Solutions to Exercises from Chapter 05

Contents

The `comphy` package is loaded once at the beginning so to make all its functions available to this exercises session.

```
library(comphy)
```

1 Exercises on least squares

1.1 Exercise 01

Study the main function for linear regression, `solveLS`, in `comphy` and apply it to estimate the parameters, a_1, a_2, a_3, a_4, of the linear model

$$y = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4$$

The data points are listed in the following table.

x_1	x_2	x_3	y
−0.959	0.829	0.419	4.327
−0.781	−0.218	0.752	6.513
−0.957	0.549	0.499	4.847
0.413	−0.184	0.919	9.838
−0.989	−0.683	0.819	6.931
0.460	−0.274	−0.362	4.606
−0.479	0.739	−0.156	2.738
−0.436	−0.229	−0.664	2.191
−0.329	0.289	0.663	6.657
−0.866	0.662	−0.252	1.630
0.988	−0.635	0.957	11.556
0.885	−0.422	−0.736	4.372
−0.760	−0.231	0.596	6.010
−0.338	0.705	0.502	5.397
−0.734	0.551	−0.991	−1.245

SOLUTION

The function `solveLS` can be studied if its source code is dumped on the screen. This is readily done simply typing `solveLS` without parentheses.

```
solveLS
#> function(x,intercept=TRUE,tol=NULL) {
#>   # Check input
#>   ans <- (is.matrix(x) | is.data.frame(x))
#>   if (!ans) {
#>     msg <- "Input has to be either a matrix or a data frame\n"
#>     cat(msg)
#>     return(NULL)
#>   }
#>
#>   # Number of data points (n) and parameters (m+1)
#>   tmp <- dim(x)
#>   n <- tmp[1]
#>   m <- tmp[2]-1
#>
#>   # Build A and y matrices
#>   ones <- matrix(rep(1,times=n),ncol=1)
#>   A <- as.matrix(x[,1:m])
#>   colnames(A) <- NULL
#>   rownames(A) <- NULL
#>   if (intercept) A <- cbind(A,ones)
#>   y <- matrix(x[,m+1],ncol=1)
#>   colnames(y) <- NULL
#>   rownames(y) <- NULL
#>
#>   # Build F and g matrices to get solution through solve
#>   F <- t(A) %*% A
#>   g <- t(A) %*% y
#>
#>   # Check whether F is singular
#>   d <- det(F)
#>   if (abs(d) < 1e-6) {
#>     msg <- paste0("There are infinite solutions to ",
#>                   "this least squares fitting.\n")
#>     cat(msg)
#>     return(NULL)
#>   }
#>
#>   # Change tolerance, if required
#>   if (is.null(tol)) tol <- 1e-17
#>
#>   # Solution
#>   a <- solve(F,g,tol=tol)
#>
#>   # Print out the sum of squared residuals:
#>   eps <- y-A %*% a
#>   d <- sum(eps^2)
#>   msg <- sprintf("Sum of squared residuals: %f\n",d)
#>   cat(msg)
#>
#>   # Reshape for output
```

```
#>   a <- as.vector(a)
#>
#>   return(a)
#> }
#> <bytecode: 0x000001d48daba040>
#> <environment: namespace:comphy>
```

The code is essentially an implementation of the matrix form of the least squares' normal equations, where a prominent role is played by the matrix A in which the last column is filled with 1's. For example, in the case of the model studied in this exercise, we have

$$A = (\mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_3 \quad \mathbf{1})$$

An exception, cleverly coded with the `intercept` parameter, is implemented for the case in which the constant term (a_{m+1}) is not part of the model (or $a_{m+1} = 0$). In this case the column filled with 1's is not included in A (check the one-liner in which this is taken care of).

Another important aspect of the code is that when the matrix F of coefficients of the normal equations is ill-conditioned, then the `solve` function can fail, unless the `tol` keyword is appropriately small. The default is `1e-17`, but the user can input smaller values if the default makes execution fail.

Let us now try and solve the regression problem. First we have to insert the tabulated data into an array structure, say a data frame.

```
# Data are saved into a data frame
x1 <- c(-0.959,-0.781,-0.957,0.413,-0.989,0.460,-0.479,-0.436,
        -0.329,-0.866,0.988,0.885,-0.760,-0.338,-0.734)
x2 <- c(0.829,-0.218,0.549,-0.184,-0.683,-0.274,0.739,-0.229,
        0.289,0.662,-0.635,-0.422,-0.231,0.705,0.551)
x3 <- c(0.419,0.752,0.499,0.919,0.819,-0.362,-0.156,-0.664,
        0.663,-0.252,0.957,-0.736,0.596,0.502,-0.991)
y <- c(4.327,6.513,4.847,9.838,6.931,4.606,2.738,2.191,
       6.657,1.630,11.556,4.372,6.010,5.397,-1.245)
x <- data.frame(x1=x1,x2=x2,x3=x3,y=y)
print(x)
#>         x1     x2     x3      y
#> 1  -0.959  0.829  0.419  4.327
#> 2  -0.781 -0.218  0.752  6.513
#> 3  -0.957  0.549  0.499  4.847
#> 4   0.413 -0.184  0.919  9.838
#> 5  -0.989 -0.683  0.819  6.931
#> 6   0.460 -0.274 -0.362  4.606
#> 7  -0.479  0.739 -0.156  2.738
#> 8  -0.436 -0.229 -0.664  2.191
#> 9  -0.329  0.289  0.663  6.657
#> 10 -0.866  0.662 -0.252  1.630
#> 11  0.988 -0.635  0.957 11.556
#> 12  0.885 -0.422 -0.736  4.372
#> 13 -0.760 -0.231  0.596  6.010
#> 14 -0.338  0.705  0.502  5.397
#> 15 -0.734  0.551 -0.991 -1.245
```

Then we can proceed with the regression. The function prints out also the sum of squared residuals (SSE).

```
a <- solveLS(x)
#> Sum of squared residuals: 0.694310
```

```r
# Present result with same precision as the data
a <- round(a,digits=3)
print(a)
#> [1]  2.009 -0.987  3.996  5.050
```

The SSE is relatively small. So it's expected that the data variability is well covered by the linear model found.

