MSG400-TMS150

Stochastic data processing and simulation 2024

Model choice in linear regression and a few last bits about parameter inference

Umberto Picchini

In the previous lecture we have recollected some notions pertaining (simple) linear regression and some basics of inference such as OLS and confidence intervals. Here we consider a few more bits about parameter inference and prediction, before moving to the topic of model complexity and model section.

Exercises 1 and 2 constitute the assignment A2a. The A2a assignment must be written as a LaTeX report and be submitted to Canvas. Notice the "recommended deadline" on the course webpage.

Please use the recommended report template provided on the course page. You must also upload separately from the report all the code you produced. And you must also embed the code inside the report, as illustrated in the report template found on the course webpage. Recall that each submitted report and the code therein is INDIVIDUAL not group work.

Finally, since you have to write a proper report: as from the provided template, you are also asked to produce some background on the methodology you use. So do not just write answers to the exercise questions. See the guidelines on report writing posted on Canvas.

For a given project report, 0.5 points will be deducted if the report is not clearly structured or is otherwise hard to understand. Likewise, 0.5 points will be deducted if the code attached to the report is not properly structured and commented. The report should not be longer than 10 pages including figures, but excluding appendices. Figures and axes labels should be big enough to be readable if printed. It is OK to use colors.

Full details on grading are on the course webpage.

1 Approximate prediction intervals

As a data-scientist, once you have put-in your fair share of effort to understand your data, and selected a model to fit after checking that assumptions are met[1], then you have finally obtained parameter estimates. You may be interested in doing some form of prediction and assess the variability of your predictions, but note the following. One thing is to estimate the *expected value of the response* y at some value $x = x^*$, which is $E(y|x^*)$, and this is of course estimated via $\hat{E}(y|x^*) = \hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x^*$, where we have assumed as usual that $E(\epsilon) = 0$. A different thing is to predict a *future response*. Predicting a response means predicting a new observation of "type y" at x^*, meaning something like

$$y_{new} = \hat{\beta}_0 + \hat{\beta}_1 x^* + \epsilon_{new} \tag{1}$$

[1] We did not discuss assumptions checking. We disregard these very important aspects for reasons of brevity, but these will be discussed at length in courses such as MSG500-MVE190.

where y_{new} is *not part of our dataset*, but is a potential new measurement (it just hasn't been observed yet) which we obtain using the estimates computed on the data we have ($\hat{\beta}_0$ and $\hat{\beta}_1$). We can also assume $\epsilon_{new} \sim N(0, s^2)$, with s based on the data we already have and computed as described in the previous notes. Do you see the difference with using (1) instead of $\hat{E}(y|x^*)$? As you well know, with the latter we estimate the average (expected) value of the response, based on the usual fitted model, eg the black line in Figure 1. In (1) instead we add some new measurement error to the already fitted model, since we want to predict a new response y_{new}. So in this case we are not interested in understanding the "average response", but we want to be able to reason about potential new measurements and also how they distribute.

For example, imagine the manager at the geology institute that employs you wants to know the width of the spread of possible future earthquake's magnitudes. She is not interested in the expected magnitude of the next earthquake, nor in inferring a confidence interval for such expected value. She is concerned about predicting a future actual magnitude and the random variation around it (uncertainty quantification). She wants you to quantify the uncertainty about the magnitude of a future earthquake.

Similarly to confidence intervals, whenever we want to construct intervals that will most likely contain actual future observations, with a confidence level $1 - \alpha$, then we have *prediction intervals*. Under the usual assumptions placed on simple linear regression, exact formulae exist to compute such intervals, see the appendix. However what I want you to do here is to use a simulation-based approach. It is extremely simple: (i) obtain estimates $(\hat{\beta}_0, \hat{\beta}_1)$ from the actual data at hand, (ii) plug those inside (1), (iii) repeatedly simulate as many $\epsilon_{new,b} \sim N(0, s^2)$ as wanted, say B times, to obtain corresponding predictions at scalar x^*

$$\hat{y}_{new,b} = \hat{\beta}_0 + \hat{\beta}_1 x^* + \epsilon_{new,b}, \qquad \epsilon_{new,b} \sim N(0, s^2), \qquad b = 1, ..., B.$$

