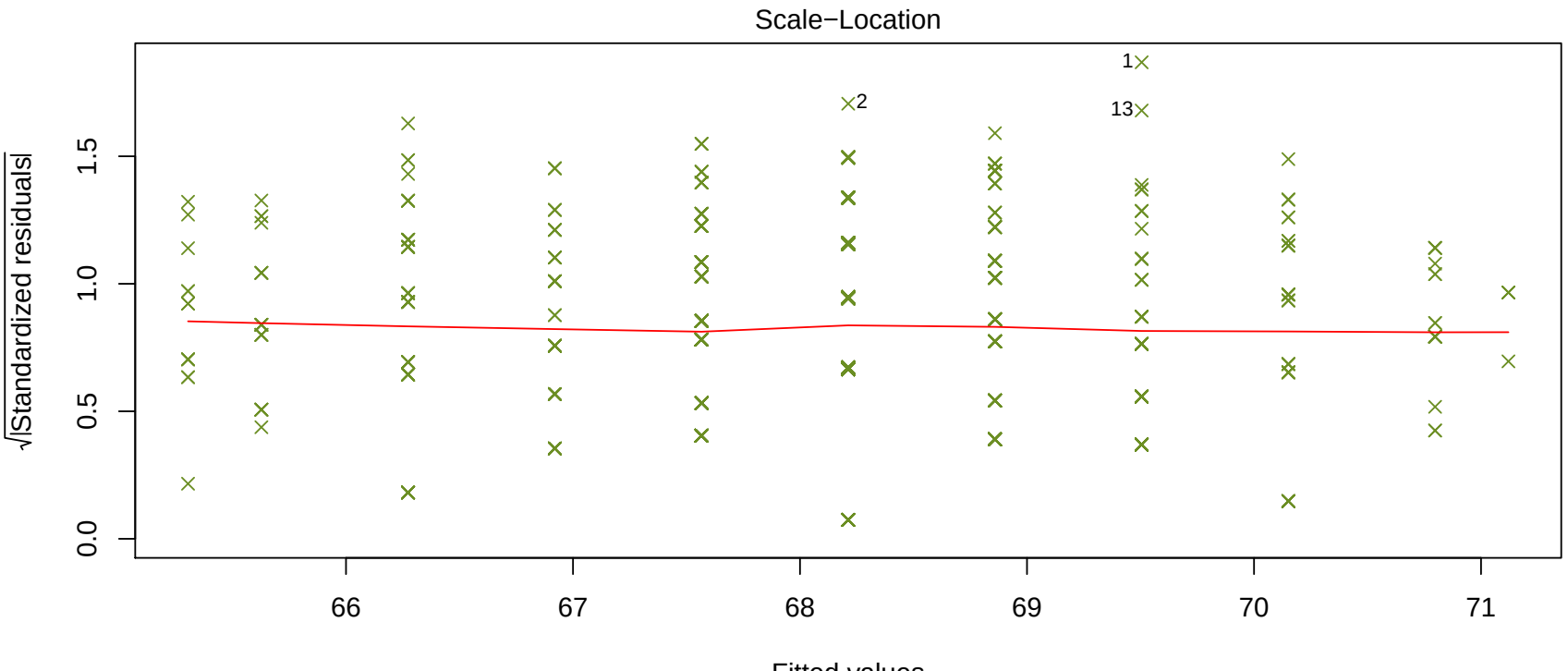
```
plot(lm.galton, which=1, pch=4, col="olivedrab")
```



Residuals variance homogenity

On the y-axis we have square root of modules of standardised residuals. The residuals should be uniformly dispersed around the mean, e.g. if for low values of $\hat{y}_i$ residuals are small and for medium values high, something went wrong.

