Topic 9

Simple linear regression

Sergey Mastitsky ©
Klaipeda, 28-30 September 2011

Why correlation is not enough?

- Correlation coefficient is good in reflecting the magnitude of association between any two numerical variables
- However, we cannot use correlation to predict the values of one variable based on the values of the other one

9. Simple linear regression

9.1. Linear regression: brushing up the theory

Simple linear regression: fits a straight line and has just one predictor

$$y_i = \alpha + \beta x_i + \varepsilon_i$$

- y_i – "dependent" variable
- α – intercept
- β - regression coefficient (=slope)
- x_i – predictor (="independent" variable)
- ε_i – errors, independent and $N(0, \sigma^2)$

Don't be confused by terminology!

"Linear" doesn't always mean a straight line, e.g.

$$y_i = \alpha + \beta_1 x_i + \beta_2 x_i^2 + \varepsilon_i$$

is not linear in x, but **is** linear in the parameters β_1 and β_2:

$$if \;\; x_i^2 = c_i$$

$$y_i = \alpha + \beta_1 x_i + \beta_2 c_i + \varepsilon_i$$

Estimation of parameters

- Parameters α, β, and σ^2 are estimated using the *method of least squares*
- The method tries to find such α, β, and σ^2 that *minimize* the sum of squared residuals (i.e. find a line that goes as close to all data points as possible):

$$SS_{res} = \sum_i (y_i - (\alpha + \beta x_i))^2$$

Author: Sergey Mastitsky

Estimation of parameters

- It can be shown the SS_{res} takes the smallest value when

$$\hat{\beta} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

$$\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$$

Estimation of parameters

The residual variance σ^2 is estimated as

$$\sigma^2 = SS_{res} / (n - 2)$$

Significance of parameters

- Apparently, regression parameters would vary if we were to take different samples
- Therefore, it is of great importance to estimate the significance of the model parameters
- Usually of prime interest is to test the null hypothesis that $\beta_0 = 0$ (i.e. a horizontal line)
- This is done with a *t*-test: $t = \dfrac{\hat{\beta} - \beta_0}{S_{\hat{\beta}}} = \dfrac{\hat{\beta}}{S_{\hat{\beta}}}$

Significance of parameters

- A similar test can be applied to the intercept
- However, in most cases it's a meaningless test because:
  - there is no natural reason to believe that the line has to go through the origin
  - or it would involve an extrapolation far outside the range of data

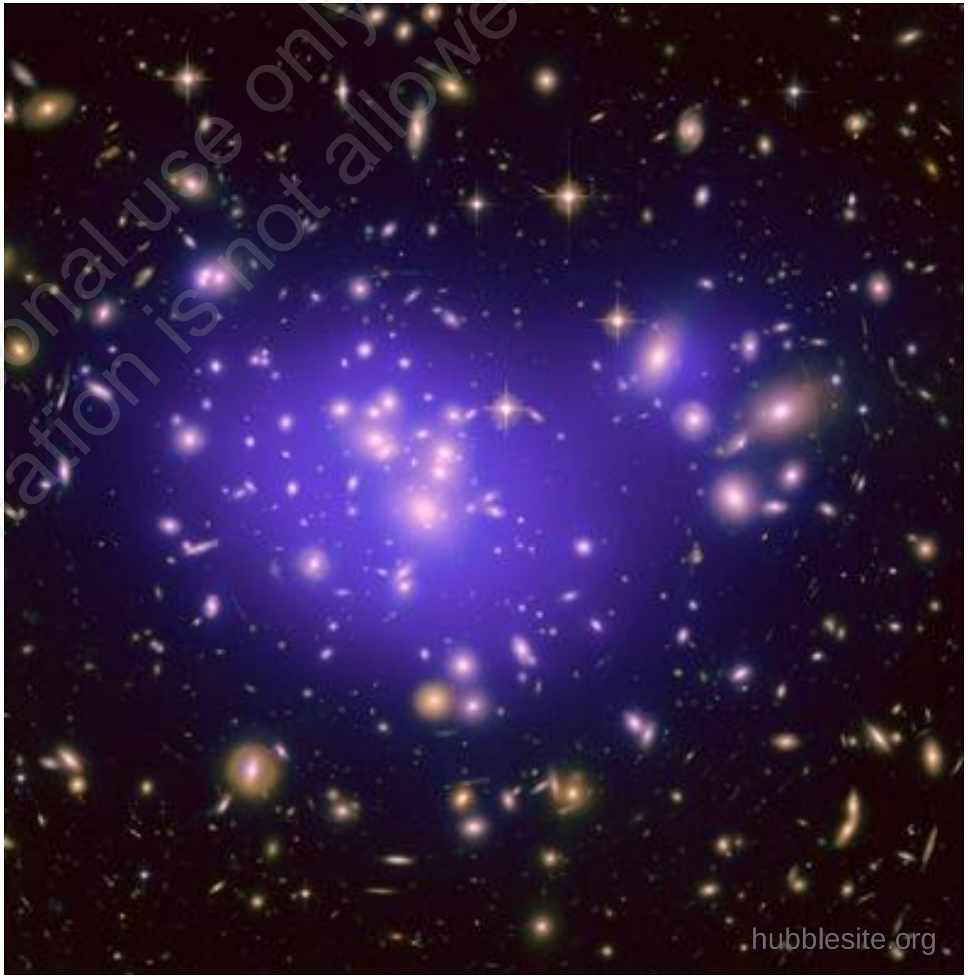
9. Simple linear regression

9.2. Fitting linear regression in R

How old is the Universe?

