Laboratory 3 - Statistical Methods

Linear regression model

Torelli N, Di Credico G, Gioia V

10/11/2023

Contents

Introduction to regression and diagnostic checks

Regression is trying to describe a **dependent variable y** through some statistically significant **predictors x**, linking them through a function $f(\cdot)$ (also referred as systematic component):

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\varepsilon},$$

where $\boldsymbol{\varepsilon}$ is a measure of *noise* implicit in modelling (also stochastic component).

Checking the assumptions of a regression model is not just a theoretical matter, but it is something related to *scientific reproducibility*, *scientific transparency* and *visualization power*. Although it is impossible to write down an exhaustive list of 'what to do for checking a model', we put here just some initial steps for visualising and inspecting your variables, as well as possible correlations between them and the inclusion of further predictors. The suggested steps refer to the **classical linear model**, $f(\mathbf{x}) = f(\mathbf{x}; \boldsymbol{\beta}) = \mathbf{X}\boldsymbol{\beta}$, but they contain principles valid for statistical modelling in general.

Before fitting a model

- Examine the distribution of each of the explanatory variables, and of the dependent variable. Thus, skewness and presence of outliers can be detected. When the distribution is skew, consider transformations inducing symmetry.
- Examine the scatterplot involving all the explanatory variables. If some pairwise plot shows non-linearity, consider some possible transformations.
- Examine the range of the variables.
- Examine the degree of correlation between the explanatory variables.

After fitting a model

- Plot residuals against fitted values and check for patterns in the residual. Also, check for homoscedasticity.
- Examine the Cook's distance for possible outliers.

Linear regression in practice

We consider the dataset `nihills` available in the `DAAG` package, containing data from the 2007 calendar for the Northern Ireland Mountain Running Association. The number of observations (races) is $n = 23$, and for each of them we register the following variables:

- Distances in miles: `dist`
- Amount of climbs expressed in feet: `climb`
- Record time in hours for males: `time`
- Record time in hours for females: `timef`.

Objective: We want to regress the times through some predictors.

In the following, we focus on the time results for males. Then, we load the data.

```
library(DAAG)
data(nihills)
n <- nrow(nihills)
```

We explore the data structure and the first six rows by means of `str`() and `head`() function, respectively.

```
str(nihills)
```

```
## 'data.frame':    23 obs. of  4 variables:
##  $ dist : num  7.5 4.2 5.9 6.8 5 4.8 4.3 3 2.5 12 ...
##  $ climb: int  1740 1110 1210 3300 1200 950 1600 1500 1500 5080 ...
##  $ time : num  0.858 0.467 0.703 1.039 0.541 ...
##  $ timef: num  1.064 0.623 0.887 1.214 0.637 ...
```

```
head(nihills)
```

```
##                    dist climb      time     timef
## Binevenagh          7.5  1740 0.8583333 1.0644444
## Slieve Gullion      4.2  1110 0.4666667 0.6230556
## Glenariff Mountain  5.9  1210 0.7030556 0.8869444
## Donard & Commedagh  6.8  3300 1.0386111 1.2141667
## McVeigh Classic     5.0  1200 0.5411111 0.6375000
## Tollymore Mountain  4.8   950 0.4833333 0.5886111
```

A summary report of the variables allows exploring the range of the variables and the skewness.

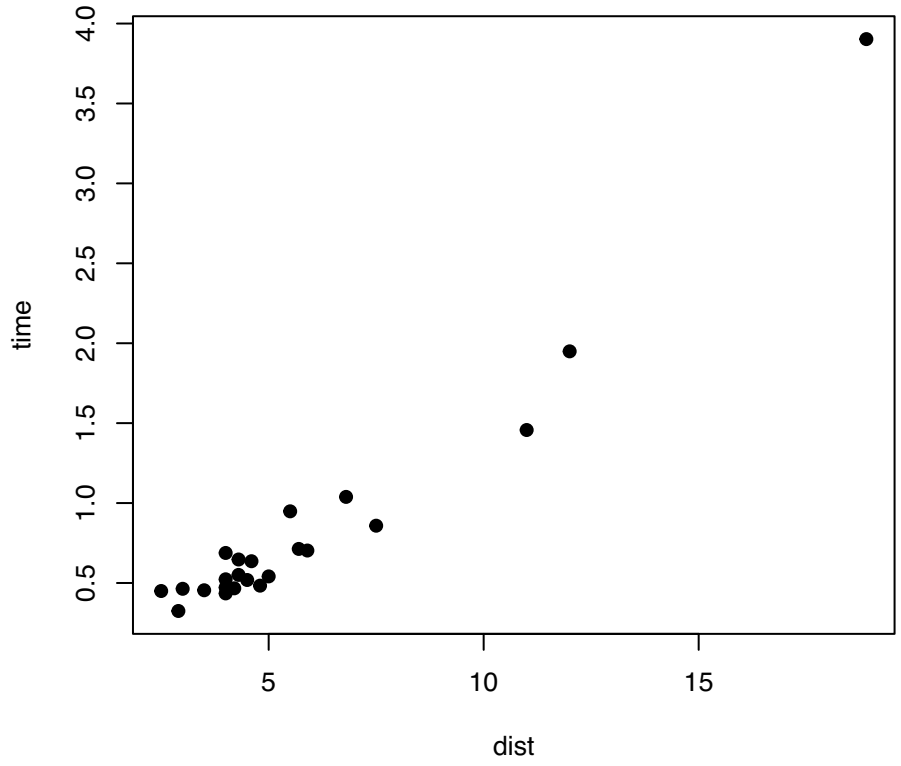
```
summary(nihills)
```

```
##       dist            climb           time            timef
##  Min.   : 2.500   Min.   : 750   Min.   :0.3247   Min.   :0.4092
##  1st Qu.: 4.000   1st Qu.:1205   1st Qu.:0.4692   1st Qu.:0.6158
##  Median : 4.500   Median :1500   Median :0.5506   Median :0.7017
##  Mean   : 5.778   Mean   :2098   Mean   :0.8358   Mean   :1.1107
##  3rd Qu.: 5.800   3rd Qu.:2245   3rd Qu.:0.7857   3rd Qu.:1.0014
##  Max.   :18.900   Max.   :8775   Max.   :3.9028   Max.   :5.9856
```

Simple Linear Regression

In a first step, we consider only the predictor `dist` for regressing the dependent variable `time`. Let us have a quick look to the data by means of a scatterplot:

```
plot(nihills$dist, nihills$time, pch = 19, xlab = "dist", ylab = "time")
```




```
# with(nihills, plot(dist, time, pch = 19))
```

Apart one, all the data times are lower than 2 hours. As an initial attempt, let's fit a simple linear model:

$$time_i = \beta_1 + \beta_2 dist_i + \varepsilon_i, \quad i = 1, \ldots, n, \quad \varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2).$$

```
lm_dist <- lm(time ~ dist, data=nihills)
```

The coefficient estimates (togheter with the information on the model fitted) can be printed simply typing

```
lm_dist
```

```
##
## Call:
## lm(formula = time ~ dist, data = nihills)
##
## Coefficients:
## (Intercept)         dist
##     -0.3253       0.2009
```

A more detailed summary of the fitting procedure can be obtained by means of:

```
summary(lm_dist)
```

```
##
## Call:
## lm(formula = time ~ dist, data = nihills)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42812 -0.12193 -0.00250  0.09232  0.43028
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.32530    0.07629  -4.264 0.000345 ***
## dist         0.20094    0.01120  17.934 3.28e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1935 on 21 degrees of freedom
## Multiple R-squared:  0.9387, Adjusted R-squared:  0.9358
## F-statistic: 321.6 on 1 and 21 DF,  p-value: 3.278e-14
```

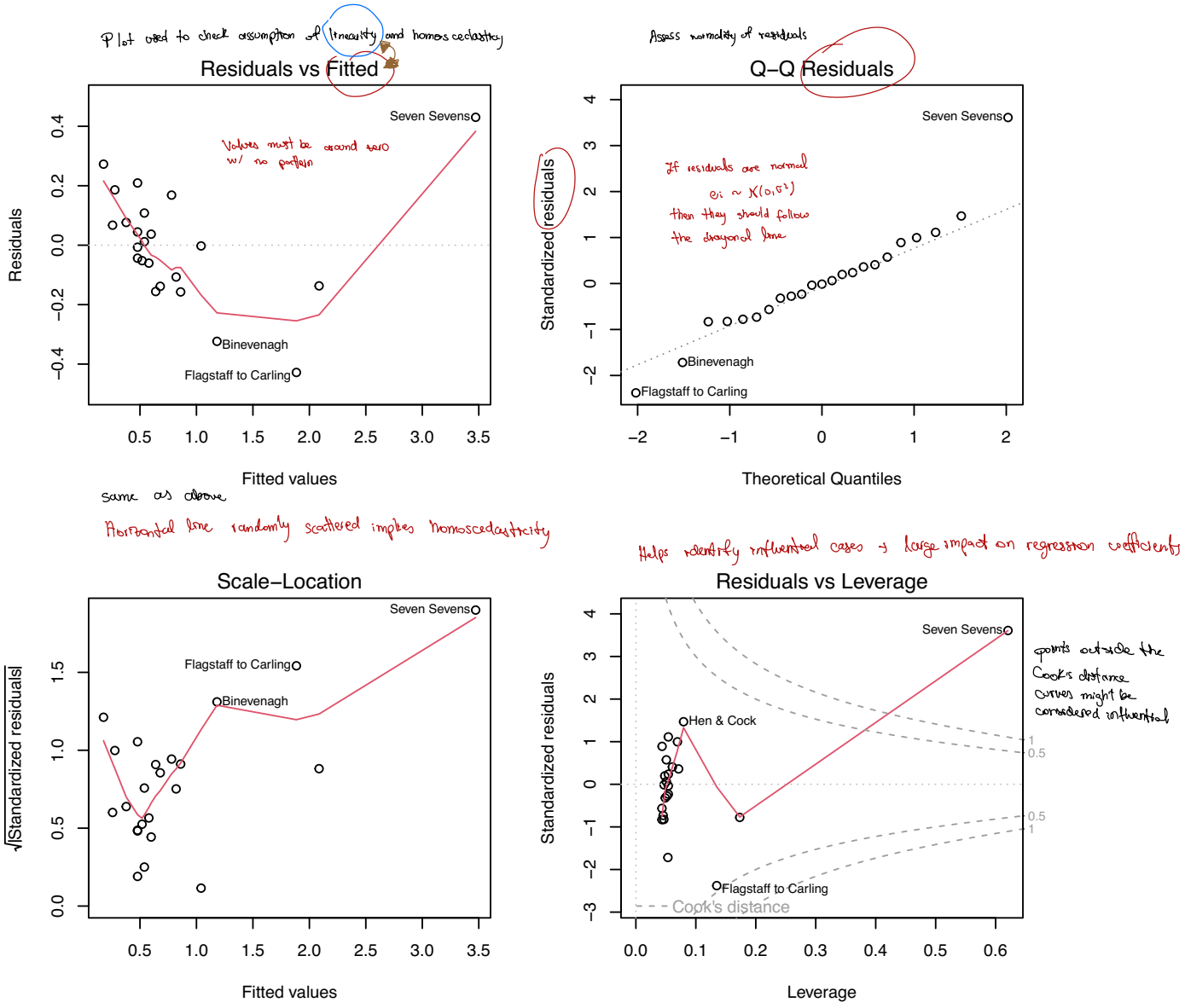
Quick question: What are the main information from the fitted model? Comment on the results of the model output