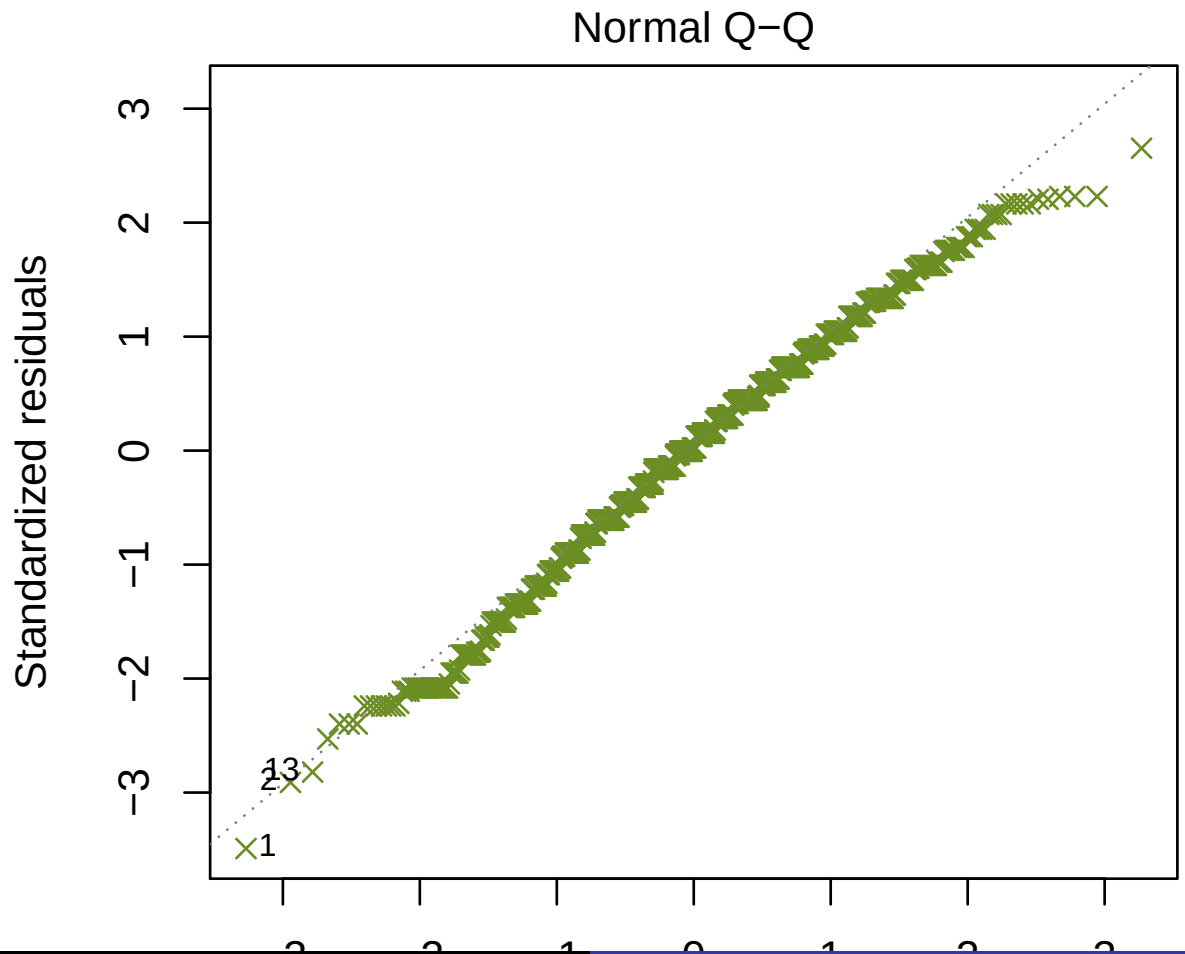
```
plot(lm.galton, which=3, pch=4, col="olivedrab")
```



Normality of residuals

Empirical quantiles for the distribution of residuals vs. theoretical quantiles for the normal distribution of residuals (QQ-plot).