1.2 Exercise 02

Simulate a set of 15 data points coming from measuring x of the following physical law,

$$x(t) = 5t + \sin(2\pi t) + \cos(2\pi t) \quad , \quad 0 \leq t \leq 3$$

for the values of t equal to

$$t = 0.0\ ,\ 0.2\ ,\ 0.6\ ,\ 1.8\ ,\ 1.9\ ,\ 2.5\ ,\ 2.6$$

and where data are affected by random errors distributed according to the normal distribution with mean 0 and standard deviation 0.5. Use the seed 5522 for the simulation. Organise the data thus simulated into a data frame and use `solveLS`to estimate the physical law, assuming the following model,

$$x = a_1 t + a_2 \sin(2\pi t) + a_3 \cos(2\pi t)$$

You should find that the estimate of a_1, a_2, a_3 is close to their theoretical value, $a_1 = 5, a_2 = 1, a_3 = 1$. What is the SSE if the model does not have the cosine components? On a same graph, plot the data points simulated and the two regression curves corresponding to the two models.

SOLUTION

The simulation is done simply by creating an array for the 7 values of t, calculating the x array based on the formula suggested, and adding 7 randomly generated values from a normal distribution.

```r
# Sampled values of t (regular grid)
t <- c(0,0.2,0.6,1.8,1.9,2.5,2.6)

# Random errors (with seed suggested)
set.seed(5522)
eps <- rnorm(7,mean=0,sd=0.5)

# Simulated data
x <- 5*t+sin(2*pi*t)+cos(2*pi*t)+eps
```

To find the estimate for the 3 parameters of the model suggested, a multilinear regression is needed with the three variables,

$$x_1 = t \quad , \quad x_2 = \sin(2\pi t) \quad , \quad x_3 = \cos(2\pi t)$$

Thus the linear model is:

$$y = a_1 x_1 + a_2 x_2 + a_3 x_3$$

It is important for the use of the `solveLS` function, to consider that the intercept term is not present in the model. The data are turned into a data frame, prior to using them in `solveLS`.

```r
# Variables for the linear model
x1 <- t
x2 <- sin(2*pi*t)
x3 <- cos(2*pi*t)

# Data frame for use in solveLS
ddd <- data.frame(x1=x1,x2=x2,x3=x3,x=x)
```

```
print(round(ddd,digits=3))
#>    x1     x2     x3      x
#> 1 0.0  0.000  1.000  1.590
#> 2 0.2  0.951  0.309  1.651
#> 3 0.6 -0.588 -0.809  1.628
#> 4 1.8 -0.951  0.309  8.390
#> 5 1.9 -0.588  0.809  9.172
#> 6 2.5  0.000 -1.000 10.558
#> 7 2.6 -0.588 -0.809 11.419

# Parameter estimation via regression
a <- solveLS(ddd,intercept=FALSE)
#> Sum of squared residuals: 0.524553
print(a)
#> [1] 4.6927321 0.4277631 1.0984756
```

The estimated coefficients are relatively close to the $5, 1, 1$ values of the simulated data. The difference is obviously due to the effect of the random differences. If the model is

$$y = a_1 x_1 + a_2 x_2,$$

i.e. if the cosine term is dropped, the regression yields:

```
# Regression without cosine term
aa <- solveLS(ddd[c(1:2,4)],intercept=FALSE)
#> Sum of squared residuals: 5.003979
print(aa)
#> [1] 4.5098132 0.3208128
```

The SSE, 5.004, is much higher than the value 0.525 found with the previous model. This indicates that the second model provides a worst fitting, compared to the first. Visually, the fit quality can be appreciated by looking at the plot of the data points and the fitting curves.

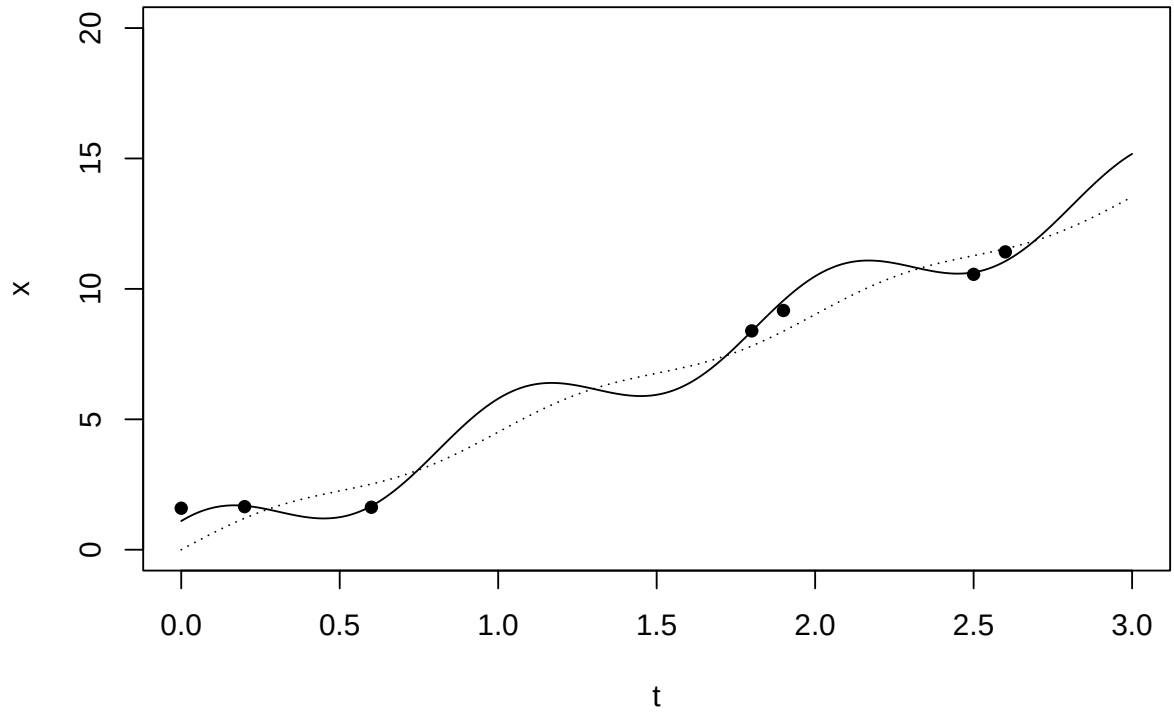
```
# Plot simulated points
plot(t,x,pch=16,xlab="t",ylab="x",
     xlim=c(0,3),ylim=c(0,20))

# Finer grid for plot
tt <- seq(0,3,length=1000)

# First curve
xx1 <- a[1]*tt+a[2]*sin(2*pi*tt)+a[3]*cos(2*pi*tt)

# Second curve
xx2 <- aa[1]*tt+aa[2]*sin(2*pi*tt)

# Plot both curves (dashed one is the second curve)
points(tt,xx1,type="l")
points(tt,xx2,type="l",lty=3)
```