The call to the `summary()` function (on a lm object) produces:

- Call: the fitted model
- Residuals: Some summaries, mainly quantiles (useful for detecting asymmetry)
- Coefficients:
  - Estimates: coefficient estimates, $\hat{\beta}_j, j = 1, \ldots, p$
  - Std. Error: estimated standard deviation of the estimators $\hat{\beta}_j$, that is $\sqrt{\hat{V}(\hat{\beta}_j)}$
  - t value: the (observed) values of the test statistic for testing the nullity of the coefficients, $H_0 : \beta_j = 0$ against $H_1 : \beta_j \neq 0$, that is $t_j = \hat{\beta}_j / \sqrt{\hat{V}(\hat{\beta}_j)}$
  - $Pr(> |t|)$: the corresponding p-values, $2P(T_{n-p} > |t_j|)$
  - Some symbols denoting where the p-values fall, *** if $0 < p \leq 0.001$, ** if $0.001 < p \leq 0.01$, * if $0.01 < p \leq 0.05$, . if $0.05 < p \leq 0.1$, otherwise nothing
- Residual standard error: the square root of the unbiased sample variance of the residuals, that is $\sqrt{s^2}$, with the corresponding degrees of freedom $n - p$
- Multiple R-squared: the R^2 coefficient
- Adjusted R-squared: the adjusted R^2 coefficient (given by $R^2 - \frac{p-1}{n-p}(1 - R^2)$ and useful for the multiple linear regression)
- F-statistic: the observed value for the test statistic comparing the full model with the null model (that is the model including only the intercept), with the corresponding degrees of freedom and the related p-values. Namely, for testing $H_0 : \beta_2 = \ldots = \beta_p = 0$ against the alternative that at least one is significantly different from zero

Then, we can visualise some aspects of the fitted model. The first step, and likely the most important one, is to explore the residuals plots

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



Then, we can produce the scatterplot including the regression line

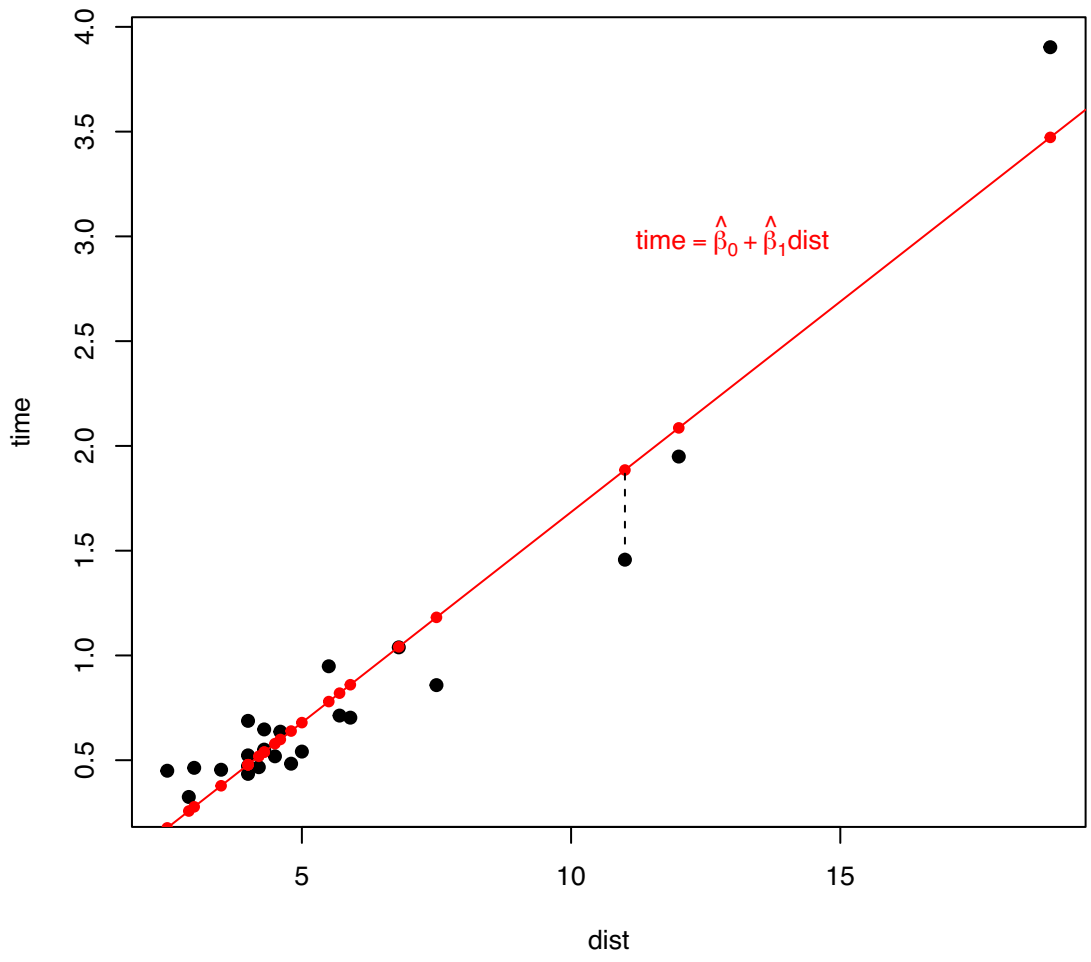
```
par(mfrow = c(1, 1))
with(nihills, plot(dist, time, pch = 19))
abline(coef(lm_dist), col = "red", lty = "solid")
# or
#curve(predict(lm_dist, data.frame(dist = x)), col = "red", lty = "solid", lwd = 2, add = TRUE)
text(13, 3, expression(time == hat(beta)[0] + hat(beta)[1] * dist), col = "red")
```

```r
points(nihills$dist, predict(lm_dist), col = "red", pch = 19, cex = 0.8)

nihills[17,]
```

```
##                       dist climb     time    timef
## Flagstaff to Carling   11  3000 1.456944 2.034444
```

```r
segments(nihills[17,]$dist, nihills[17,]$time,
         nihills[17,]$dist, fitted(lm_dist)[17], lty = "dashed")
```



Of course, the fitting seems good enough. However, the relationship between `time` and `dist` from the first plot above seems not to be purely linear...and residual plots seem to suggest a lack of fit for a few points.

Extra: the summary output on a `lm` object

Here, we obtain the quantities of the output of the `summary()` function applied to an `lm` object by hand. In the following, the computations are done using matrix algebra (albeit we are working with a simple linear model). In this way, the following lines of code can be used to (eventually) carry out some checks using the multiple linear model.

```
sum_lm_dist <- summary(lm_dist)

X <- model.matrix(~ dist, data = nihills)
y <- nihills$time
n <- nrow(X)
p <- ncol(X)
```

```
beta.hat <- solve(t(X) %*% X) %*% t(X) %*% y
beta.hat
```

```
##                   [,1]
## (Intercept) -0.3252964
## dist         0.2009417
```

```
sum_lm_dist$coefficients[,1] # Check
```

```
## (Intercept)        dist
##  -0.3252964   0.2009417
```

```
###################################

y.hat <- X %*% beta.hat
max(abs(predict(lm_dist) - y.hat)) # Check
```

```
## [1] 2.220446e-15
```

```
###################################

residuals <- y - y.hat
max(abs(residuals(lm_dist) - residuals)) # Check
```

```
## [1] 2.275957e-15
```

```
cbind(summary(residuals), summary(sum_lm_dist$residuals)) # Check
```

```
##        V1
##  "Min.   :-0.428118  " "-0.428117751172833"
##  "1st Qu.:-0.121926  " "-0.121926447481956"
##  "Median :-0.002496  " "-0.00249597962166161"
##  "Mean   : 0.000000  " "-3.2883144803112e-17"
##  "3rd Qu.: 0.092318  " "0.0923182594652581"
##  "Max.   : 0.430276  " "0.430276218211075"
```

```
###################################

s2 <- sum(residuals^2)/(n - p); s2
```

```
## [1] 0.03744742
```

```
sum_lm_dist$sigma^2 # Check
```

```
## [1] 0.03744742
```

```
################################
```

```
stdb <- sqrt(diag(s2 * solve(t(X) %*% X))); stdb
```

$\sqrt{S^2 \cdot (X^T X)^{-1}}$

```
## (Intercept)        dist
##  0.07628629  0.01120431
```

$\hat{\beta} \sim N(\beta_1, \hat{\sigma}^2 (X^T X)^{-1})$

```
sum_lm_dist$coefficients[, 2] ## Check
```

```
## (Intercept)        dist
##  0.07628629  0.01120431
```

```
################################
```

```
beta.hat/stdb
```

```
##                  [,1]
## (Intercept) -4.264153
## dist        17.934328
```

t-statistic

```
sum_lm_dist$coefficients[, 3]  ## Check
```

```
## (Intercept)        dist
##   -4.264153   17.934328
```

```
##########################
```

```
2 * pt(abs(beta.hat/stdb), df = n - p, lower.tail = FALSE)
```

```
##                      [,1]
## (Intercept) 3.454758e-04
## dist        3.278480e-14
```

p-value

```
sum_lm_dist$coefficients[,4] ## Check
```

```
##  (Intercept)         dist
## 3.454758e-04 3.278480e-14
```

```
##########################
```

```
Tot_SS <- sum((y - mean(y))^2)
Res_SS <- sum((y.hat - y)^2)
Mod_SS <- sum((y.hat - mean(y))^2)

R_sq <- Mod_SS/Tot_SS
R_sq
```

```
## [1] 0.9387112
```

R^2-squared

```
with(nihills, cor(time, dist))^2
```

```
## [1] 0.9387112
```

```
sum_lm_dist$r.squared  # Check
```

```
## [1] 0.9387112
```

```
#########################

1 - (Res_SS/(n - p))/(Tot_SS/(n - 1))
```

```
## [1] 0.9357927
```

```
R_sq - (1 - R_sq)*(p - 1)/(n - p)      # Check
```

```
## [1] 0.9357927
```

```
sum_lm_dist$adj.r.squared # Check
```

```
## [1] 0.9357927
```

Adjusted R^2-squared

```
############################
X0 <- X[, 1]
p0 <- 1
beta.hat0 <- solve(t(X0) %*% X0) %*% t(X0) %*% y
beta.hat0
```

$$\hat{\beta}_0 = (X_0^T X_0)^{-1} X_0^T y = (n)^{-1} \mathbb{1}^T y = \frac{\sum y}{n} = \bar{y}$$

```
##           [,1]
## [1,] 0.8357971
```