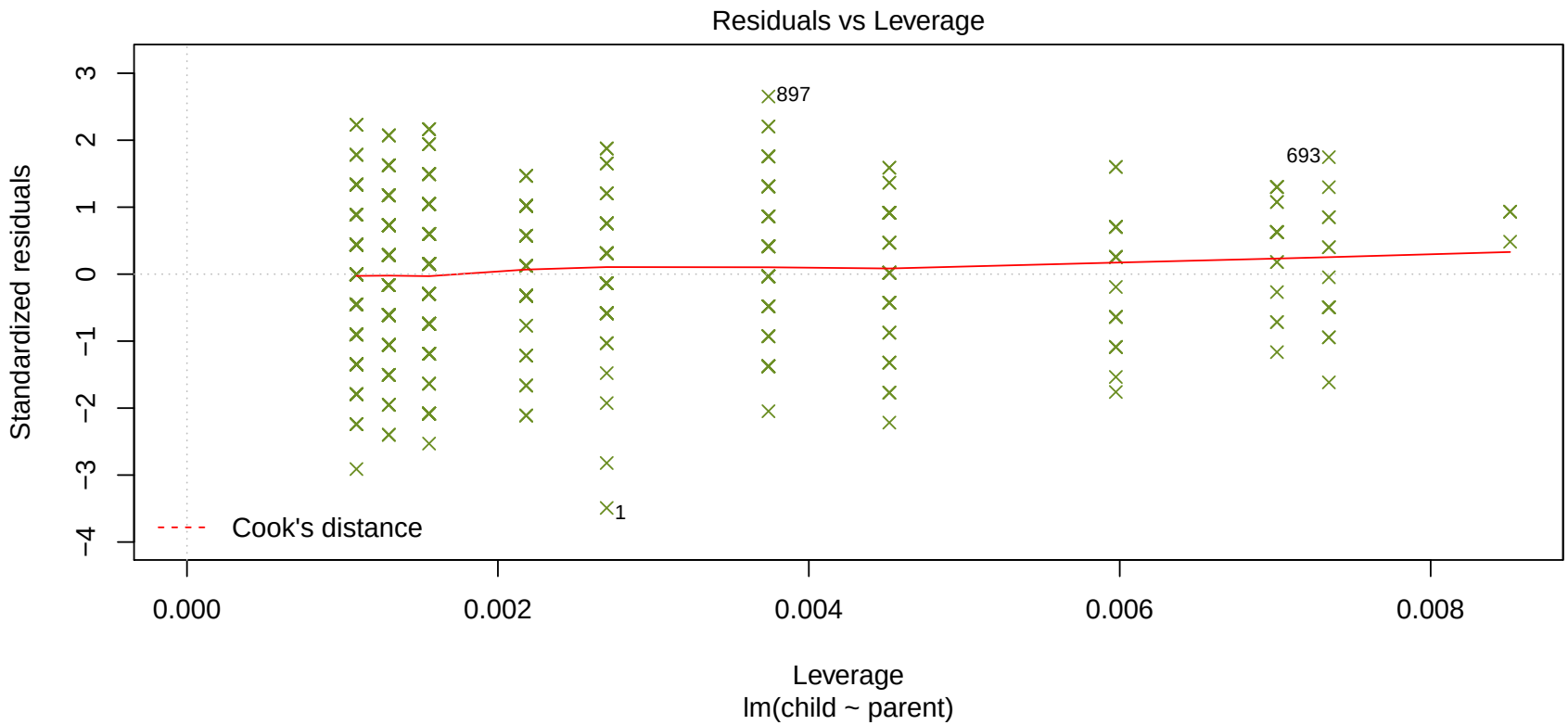
```
plot(lm.galton, which=2, pch=4, col="olivedrab")
```



Outliers – leverage

Leverage h_i measures the influence of value y_i on the estimated $\hat{y}_i$. Thus, indirectly, it shows also what is the influence of y_i value on $\hat{\beta}_i$. There should not be any outstanding leverages.

