V. Statistical analysis in R (presentation)

Data Science Lab, University of Copenhagen

14 March, 2022

Structure of a biostatistical analysis in R

The very basic structure of an R script doing a classical statistical analysis is as follows:

- Load packages that you will be using.

- Read the dataset to be analyzed. Possibly also do some data cleaning and manipulation.

- Visualize the dataset by graphics and other descriptive statistics.

- Fit and validate a statistical model.

- Hypothesis testing. Possibly also post hoc testing.

- Report model parameters and/or model predictions.

- Visualize your results.

Of course there are variants of this set-up, and in practice there will often be some iterations of the steps. In this manuscript we will exemplify the proposed steps in the analysis of two simple datasets:

- One-way ANOVA of the expression of the IGFL4 gene against the skin type in psoriasis patients.
- Multiple linear regression of volume of cherry trees against their diameter and height.

Load packages For both examples we will use **ggplot2** to make plots, and to be prepared for data manipulations, we simply load this together with the rest of the **tidyverse**. Moreover, we will also use extra plotting functionalities from the **GGally** package.

The psoriasis data are provided in an Excel sheet, so we also load **readxl**. Finally, we will use the package **emmeans** to make post hoc tests and to report model predictions.

Remember that you should install the wanted packages before they can be used (but you only need to install the packages once!).

Thus,

```r
#install.packages("tidyverse")
#install.packages("GGally")
#install.packages("readxl")
#install.packages("emmeans")
library(tidyverse)
library(GGally)
library(readxl)
library(emmeans)
```

Now we have done step 1 for both analyses. Next we will look specifically at the two examples. Finally, we conclude with a brief outlook on other statistical models in R.

Example A: Analysis of variance

Step 1: Data   Psoriasis is an immune-mediated disease that affects the skin. Researchers carried out a micro-array experiment with skin from 37 people in order to examine a potential association between the disease and a certain gene (IGFL4). For each of the 37 samples the gene expression was measured. Fifteen skin samples were from psoriasis patients and from a part of the body affected by the disease (`psor`); 15 samples were from psoriasis patients but from a part of the body not affected by the disease (`psne`); and 7 skin samples were from healthy people (`control`).

The data are saved in the file **psoriasis.xlsx**. At first the variable `type` is stored as a character variable, we change it to a factor (and check that indeed there are 15, 15 and 7 patients in the three groups).

```r
psoriasisData <- read_excel("psoriasis.xlsx")

# Dataset1
psorData  <- select(psoriasisData, type, IGFL4)

# Dataset2
psorDataG <- select(psoriasisData, -IGFL4)


psorData <- mutate(psorData, type = factor(type))
count(psorData, type)
```

```
## # A tibble: 3 x 2
##   type        n
##   <fct>   <int>
## 1 healthy     7
## 2 psne       15
## 3 psor       15
```