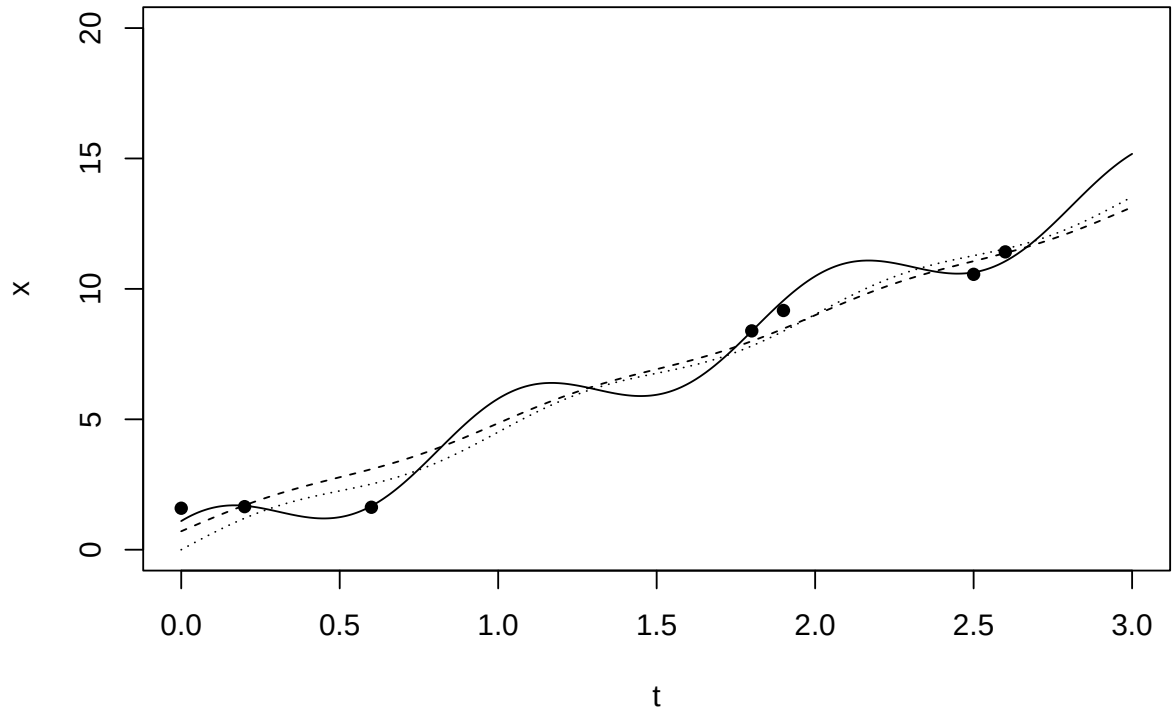


It might be asked whether the intercept term made things better, in the second case.

```
# Regression without cosine term, but with intercept term
aaa <- solveLS(ddd[c(1:2,4)])
#> Sum of squared residuals: 3.816207
print(aaa)
#> [1] 4.1421249 0.1752267 0.7074379

# New curve
xx3 <- aaa[1]*tt+aaa[2]*sin(2*pi*tt)+aaa[3]

# Plot all curves
# (dashed one is the second curve, dotted one the new curve)
# Plot simulated points
plot(t,x,pch=16,xlab="t",ylab="x",
     xlim=c(0,3),ylim=c(0,20))
points(tt,xx1,type="l")
points(tt,xx2,type="l",lty=3)
points(tt,xx3,type="l",lty=2)
```



The SSE is definitely better (3.816), but the curve is still struggling do account for the data properly. In situations like this, it is clear that the model has one or more terms missing, or that it is entirely wrong.

1.3 Exercise 03

Using the simulated data of Exercise 02, find out which polynomial fits them best, using the variance σ_e^2 as criterion.

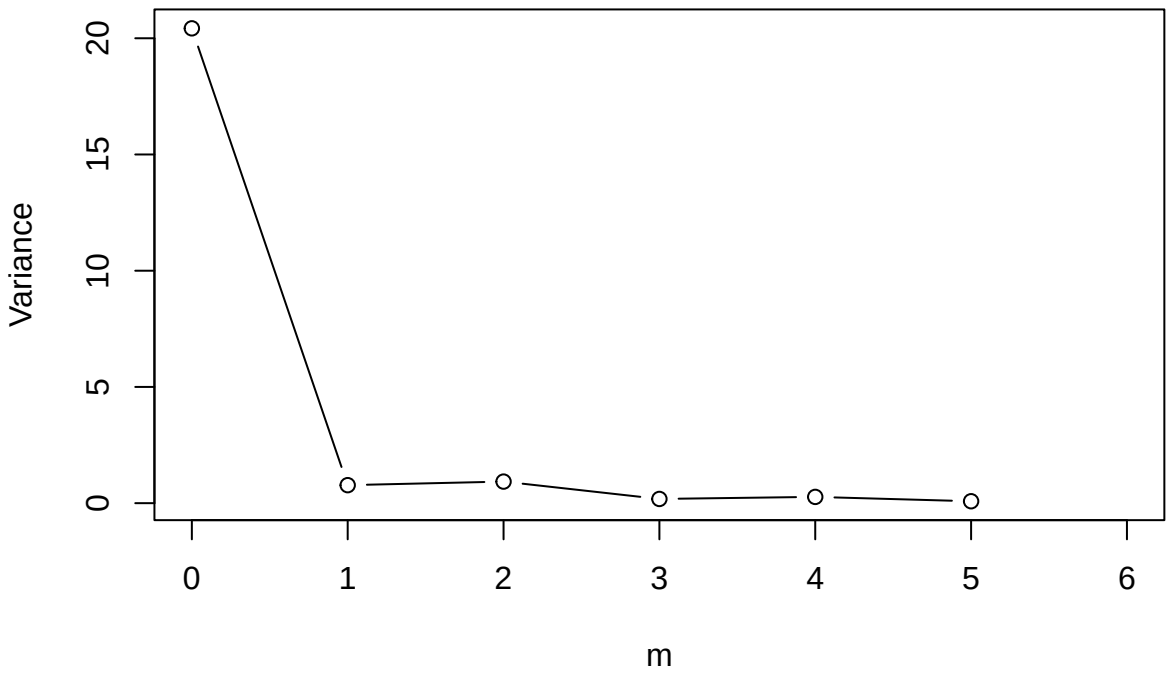
SOLUTION

The criterion makes use of the plot m versus σ_e^2, where m is the polynomial's degree. We can use the `comphy` function `which_poly` to do that.

```
# Reproduce simulated data of Exercise 02
t <- c(0,0.2,0.6,1.8,1.9,2.5,2.6)
set.seed(5522)
eps <- rnorm(7,mean=0,sd=0.5)
x <- 5*t+sin(2*pi*t)+cos(2*pi*t)+eps

# Data frame for input
pts <- data.frame(t=t,x=x)

# Test for best polynomial (plot provided by default)
# The output is a data frame
dtmp <- which_poly(pts)
```