You can then produce histograms of the $[y_{new,1}, ..., y_{new,B}]$ for B fairly large (e.g. $B = 2,000$). But you can also take the 2.5 and 97.5 empirical quantiles of $[y_{new,1}, ..., y_{new,B}]$, and these quantiles will be the boundaries of an approximate 95% prediction interval for future observations, when $x = x^*$. If you connect with a line the lower bounds at several attempted values of x^*, and with another line you connect the upper bounds, you may obtain something resembling Figure 1, for an illustrative example. There researchers had "mean platelet volume" (MPV) as y variable and running time as x variable[2]. From the figure we can say that (if the model is appropriate) at 100 min of running time we expect a new MPV measurement to be included in the interval [7.7,11.4] with 95% confidence.

[2]Mean platelet volume is a measure of the average size of your platelets, a type of blood cell that helps prevent bleeding

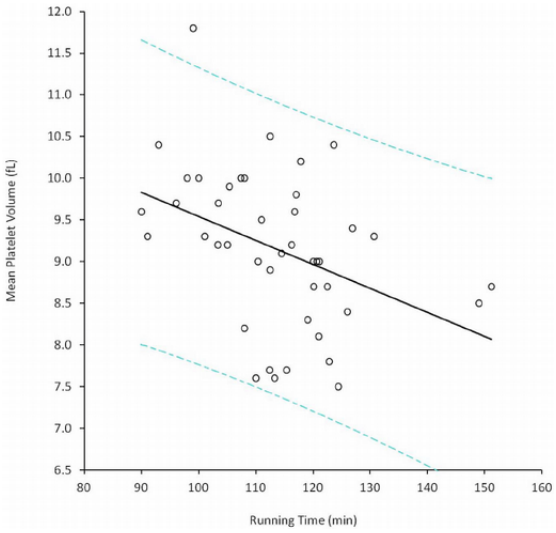


Figure 1: Just an example of 95% prediction intervals (light blue) and the regression fit $\hat{E}(y|x)$ (black line).

Exercise 1 (=1.5 points) : Here we learn how to quantify uncertainty around predicted data, and understand how serious is the impact of using a "bad" model when producing scientific considerations.

(i) (1 point) Using the `sleeptab` data, by using the parameter estimates obtained on the dataset by fitting the model $E(\log(brwt)) = \beta_0 + \beta_1 \log(bwt)$ on all observations, compute $B = 2,000$ predictions of **new observations** (hence with added measurement error) $\widehat{\log(brwt)}_{new}$ at 16 equispaced values of $\log bwt \in [-5, -4, ..., 10]$ (that is 2,000 predictions for log(bwt) $= -5$, then 2,000 predictions for log(bwt)$= -4$, etc). Then for each set comprising 2,000 values, compute the approximate 95% prediction intervals. Therefore you get equipped with sixteen upper bounds and sixteen lower bounds.

Produce the scatterplot of the actual data `log(brwt)` vs `log(bwt)` and on top of this add a line connecting the several lower bounds obtained, and then another line connecting the upper bounds obtained (similarly to Figure 1). You should see three observations outside the intervals. What do we conclude? Are these three observations outside the intervals by a considerable amount? Motivate (no Yes/No answers). *[**Note**: just because it will be easier to grade, when simulating data, please place `set.seed(321)` before the outermost for loop. So we get the same results.]*

(ii) (0.5 points) Do the same as in part (i) but this time using non-transformed data (bwt,brwt). Compute $B = 2,000$ predictions of new observations $\widehat{bwt}_{new}$ using 701 equispaced values of $bwt \in [0, 10, 20, ..., 7000]$. What do we conclude in this case? Once more, here you find that some observations are outside the intervals. Are these outside by a considerable amount? **Comment on your findings**. *[**Note**: also here, please place `set.seed(321)` before the outermost for loop.]*

2 Model complexity and model selection

So far we have reasoned about a model that was *given*, and hence we assumed that we were satisfied with the model and made inferences from it. However the problem of model selection is clearly relevant. It is a quite complex problem and here we just scratch the surface of it.