Step 2: Descriptive plots and statistics   To get an impression of the data we make two plots and compute group-wise means and standard deviations.

```r
ggplot(psorData, aes(x=type, y=IGFL4)) +
  geom_point() +
  labs(x="Skin type", y="IGFL4")

ggplot(psorData, aes(x=type, y=IGFL4)) +
  geom_boxplot() +
  labs(x="Skin type", y="IGFL4")

psorData %>%
  group_by(type) %>%
  summarise(avg=mean(IGFL4), median=median(IGFL4), sd=sd(IGFL4))
```

```
## # A tibble: 3 x 4
##   type      avg median    sd
##   <fct>   <dbl>  <dbl> <dbl>
## 1 healthy 1.34    1.27 0.230
## 2 psne    1.39    1.32 0.363
## 3 psor    0.955  0.909 0.255
```
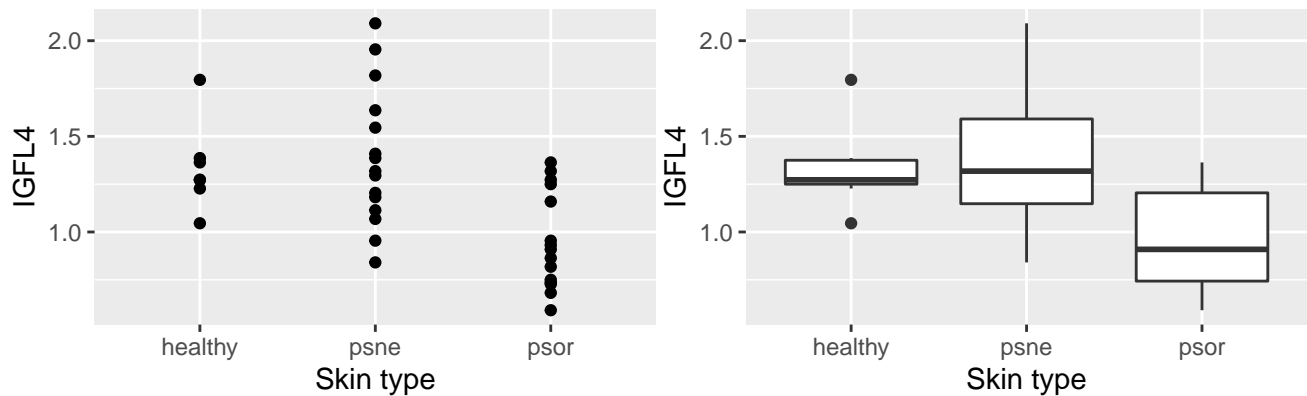
Step 3: Fit of oneway ANOVA, model validation The scientific question is whether the gene expression level of IGFL4 differs between the three types/groups. Thus, the natural type of analysis is a oneway analysis of variance (ANOVA). The oneway ANOVA is fitted with `lm`. It is a good approach to assign a name (below *oneway*) to the object with the fitted model. This object contains all relevant information and may be used for subsequent analysis. Note that we have logarithmic transformed the response as intensities are often on a multiplicative scale.

```
oneway <- lm(log(IGFL4) ~ type, data=psorData)
oneway
```

```
##
## Call:
## lm(formula = log(IGFL4) ~ type, data = psorData)
##
## Coefficients:
## (Intercept)      typepsne      typepsor
##     0.27910       0.01724      -0.35794
```

Step 4: Hypothesis test + Post hoc tests It is standard to carry out an *F*-test for the overall effect of the explanatory (i.e. independent) variable. To be precise, the hypothesis is that the expected values are the same in *all* groups. The most easy way to do this test is to use `drop1`. The option `test="F"` is needed to get the *F*-test:

```
drop1(oneway,test="F")
```

```
## Single term deletions
##
## Model:
## log(IGFL4) ~ type
##          Df Sum of Sq    RSS     AIC F value    Pr(>F)
## <none>                2.0924 -100.286
## type      2    1.2204 3.3128  -87.286  9.9153 0.0004052 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Thus, the overall test for homogeneity between the groups show, that the groups are not all the same. But it might be that the gene expression in two of the three groups, say, are not significantly different. To investigate that we do post hoc testing. This is nicely done within the framework of *estimated marginal means* using the **emmeans** package. Here `emmeans` makes the estimated marginal means (that is the predicted gene expression IGFL4 on the log scale), and the `pairs` command provide post hoc pairwise comparisons (package automatically adjusts for multiple comparisons using the default tukey method):

```
emmeans(oneway,~type)
```

```
##   type       emmean      SE df lower.CL upper.CL
##   healthy   0.2791 0.0938 34   0.0885    0.4696
##   psne      0.2963 0.0641 34   0.1662    0.4265
##   psor     -0.0788 0.0641 34  -0.2090    0.0513
##
## Results are given on the log (not the response) scale.
## Confidence level used: 0.95
```

```
pairs(emmeans(oneway,~type))
```

```
##   contrast        estimate     SE df t.ratio p.value
##   healthy - psne   -0.0172 0.1136 34  -0.152  0.9874
##   healthy - psor    0.3579 0.1136 34   3.152  0.0092
##   psne - psor       0.3752 0.0906 34   4.142  0.0006
##
## Results are given on the log (not the response) scale.
## P value adjustment: tukey method for comparing a family of 3 estimates
```

Step 5: Report of model parameters In this example, it is natural to report the estimated intensities as well as the comparisons between the 3 groups. As the analysis was done using a log transformation it is also advisable to backtransform. Both things can be done automatically by the **emmeans** package, where the option `type="response"` (here `type` is the name of an option inside the R command, and it is a coincidence that it has the same name as the variable *type*) requests the backtransformation:

```
emmeans(oneway,~type,type="response")
```

```
##   type     response      SE df lower.CL upper.CL
##   healthy    1.322 0.1239 34    1.093     1.60
##   psne       1.345 0.0861 34    1.181     1.53
##   psor       0.924 0.0592 34    0.811     1.05
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
```