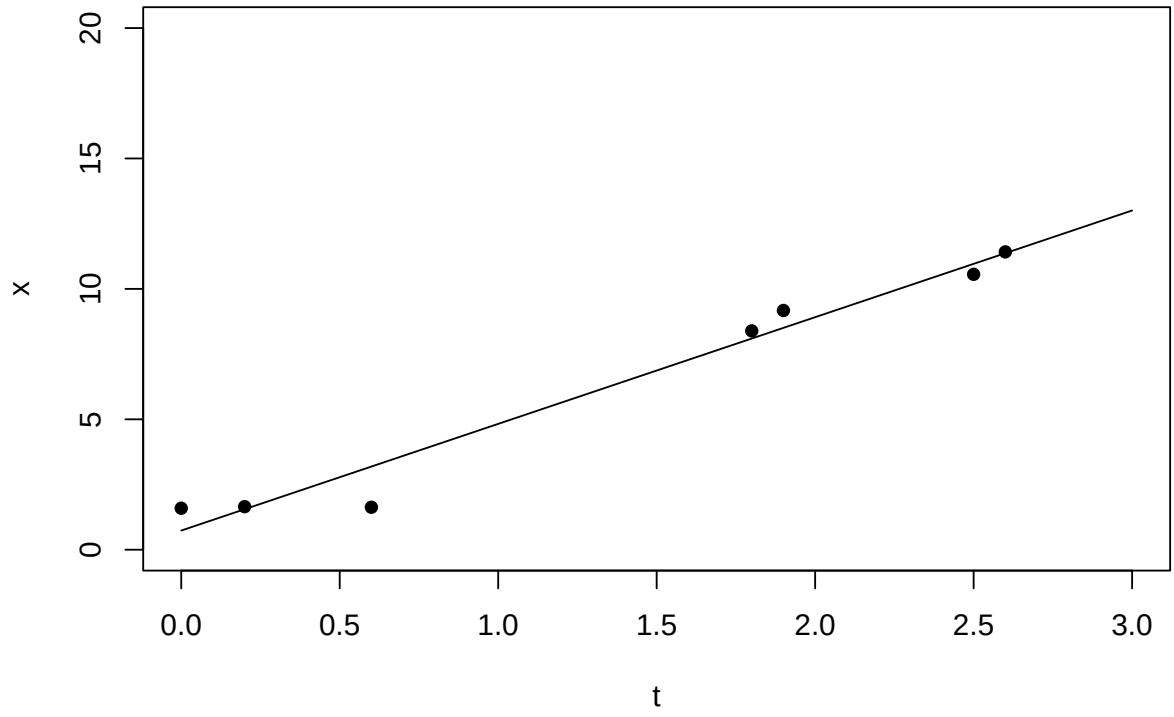
From the plot is very clear that a polynomial of degree one can account for most data variability, degrees higher than one would most likely model data noise. The fit is, subsequently, carried out with the function `polysolveLS`.

```
# Linear regression with a straight line
ltmp <- polysolveLS(pts,1)
a <- ltmp$a
print(a)
#> [1] 4.0902289 0.7344686
print(ltmp$SSE)
#> [1] 3.87083

# Finer grid for plotting the regression curve
tt <- seq(0,3,length=1000)

# Regression curve
xx <- a[1]*tt+a[2]

# Plot
plot(t,x,pch=16,xlab="t",ylab="x",
     xlim=c(0,3),ylim=c(0,20))
points(tt,xx,type="l")
```



The sum of squared residuals is SSE= 3.871, even better than the values found in the last models of Exercise 02. This means that, unless an additional trigonometric contribution is known to be part of the model, it is better to avoid it as an unnecessary complication. One should also notice that the estimate of a_1 is close to the value 5 used to simulate the data, which makes sense because $5t$ is the trend dominating the data in this case.

1.4 Exercise 04

Consider the dampened oscillations of a body of mass $m = 5$ kg, attached to a spring of unknown constant k, and oscillating horizontally, with angular frequency $\omega = 1.998$ (rad/sec), on a smoothed plane which progressively decreases the amplitude of the oscillation. The motion of the body is described by the equation

$$x = x(t) = Ae^{-bt/(2m)} \cos(\omega t),$$

which describes the position of the body around its equilibrium position. In the above equation,

$$\omega = \sqrt{\frac{k}{m} - \left(\frac{b}{2m}\right)^2}$$

In order to calculate the spring constant, k, and the coefficient b that causes the dampening, a series of $n = 11$ regular measurements of the body's position is taken at regular time intervals between 0 seconds and

20 seconds. The values are summarised in the following table.

t (sec)	x (m)
0	16.218
2	−11.947
4	−1.979
6	5.046
8	−4.620
10	2.821
12	2.026
14	−3.459
16	2.597
18	0.959
20	−0.982

Using linear regression, estimate the value of both k (N/m) and b (N sec/m).

SOLUTION

When the natural logarithm of both sides of the formula describing the oscillation is taken, the result is:

$$\log(x) = \log(A) - \frac{b}{2m}t + \log(\cos(\omega t))$$

As both x and ωt are known at the 11 sampled points, the above relation can be re-written as,

$$\log(x) - \log(\cos(\omega t)) = -\frac{b}{2m}t + \log(A)$$

This is equivalent to the linear model,

$$y = a_1 t + a_2,$$

where,

$$y \equiv \log(x) - \log(\cos(\omega t)) \quad , \quad a_1 = -\frac{b}{2m} \quad , \quad a_2 = \log(A)$$

Let us load the data in memory so that all calculation can be carried out using R.

```
t <- c(0,2,4,6,8,10,12,14,16,18,20)
x <- c(16.218,-11.947,-1.979,5.046,-4.620,
        2.821,  2.026,-3.459,2.597, 0.959,-0.982)

# Omega is 1.99750
#om <- 1.99750
om <- 1.998

# Other constants
m <- 5

# New variable y
y <- log(x)-log(cos(om*t))
#> Warning in log(x): NaNs produced
#> Warning in log(cos(om * t)): NaNs produced
```