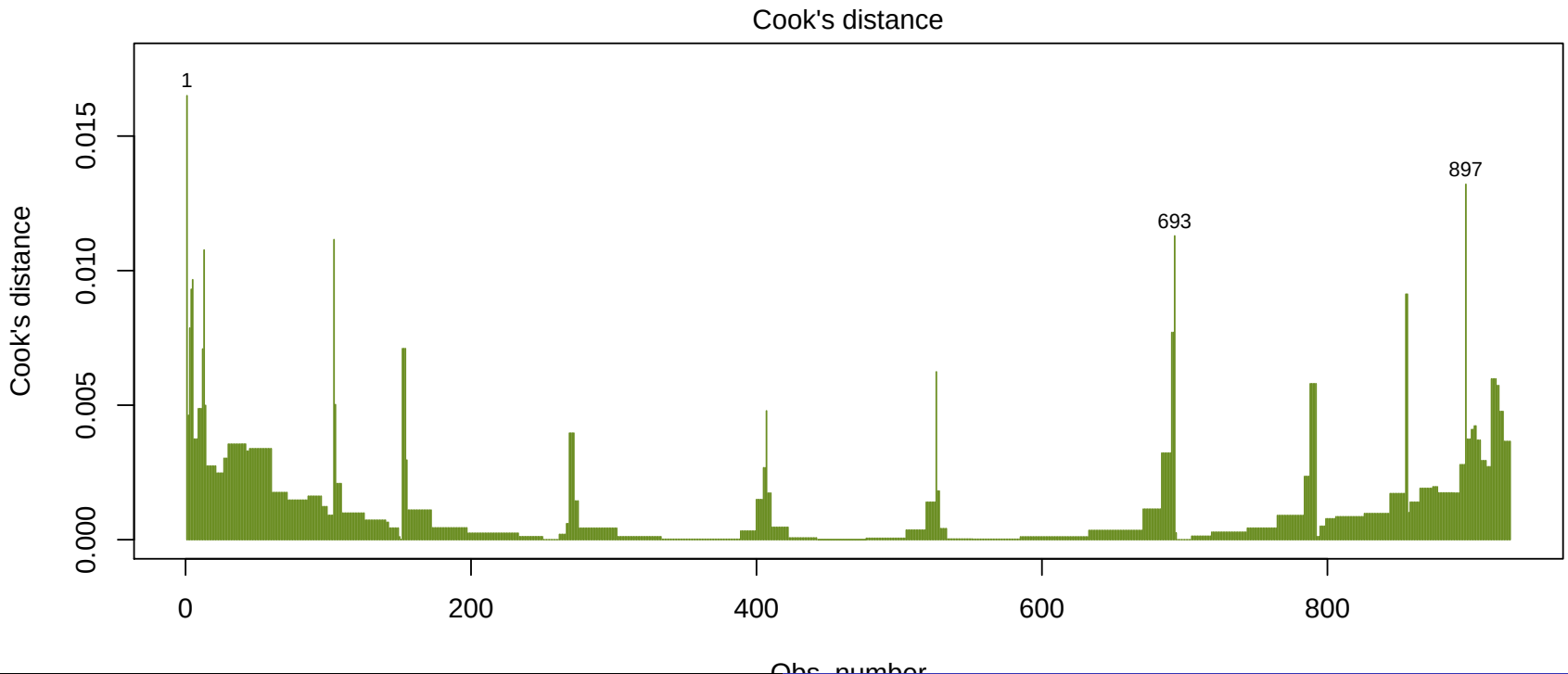
```
plot(lm.galton, which=5, pch=4, col="olivedrab")
```



Cook's distance

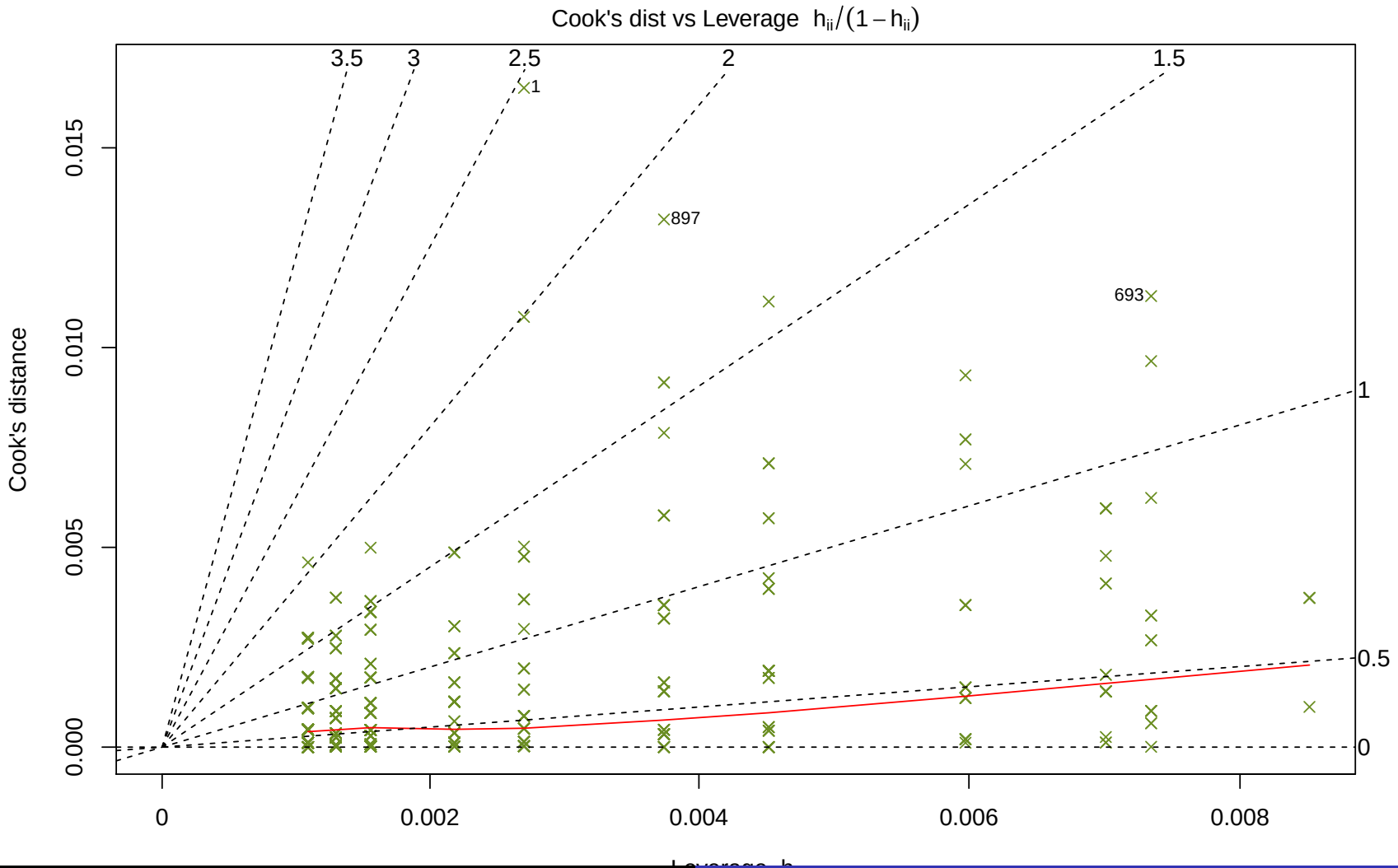
Cook proposed a measure of influence based on the extent to which parameter estimates would change if one omitted the i-th observation. An observation with Cook's distance **greater than 1** is unusual.

```
plot(lm.galton, which=4, pch=4, col="olivedrab")
```