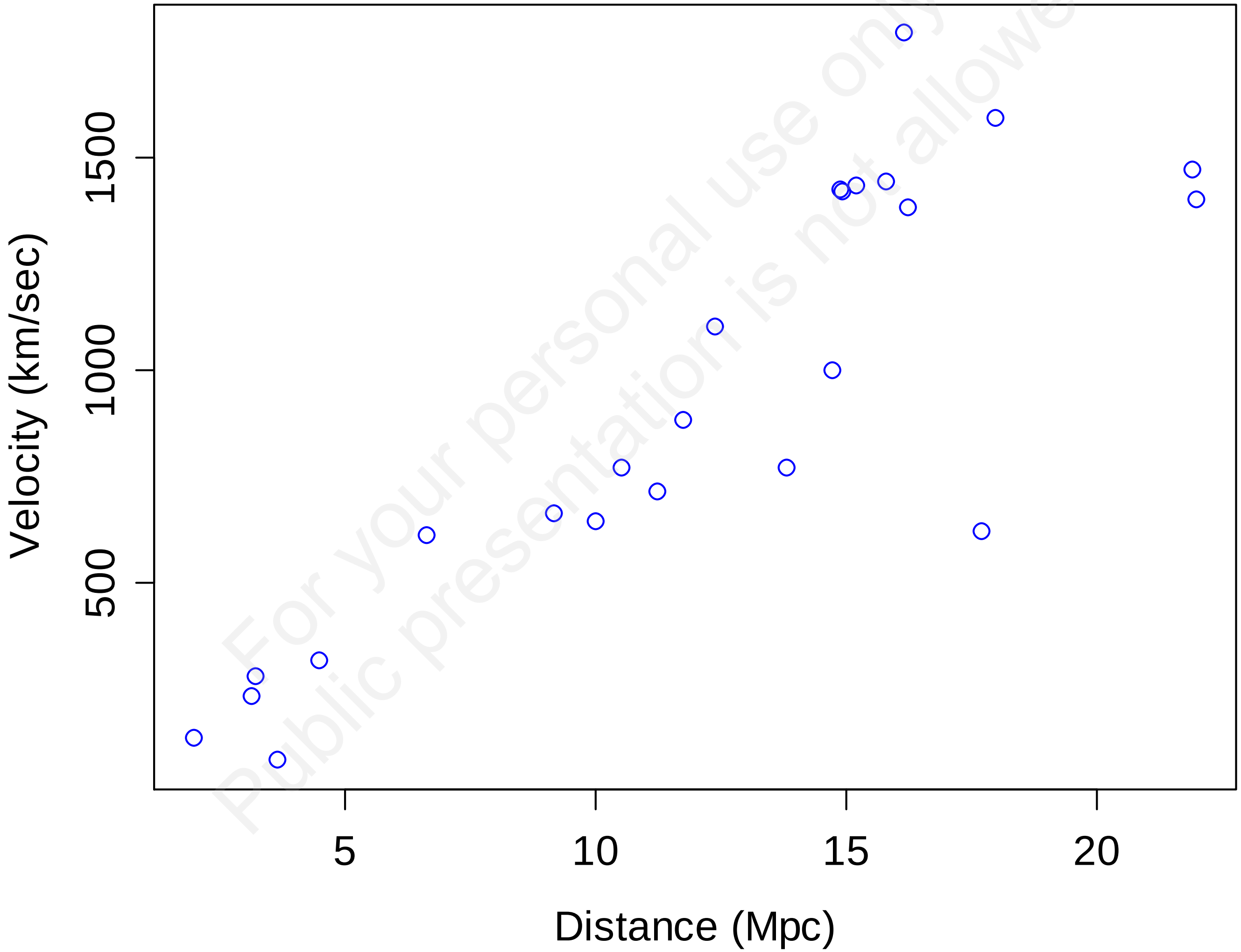
- Freedman et al. (2001)* report data on distance to 24 galaxies (Mpc) measured with the Hubble Telescope
- Velocities assessed by measuring the Doppler effect red shift are also reported (km/sec)

*Freedman WL et al. (2001) The Astrophysical Journal 553: 47-72



Author: Sergey Mastitsky

Data from Friedman et al. (2001)



Author: Sergey Mastitsky

How old is the Universe?

- The Big-Bang Theory says that the Universe expands uniformly according to the Hubble's law:

$$y = \beta x$$

where y is the relative velocity of any two galaxies separated by distance x

- β^{-1} gives the approximate age of the Universe, but it is unknown
- Data from Friedman et al. (2001) can be used to estimate β: $y_i = \beta x_i + \varepsilon_i$

Loading Hubble Telescope data

- Use the command

```
> setwd("~/Introductory R Course/R_Course_Datasets")
```

- Or in **RStudio** do

Tools -> Set Working Directory -> Choose Directory -> …your Desktop -> folder "`Introductory R Course`" -> folder "`R_Course_Datasets`"

Loading Hubble Telescope data

```
> hub.data <- read.table(
file = "hubble_data.txt",
header = TRUE,
sep = "\t")
```

Examine the data:

```
> head(hub.data)
```

Specifying linear regression in R

The function **lm()** is used to estimate the regression parameters

Model formula:
“ ~ “ is called “tilde”
“ -1” means the model has no intercept term

Object that contains the results of analysis

```
> hub.mod <- lm(y ~ x - 1,
         data = hub.data)
```

Name of the data frame in which the variables are to be found

Author: Sergey Mastitsky

Summary of the analysis

```
> summary(hub.mod)
Call:
lm(formula = y ~ x - 1, data = hub.data)

Residuals:
   Min      1Q  Median      3Q     Max
-736.5 -132.5   -19.0   172.2   558.0

Coefficients:
  Estimate Std. Error t value Pr(>|t|)
x   76.581      3.965   19.32 1.03e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 258.9 on 23 degrees of freedom
Multiple R-squared: 0.9419, Adjusted R-squared: 0.9394
F-statistic: 373.1 on 1 and 23 DF,  p-value: 1.032e-15
```

“Dissection” of the `summary(hub.mod)` output

```
Call:
lm(formula = y ~ x - 1, data = hub.data)
```

- Repeat of the function call
- Useful when many models are fitted during one R session

“Dissection” of the `summary(hub.mod)` output

```
Residuals:
    Min      1Q  Median      3Q     Max 
-736.5 -132.5   -19.0   172.2   558.0 
```

- A numerical summary of the distribution of the model residuals
- Can be used as a quick check of the distributional assumptions
- For example, the `Median` has to be close to 0, and the `Max` and `Min` should be roughly equal (in absolute value)

"Dissection" of the `summary(hub.mod)` output

```
Coefficients:
  Estimate Std. Error t value Pr(>|t|)
x   76.581      3.965   19.32 1.03e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The regression coefficient (`76.581`), along with its S.E. and P-value of the t-test of significance
- `*` - indicators of significance
- Below the table – definition of these indicators, e.g. three stars means 0 < P < 0.001
- Stars can be turned off with the comand

`options(show.signif.stars = FALSE)`

"Dissection" of the `summary(hub.mod)` output

```
Residual standard error: 258.9 on 23 degrees of freedom
```

- The residual variation, i.e. the variation of observations around the regression line
- σ

“Dissection” of the `summary(hub.mod)` output

```
Multiple R-squared: 0.9419, Adjusted R-squared: 0.9394
```

- `Multiple R-squared` (= coefficient of determination) is the squared Pearson correlation
- `Multiple R-squared` × 100% = percent of variation explained by the model; always increases with the number of predictors
- `Adjusted R-squared` is the `Multiple R-squared` adjusted in a certain way to account for the d.f.

“Dissection” of the `summary(hub.mod)` output