```
confint(pairs(emmeans(oneway,~type,type="response")))
```

```
##   contrast        ratio     SE df lower.CL upper.CL
##   healthy / psne  0.983 0.112 34    0.744     1.30
##   healthy / psor  1.430 0.162 34    1.083     1.89
##   psne / psor     1.455 0.132 34    1.166     1.82
##
## Confidence level used: 0.95
## Conf-level adjustment: tukey method for comparing a family of 3 estimates
## Intervals are back-transformed from the log scale
```

```
pairs(emmeans(oneway,~type,type="response"))
```

```
##   contrast        ratio     SE df null t.ratio p.value
##   healthy / psne  0.983 0.112 34    1  -0.152  0.9874
##   healthy / psor  1.430 0.162 34    1   3.152  0.0092
##   psne / psor     1.455 0.132 34    1   4.142  0.0006
##
## P value adjustment: tukey method for comparing a family of 3 estimates
## Tests are performed on the log scale
```

Although we don't recommend usage of standard errors in the context of backtransformed parameters, we see that a standard error is provided on the backtransformed parameters.

Example B: Simple and multiple linear regression

Step 1: Data   For the first part of the presentation we will use data from 31 cherry trees available in the built-in dataset `trees`: Girth, height and tree volume have been measured for each tree. Please note, that the help page `?trees` states, that girth is the tree diameter. Interest is in the association between tree volume on the one side, and girth and height on the other side.

```
data(trees)
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
## 7   11.0     66   15.6
## 8   11.0     75   18.2
## 9   11.1     80   22.6
## 10  11.2     75   19.9
## 11  11.3     79   24.2
## 12  11.4     76   21.0
## 13  11.4     76   21.4
## 14  11.7     69   21.3
## 15  12.0     75   19.1
## 16  12.9     74   22.2
## 17  12.9     85   33.8
## 18  13.3     86   27.4
## 19  13.7     71   25.7
## 20  13.8     64   24.9
## 21  14.0     78   34.5
## 22  14.2     80   31.7
## 23  14.5     74   36.3
## 24  16.0     72   38.3
## 25  16.3     77   42.6
## 26  17.3     81   55.4
## 27  17.5     82   55.7
## 28  17.9     80   58.3
## 29  18.0     80   51.5
## 30  18.0     80   51.0
## 31  20.6     87   77.0
```

```
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
```
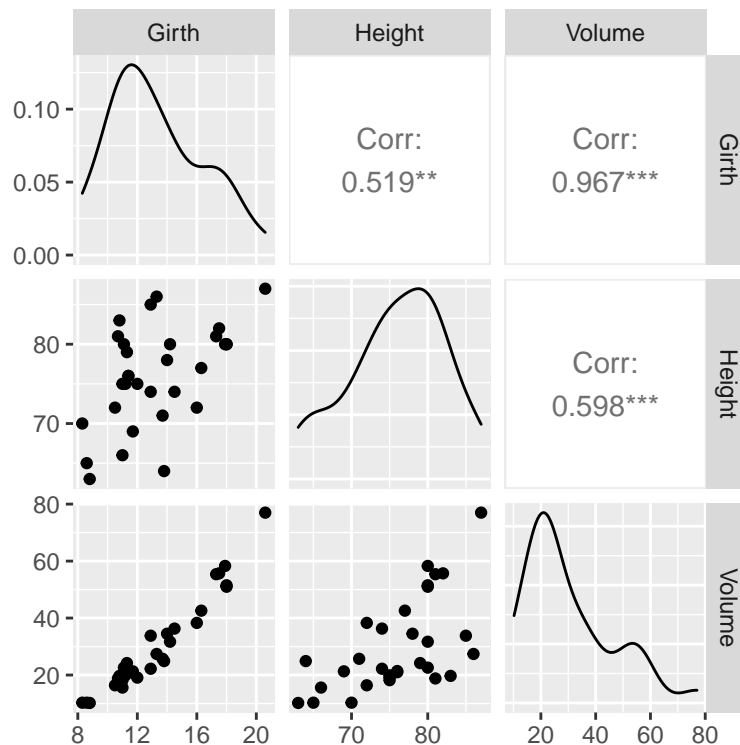
```
plot(trees)
```
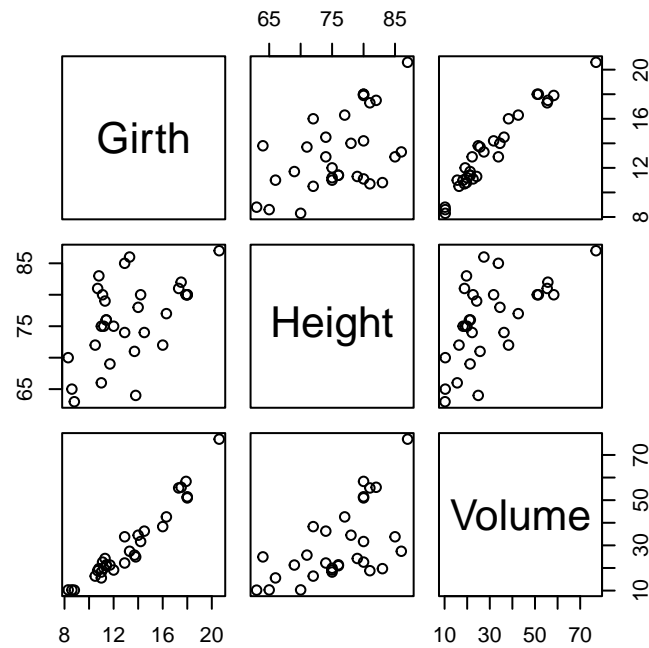

Step 2: Visualization of raw data

A more fancy version of this is available in **GGally**, where the diagonal is used to visualize the marginal distribution of the three tree variables:

```
ggpairs(trees)
```


Step 3: Fitting and validating a simple linear regression model   We start out with a simple linear regression with volume as outcome (i.e. dependent variable) and girth as covariate (i.e. independent variable).

This model is fitted with the `lm` function. It is a good approach to assign a name (below *linreg1*) to the object with the fitted model. This object contains all relevant information and may be used for subsequent analysis.

```
linreg1 <- lm(Volume ~ Girth, data=trees)
linreg1
```

```
##
## Call:
## lm(formula = Volume ~ Girth, data = trees)
##
## Coefficients:
## (Intercept)        Girth
##     -36.943        5.066
```

Mathematically this model makes assumptions about linearity, variance homogeneity, and normality. These assumptions are checked visually via four plots generated from the model residuals. In R, it is easy to make these plots; simply apply `plot` on the `lm`-object.

```
par(mfrow=c(2,2))    # makes room for 4=2x2 plots!
plot(linreg1)
```
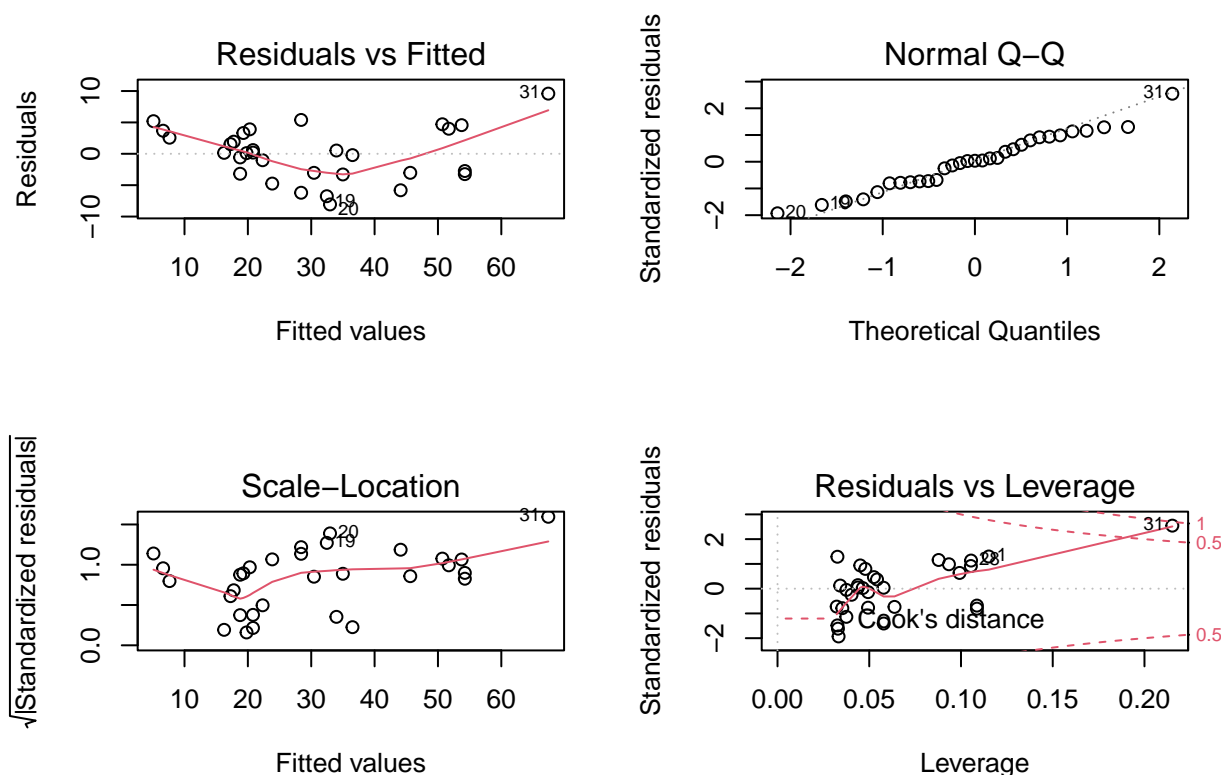

In this case there appears to be problems with the model. In particular, the upper left plot shows a quadratic tendency suggesting that the linearity assumption is not appropriate. Datapoint 31 seems to be an outlier as well. There seems to be some potential skewness. Perhaps a data transformation can help us out.

Step 4 iterated: Transformation The linear regression model above says that an increment of Δ_g of girth corresponds to an increment of volume by $\beta\Delta_g$ where β is the slope parameter (i.e. the coefficient), and hence, the interpretation is concerned with *absolute changes*. Perhaps it is more reasonable to talk about *relative changes*: An increment of girth by a factor c_g corresponds to an increment of volume of γc_g. This corresponds to a linear regression model on the log-transformed variables, which is easily fitted. Notice that `log` in R is the natural logarithm.