Cook's distance vs. leverage:

```
plot(lm.galton, which=6, pch=4, col="olivedrab")
```



Some statistical tests useful in model diagnostics

Although visual examination of the plots is a quick and easy way of checking validity of a model, hypotheses should be verified with proper statistical tests. Some of the tests useful in diagnostics of a linear model are discussed below.

Shapiro-Wilk test of normality

Residuals should be normally distributed. We have checked this by looking at Q-Q plot. Let us verify our finding using Shapiro-Wilk for normality.

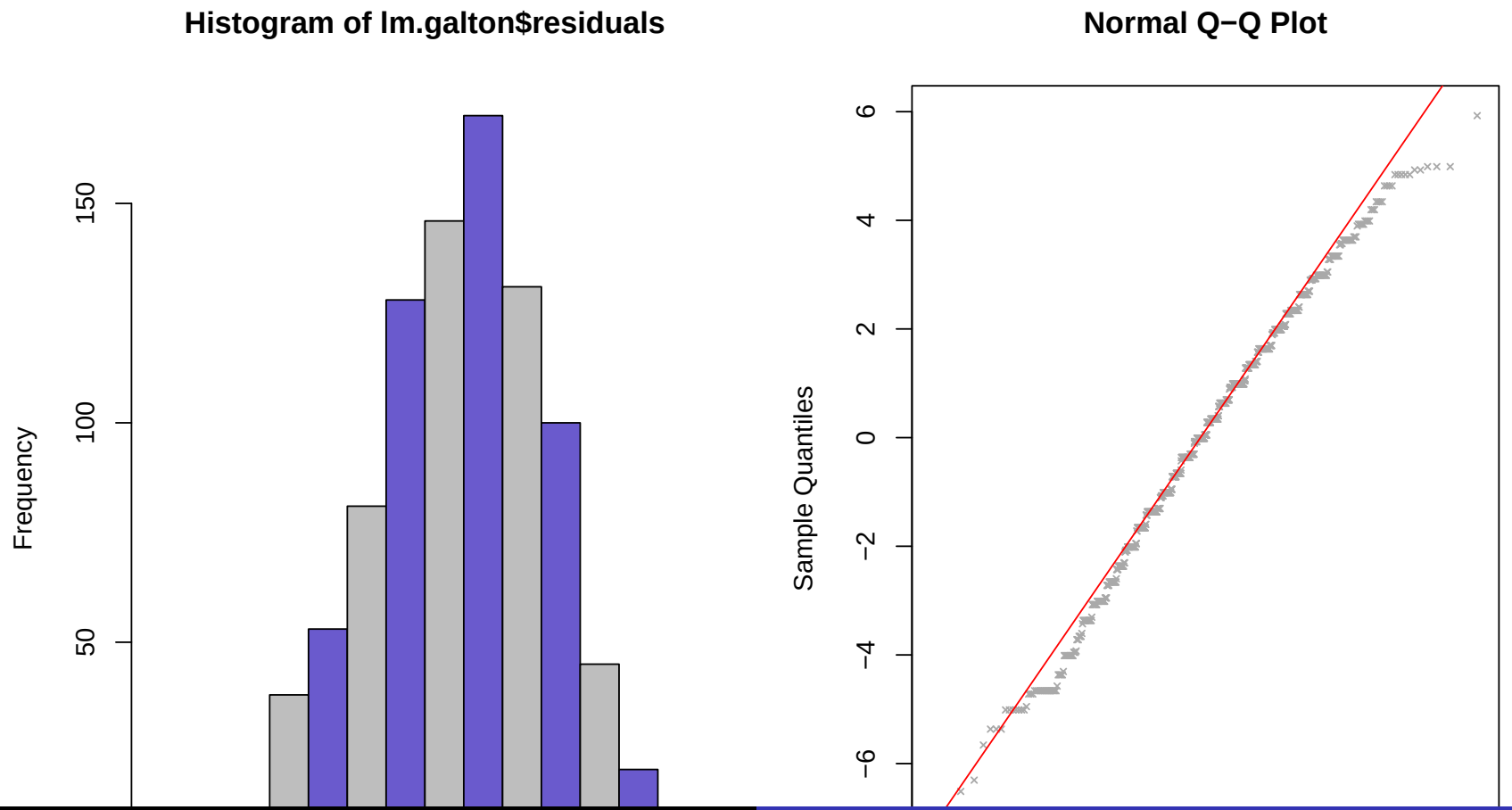
```
shapiro.test(lm.galton$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  lm.galton$residuals
## W = 0.99275, p-value = 0.0001697
```