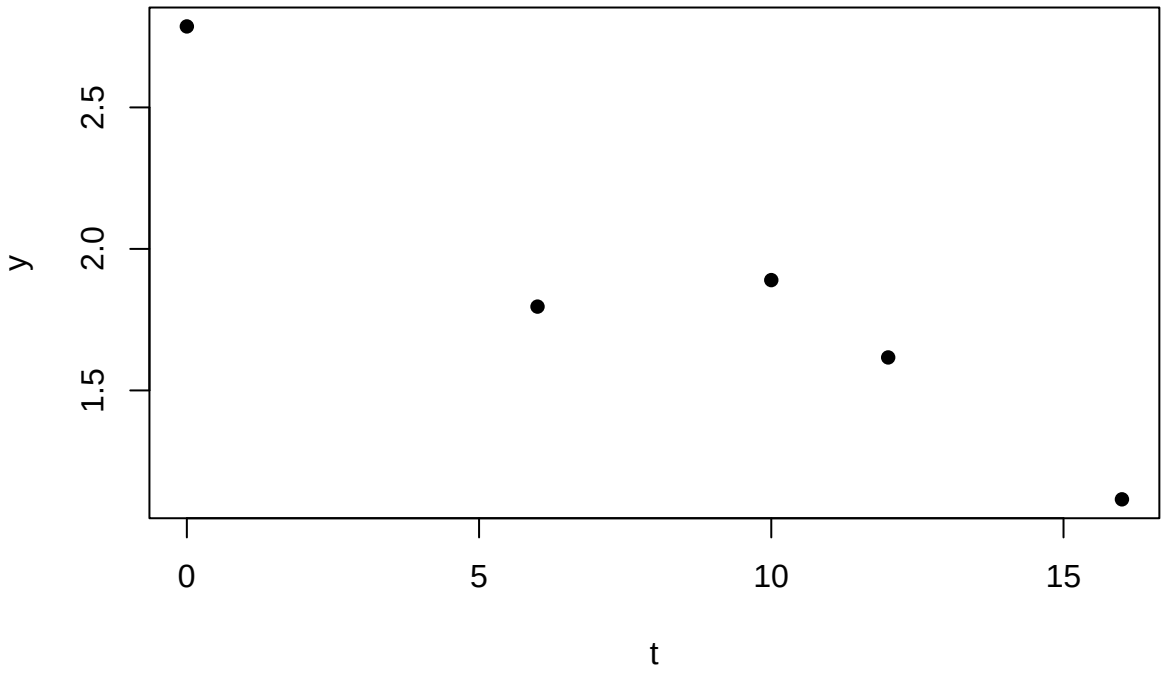
The warning printed out is reminding us that some of the arguments of the logarithm are non positive, which is not allowed. We will have, therefore to make a selection of the data, if we want to persevere with the original plan of performing linear regression. The data are not many, but this is our only option. This is how only the positive values can be selected, remembering that also $\cos(\omega t)$ must be positive.

```
idx <- which(x > 0 & cos(om*t) > 0)

# New data
newt <- t[idx]
newx <- x[idx]

# Data for regression
newy <- log(newx)-log(cos(om*newt))
pts <- data.frame(t=newt,y=newy)

# The data plot should give the impression of a
# negative gradient
plot(pts,pch=16,xlab="t",ylab="y")
```

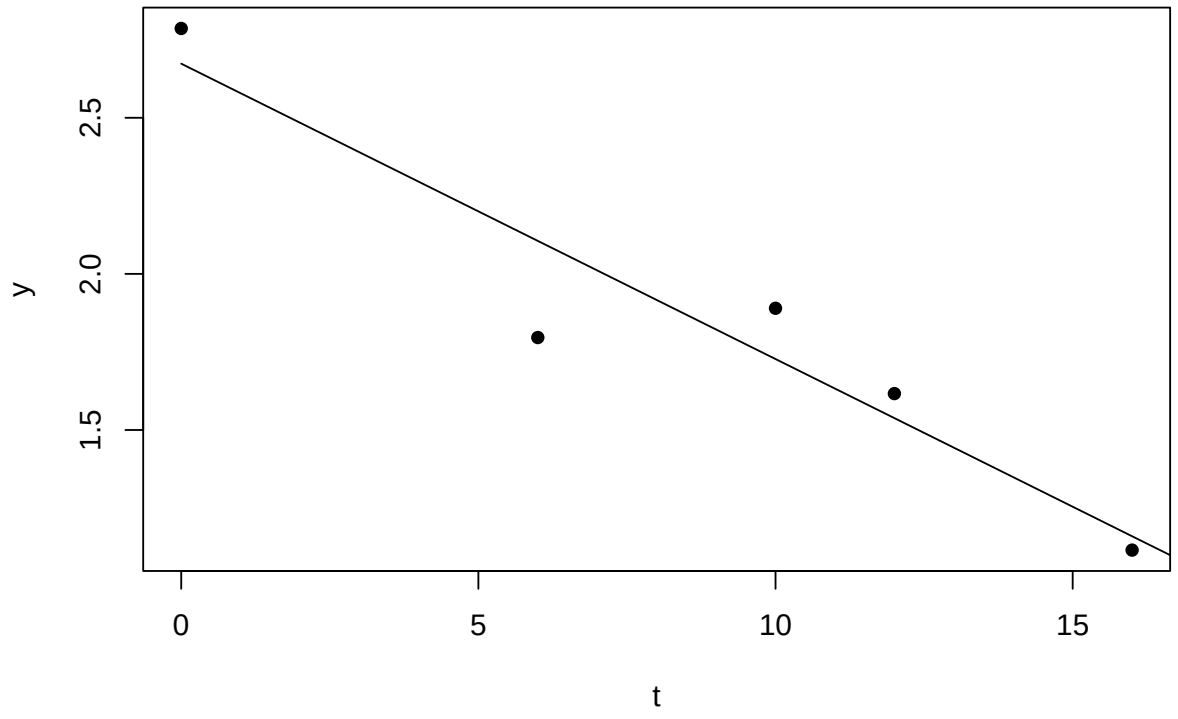


The regression can be carried out using `solveLS`. The regression curve is shown here, overlapped with the few data points available.

```
# Regression
a <- solveLS(pts)
#> Sum of squared residuals: 0.143123
print(a)
#> [1] -0.09460272  2.67325789

# Regression curve
tt <- seq(0,20,length=1000)
yy <- a[1]*tt+a[2]
```

```
plot(pts,pch=16,xlab="t",ylab="y")
points(tt,yy,type="l")
```




```
# Value of b
b <- -2*m*a[1]
print(b)
#> [1] 0.9460272

# Value of A
A <- exp(a[2])
print(A)
#> [1] 14.48709
```

The fit appears to be reasonable and the SSE is also relatively small (0.141). The advantage of having carried out the regression is that now we have an estimate of A and b. Knowledge of b implies knowledge of k:

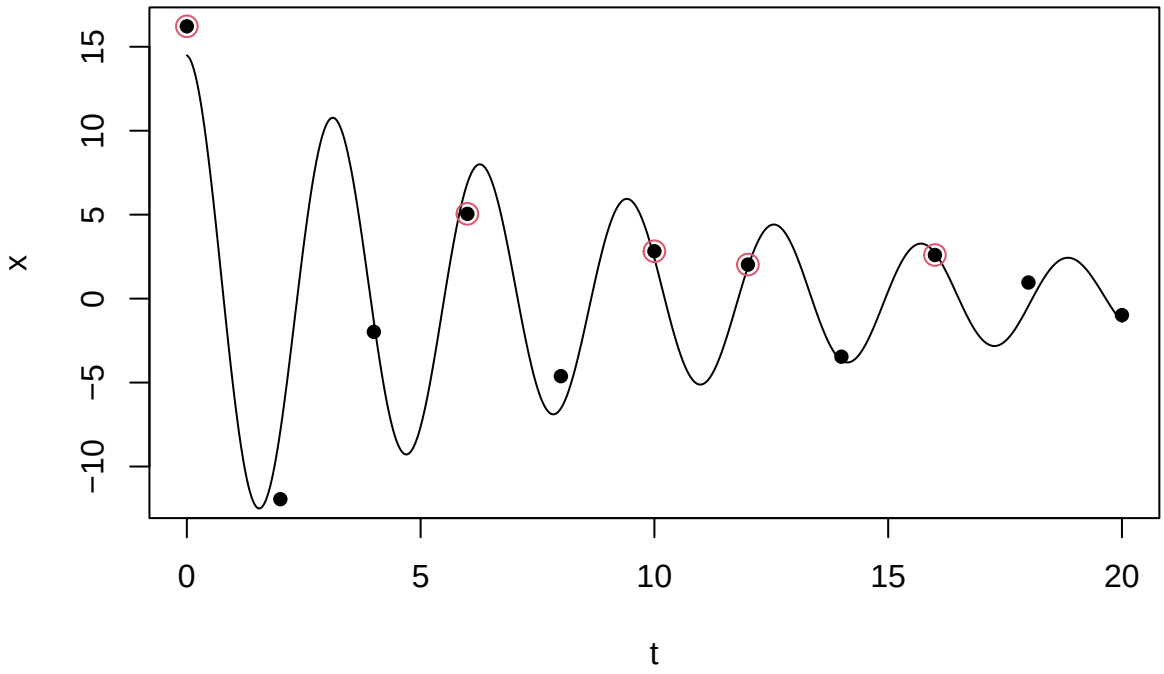
$$\omega = \sqrt{\frac{k}{m} - \left(\frac{b}{2m}\right)^2} \quad \rightarrow \quad k = m\omega^2 + \frac{b^2}{4m}$$

All parameters needed to determine the model are thus available and can be used to check how the curve determined fits all data.

```
# Value of k
k <- m*om^2+b^2/(4*m)
print(k)
#> [1] 20.00477
```

```r
# Final estimated curve overlapped on data
xx <- A*exp(-b*tt/(2*m))*cos(om*tt)

# Plot
# (open red circles are the data used for initial estimate)
plot(t,x,pch=16,xlab="t",ylab="x")
points(tt,xx,type="l")
points(t[idx],x[idx],cex=1.5,col=2)
```



The important feature of the plot above is that the points that have not been used for the regression are relatively close to the estimated curve. This is a good indication that the fitting is relatively accurate. The unused data kind of *validate* the goodness of fit, a feature known as *cross validation* in data analysis.

1.5 Exercise 05

Water is discharged from a container, according to the following dynamical formula:

$$h(t) = a + \frac{b}{1+\sqrt{t}},$$

where h is the height of the discharging water surface above the ground, and a and b two positive constants related to the water's height before discharging starts (time $t = 0$) and after it has finished (time $t = \infty$).

The level has been measured at random, specified times and the values are reported in the following table.

t (sec)	h (cm)
3.5	117.6
7.1	113.7
14.1	110.5
28.2	107.9
31.8	107.5
49.4	106.2
52.9	106.0
56.5	105.9

Using least squares regression, find the two constants a, b and the water's height before discharging starts and after it has finished.

SOLUTION

The levels of water before discharging has started, and after it has finished are:

$$h_0 = h(0) = a + b \qquad , \qquad h_f = \lim_{t\to\infty} h(t) = a$$

These levels can be calculated once the parameters a and b have been estimated. This can be done *via* linear regression because the main relation between t and h can be transformed and made linear. It will suffice to take $1/(1+\sqrt{t})$ as the new independent regression variable, while leaving h as the dependent variable.

$$h = a + \frac{b}{1+t} \quad \rightarrow \quad h = a + b\tau \quad \text{, with} \quad \tau = \frac{1}{1+t}$$

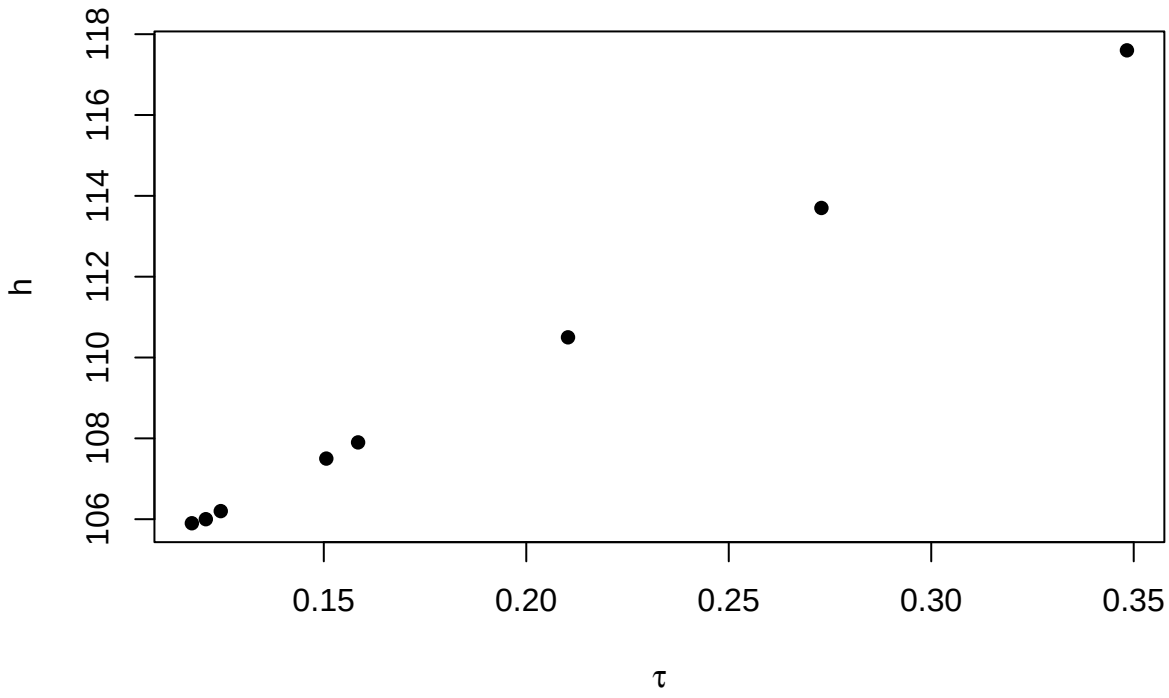
Let us load the data in memory and calculate how they are transformed.

```
# Experimental data
t <- c(3.5,7.1,14.1,28.2,31.8,49.4,52.9,56.5)
h <- c(117.6,113.7,110.5,107.9,107.5,106.2,106.0,105.9)

# Transformed data
newt <- 1/(1+sqrt(t))
newh <- h

# Save them in a data frame
pts <- data.frame(newt,newh)
print(pts)
#>        newt  newh
#> 1 0.3483315 117.6
#> 2 0.2728824 113.7
#> 3 0.2103051 110.5
#> 4 0.1584694 107.9
#> 5 0.1506217 107.5
#> 6 0.1245561 106.2
#> 7 0.1208716 106.0
#> 8 0.1174171 105.9

# Plot data to verify visually they are roughly
# located around a straight line
plot(pts,pch=16,xlab=expression(tau),ylab=expression(h))
```