Useless

$$\hat{y}_0 = X^T \hat{\beta}_0 = \mathbb{1}^T \hat{\beta}_0 = \mathbb{1}^T \bar{y}$$

```
beta.hat0 <- mean(y)

y.hat0 <- mean(y)

residuals0 <- y - y.hat0

F <- ((sum(residuals0^2) - sum(residuals^2))/(p - p0))/(sum(residuals^2)/(n - p))
F
```

$$F = \frac{(SSE_0 - SSE_1)/(p - p_0)}{SSE_1/(n - p)}$$

Nested Model

```
## [1] 321.6401
```

```
sum_lm_dist$coefficients[2, 3]^2   # Check
```

```
## [1] 321.6401
```

```
sum_lm_dist$fstatistic # Check
```

```
##    value    numdf    dendf
## 321.6401   1.0000  21.0000
```

```
pf(F, p0, n - p, lower.tail = FALSE)
```

```
## [1] 3.27848e-14
```

Multiple linear regression

Polynomial regression

A natural way to introduce the multiple linear regression model is to expand the previous simple linear regression model `lm_dist` including a quadratic term of the variable `dist`, that is modelling the explanatory variable `dist` as a polynomial of degree two.

$$time_i = \beta_1 + \beta_2 dist_i + \beta_3 dist_i^2 + \varepsilon_i, \quad i = 1, \ldots, n, \quad \varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2).$$

The model could also be extended by considering polynomials of higher order, but let us start fitting the model above in R

```
lm_dist2 <- lm(time ~ dist + I(dist^2), data = nihills)
```

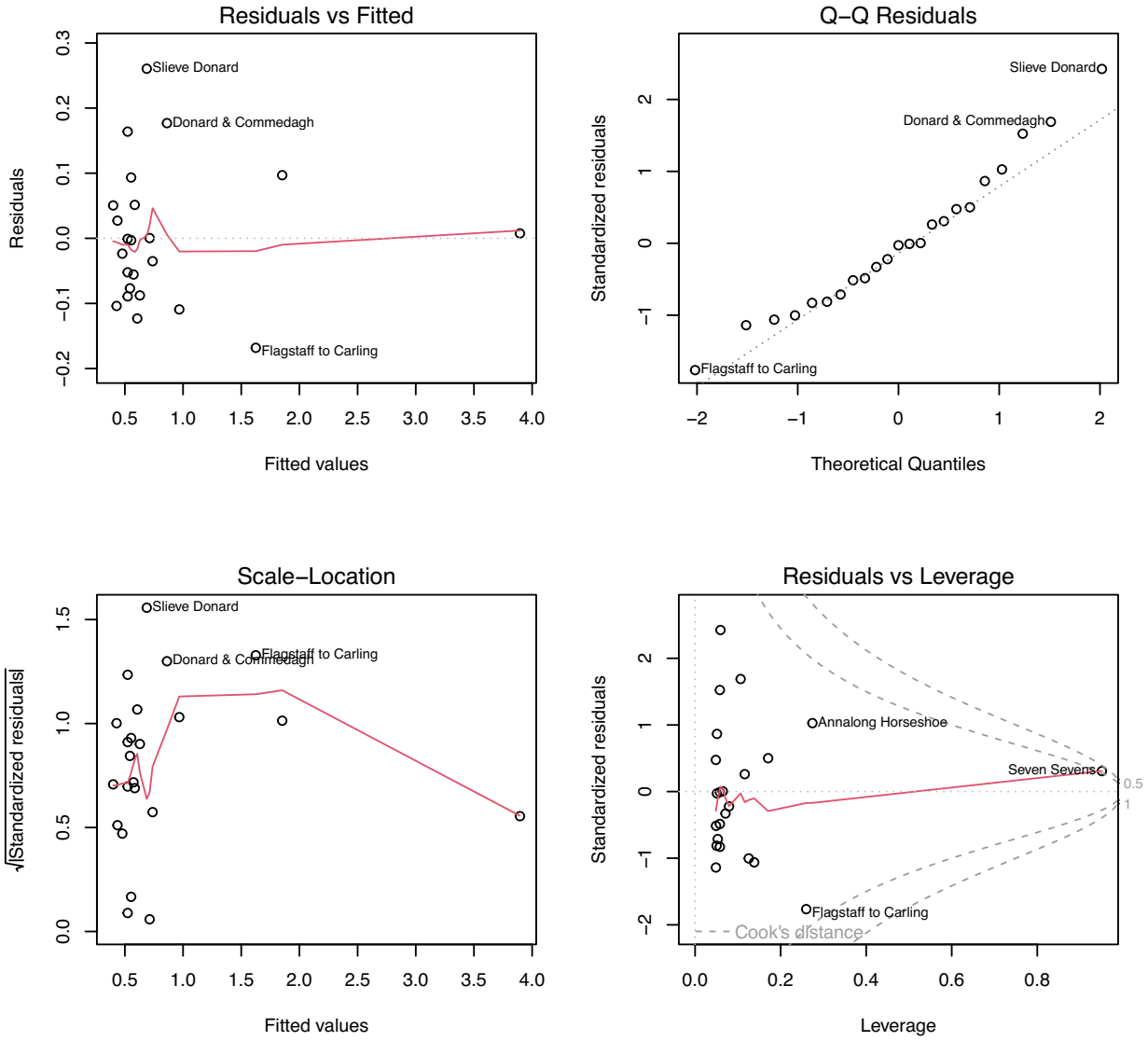
We can explore the model output

```
summary(lm_dist2)
```

```
##
## Call:
## lm(formula = time ~ dist + I(dist^2), data = nihills)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.168138 -0.082280 -0.002996  0.050920  0.260472
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.278699   0.100906   2.762    0.012 *
## dist        0.026388   0.027061   0.975    0.341
## I(dist^2)   0.008728   0.001315   6.640 1.83e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1108 on 20 degrees of freedom
## Multiple R-squared:  0.9809, Adjusted R-squared:  0.979
## F-statistic: 512.8 on 2 and 20 DF,  p-value: < 2.2e-16
```

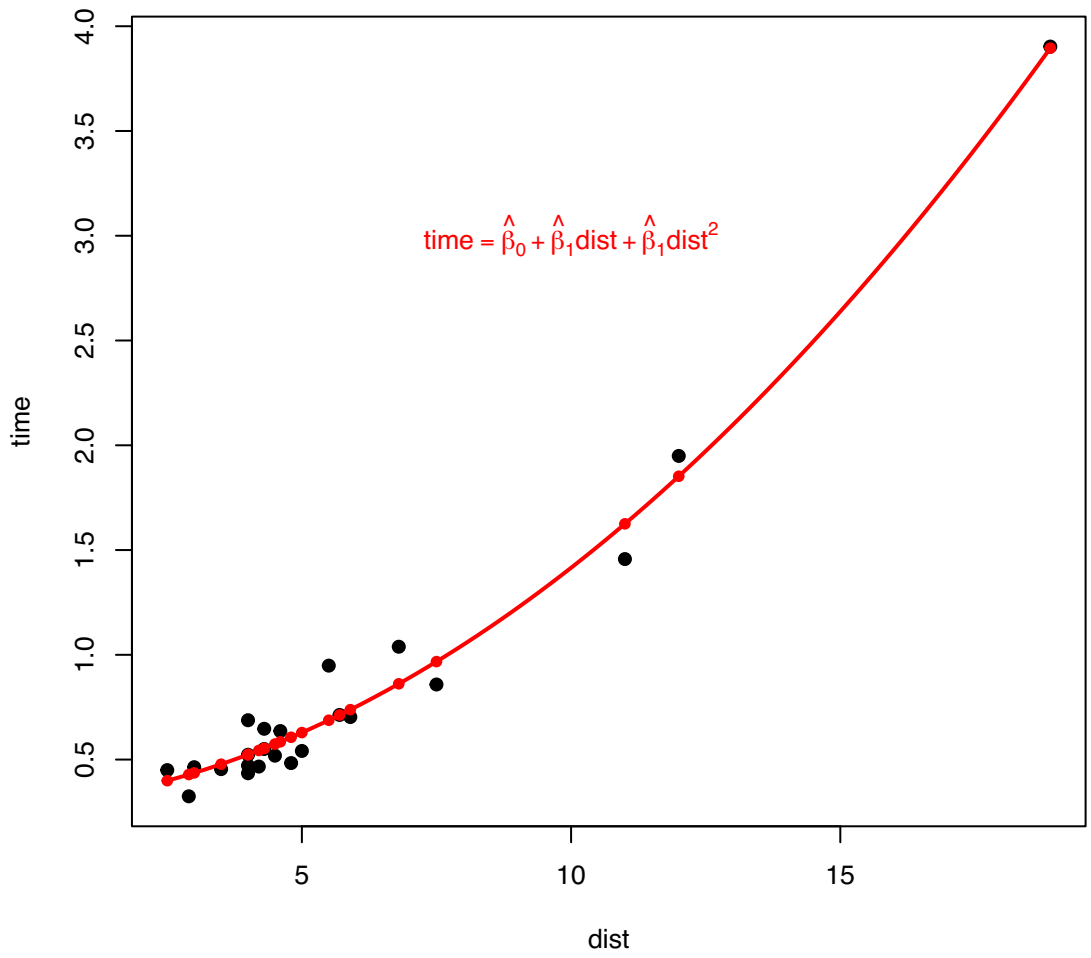
We start exploring the residuals plots

```
par(mfrow = c(2,2))
plot(lm_dist2)
```



Finally, we visualise the scatterplot including the fitted curve

```
par(mfrow = c(1,1))
with(nihills, plot(dist, time, pch=19))
curve(predict(lm_dist2, data.frame(dist = x)), col = "red",
      lty = "solid", lwd = 2, add = TRUE)
text(10, 3, expression(time == hat(beta)[0] + hat(beta)[1] * dist +
                         hat(beta)[1] * dist^2), col = "red")
points(nihills$dist, predict(lm_dist2), col = "red", pch = 19, cex = 0.8)
```