We now touch upon the concept of *model complexity* in linear models. This tries to answer the

question "how many variables/covariates shall we insert in the right-hand-side of $E(y) = f(\cdot)$?" Clearly this is not a discussion that pertains *simple* linear regression, where there is exactly one covariate to consider, but is instead a discussion that makes sense when we make use of several variables/terms to fit the response y. These multiple terms could be p different covariates $[x_1, x_2, ..., x_p]$, and in such case we have a *multiple linear regression model*

$$E(y|x_1, ..., x_p) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p.$$

If we denote with $[x_{i1}, ..., x_{ip}]$ the covariates observed on case i $(i = 1, ..., n)$, the expected value for response y_i can be written as

$$E(y_i|x_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}.$$

For example, if we were to reconsider the cars example we analysed in the previous lecture, we could try to fit the distance `dist` using as covariates the weight of the car, its brand, the number of seconds required to accelerate from 0 to 100 miles/sec, etc (note: these information are not provided in the `cars` dataset. It was just to name possibilities if we actually had those variables in the dataset). However, multiple linear regression requires a number of lectures to be properly introduced and we do not have enough time, so we move to a simpler possibility. Even if we have only one x variable to use to make sense of y, we can postulate several *multiple regression* models given by polynomials of x. That is we may consider

$$E(y|x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p, \qquad p = 1, 2, ...$$

with $p \geq 1$ an integer representing the degree of a polynomial. Of course we can also write

$$E(y_i|x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_p x_i^p, \qquad p = 1, 2, ... \quad i = 1, ..., n$$

Let's recall the example we are already familiar with. Look at Figure 2. Who said that $E(distance|speed) = \beta_0 + \beta_1 speed$ is a great choice? It's certainly not absurd. It does make some sense, but is it better than a quadratic relationship? Or a cubic one? We even have a polynomial of order five in the figure. Is it order five "too much" to explain these data? Is order one "too little"?

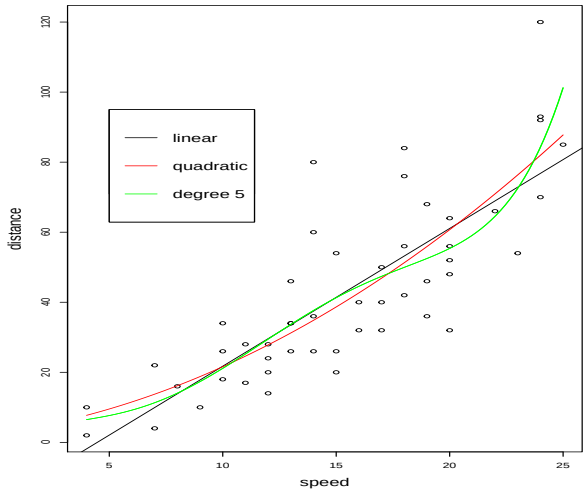


Figure 2: Linear, quadratic and a degree-five polynomial fit.

So how "complex" should the model be? We'll see we need to find a balance between interpretability of the model (a simpler model is easier to understand) and predictive ability of the model (a more complex model tends to "fit better" the current data, but might also "overfit" it).

We try to follow the Occam's principle, or "Occam's razor": *when presented with competing hypotheses that make the same predictions, one should select the solution with the fewest assumptions*. That is, there is no need to build a model that is unnecessarily complex, if a simpler one is "good enough". Obviously, we need an objective and scientific tool to decide. But first, supposing model complexity is not an issue, can we construct an index of "goodness of fit" of the model to the data?

3 Recap: goodness-of-fit of a linear model, the R^2 index

Is there a way to measure how well the estimated regression model fits the data? Indeed the R^2 index[3] provides such a measure. This is something that has also been treated in the MSG110 course.