The alignment of the data points along a seemingly straight line is a strong indication that the relation used is correct. Next, the regression coefficients can be estimated via least squares.

```
# Linear regression
coeffs <- solveLS(pts)
#> Sum of squared residuals: 0.010183
print(coeffs)
#> [1] 50.77876 99.86955
```

The SSE is very small, as expected (SSE=0.010). The estimated coefficients are,

$$a = 99.9 \qquad , \qquad b = 50.8$$

Therefore in the full container the water has level 150.7 cm, while once it is fully discharged (at $t = \infty$, meaning after a sufficiently long time) the level is 99.9 cm.

2 Exercises on statistical linear regression

2.1 Exercise 06

Using the `lm` function, find the estimated parameters for the following models:

1. $y = 2x + 1$
2. $y = 6x_1 - 2x_2 + 3x_3 + 1$

The independent variables, x, x_1, x_2, x_3 have the following values:

$$\begin{aligned} x &= \{0.23, 0.31, 0.34, 0.64, 0.65, 1.42, 2.07, 3.03, 3.17, 3.27, 4.96\} \\ x_1 &= \{0.09, 0.34, 1.00, 1.04, 2.05, 2.05, 2.59, 3.44, 4.44, 4.59, 4.99\} \\ x_2 &= \{1.39, 4.51, 4.63, 5.12, 5.50, 7.89, 8.03, 9.48, 9.64, 9.64, 9.94\} \\ x_3 &= \{5.13, 5.41, 5.45, 5.49, 5.62, 5.67, 5.93, 6.41, 6.79, 6.97, 7.77\} \end{aligned}$$

The errors (residuals) for the first simulation are generated by a random distribution with mean 0 and variance $\sigma^2 = 0.01$ (use seed 1132), while for the second simulation the mean is still 0 and the variance $\sigma^2 = 0.04$ (use seed 2311).

SOLUTION

Let us start with the first simulation.

```
# Simulating y=2x+1
x <- c(0.23,0.31,0.34,0.64,0.65,1.42,2.07,3.03,3.17,3.27,4.96)
n <- length(x)
s <- sqrt(0.01)
set.seed(1132)
y <- 2*x+1+rnorm(n,mean=0,sd=s)

# Regression
md1 <- lm(y ~ x)
slm1 <- summary(md1)
print(slm1)
#>
#> Call:
#> lm(formula = y ~ x)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -0.16455 -0.05798  0.02546  0.05808  0.12603
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  1.06220    0.04629   22.94 2.70e-09 ***
#> x            1.97191    0.01951  101.06 4.61e-15 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.09801 on 9 degrees of freedom
#> Multiple R-squared:  0.9991, Adjusted R-squared:  0.999
#> F-statistic: 1.021e+04 on 1 and 9 DF,  p-value: 4.614e-15

# Estimated variance
print(slm1$sigma^2)
#> [1] 0.009605766
```

So the estimates for the first model, $y = a_1 x + a_2$ are 1.97191 for a_1 (close to its correct value, 2) and 1.06220 for a_2 (close to its correct value, 1). The estimated variance is 0.00961, relatively close to its correct value, 0.01.

For the second simulation we have.

```
# Simulating y=6x1-2x2+3x3+1
x1 <- c(0.09,0.34,1.00,1.04,2.05,2.05,2.59,3.44,4.44,4.59,4.99)
n <- length(x1)
```

```r
x2 <- c(1.39,4.51,4.63,5.12,5.50,7.89,8.03,9.48,9.64,9.64,9.94)
x3 <- c(5.13,5.41,5.45,5.49,5.62,5.67,5.93,6.41,6.79,6.97,7.77)
s <- sqrt(0.04)
set.seed(2311)
y <- 6*x1-2*x2+3*x3+1+rnorm(n,mean=0,sd=s)

# Regression
md2 <- lm(y ~ x1+x2+x3)
slm2 <- summary(md2)
print(slm2)
#>
#> Call:
#> lm(formula = y ~ x1 + x2 + x3)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -0.40968 -0.11751  0.03982  0.10186  0.28137
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  2.95958    1.68074   1.761    0.122
#> x1           6.12695    0.21405  28.624 1.63e-08 ***
#> x2          -1.97420    0.07677 -25.714 3.44e-08 ***
#> x3           2.59066    0.31827   8.140 8.16e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.2387 on 7 degrees of freedom
#> Multiple R-squared:  0.9994, Adjusted R-squared:  0.9991
#> F-statistic:  3637 on 3 and 7 DF,  p-value: 1.553e-11

# variance
print(slm2$sigma^2)
#> [1] 0.05696041
```

If the model is written as $y = a_1x_1 + a_2x_2 + a_3x_3 + a_4$, the estimate for a_1, a_2, a_3 are relatively good (by looking at the number of stars at the end of the corresponding lines, the chances of those values being accidental are really low). But the estimate of a_4 is not very reliable and, in fact, the estimate is not close to the correct value, 1. The estimate of the variance σ^2 is 0.05696, which is slightly higher than the correct value, 0.04.