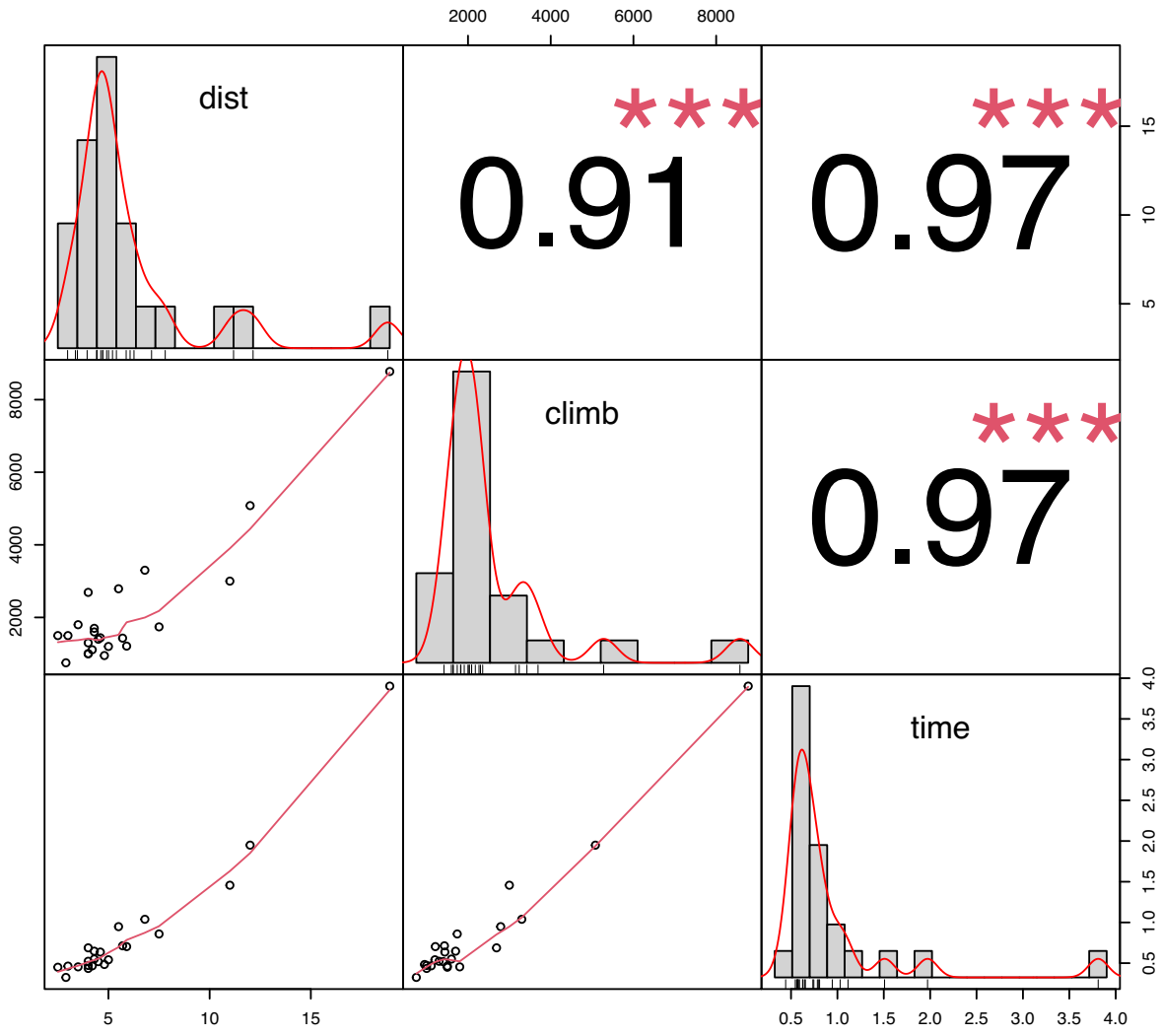


It seems we improved the model, but we can do better...

Adding other predictors

Up to now, we limited to modelling the time by using only the predictor `dist`. However, the `climb` variable is in our availability. Consider that checking the model fit is of vital importance, and many graphical tools are available for this purpose. We may start exploring the variables with a scatterplot matrix.

```
library(PerformanceAnalytics)
chart.Correlation(nihills[, c("dist", "climb", "time")])
```



Some comments:

- All the variables are seen to be skewed
- It seems to exist a linear relationship between `time` and both the variables `climb` and `dist`, apart from a possible outlier.
- The correlation between `time` and both `climb` and `dist`, as well as the correlation between `climb` and `dist`, is extremely high.

We fit the multiple linear model, with the dependent variable `time` explained by `dist` and `climb`:

$$time_i = \beta_1 + \beta_2 dist_i + \beta_3 climb_i + \varepsilon_i, \quad i = 1, \ldots, n, \quad \varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2).$$

```
lm_distclimb <- lm(time ~ dist + climb, data = nihills)
```

```
summary(lm_distclimb)
```

```
##
## Call:
## lm(formula = time ~ dist + climb, data = nihills)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.19857 -0.04824  0.01701  0.05539  0.21083
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.286e-01  4.025e-02  -5.679 1.47e-05 ***
## dist         1.008e-01  1.382e-02   7.293 4.72e-07 ***
## climb        2.298e-04  2.893e-05   7.941 1.31e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0973 on 20 degrees of freedom
## Multiple R-squared:  0.9852, Adjusted R-squared:  0.9838
## F-statistic: 667.6 on 2 and 20 DF,  p-value: < 2.2e-16
```

The estimated regression is then:

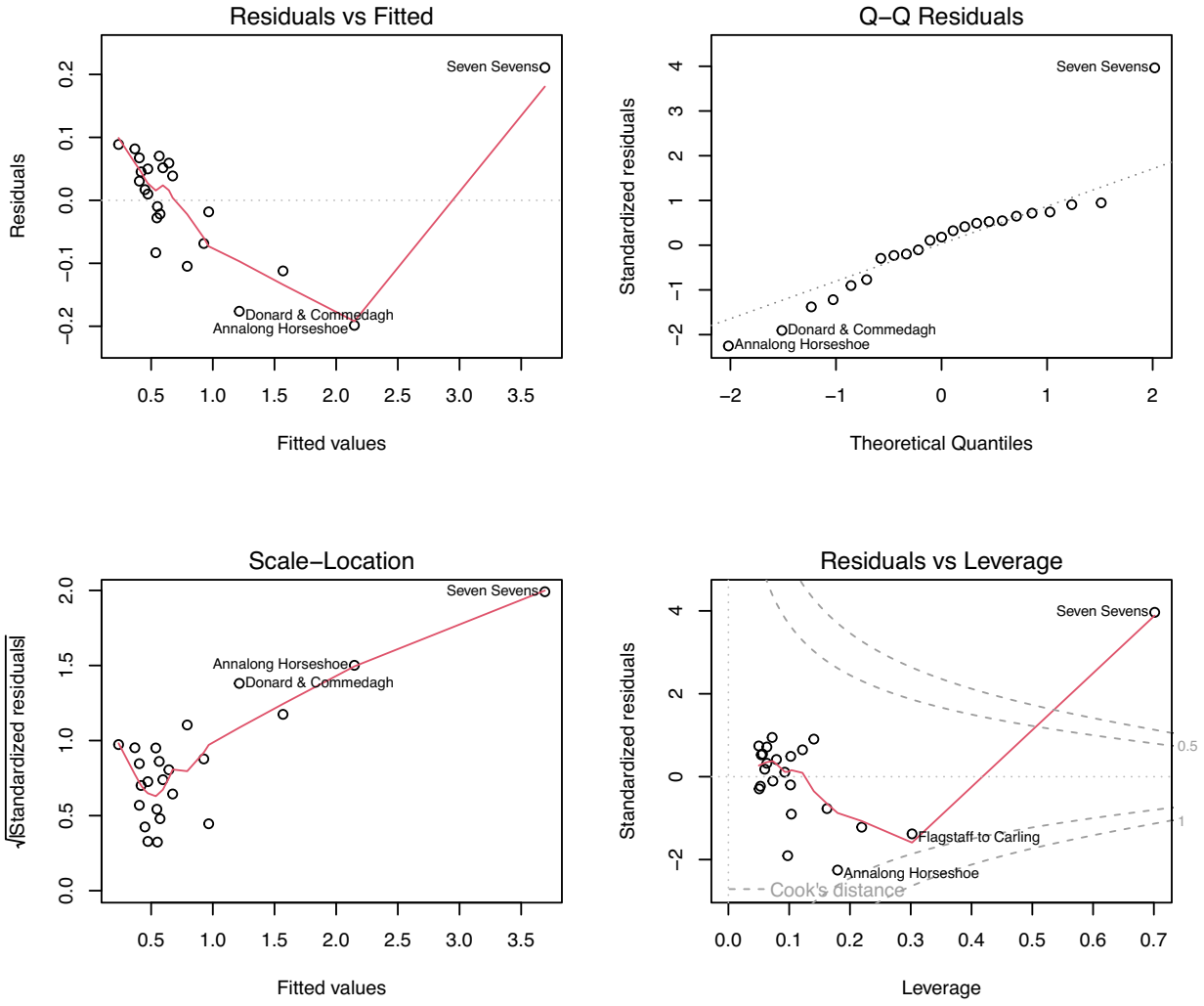
$$time = -0.2286 + 0.1008 dist + 2 \times 10^{-4} climb$$

According to the summary:

- For fixed values of `climb`, the time depends on the distance (p-value extremely low). And the time is estimated to increase on average of 0.1 hour (6 minutes), for a unitary increase of `dist` (that is for each additional mile)
- For fixed values of `dist`, the time depends on the climbs (p-value extremely low). And the time is estimated to increase on average of 0.0002 hour, for a unitary increase of `climb` (that is for each additional feet)
- The adjusted R^2 is 0.9838, that is the model explain 98.38% of the variability of the outcome.
- The complete model is clearly better than the model with only the intercept.
- The joint effect of `dist` and `climb` explains much more than including only `dist`

We may also produce some **diagnostic plots**

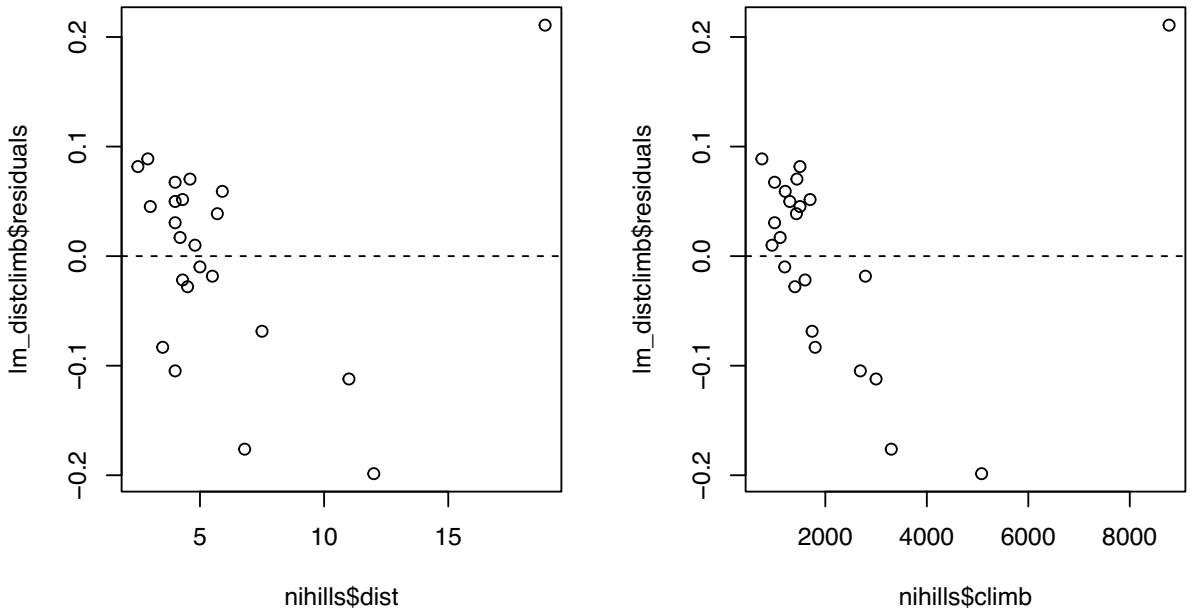
```
par(mfrow = c(2, 2))
plot(lm_distclimb)
```



From the first plot (top left) and the third plot (bottom left), we can suspect that the homoscedasticity assumption is violated. Also, there is evidence of a departure from the linearity. Further, the race `Seven Sevens` is an outlier according to the QQ plot (top right) and the bottom right plot.