Without loss of generality, in this section suppose we have a polynomial of degree $p = 1$. Under the assumption that the pairs $(x_i, y_i)_{i=1,\dots,n}$ do show an approximately linear relationship (and as such it makes sense to fit a linear model to data), then we can compute the *coefficient of determination*, or R^2, as

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} = 1 - \frac{\sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}.$$

We have that $0 \leq R^2 \leq 1$. The higher the value, the better the fit (if a linear model is appropriate), this is why R^2 is a "goodness-of-fit" index. The maximum value it can take is 1, which in practice is never achieved unless there is a perfect linear correspondence between the covariate and the response, which is rather unrealistic in practical experiments. The smaller the value of R^2 the lower the *linear association* between y and x. With the extreme case of y completely independent of x, where $R^2 = 0$.
Basically the fit improves when the *residuals* $(y_i - \hat{y}_i)$ are small, because this indicates that predictions $\hat{y}_i$ are close to the observations. See Figure 3.

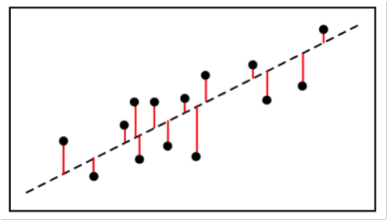


Figure 3: An illustration of linear regression: linear fit (dashed line), data (circles) and residuals (red lines).

And in Figure 4 we can see that the less linear the relationship, the lower the value of R^2. Just remember that x can be a transformation of some variable, as mentioned in the first lecture, it does not have to be the originally measured variable. For example, if we notice a parabolic relationship in the data, then we should consider a quadratic regression model (recall, this is still a linear model in the sense we introduced).

But why is that $R^2 \in [0, 1]$? The reason is that, for linear models only, the following holds:

$$\sum_{i=1}^n (y_i - \bar{y})^2 = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 + \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

so clearly $\sum_{i=1}^n (y_i - \hat{y}_i)^2 \leq \sum_{i=1}^n (y_i - \bar{y})^2$.
Furthermore, notice the following

$$\underbrace{\sum_{i=1}^n (y_i - \bar{y})^2}_{\text{total variability}} = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 + \underbrace{\sum_{i=1}^n (y_i - \hat{y}_i)^2}_{\text{sum squared residuals}} .$$

[3]Here the "R" in R^2 has nothing to do with the software.

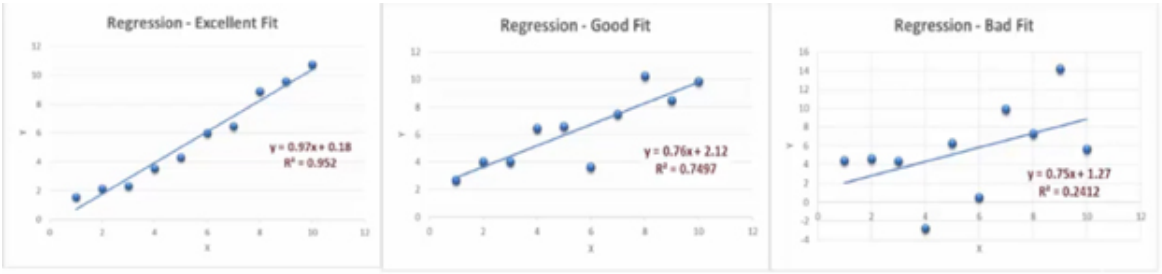


Figure 4: Three examples of linear fit.

So if we define $\sum_{i=1}^n (y_i - \bar{y})^2$ as *total variability of the response* y, then this variability has a fraction $\sum_{i=1}^n (y_i - \hat{y}_i)^2$ that is *not explained* by the regression model. What is left is the fraction of the response y variability that *is* explained by the regression model, namely $\sum_{i=1}^n (\hat{y}_i - \bar{y})^2$.

Obviously we can equivalently write

$$R^2 = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \in [0, 1],$$

and basically we wonder if the numerator can do any better than the denominator, the latter being represented by the red line in Figure 5. Hopefuly the answer is "yes" as the red line just doesn't care of x, it's simply the average $\bar{y}$, which doesn't take x into account.