```
linreg2 <- lm(log(Volume) ~ log(Girth), data=trees)
par(mfrow=c(2,2))
plot(linreg2)
```
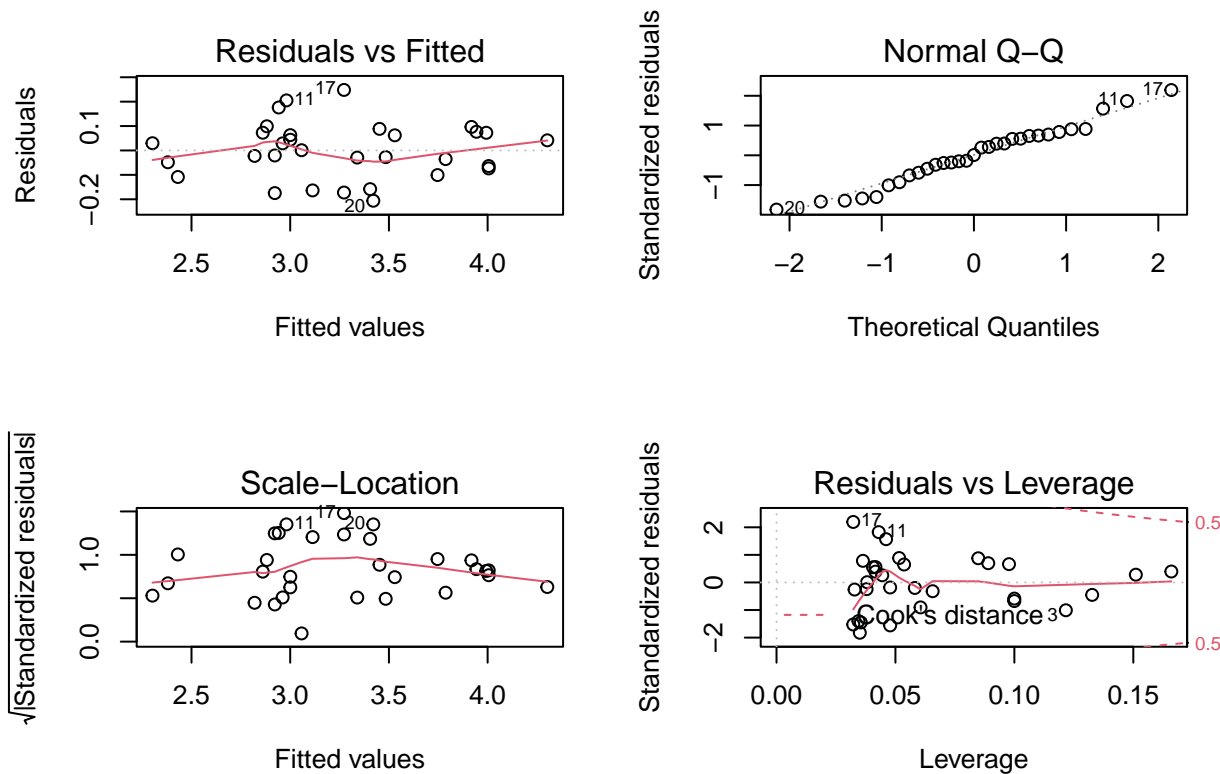

This model looks better! Residuals vs. fitted values as well as no outliers in the leverage plot!

Step 5: Report of the model Parameter estimates, standard errors are computed and certain hypothesis tests are carried out by the `summary` function:

```
summary(linreg2)
```

```
##
## Call:
## lm(formula = log(Volume) ~ log(Girth), data = trees)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.205999  -0.068702   0.001011   0.072585   0.247963
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.35332    0.23066  -10.20 4.18e-11 ***
## log(Girth)   2.19997    0.08983   24.49  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.115 on 29 degrees of freedom
## Multiple R-squared:  0.9539, Adjusted R-squared:  0.9523
## F-statistic: 599.7 on 1 and 29 DF,  p-value: < 2.2e-16
```

The coefficient table is the most important part of the output. It has two lines — one for each parameter in

the model — and four columns. The first line is about the intercept, the second line is about the slope. The columns are

- `Estimate`: The estimated value of the parameter (intercept or slope).
- `Std. Error`: The standard error (estimated standard deviation) associated with the estimate in the first column.
- `t value`: The t-test statistic for the hypothesis that the corresponding parameter is zero. Computed as the estimate divided by the standard error.
- `Pr(>|t|)`: The p-value associated with the hypothesis just mentioned. In particular the p-value in the second line is for the null hypothesis that there is no effect of girth on volume.

If you would like to report confidence intervals for the parameters (intercept and slope), then use `confint`:

`confint(linreg2)`

```
##                 2.5 %     97.5 %
## (Intercept) -2.825083 -1.881566
## log(Girth)   2.016238  2.383702
```

Finally, don't forget to make the mathematical interpretation of the model and its parameter estimates! Thus, the model postulates the relation

$$\ln V \approx \alpha + \beta \ln G$$

with estimates $\hat{\alpha} = -2.353$ and $\hat{\beta} = 2.200$. This corresponds to $V \approx e^{\alpha} \cdot G^{\beta}$, and we see that an increment of girth by 10%, say, corresponds to an increment of volume by a factor of $1.1^{2.2} = 1.23$, that is, by 23%.

Step 6: Visualization of the model   An excellent synthesis of a statistical analysis is often provided by a plot combining the raw data with the model fit.

The estimated regression line can be added to a scatter plot in different ways (and with/without corresponding standard error curves):
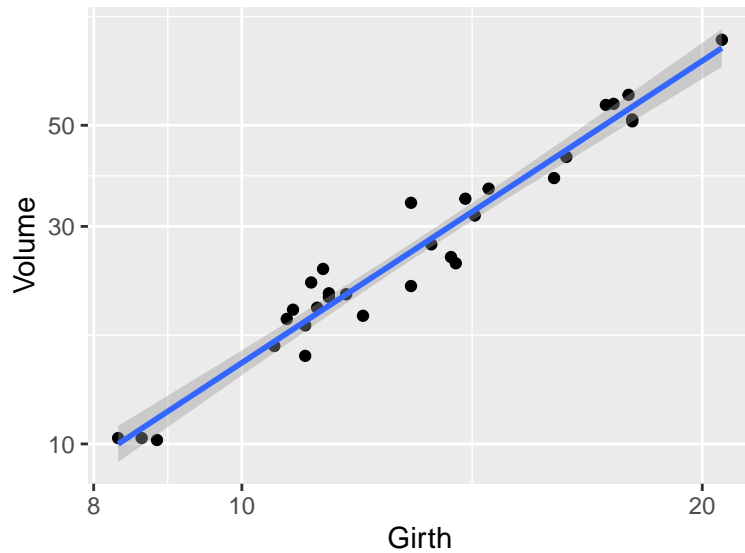
```
ggplot(trees, aes(x = log(Girth), y = log(Volume))) +
  geom_point() +
  geom_abline(intercept=-2.35332, slope=2.19997, col="red")
```
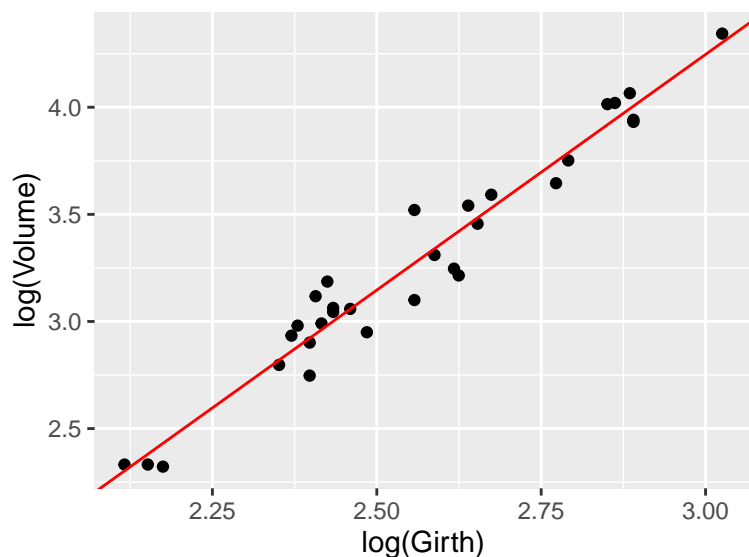