Plotting the residuals against the continuous explanatory variables is another good check to highlight (if there are) structures/patterns in the residuals.

```
par(mfrow = c(1,2))
plot(nihills$dist, lm_distclimb$residuals)
abline(h = 0, lty = "dashed")
plot(nihills$climb, lm_distclimb$residuals)
abline(h = 0, lty = "dashed")
```

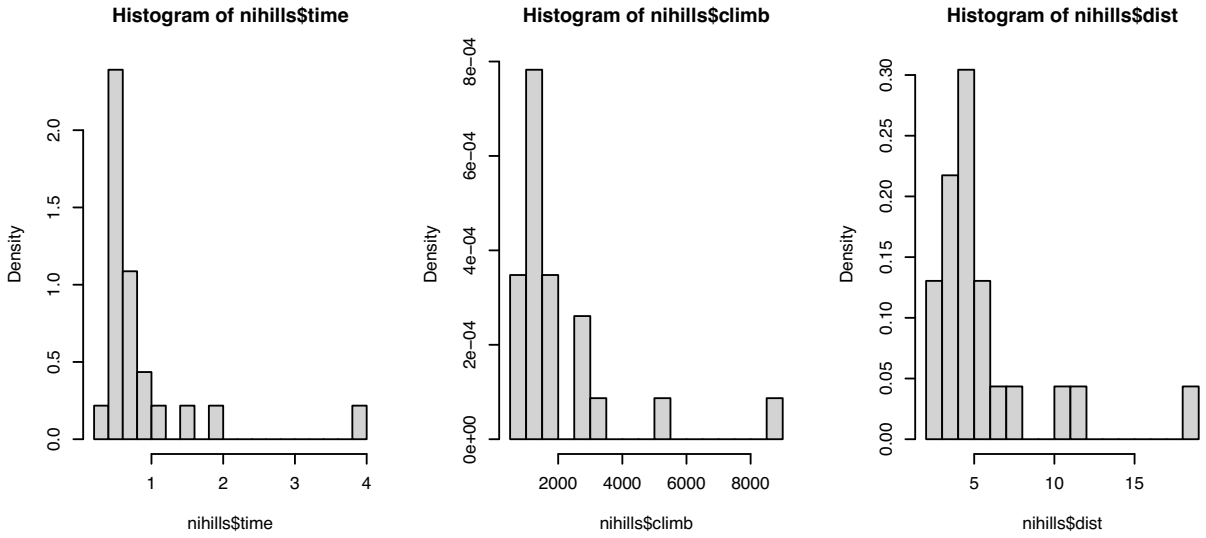


There are clear patterns in the residuals.

Trasforming variables

The analysis of the marginal distribution of the variables also highlighted a marked asymmetry. We reported again the histograms for sake of simplicity

```
par(mfrow = c(1, 3))
hist(nihills$time, probability = TRUE, breaks = 15)
hist(nihills$climb, probability = TRUE, breaks = 15)
hist(nihills$dist, probability = TRUE, breaks = 15)
```



Thus, it may be convenient to consider a sensible transformation of the data. However, statistical modelling is not an exact science, and motivations for transforming data may vary among the different datasets and situations.

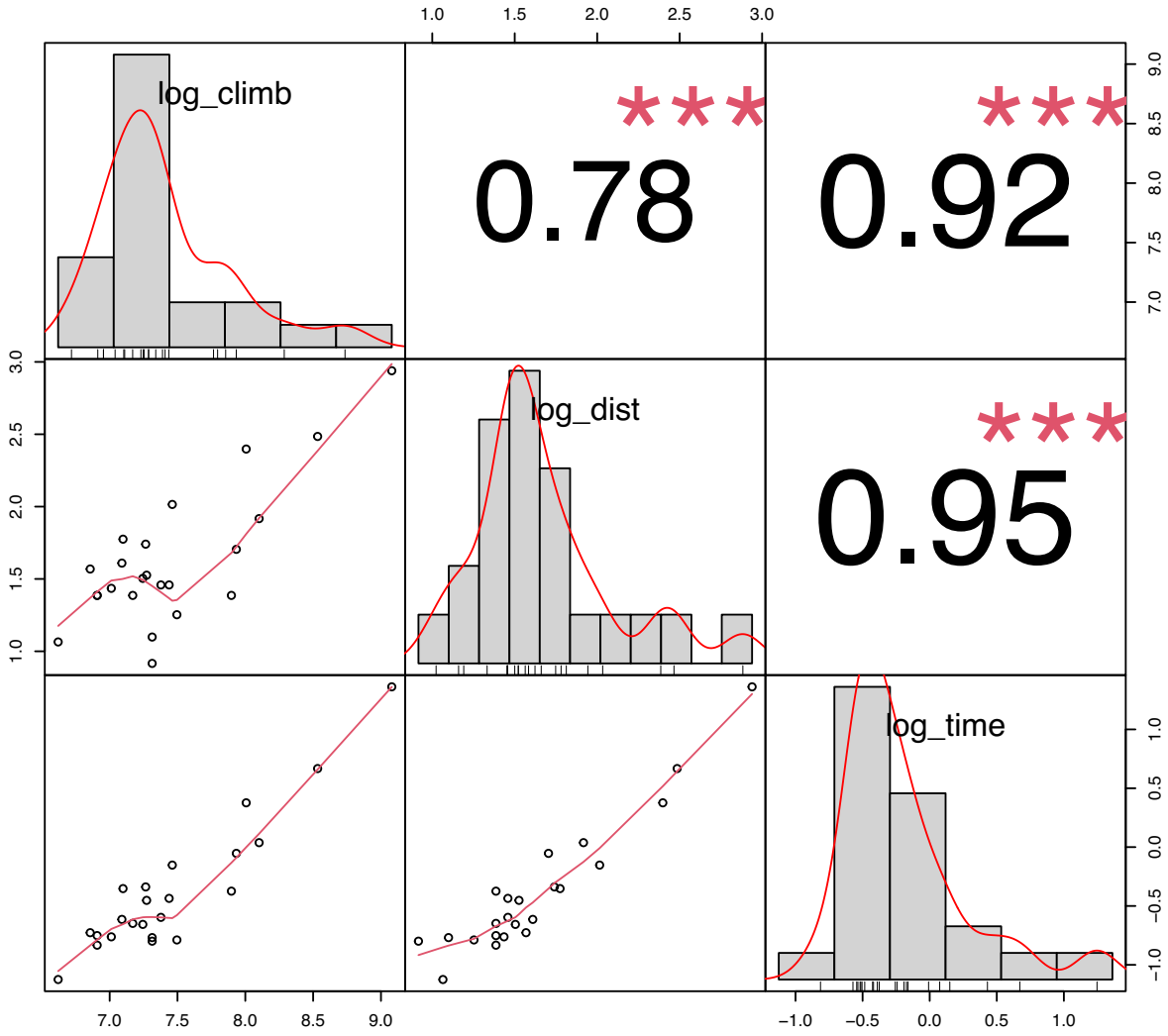
Remember that the last fitted linear model relies on the assumption:

$$time_i \overset{ind}{\sim} \mathcal{N}(\beta_1 + \beta_2 dist_i + \beta_3 climb_i, \sigma^2), \quad i = 1, \dots, n$$

According to the (left) histogram above, the distribution has a long right tail, and lot of values are shrunk towards zero. Furthermore, the extreme point on the right tail —the `Seven sevens` race— influences a lot the estimation of the equation line, it has large *leverage*. Maybe, we need a more symmetric distribution, and the transformation $\log(time_i)$ seems to be a natural choice. Further, it is also sensible taking the logarithm of the explanatory variables.

```
nihills$log_time <- log(nihills$time)
nihills$log_climb <- log(nihills$climb)
nihills$log_dist <- log(nihills$dist)

chart.Correlation(nihills[, c("log_climb", "log_dist", "log_time")])
```

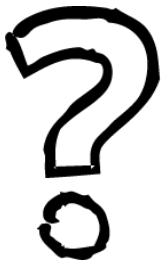


According to the transformations above, we can set up the linear model:

$$\log(time_i) = \beta_1 + \beta_2 \log(dist_i) + \beta_3 \log(climb_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2), \quad i = 1, \ldots, n$$

and it is clear that the effects of the explanatory variables on the response variable is not linear, indeed

$$time_i = e^{\beta_1} dist_i^{\beta_1} climb_i^{\beta_2} \varepsilon_i', \quad i = 1, \ldots, n$$



We fit the model with the transformed variables and we analyse the summary

```
lm_ldist_lclimb <- lm(log_time ~ log_dist + log_climb, data=nihills)
summary(lm_ldist_lclimb)
```

```
##
## Call:
## lm(formula = log_time ~ log_dist + log_climb, data = nihills)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.17223 -0.04229 -0.02538  0.05222  0.13150
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.96113    0.27387  -18.11 7.09e-14 ***
## log_dist     0.68136    0.05518   12.35 8.19e-11 ***
## log_climb    0.46576    0.04530   10.28 1.98e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0766 on 20 degrees of freedom
## Multiple R-squared:  0.9831, Adjusted R-squared:  0.9814
## F-statistic: 582.7 on 2 and 20 DF,  p-value: < 2.2e-16
```