```
F-statistic: 373.1 on 1 and 23 DF,  p-value: 1.032e-15
```

- F-test of the null hypothesis that the data were generated from a model with only an intercept term
- Test of the overall usefulness of the model

Preliminary conculsion

- In overall, the model is highly significant ($P < 0.001$, F-test)
- Distance to a galaxy seems to be an important predictor of its observed velocity ($P < 0.001$, t-test)
- The fitted models is as follows:

$$y = 76.581x$$

9. Simple linear regression

9.3. Validation of the model

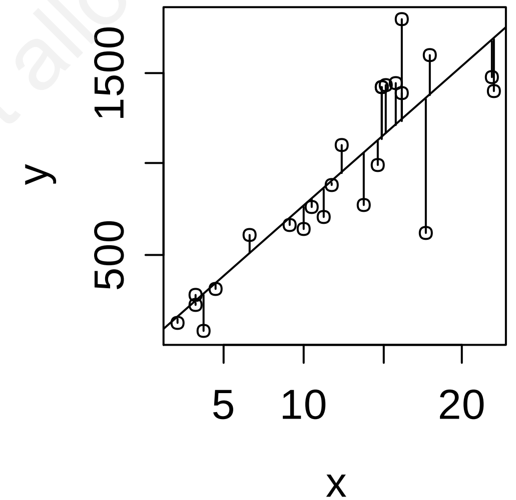
Validation of the model

- Highly significant P-values don't guarantee that the model correctly describes the process under study
- Once the model is fitted, one has to check if its assumptions are met
- In particular, one has to check if the model residuals are normally distributed and if there are any "influential" observations that distort the real picture

The `residuals()` function

resid() extracts the residuals from the model object:

```
> resid(hub.mod)
```

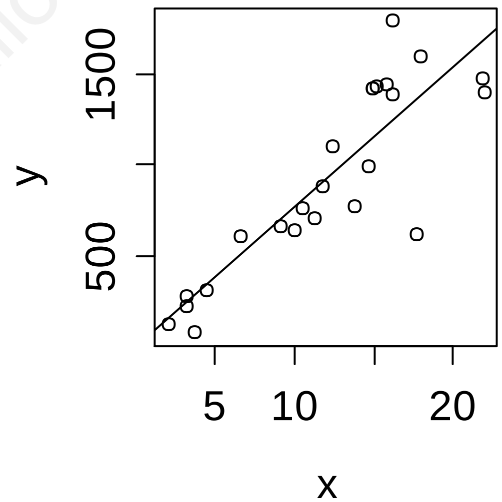



```
         1          2          3          4          5          6 
 -20.162344  -37.483536  557.979883  219.367962 -202.596044   31.408626 
         7          8          9         10         11         12 
-145.240750  -17.828771 -197.989654 -284.820174 -123.811720  -37.633930 
        13         14         15         16         17         18 
 100.501018  268.200373 -736.486745  286.003784  141.853390  235.549105 
        19         20         21         22         23         24 
 279.643102 -280.254161  156.456714  -25.849462   -9.230692 -128.274852 
```

The fitted() function

fitted() extracts model-fitted values of the response variable:

```
> fitted(hub.mod)
```



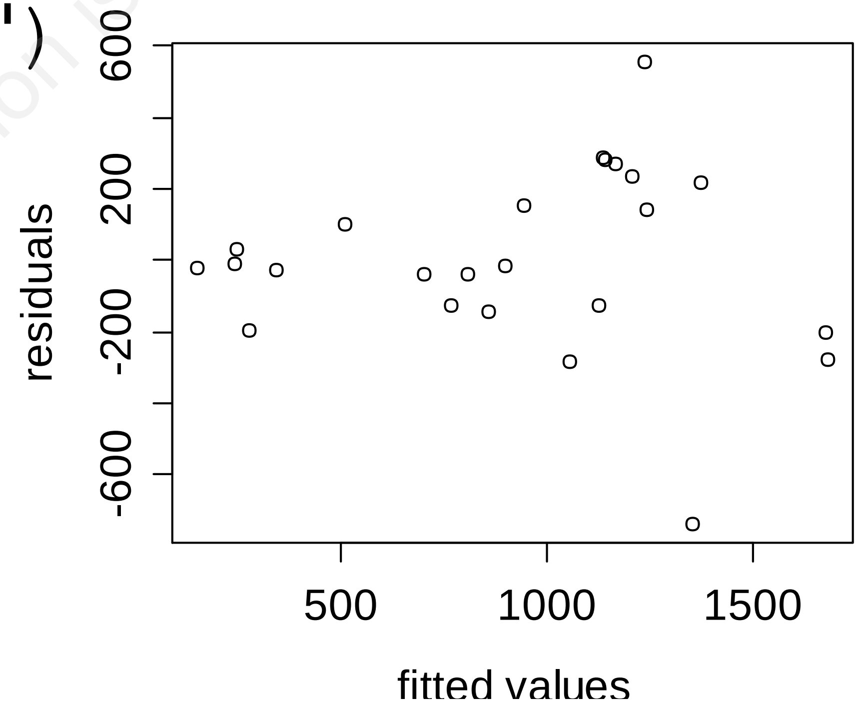

```
        1         2         3         4         5         6         7 
 153.1623  701.4835 1236.0201 1374.6320 1675.5960  246.5914  859.2408 
        8         9        10        11        12        13        14 
 899.8288  277.9897 1056.8202  765.8117  805.6339  508.4990 1164.7996 
       15        16        17        18        19        20        21 
1355.4867 1137.9962 1242.1466 1208.4509 1143.3569 1683.2542  946.5433 
       22        23        24 
 343.8495  241.2307 1127.2749 
```


Author: Sergey Mastitsky

Are the residuals distributed normally?