```
ggplot(trees, aes(x = Girth, y = Volume)) +
  geom_point() +
  geom_smooth(method = "lm", se = TRUE) +
```

```
  scale_x_log10() +
  scale_y_log10()
```

## `geom_smooth()` using formula 'y ~ x'

Step 7 iterated: Multiple linear regression Several covariates can be included in a regression model, corresponding to multiple linear regression; simply write a plus between the covariates. Here is model fit and validation with log-girth as well as log-height included in the model:

```
linreg3 <- lm(log(Volume) ~ log(Girth) + log(Height), data=trees)
summary(linreg3)$coefficients
```

```
##                Estimate Std. Error    t value      Pr(>|t|)
## (Intercept) -6.631617 0.79978973 -8.291701 5.057138e-09
## log(Girth)   1.982650 0.07501061 26.431592 2.422550e-21
## log(Height)  1.117123 0.20443706  5.464388 7.805278e-06
```

Don't forget to do model validation:

```
par(mfrow=c(2,2))
plot(linreg3)
```
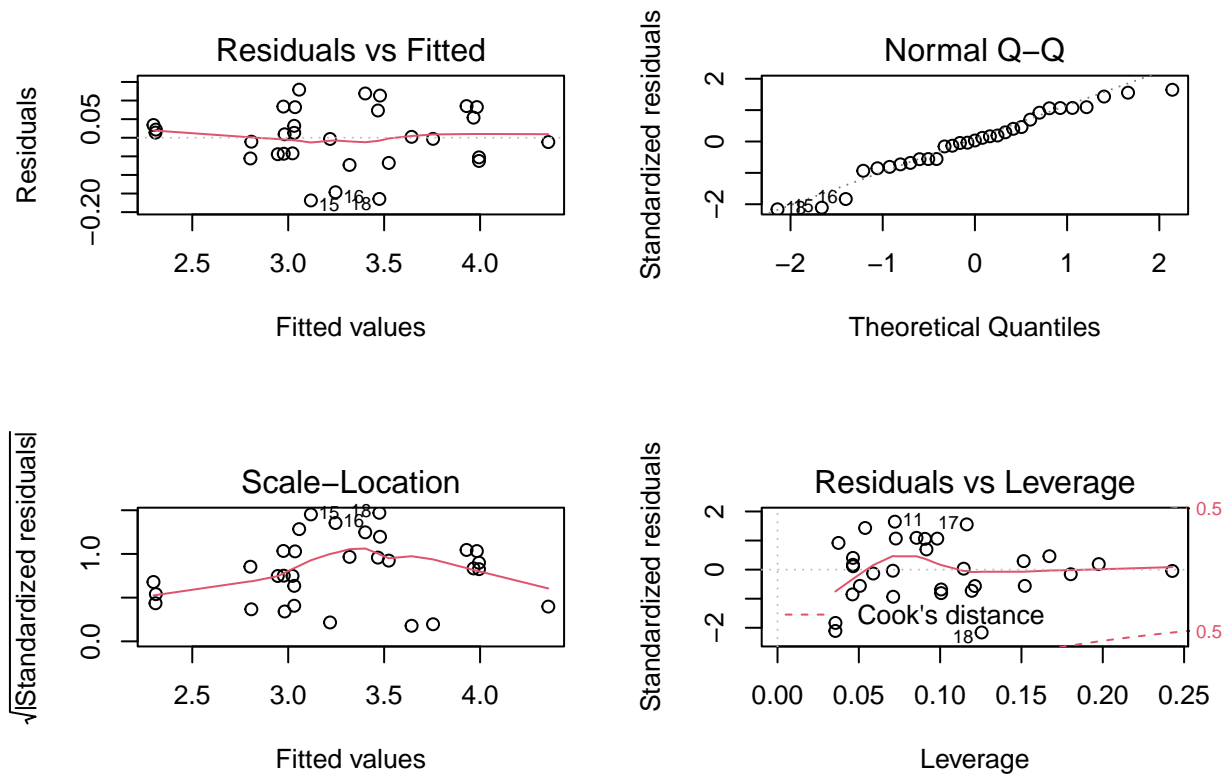
Step 8: Hypothesis test   The *F*-test for comparison of two nested models may be carried out by the `anova` function. For example, `linreg3` is nested in `linreg2`, and the comparison between the two corresponds to the null hypothesis that there is no extra information provided by height when girth is included (on log-scale) in the model.