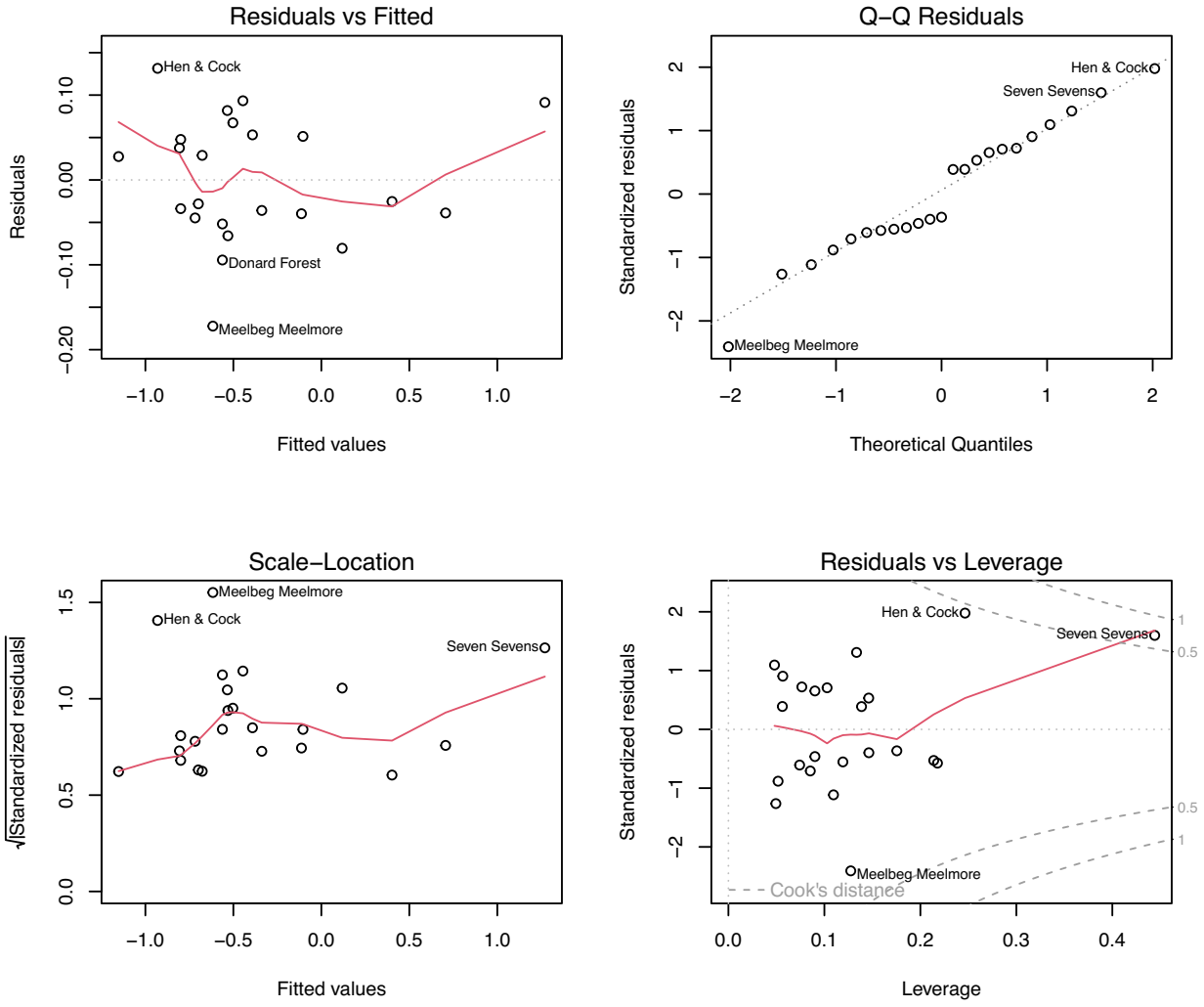
Note: the estimated coefficients/effects must be interpreted on the transformed scale

In terms of R^2 (adjusted), the model in log scale behaves analogously to the model `lm_distclimb`, ($R^2 = 0.984$) against `lm_ldist_lclimb` with $R^2 = 0.986$ (on the original scale). Notice that both these multiple linear models—the one with not transformed data and the other with log data transformation—provide a better fit than the simple linear model `lm_dist` with one predictor, whose R^2 was 0.936. While recall that the model with the quadratic term of the distance get an R^2 of 0.979.

Exercise Note that the R^2 of the the model in log-scale reported above is on the original scale (different from $R^2 = 0.9814$ reported in the output). Obtain it.

Despite the similarity in terms of R^2, the analysis of the residuals for the model in log scale do not show any kind of particular misfit. From the pairs plot above we notice that the data distributions are now less shrunk towards zero and the outliers likely to be less influential.

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



Extra: Computational notes

Performing matrix operations in R requires to be careful: some operations can be obtained more efficiently. In this respect, consider generating some data to be fitted by a linear model. For instance

```
set.seed(123)
error <- rnorm(1000)
X <- cbind(1, matrix(rnorm(1000 * 9), 1000, 9))
beta <- rnorm(10, 0, 2)
y <- X %*% beta + error
```

First, note that the matrix product $X^\top X$ can be computed more efficiently by using the `crossprod()` function. In order to compare the equivalent implementations we can use the `microbenchmark()` function of the `microbenchmark` package, which performs several times the evaluation of the operations/functions specified (by default `times = 100`).

```
library(microbenchmark)
microbenchmark(t(X) %*% X, crossprod(X))
```

```
## Unit: microseconds
##          expr  min     lq    mean median     uq    max neval cld
##    t(X) %*% X 77.0 144.25 375.224  215.6 358.40 3041.5   100  a
##  crossprod(X) 32.6  39.20  49.387   41.6  47.75  134.0   100   b
```

Second, note the differences in time evaluation to carry out the parameter estimates of the linear model according to the following implementations.

```
w1 <- function(){
  XTX <- t(X) %*% X
  solve(t(X) %*% X) %*% t(X) %*% y
}

w2 <- function(){
  XTX <- crossprod(X)
  solve(XTX) %*% t(X) %*% y
}

w3 <- function(){
  XTX <- crossprod(X)
  solve(XTX) %*% (t(X) %*% y)
}

w4 <- function(){
  XTX <- crossprod(X)
  solve(XTX, t(X) %*% y)
}

microbenchmark(w1(), w2(), w3(), w4())
```

```
## Unit: microseconds
##  expr   min     lq    mean median     uq    max neval cld
##  w1() 185.5 236.15 272.765 243.95 266.65 2219.8   100   a
##  w2() 101.6 129.85 152.404 133.45 144.95 1538.3   100   a
##  w3()  58.3  70.65  91.457  72.55  80.10 1580.7   100   a
##  w4()  53.9  67.30 210.336  70.55  78.10 8444.6   100   a
```

We can do better using the QR decomposition. See Section 7.1.1. of the book Core Statistics (Wood, 2015)

Extra: Linear model with categorical predictors

Let us suppose that the `dist` variable is expressed in categories ("Low", "Medium", "High"). So we manage to transform the dist variable in a categorical one, having three levels (we consider as cutoffs the quantiles of order 1/3 and 2/3). Take a look to the model matrix that can be obtained by means of the `model.matrix()` function

```
qdist <- with(nihills, quantile(dist, prob = c(1/3, 2/3)))

nihills$distdisc <- rep("Low", n)
nihills$distdisc[nihills$dist > qdist[1] & nihills$dist <= qdist[2]] <- "Medium"
nihills$distdisc[nihills$dist > qdist[2]] <- "High"

nihills$distdisc <- factor(nihills$distdisc, level = c("Low", "Medium", "High"))
model.matrix(~ distdisc, data = nihills)
```

```
##                          (Intercept) distdiscMedium distdiscHigh
## Binevenagh                         1              0            1
## Slieve Gullion                     1              1            0
## Glenariff Mountain                 1              0            1
## Donard & Commedagh                 1              0            1
## McVeigh Classic                    1              1            0
## Tollymore Mountain                 1              1            0
## Slieve Martin                      1              1            0
## Moughanmore                        1              0            0
## Hen & Cock                         1              0            0
## Annalong Horseshoe                 1              0            1
## Monument Race                      1              0            0
## Loughshannagh Horseshoe            1              1            0
## Rocky                              1              0            0
## Meelbeg Meelmore                   1              0            0
## Donard Forest                      1              1            0
## Slieve Donard                      1              0            1
## Flagstaff to Carling               1              0            1
## Slieve Bearnagh                    1              0            0
## Seven Sevens                       1              0            1
## Lurig Challenge                    1              0            0
## Scrabo Hill Race                   1              0            0
## Slieve Gallion                     1              1            0
## BARF Turkey Trot                   1              0            1
## attr(,"assign")
## [1] 0 1 1
## attr(,"contrasts")
## attr(,"contrasts")$distdisc
## [1] "contr.treatment"
```

Exercise: Fit a linear model and interpret the coefficient estimates.

Quick questions:

1. If I regress the `distdisc` variable on time, how many parameters are needed for specifying the linear model? 3 w/ an intercept they are relative from one factor to another
2. And suppose to consider also the `climb` variable categorised in three levels ("Low", "Medium", "High"), according to the same strategy as above, and including in the linear model together with `dist` regressing on the `time`, how many parameter are needed for specifying the linear model? 5

$t = \beta_0 + \beta_{dist} + \beta_{2\,C.low} + \beta_{3\,C.med.} + \beta_{4\,C.high} + \varepsilon$

Model selection

Model selection for nested models

Once we propose competing nested models, *analysis of deviance* is required for assessing whether the more complex model better describe data at hand. Let us set up a test for `nihills` races, where the dependent variable is still $\log(time)$, and possible exploratory variables are $\log(dist)$ and $\log(climb)$. Two nested competing models are:

$$\begin{cases} \mathcal{M}_1: & \log(y_i) = \beta_o + \beta_1 \log(dist_i) + \varepsilon_i, \quad i = 1, \dots, n \\ \mathcal{M}_2: & \log(y_i) = \beta_o + \beta_1 \log(dist_i) + \beta_2 log(climb_i) + \varepsilon_i, \quad i = 1, \dots, n \end{cases}$$

We can then set up the following test:

$$\begin{cases} H_0: & \beta_2 = 0 \\ H_1: & \beta_2 \neq 0 \end{cases}$$

using as a test statistic the following ratio:

$$F = \frac{\frac{RSS_1 - RSS_2}{p_2 - p_1}}{\frac{RSS_2}{n - p_2}}, \qquad F = \frac{(SSE_0 - SSE_1)/(p_1 - p_0)}{SSE_1/(n - p_1)}$$

where p_1 is the number of parameters in $\mathcal{M}_1$, p_2 the number of parameters in $\mathcal{M}_2$, with $p_2 > p_1$ (in this case, 3>2). High values for the statistic F suggest to select the complex model in place of the simplest one. Under H_0 the test statistic is distributed as $F_{p_2 - p_1, n - p_2}$. The above formula is more general and allows testing the nullity of subgroup of coefficients, $\tilde{\boldsymbol{\beta}} \in \mathbb{R}^{p_2 - p_1}$ that is for testing the hypothesis system

$$\begin{cases} H_0: & \tilde{\boldsymbol{\beta}} = 0 \\ H_1: & \tilde{\boldsymbol{\beta}} \neq 0 \end{cases}$$

Let us start fitting the model under $\mathcal{M}_O$. By means of the `anova()` function we recover the same results reported in the `summary()` function of the lm function. Namely, we are comparing the full model (with only the covariate $\log(dist_i)$) with the null model

```
m1 <- lm(log_time ~ log_dist, data = nihills)
summary(m1)
```

```
##
## Call:
## lm(formula = log_time ~ log_dist, data = nihills)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.27988 -0.13697 -0.02603  0.11445  0.38119
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.21012    0.14285  -15.47 5.91e-13 ***
## log_dist     1.12389    0.08447   13.30 1.06e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1874 on 21 degrees of freedom
## Multiple R-squared:  0.894,  Adjusted R-squared:  0.8889
## F-statistic:   177 on 1 and 21 DF,  p-value: 1.06e-11
```

```r
anova(m1)
```

```
## Analysis of Variance Table
##
## Response: log_time
##           Df Sum Sq Mean Sq F value   Pr(>F)
## log_dist   1 6.2174  6.2174  177.03 1.06e-11 ***
## Residuals 21 0.7375  0.0351
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

By running `anova()` on the extended model ($\mathcal{M}_2$, with $\log(dist_i)$ and $\log(climb_i)$). The table below first computes the F statistic for the null model $\mathcal{M}_0$ with the intercept only and $\mathcal{M}_1$, with a corresponding p-value ≈ 0 ($\mathcal{M}_1$ preferred to $\mathcal{M}_0$, and difference of degrees of freedom `df`=2-1 =1); then, the test suggests to accept $\mathcal{M}_2$ in place of $\mathcal{M}_1$, since p-value ≈ 0. Here the difference of degrees of freedom `df` is always 1, while `residuals` is the number of degrees of freedom of the final model $\mathcal{M}_2$, $= n - p_2 = 23 - 3 = 20$.