```
> plot(resid(hub.mod) ~
  fitted(hub.mod),
  xlab = "fitted values",
  ylab = "residuals")
```

The spread of residuals is increasing with larger fitted values – not good!



Author: Sergey Mastitsky

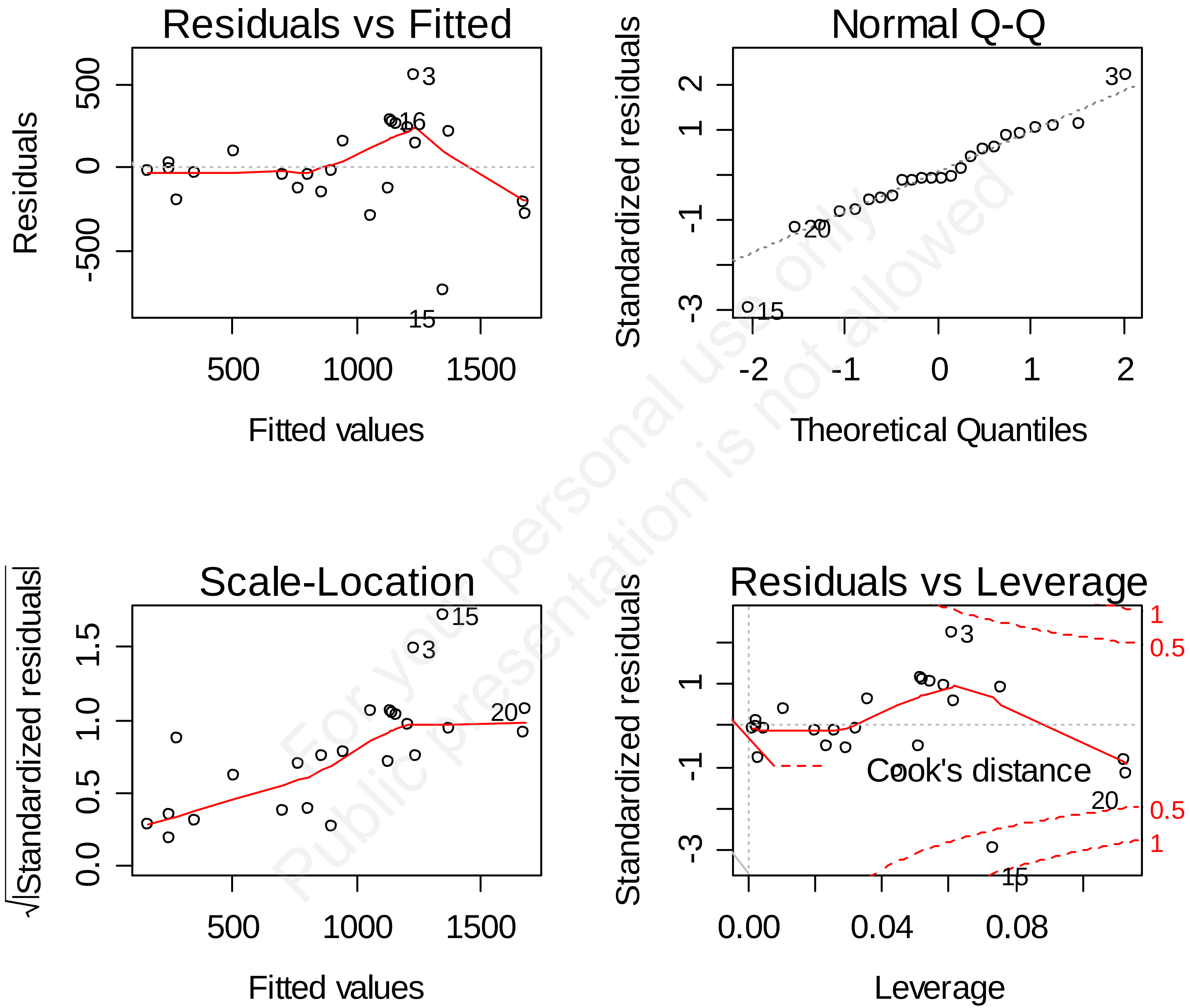
Getting four diagnostic plots with one command