2.2 Exercise 07

The estimation of parameters in multilinear regression can be problematic when one or more independent variables are correlated (*multicollinearity*). This exercise is about perfect collinearity between two independent variables in a linear model of the type,

$$y = a_1x_1 + a_2x_2 + a_3$$

The exercise (and its solution) provides insights on the problem of multicollinearity and on how this can be explained and managed.

The data points are listed in the following table:

x_1	x_2	y
0.0	−5.0	−7.16
0.5	−4.5	−3.06
1.0	−4.0	0.96
1.5	−3.5	5.19
2.0	−3.0	9.02
2.5	−2.5	13.02
3.0	−2.0	17.10
3.5	−1.5	21.17
4.0	−1.0	24.86
4.5	−0.5	29.27
5.0	0.0	32.82
5.5	0.5	37.00
6.0	1.0	40.90
6.5	1.5	44.95
7.0	2.0	48.91
7.5	2.5	53.08
8.0	3.0	56.95
8.5	3.5	60.81
9.0	4.0	65.18
9.5	4.5	69.22
10.0	5.0	73.04

a. Try the least squares regression on these data, using the function `lm`. You will not be able to proceed. Can you explain why?

b. Calculate, using the function `cor`, the correlation between x_1 and x_2.

c. Using the result from part b, attempt a way out the problem met in part a, and finally find an estimate for a_1, a_2, a_3.

SOLUTION

Part a

First, we should try and load the data in memory.

```
# Data points
x1 <- c(0.0,0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0,
        5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10.0)
x2 <- c(-5.0,-4.5,-4.0,-3.5,-3.0,-2.5,-2.0,-1.5,-1.0,-0.5,0.0,
        0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0)
y <- c(-7.16,-3.06,0.96,5.19,9.02,13.02,17.10,21.17,24.86,29.27,
       32.82,37.00,40.90,44.95,48.91,53.08,56.95,60.81,65.18,69.22,73.04)
```

Then we use `lm` with the appropriate linear model.

```
# Regression
# summary returns an informative message
model <- lm(y ~ x1+x2)
summary(model)
#>
#> Call:
#> lm(formula = y ~ x1 + x2)
#>
#> Residuals:
```

```
#>      Min       1Q   Median       3Q      Max
#> -0.21813 -0.11077 -0.01341  0.09887  0.26150
#>
#> Coefficients: (1 not defined because of singularities)
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -7.01359    0.05898  -118.9   <2e-16 ***
#> x1           8.00491    0.01009   793.4   <2e-16 ***
#> x2                NA         NA      NA       NA
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.14 on 19 degrees of freedom
#> Multiple R-squared:      1,  Adjusted R-squared:      1
#> F-statistic: 6.294e+05 on 1 and 19 DF,  p-value: < 2.2e-16
```

One of the parameters has not been determined and a message is printed, warning us that one of them is not defined because of singularities. This is due to the independent variables not being really independent, but being, in fact, collinear.

Part b

In order to check how severe the multicollinearity is, we can calculate the correlation between x_1 and x_2.

```
c
#> function (...)  .Primitive("c")
# Correlation between x1 and x2
print(cor(x1,x2))
#> [1] 1
```

The correlation is exactly 1. This can only happen when one of the two variables is a linear function of the other, i.e.

$$x_2 = \alpha x_1 + \beta$$

Part c Given what found in part c, we can eliminate x_2 from the linear model and carry out regression analysis with two, rather than three, parameters. To find the coefficients α and β, we can use two of the data points (x_1, x_2) provided. So, using,

$$(x_1, x_2) = (0, -5) \qquad \text{and} \qquad (x_1, x_2) = (1, -4)$$

we have:

$$\begin{cases} -5 &= 0\alpha + \beta \\ -4 &= 1\alpha + \beta \end{cases} \quad \Rightarrow \quad \begin{cases} \alpha &= 1 \\ \beta &= -5 \end{cases}$$

The transformation is, therefore,

$$x_2 = x_1 - 5$$

What we have to do, next, is to ignore x_2 and carry out the regression only using x_1 and y. The parameters found can then be used to recover the parameters of the original regression. In fact,

$$y = a_1 x_1 + a_2 x_2 + a_3$$

$$\Downarrow$$

$$y = a_1 x_1 + a_2 (x_1 - 5) + a_3$$

$$\Downarrow$$

$$y = (a_1 + a_2) x_1 + (-5a_2 + a_3) \equiv a_1^{'} x_1 + a_2^{'},$$

where $a_1^{'}$ and $a_2^{'}$ are the parameters of the new regression. These are only two, not enough to recover the three parameters of the original model. Inverting the transformation, we have thus

$$\begin{array}{rcl} a_1^{'} &=& a_1 + a_2 \\ a_2^{'} &=& -5a_2 + a_3 \end{array} \quad \Rightarrow \quad \begin{array}{rcl} a_1 &=& a_1^{'} - \gamma \\ a_2 &=& \gamma \\ a_3 &=& a_2^{'} + 5\gamma, \end{array}$$

where γ is an arbitrary parameter.

```
# New regression (y=a'_1x_1+a'_2)
model2 <- lm(y ~ x1)
summary(model2)
#>
#> Call:
#> lm(formula = y ~ x1)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -0.21813 -0.11077 -0.01341  0.09887  0.26150
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -7.01359    0.05898  -118.9   <2e-16 ***
#> x1           8.00491    0.01009   793.4   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.14 on 19 degrees of freedom
#> Multiple R-squared:      1,  Adjusted R-squared:      1
#> F-statistic: 6.294e+05 on 1 and 19 DF,  p-value: < 2.2e-16
```

The regression returns clear values for both $a_1^{'}$ and $a_2^{'}$. We have, therefore:

$$\begin{array}{rcll} a_1 &=& 8.005 - \gamma & \\ a_2 &=& \gamma & \text{with} \quad \gamma \in \mathbb{R} \\ a_3 &=& -7.014 + 5\gamma & \end{array}$$

The set of parameters satisfying the above relations are the coefficients we were looking for. The equation,

$$y = a_1 x_1 + a_2 x_2 + a_3$$

represents a plane in the Cartesian system in which the coordinates are represented on the x_1 axis, the x_2 axis and the y axis. Due to the collinearity of x_1 and x_2, the plane is not unique. It is, in fact, represented by parametric equations, where the free parameter is the γ introduced earlier. All the planes intersect at a straight line described by the parametric equations:

$$x_1 = \delta, \; x_2 = \delta - 5, \; y = 8.005\delta - 7.014$$

These parametric equations can be obtained simply as intersection of any two planes of the parametric family above (say the one with $\gamma = 1.005$ and the one with $\gamma = 0$), where $x_1 = \delta$ is the parameter used.