```r
anova(lm_ldist_lclimb)
```

```
## Analysis of Variance Table
##
## Response: log_time
##           Df Sum Sq Mean Sq F value    Pr(>F)
## log_dist   1 6.2174  6.2174  1059.7 < 2.2e-16 ***
## log_climb  1 0.6202  0.6202   105.7 1.981e-09 ***
## Residuals 20 0.1173  0.0059
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Another possible use of `anova()` allows to pass both the models specified under $\mathcal{M}_1$ and $\mathcal{M}_2$.

```r
anova(m1, lm_ldist_lclimb)
```

```
## Analysis of Variance Table
##
## Model 1: log_time ~ log_dist
## Model 2: log_time ~ log_dist + log_climb
##   Res.Df     RSS Df Sum of Sq     F    Pr(>F)
## 1     21 0.73752
## 2     20 0.11734  1   0.62017 105.7 1.981e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the latter value of the test statistic can be obtained by squaring the value of the test statistic of the corresponding coefficient, that is

```r
summary(lm_ldist_lclimb)$coefficients[3,3]^2
```

```
## [1] 105.7033
```

Models are comparable if one is included inside another and models w/ less parameters are preferable

Likelihood-based Information Criteria

R^2 and adjusted R^2 represent the easiest way for comparing models, or, at least, for assessing whether a given model fits or not the data. But some other useful statistics exist, such as the Akaike Information Criterion (AIC) and some related statistics, which are designed to choose, from among a small number of alternatives, the model with the best predictive power and a good quality of the model fit. AIC is defined as:

$$\text{AIC} = -2 \log \mathcal{L}(\boldsymbol{\beta}; y) + k * edf,$$

where $\log \mathcal{L}(\boldsymbol{\beta}; y)$ is the log-likelihood for the multiple linear model, edf is the number of equivalent degrees of freedom for the fitted model and k the weight associated to edf. AIC and related statistics may be seen as a sum between two components: the first one refers to a pure **measure of fit**, the second one to the **model complexity**. For an equal (or similar) measure of fit, these statistics will favor the most parsimonious models.

```
AIC(lm_dist)
```

```
## [1] -6.371981
```

```
-2 * logLik(lm_dist)[1]    + 2 * (length(coef(lm_dist)) + 1)
```

```
## [1] -6.371981
```

```
BIC(lm_dist)
```

```
## [1] -2.965498
```

```
-2 * logLik(lm_dist)[1]  + log(nrow(nihills)) * (length(coef(lm_dist)) + 1)
```

```
## [1] -2.965498
```

From the previous code, you can note that I added 1 to the number of parameters. Indeed, AIC on a `lm` object accounts for the estimation of the unknown variance of the error (i.e., scale).

Then, we can extract the AIC and BIC from each model we fitted and compare such models

```
AIC <- rbind(AIC(lm_dist), AIC(lm_distclimb),
             AIC(lm_dist2), AIC(lm_ldist_lclimb))
BIC <- rbind(AIC(lm_dist, k = log(n)), AIC(lm_distclimb, k = log(n)),
             AIC(lm_dist2, k = log(n)), AIC(lm_ldist_lclimb, k = log(n)))
cbind(AIC, BIC)
```

```
##            [,1]       [,2]
## [1,]  -6.371981  -2.965498
## [2,] -37.119503 -32.577526
## [3,] -31.155136 -26.613159
## [4,] -48.126387 -43.584410
```

Remember: as for many scoring rules, lower is the best

The lower the AIC, and the better is the predictive power of the model. The linear model in log-scale `lm_ldist_lclimb` yields the lowest AIC.

Exercise The `R` function `extractAIC` computes the generalized AIC criterion. Explore the help of `extractAIC`() function recognising the main difference w.r.t. the `AIC`() function.

AIC allows us to compare different models and not only nested models.

Multicollinearity

A common feature in regression modelling is that some predictors may be lineary related to combinations of one or more of the other explanatory variables, i.e. $x_1 = a + bx_2$. In some sense, we would avoid this behaviour, and we should break up the multicollinearity existing in our predictors.

The variance inflation factor (**VIF**) measures the effect of correlation with other variables in increasing the standard error of a regression coefficient. Let suppose to have the following linear model:

$$y_i = \beta_1 + \beta_2 x_{i1} + \beta_3 x_{i2} + \beta_4 x_{i3} + \varepsilon_i, \quad i = 1, \ldots, n.$$

The VIF is defined as :

$$VIF(\beta_j) = \frac{1}{1 - R_j^2},$$

where R_j^2 is the multiple R^2 for the regression of x_j on the other covariates (**a regression that does not involve the response variable Y**). Then the VIF reflects all other factors that influence the uncertainty in the coefficient estimates. As a rule of thumb, we can detect an high multicollinearity if $VIF(\hat{\beta}_j) > 10$. While when the vector x_j is orthogonal to each column of the design matrix, the VIF is equal to 1. Then, a side effect is that the estimated variance of the estimator of $\hat{\beta}_j$ increases under multicollinearity.

There is an R function, the `vif()` function, which allows to compute the variance inflation factor. Consider the model in log-scale

```
vif(lm_ldist_lclimb)
```

```
##  log_dist log_climb
##    2.5543    2.5543
```

In this case there is no suggestion of multicollinearity between `log(dist)` and `log(climb)`.

To compute the VIF manually, we must regress the climb (in log scale) on the distance (in log scale)

```
lm_log_dist_climb <- lm(log_dist ~ log_climb, data = nihills)
1/(1 - summary(lm_log_dist_climb)$r.squared)
```

```
## [1] 2.554308
```

US Crime dataset

The `US Crime` dataset includes some socio-demographic characteristics of 47 states of the USA, collected in the 1960. The crime rate represents the number of crimes reported per million inhabitants.

Objective: investigate the relationship between crime rate (thereafter the outcome, `crRate`) and socio-demographic characteristics (predictors) of the states.

First, load the data and look at the first rows of the data matrix

```
USC <- read.table("UScrime.dat", header = TRUE)
colnames(USC) <- c("crRate", "age", "region", "edu", "expPol0", "expPol1",
                   "labForce", "sex", "pop", "race", "unemp0", "unemp1", "her", "poor")
head(USC)
```

```
##   crRate age region edu expPol0 expPol1 labForce  sex pop race unemp0 unemp1
## 1   79.1 151      1  91      58      56      510  950  33  301    108     41
## 2  163.5 143      0 113     103      95      583 1012  13  102     96     36
## 3   57.8 142      1  89      45      44      533  969  18  219     94     33
## 4  196.9 136      0 121     149     141      577  994 157   80    102     39
## 5  123.4 141      0 121     109     101      591  985  18   30     91     20
## 6   68.2 121      0 110     118     115      547  964  25   44     84     29
##   her poor
## 1 394  261
## 2 557  194
## 3 318  250
## 4 673  167
## 5 578  174
## 6 689  126
```

Here, there is a description of the covariates

- age: males aged 14-24 per 1000 inhabitants
- region: indicator for "Southern States" (0 = No, 1 = Yes)
- edu: 10 × Average number of school years for persons of the same age or more than 25 years
- expPol0: 1960 expense (per capita) for the police (state and local government)
- expPol1: 1959 expense per capita for the police (state and local government)
- labForce: Labour force participation per 1000 males
- sex: number of males per 1000 females
- pop: population of the state (per 100000)
- race: number of non-white per 1000
- unemp0: Unemployed per 1000 males aged 14-24 in cities
- unemp1: Unemployed per 1000 males aged 35-39 in cities
- her: Median value of family assets (per 10$)
- poor: number of households out of 1000 earning less than half a median (state) income

Take a look to the summary of the varibales

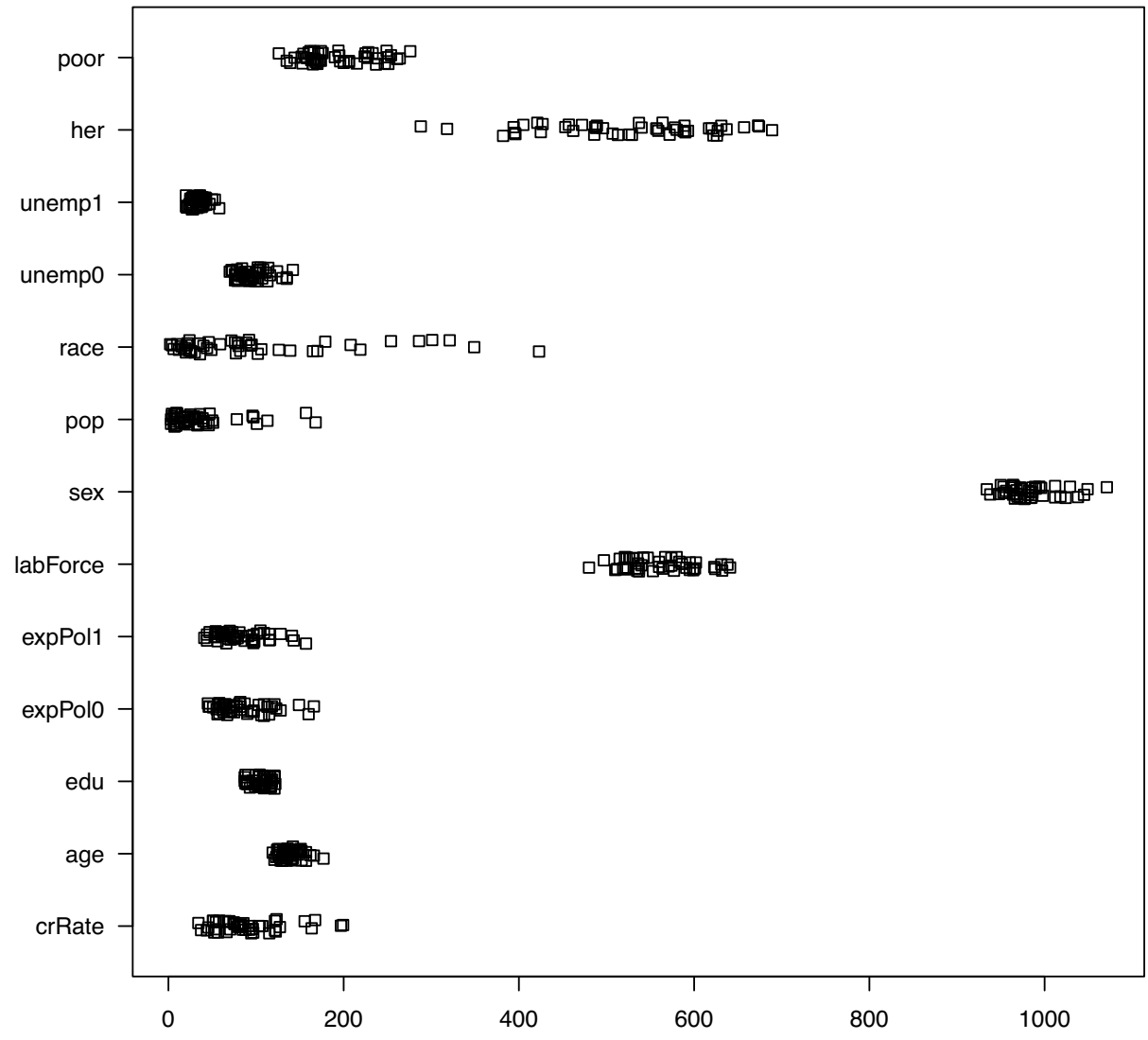
```
summary(USC)
```