```
anova(linreg3, linreg2)
```

```
## Analysis of Variance Table
##
## Model 1: log(Volume) ~ log(Girth) + log(Height)
## Model 2: log(Volume) ~ log(Girth)
##   Res.Df     RSS Df Sum of Sq     F    Pr(>F)
## 1     28 0.18546
## 2     29 0.38324 -1  -0.19778 29.86 7.805e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

However, as an alternative to `anova` it is more pratical to use the `drop1` command, which makes all the tests corresponding to removal of one of the explanatory variables (i.e. independent variables). If you want *p*-values, then ask for *F*-tests:

```
drop1(linreg3,test="F")
```

```
## Single term deletions
##
## Model:
## log(Volume) ~ log(Girth) + log(Height)
##           Df Sum of Sq    RSS     AIC F value    Pr(>F)
## <none>                 0.1855 -152.685
## log(Girth)  1    4.6275 4.8130  -53.743  698.63 < 2.2e-16 ***
```

11

```
## log(Height)  1    0.1978 0.3832 -132.185   29.86 7.805e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that `anova` and `drop1` give the same p-value ($p = 7.8 * 10^{-6}$) for the effect of log(Height). And in a Gaussian model (like this) this coincides with the p-value from the t-test for the null hypothesis that there is no effect of a single numerical covariate i.e. height in our example. Thus,

```
summary(linreg3)$coefficients
```

```
##                 Estimate Std. Error    t value      Pr(>|t|)
## (Intercept) -6.631617 0.79978973 -8.291701 5.057138e-09
## log(Girth)   1.982650 0.07501061 26.431592 2.422550e-21
## log(Height)  1.117123 0.20443706  5.464388 7.805278e-06
```

So should you do all three tests? Definitely not!

- In general, it is not recommendable to use `summary` for doing tests: The F-tests and t-tests only coincide for hypothesis on a single parameter. And you may also be deceived by the model parametrizations when interpreting t-tests.
- To use `drop1` you only need to fit the large model (here *linreg3*). Thus, it gives shorter and more easily readable R code.

Outlook: Other analyses

The `lm` function is used for linear models, that is, models where data points are assumed to be independent with a Gaussian (i.e. normal) distribution (and typically also with the same variance). Obviously, these models are not always appropriate, and there exists functions for many, many more situations and data types. Here, we just mention a few functions corresponding to common data types and statistical problems.

- `glm`: For independent, but non-Gaussian data. Examples are binary outcomes (logistic regression) and outcomes that are counts (Poisson regression). glm is short for generalized linear models, and the `glm` function is part of the base installation of R.
- `lmer` and `glmer`: For data with dependence structures that can be described by random effects, e.g., block designs. lme is short for linear mixed effects (Gaussian data), glmer is short for generalized linear mixed effects (binary or count data). Both functions are part of the **lme4** package.
- `nls`: For non-linear regression, e.g., dose-response analysis. nls is short for non-linear least squares. The function is included in the base installation of R.

The functions mentioned above are used in a similar way as `lm`: a model is fitted with the function in question, and the model object subsequently examined with respect to model validation, estimation, computation of confidence limits, hypothesis tests, prediction, etc. with functions `summary`, `confint`, `drop1`, `emmeans`, `pairs` as indicated above.