```
> par(mfrow = c(2, 2))
```

The comand

```
> plot(hub.mod)
```

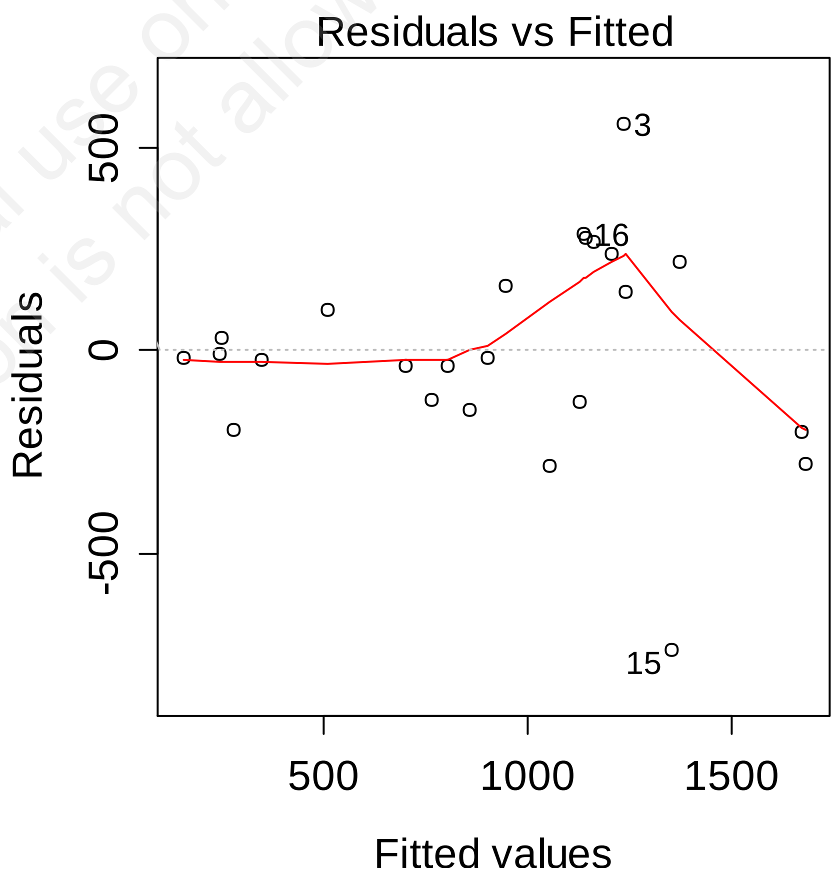
will produce four types of plots commonly used in diagnostics of the regression model validity



Author: Sergey Mastitsky

“Residuals vs. Fitted” plots

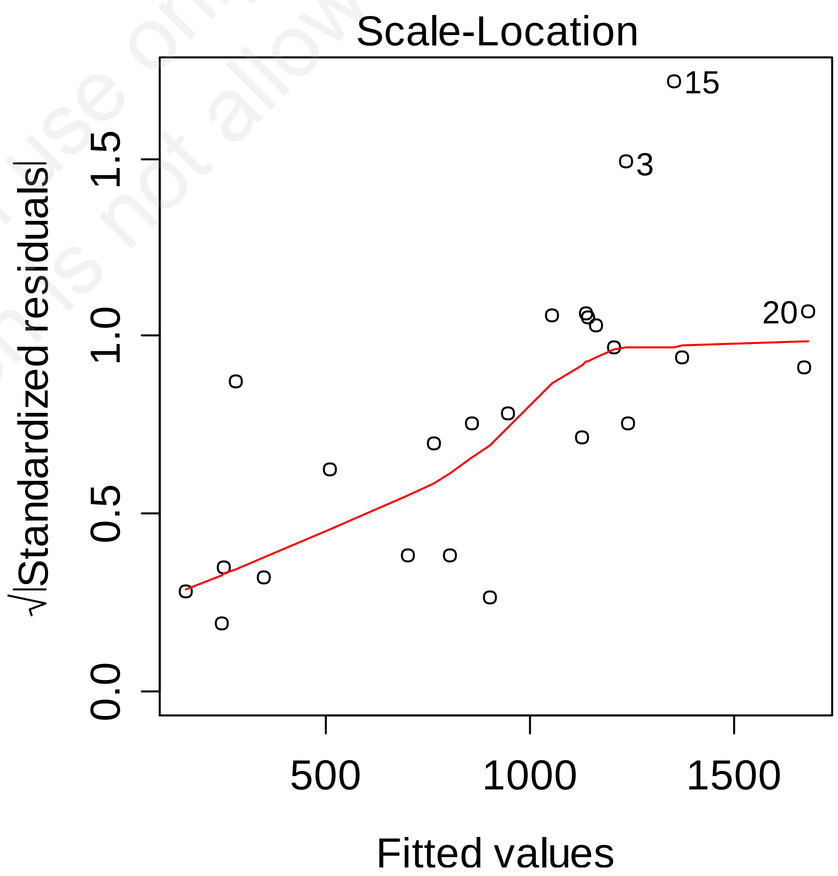
- The same plot that we already saw before
- Red line shows the trend in the distribution of residuals
- There should be no pattern in the spread of residuals
- Potentially influential data points are labeled with their index numbers



Author: Sergey Mastitsky

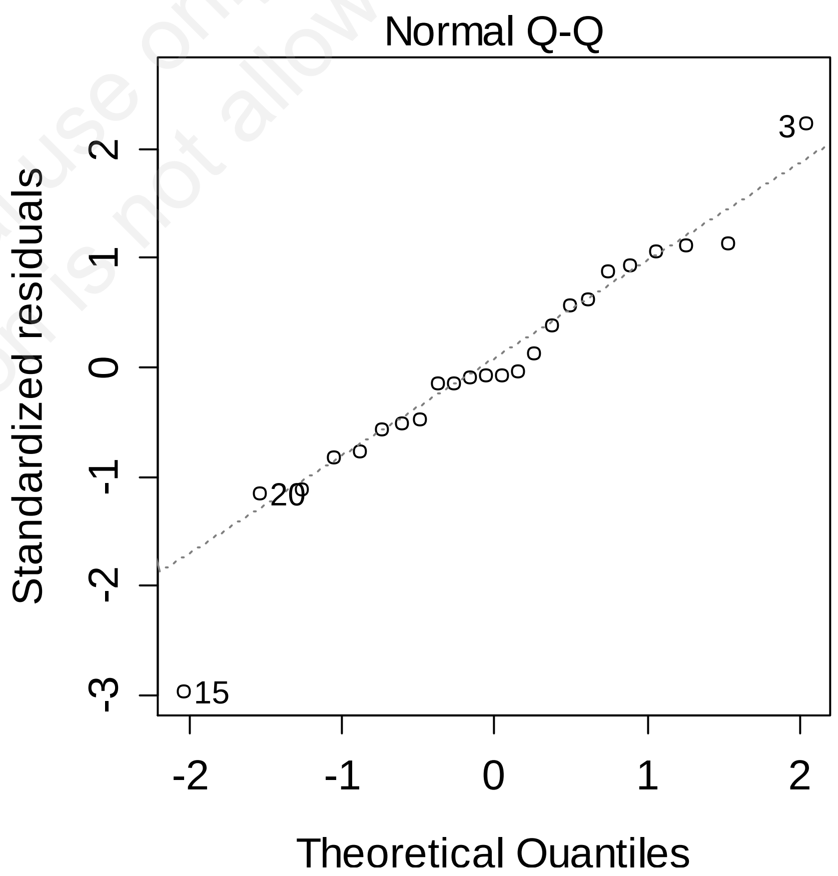
"Scale-location" plots

- The residuals are standardized by dividing by their estimated standard deviation
- Square root of the absolute value of each std. residual is plotted against the equivalent fitted value
- This plot makes it easier to check the constant variance assumption
- In this case the assumption is not met



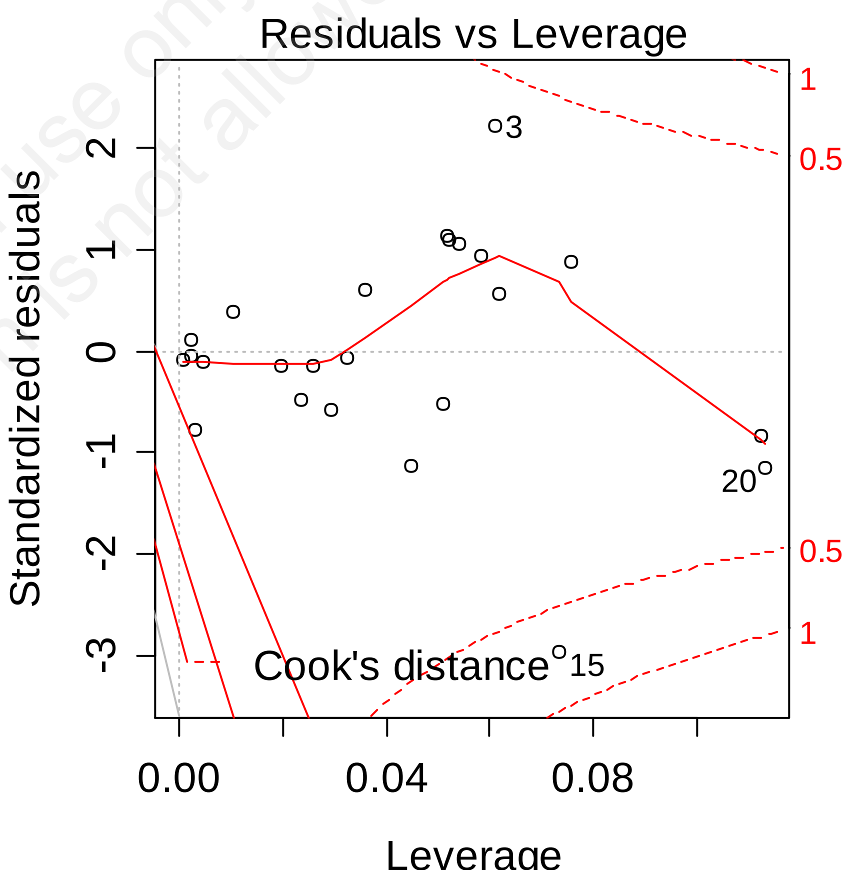
"Normal Q-Q" plot of residuals