```
##      crRate            age            region            edu       
##  Min.   : 34.20   Min.   :119.0   Min.   :0.0000   Min.   : 87.0  
##  1st Qu.: 65.85   1st Qu.:130.0   1st Qu.:0.0000   1st Qu.: 97.5  
##  Median : 83.10   Median :136.0   Median :0.0000   Median :108.0  
##  Mean   : 90.51   Mean   :138.6   Mean   :0.3404   Mean   :105.6  
##  3rd Qu.:105.75   3rd Qu.:146.0   3rd Qu.:1.0000   3rd Qu.:114.5  
##  Max.   :199.30   Max.   :177.0   Max.   :1.0000   Max.   :122.0  
##     expPol0          expPol1          labForce          sex       
##  Min.   : 45.0   Min.   : 41.00   Min.   :480.0   Min.   : 934.0  
##  1st Qu.: 62.5   1st Qu.: 58.50   1st Qu.:530.5   1st Qu.: 964.5  
##  Median : 78.0   Median : 73.00   Median :560.0   Median : 977.0  
##  Mean   : 85.0   Mean   : 80.23   Mean   :561.2   Mean   : 983.0  
##  3rd Qu.:104.5   3rd Qu.: 97.00   3rd Qu.:593.0   3rd Qu.: 992.0  
##  Max.   :166.0   Max.   :157.00   Max.   :641.0   Max.   :1071.0  
##       pop              race          unemp0          unemp1     
##  Min.   :  3.00   Min.   :  2.0   Min.   : 70.00   Min.   :20.00  
##  1st Qu.: 10.00   1st Qu.: 24.0   1st Qu.: 80.50   1st Qu.:27.50  
##  Median : 25.00   Median : 76.0   Median : 92.00   Median :34.00  
##  Mean   : 36.62   Mean   :101.1   Mean   : 95.47   Mean   :33.98  
##  3rd Qu.: 41.50   3rd Qu.:132.5   3rd Qu.:104.00   3rd Qu.:38.50  
##  Max.   :168.00   Max.   :423.0   Max.   :142.00   Max.   :58.00  
##       her             poor       
##  Min.   :288.0   Min.   :126.0  
##  1st Qu.:459.5   1st Qu.:165.5  
##  Median :537.0   Median :176.0  
##  Mean   :525.4   Mean   :194.0  
##  3rd Qu.:591.5   3rd Qu.:227.5  
##  Max.   :689.0   Max.   :276.0
```

All the variables are numeric, but `region` is a dichotomous variable and should be translated to a factor variable.

Instead of representing the variables by using histograms, we explore the marginal distributions (excluding region) by means of `stripchart` (considering the argument `method = "jitter"`). From the following plot, we can note that the distribution of the outcome is asymmetric (also other covariates).

```r
idx <- which(colnames(USC) == "region")
par(mar = c(2, 5, 1, 1))
stripchart(USC[, -idx], method = "jitter", las = 1)
```



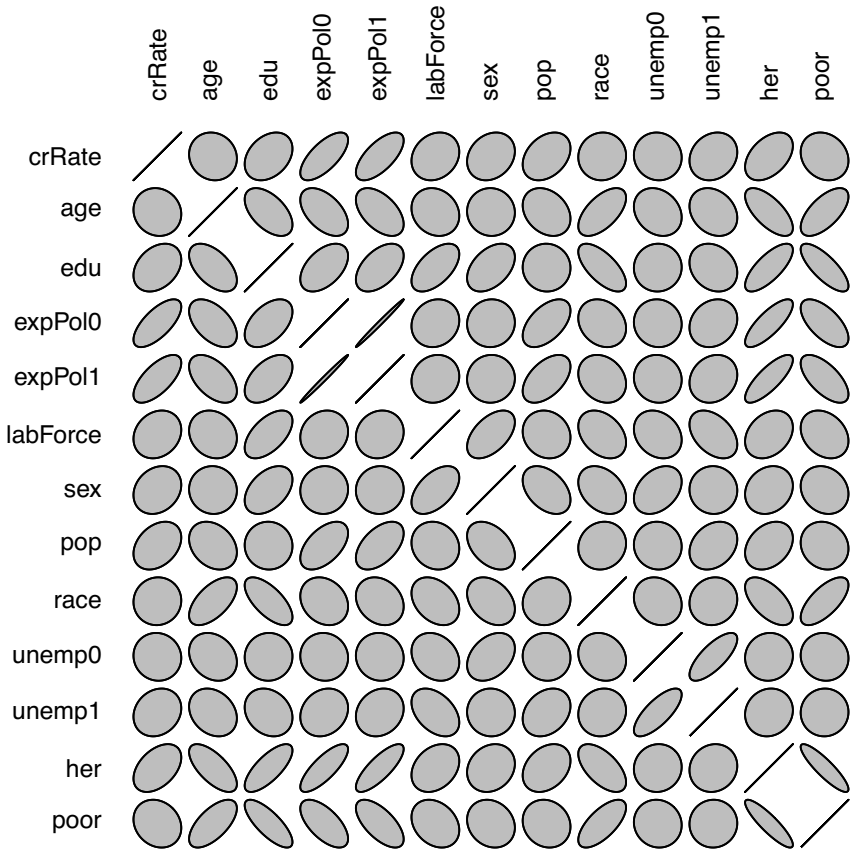
In order to explore the pairwise distributions, as an alternative to using the `chartCorrelation`() function, we will compute the correlation matrix

```
round(cor(USC[,-idx]), 2)
```

```
##          crRate   age   edu expPol0 expPol1 labForce   sex   pop  race unemp0
## crRate     1.00 -0.09  0.32    0.69    0.67     0.19  0.21  0.34  0.03  -0.05
## age       -0.09  1.00 -0.53   -0.51   -0.51    -0.16 -0.03 -0.28  0.59  -0.22
## edu        0.32 -0.53  1.00    0.48    0.50     0.56  0.44 -0.02 -0.66   0.02
## expPol0    0.69 -0.51  0.48    1.00    0.99     0.12  0.03  0.53 -0.21  -0.04
## expPol1    0.67 -0.51  0.50    0.99    1.00     0.11  0.02  0.51 -0.22  -0.05
## labForce   0.19 -0.16  0.56    0.12    0.11     1.00  0.51 -0.12 -0.34  -0.23
## sex        0.21 -0.03  0.44    0.03    0.02     0.51  1.00 -0.41 -0.33   0.35
## pop        0.34 -0.28 -0.02    0.53    0.51    -0.12 -0.41  1.00  0.10  -0.04
## race       0.03  0.59 -0.66   -0.21   -0.22    -0.34 -0.33  0.10  1.00  -0.16
## unemp0    -0.05 -0.22  0.02   -0.04   -0.05    -0.23  0.35 -0.04 -0.16   1.00
## unemp1     0.18 -0.24 -0.22    0.19    0.17    -0.42 -0.02  0.27  0.08   0.75
## her        0.44 -0.67  0.74    0.79    0.79     0.29  0.18  0.31 -0.59   0.04
## poor      -0.18  0.64 -0.77   -0.63   -0.65    -0.27 -0.17 -0.13  0.68  -0.06
##          unemp1   her  poor
## crRate     0.18  0.44 -0.18
## age       -0.24 -0.67  0.64
## edu       -0.22  0.74 -0.77
## expPol0    0.19  0.79 -0.63
## expPol1    0.17  0.79 -0.65
## labForce  -0.42  0.29 -0.27
## sex       -0.02  0.18 -0.17
## pop        0.27  0.31 -0.13
## race       0.08 -0.59  0.68
## unemp0     0.75  0.04 -0.06
## unemp1     1.00  0.09  0.02
## her        0.09  1.00 -0.88
## poor       0.02 -0.88  1.00
```

We can plot the pairwise correlation into a matrix by using the `plotcorr()` function of th `ellipse` package

```
library(ellipse)
plotcorr(cor(USC[,-idx]))
```



From the first row of the correlation matrix (or analysing the plot) we can see there is an high correlation between the outcomes and the variables `her`, `expPol0` and `expPol1`. In addition, there is an high correlation between the couples of variables (`expPol0`, `expPol1`), (`poor`, `her`)

Let's take a look to the summary of the models including separately the explanatory variables `expPol0` and `expPol1`, and the summary of the model including simultaneously such variables.

```
fit0 <- lm(crRate ~ expPol0, data = USC)
fit1 <- lm(crRate ~ expPol1, data = USC)
fit01 <- lm(crRate ~ expPol0 + expPol1, data = USC)

summary(fit0)$coefficients
```

```
##               Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 14.4463983 12.6692558 1.140272 2.602057e-01
## expPol0      0.8948484  0.1408615 6.352683 9.338016e-08
```

```
summary(fit1)$coefficients
```

```
##               Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 16.5164241 13.0426970 1.266335 2.119092e-01
## expPol1      0.9222031  0.1536808 6.000769 3.114182e-07
```

```
summary(fit01)$coefficients
```

```
##              Estimate Std. Error   t value   Pr(>|t|)
## (Intercept) 15.826462  12.592642  1.256802 0.21545640
## expPol0      2.561526   1.234173  2.075500 0.04381721
## expPol1     -1.782880   1.311753 -1.359158 0.18102255
```

If we take separately the variables we have significant and positive effects on the crime rate (the crime rate grows with the increase of the expense for the police). Considering together the variables we get opposite effects and only one is significant (in contrast to the findings of the previous models). The presence of both variables thus leads to inconsistent results. This is the multicollinearity problem. Further, note the increase of the estimated variance of the estimator of the parameter β, which is a side effect of the multicollinearity problem. Indeed, there is a 9/10 fold increase of the standard error by considering the model `fit01`. To conclude, we compute the VIF.

```
vif(fit01)
```

```
## expPol0 expPol1
##  78.211  78.211
```

Further comments:

- The multicollinearity problem is not ensured to be solved during model/variable selection. Thus, strongly correlated variables could appear in the final model. Maybe a solution could be exclude one of the tricky variable.
- In addition, the inclusion in the model of the variables `expPol0` or `expPol1` determines an interpretation problem. If the aim is identifying the social and economic factors influencing crime, the presence of expenditure on public security is likely to be a consequence of the level of crime and not a determining factor. Its presence in the model could mask the effect of some determinants, so it makes sense to estimate the model excluding both such variables.

Exercise: Try to explore the multicollinearity problem analysing the variables `poor` and `her`

Exercise: Try to propose a sensible model for analysing the US Crime dataset, following at a certain degree the workflow used for the `nihills` dataset.