Low p-value tells us, we should reject $H_0 : \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon)$. However, some normality tests (including Shapiro-Wilk) are, especially for large number of data points, very sensitive to slight departures from normality. Therefore Q-Q plot may still be a better alternative.

A Q-Q plot

```
par(mfrow=c(1,2))
hist(lm.galton$residuals, col=c("slateblue","grey"))
qqnorm(lm.galton$residuals, pch=4, cex=.5, col="darkgrey")
qqline(lm.galton$residuals, col="red")
```



Some statistical tests useful in model diagnostics

Although visual examination of the plots is a quick and easy way of cheching validity of a model, hypotheses should be verified with proper statistical tests. Some of the tests useful in diagnostics of a linear model are discussed below.

Shapiro-Wilk test of normality

Residuals should be normally distributed. We have checked this by looking at Q-Q plot. Let us verify our finding using Shapiro-Wilk for normality.

```
shapiro.test(lm.galton$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  lm.galton$residuals
## W = 0.99275, p-value = 0.0001697
```

Low p-value tells us, we should reject $H_0 : \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon)$. However, some normality tests (including Shapiro-Wilk) are, especially for large number of data points, very sensitive to slight departures from normality. Therefore Q-Q plot may still be a better alternative. . .

Variance homogenity of residuals

Now, we will perform the *Breusch-Pagan* test of variance uniformity to check whether residuals have uniform variance across all i. This test (and the tests used later) are implemented in package `lmtest`.

```
library(lmtest)
bptest(child~parent, data=galton)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  child ~ parent
## BP = 0.34256, df = 1, p-value = 0.5584
```

High p-value tells us there is no violation of variance uniformity.

Are residuals independent?

When fitting the model, we have also assumed the residuals are independent. We can check whether there is any auto-correlation using, e.g. Durbin-Watson test:

```
dwtest(child~parent, order.by=~parent, data=galton)
```

```
##
##  Durbin-Watson test
##
## data:  child ~ parent
## DW = 0.20175, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater
```

There is some auto-correlation between residuals sorted by parents' height.

Is the model truly linear?

We can also check whether linear model is appropriate here. Using *rainbow* test, we can see whether different models could be fitted for, e.g. tall and short parents (test will compare two linear models: one for the shorter 50% parents and another for the remaining "tall" parents):

```
raintest(child~parent, order.by=~parent, data=galton)
```

```
##
##  Rainbow test
##
## data:  child ~ parent
## Rain = 1.7378, df1 = 464, df2 = 462, p-value = 1.773e-09
```

Well, it seems that we could do better than linear model. We are getting different $\hat{\beta}$s for tall and short parents. . . However, splitting data half-way may not be the best idea?