- The standardized residuals are plotted against the quantiles of a standard normal distribution
- The resulting plot should look like a straight line
- In this case, we can see that the observations #3 and #15 distort the straight line relationship



"Residuals vs. Leverage" plot

- The standardized residuals are plotted against the *leverage* of each data point
- Leverage measures the *potential* of a data point to influence the overall model fit
- If a data point has a large residual in combination with a large leverage, it can be considered influential
- *Cook's distance* is a measure of the *actual influence* each data point has on the model fit
- If a point is outside of the Cook's distance-leverage contour line of ~ 0.5, that point is likely to be highly influential



Author: Sergey Mastitsky

Refitting the `hub.mod`

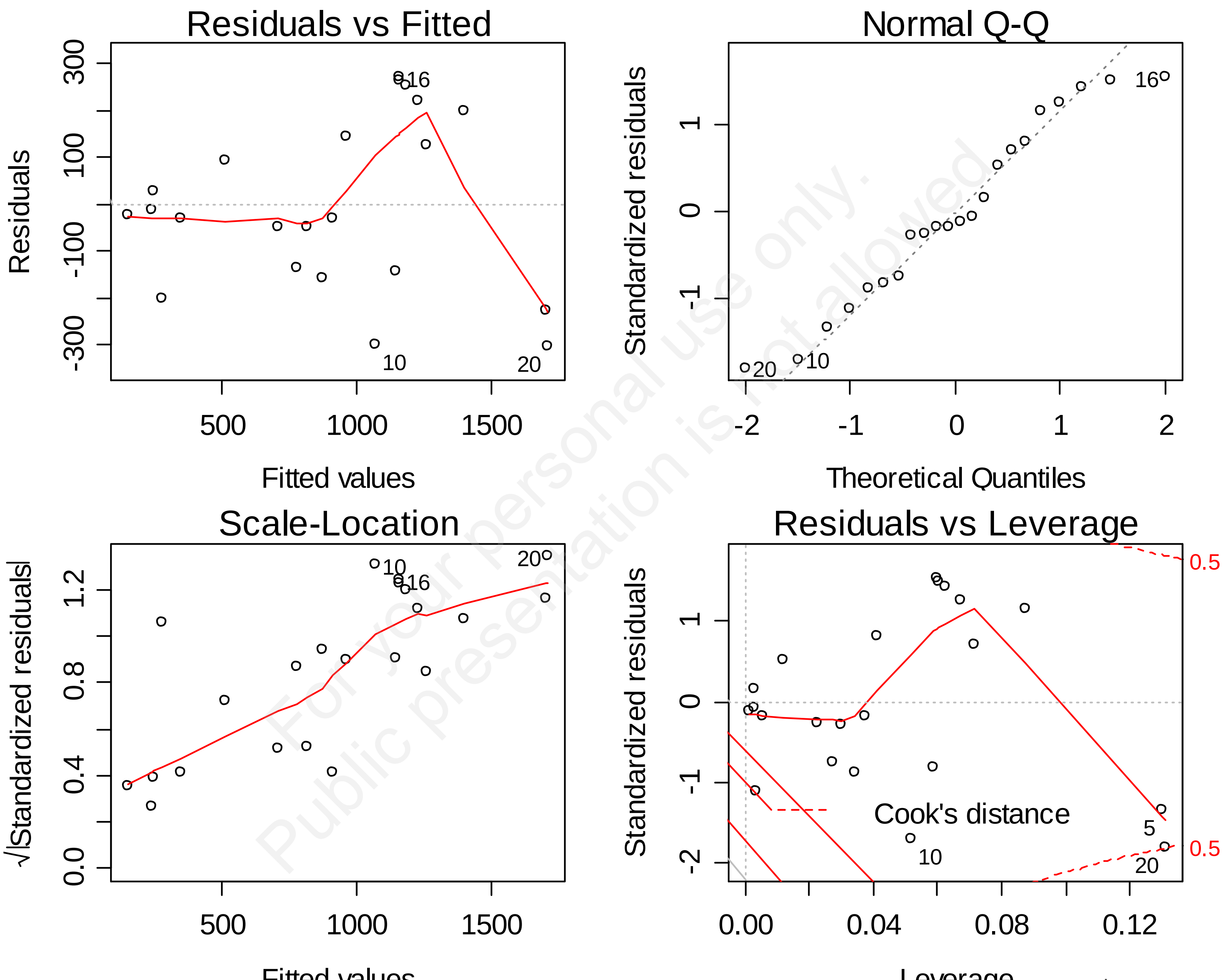
- Diagnostic plots suggest that the observations #3 and #15 have too much influence on the model fit. It would be prudent to refit the model without these offending points:

```
> hub.mod1 <- lm(y ~ x - 1,
    data = hub.data[-c(3, 15), ])
```

Summary on the hub.mod1

```
> summary(hub.mod1)

Call:
lm(formula = y ~ x - 1, data = hub.data[-c(3, 15), ])

Residuals:
   Min     1Q Median     3Q    Max 
-304.3 -141.9  -26.5  138.3  269.8 

Coefficients:
  Estimate Std. Error t value Pr(>|t|)    
x    77.67       2.97   26.15   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 180.5 on 21 degrees of freedom
Multiple R-squared: 0.9702, Adjusted R-squared: 0.9688
F-statistic: 683.8 on 1 and 21 DF,  p-value: < 2.2e-16
```

```
> plot(hub.mod1)
```



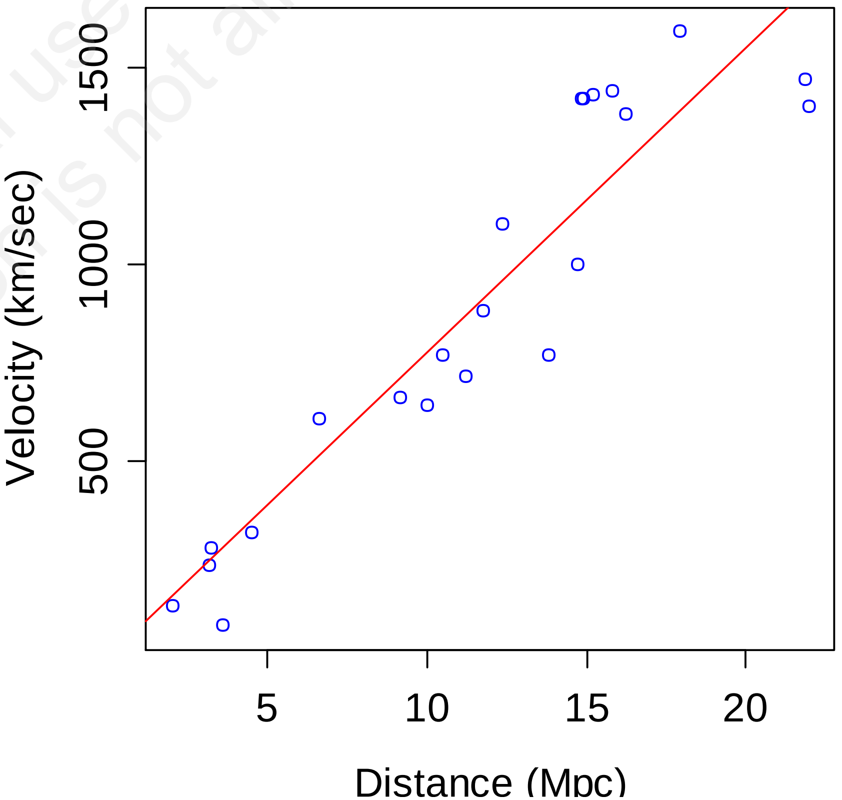
Author: Sergey Mastitsky

So, how old is the Universe?

- One Mega-parsec is 3.09×10^{19} km, so we need to divide β by this amount to obtain the Hubble's constant with units of s^{-1}:

```
> hub.const <-
  coef(hub.mod1)/3.09e19
# age in seconds:
> age <- 1/hub.const
# age in years:
> age/(60^2*24*365)
12614854757
```

- Answer: ~ 13 billion years



9. Simple linear regression

9.4. Confidence intervals of the regression coefficient

Confidence intervals for linear regression parameters

- Our estimate of the age of the Universe is based on a sample of 22 particular galaxies
- If we had a sample of different galaxies, the estimate would almost for sure be slightly different
- Thus, given the data we have, how can we estimate the range of possible true values of the age?
- Answer: calculate the confidence interval for the estimated Hubble's constant

Author: Sergey Mastitsky

Confidence interval for β

- Recall how we estimate the significance of the regression coefficient:

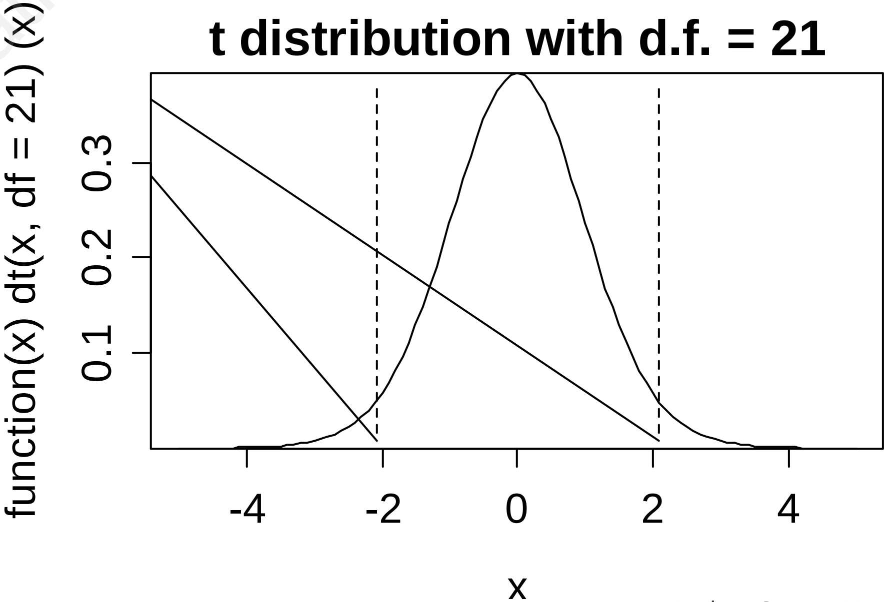
$$t = \frac{\hat{\beta} - \beta_0}{S_{\hat{\beta}}}$$

- We would reject H_0: $\beta_0 = 0$ if t was outside its acceptance region

Confidence interval for β

- The function **qt()** calculates the acceptance region of *t* for a certain significance level and d.f.:

```
> qt(p = c(0.025, 0.975), df = 21)
[1] -2.08  2.08
```



Confidence interval for β

Thus, we would accept any β_0 that fulfills

$$-2.08 \leq \frac{\hat{\beta} - \beta_0}{S_{\hat{\beta}}} \leq 2.08$$

which re-arranges to give the 95% confidence interval for β_0

$$\hat{\beta} - 2.08 S_{\hat{\beta}} \leq \beta_0 \leq \hat{\beta} + 2.08 S_{\hat{\beta}}$$

Confidence interval for the Hubble's constant

```
> bError <-
  summary(hub.mod1)$coefficients[2]
> ci <- coef(hub.mod1)+
  qt(c(0.025,0.975), df = 21)*bError
> ci
[1] 71.49588 83.84995
```

Confidence interval for the age of the Universe

```
> U.ci <-
  ci*60^2*24*365.25/3.09e19
> 1/U.ci
[1] 13695361072 11677548698
```

or better yet

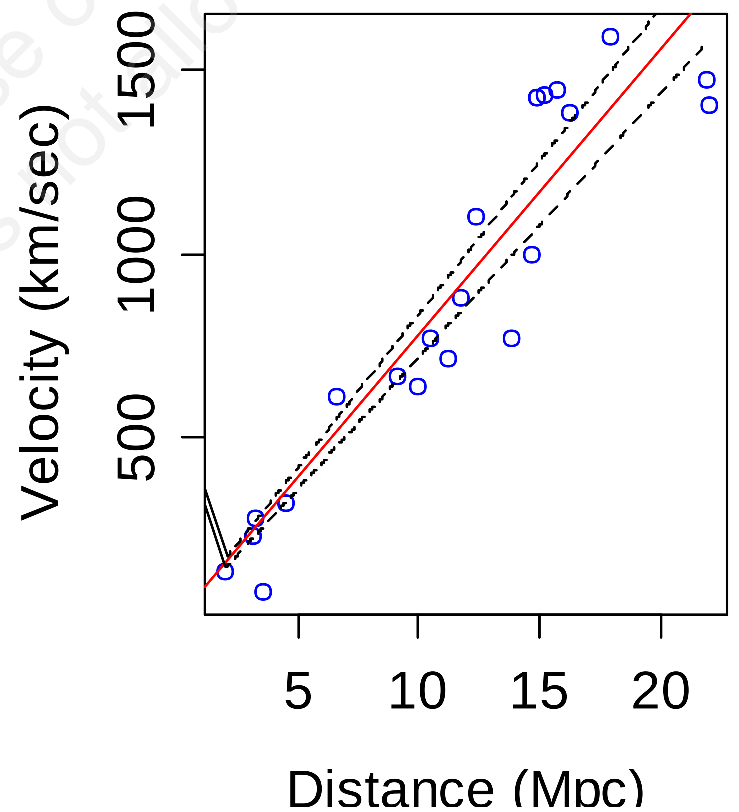
```
> sort(1/U.ci)
[1] 11677548698  13695361072
```

Thus, with the probability of 95%, the true age of the Universe is within the interval of values from 11.7 to 13.7 billion years

9.5. Confidence bands of the regression line

Confidence bands of the regression line

- As we have just seen, the regression coefficient cannot be estimated exactly
- Thus, there is an uncertainty about how the true regression line goes
- This uncertainty is usually demonstrated graphically by plotting confidence bands at both sides of the regression line



Author: Sergey Mastitsky

Calculating and plotting confidence bands of the regression line

The function predict() makes all the magic; the command

```
> predict(hub.mod1)
```

would return just the model fitted values of y:

```
        1         2         4         5         6         7         8 
 155.3458  711.4839 1394.2289 1699.4834  250.1068  871.4901  912.6568 
        9        10        11        12        13        14        16 
 281.9527 1071.8863  776.7292  817.1191  515.7482 1181.4051 1154.2196 
       17        18        19        20        21        22        23 
1259.8547 1225.6786 1159.6567 1707.2507  960.0373  348.7514  244.6697 
       24 
1143.3453 
```

Calculating and plotting confidence bands of the regression line

We can pass a new, artificial dataset with many values to predict(), and also ask it to calculate the standard errors for each of the newly predicted values

```
> newdat <- data.frame(
x = seq(min(hub.data$x),
  max(hub.data$x), 0.05) )
> dim(newdat)
[1] 400  1
```

How to calculate and plot confidence bands of the regression line

Predict new values and their SE's:

```
> Prediction <- predict(hub.mod1,
  se.fit = TRUE, newdata = newdat)
```

Check the structure of the new object:

```
> str(Prediction)
```

```
List of 4
 $ fit           : Named num [1:400] 155 159 163 167 171 ...
  ..- attr(*, "names")= chr [1:400] "1" "2" "3" "4" ...
 $ se.fit        : num [1:400] 5.94 6.09 6.24 6.39 6.53 ...
 $ df            : int 21
 $ residual.scale: num 180
```

Plotting the regression line and its confidence bands

Plot the raw data:

```
> plot(hub.data$x, hub.data$y,
  col = "blue",
  xlab ="Distance (Mpc)",
  ylab = "Velocity (km/sec)")
```

Plot the fitted regression line:

```
> abline(hub.mod1)
```

Plotting the regression line and its confidence bands

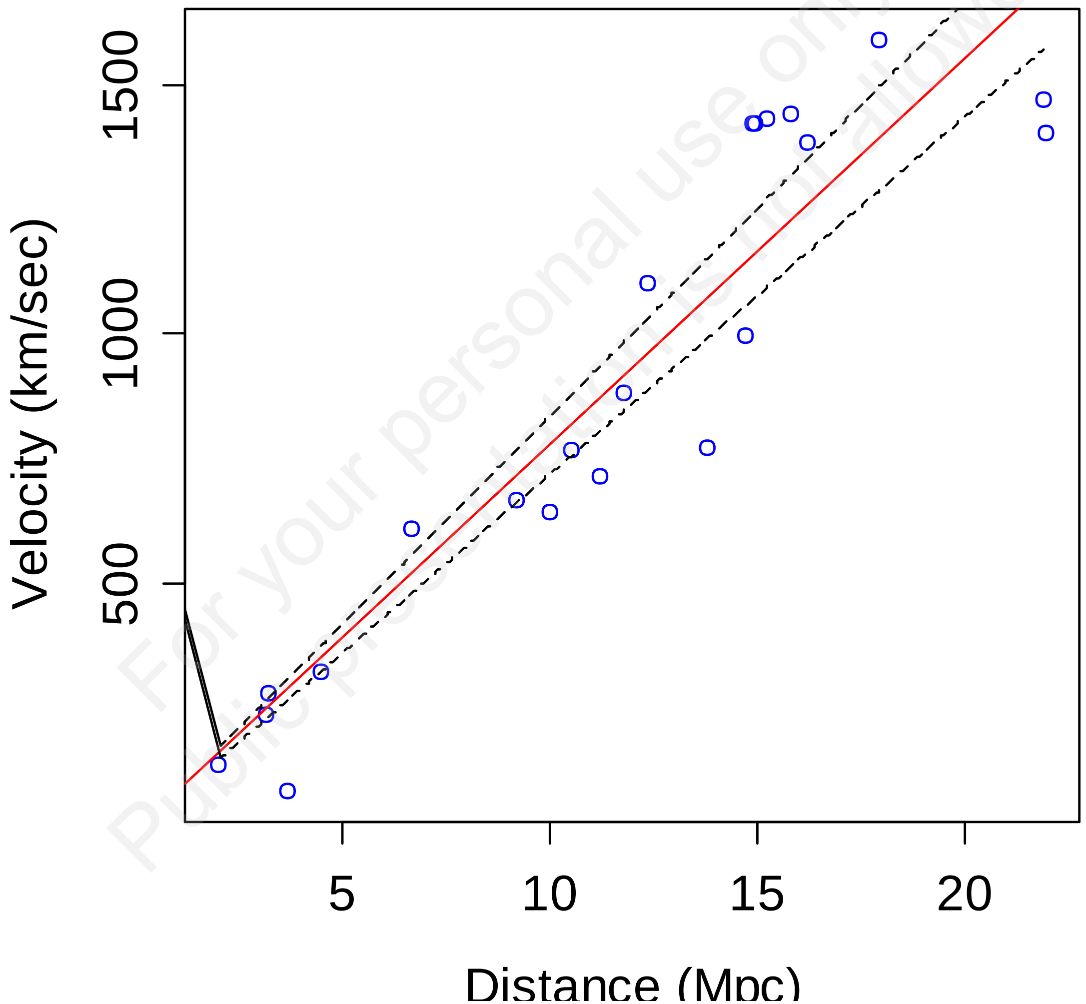
Plot the upper 95% confidence band:

```
> lines(newdat$x,
  Prediction$fit +
  1.96*Prediction$se.fit,
 lty = 2)
```

Plot the lower 95% confidence band:

```
> lines(newdat$x,
  Prediction$fit -
  1.96*Prediction$se.fit,
 lty = 2)
```

Regression line and its 95% confidence bands



Author: Sergey Mastitsky