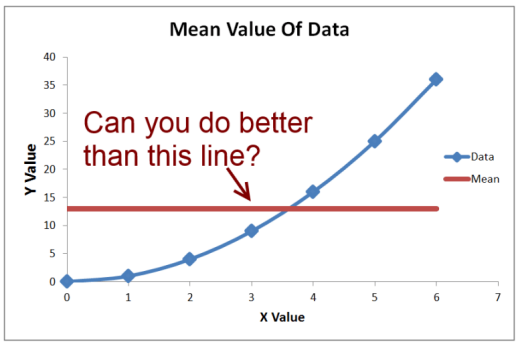


Figure 5: Red line is $\bar{y}$.

You can obtain the R^2 using `summary(mymod)$r.squared`. Or you can just look into the entire output of `summary`. For example for the cars data, we have previously run `summary(mymod)` and if you look towards the bottom of the output you find

```
...
Multiple R-squared: 0.6511
```

that is $R^2 = 0.6511$.

4 A larger model is not necessarily a better model

For the case of a single covariate x and a polynomial model of generic degree p, R^2 becomes

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} = 1 - \frac{\sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i - \cdots - \hat{\beta}_p x_i^p)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \in [0, 1].$$

The higher the value of R^2, the better the fit. However, we will see that **we should not seek a higher R^2 at all costs**.

A major problem with R^2 is that its value will always increase with p. Why is this a problem? Well, R^2 will increase no matter what, even if we plug in the model covariates that are totally irrelevant to predict y!

To predict `dist` in the cars dataset, in addition to `speed` shall we use as covariate the number of ice-creams each driver has eaten in the month previous to the driving test? Sounds silly, doesn't it?! But if you add this information, you will see the R^2 increasing by a very tiny quantity. The naive analyst would think that it is worth including ice-creams consumption just because of the R^2 increase. I know this looks like a rather extreme example, but it gives you a feeling of the importance of not just look at higher R^2 values. Let's discuss this aspect a bit more.

Here I create a fictitious example, where $n = 100$ observations are simulated from the model $y = f(x)+\epsilon$ with $f(x) = 2+50x-5x^2+0.09x^3$, where each x_i is sampled independently and uniformly as $x_i \sim U(1, 50)$. Each ϵ_i is an independent realization from the Gaussian $\epsilon_i \sim N(0, 150^2)$, i.e. $\sigma = 150$. So in this case **we know the true model** generating the data, since the observations are simulated (we never know the true model in real-data case studies). Clearly, the true model is a polynomial of order 3. The code pertaining this example is available as `demo_poly.R` on the course website.

We want to study the consequences of mindlessly increasing the degree p of a polynomial when fitting these simulated data. Notice the data are plotted in Figure 6 together with the *true* noiseless model $f(x) = 2 + 50x - 5x^2 + 0.09x^3$.

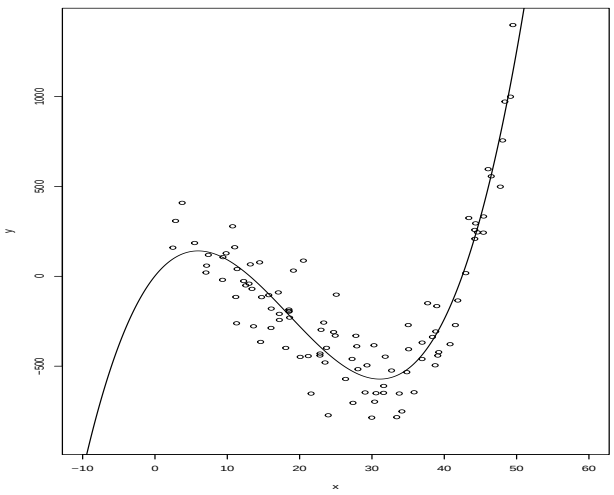


Figure 6: Simulated data together with $f(x) = 2 + 50x - 5x^2 + 0.09x^3$.

Now we try to fit the data with polynomials of order $p = 2$, 3, 5, 10 and 20. For ease of display, Figure 7 shows the cases $p = 2$ and 3, in addition to the true $f(x)$. Not unexpectedly, the cubic fit (green) is quite good, as the true generating model is also cubic. The quadratic one okish: it is a bit shifted towards the left side of the plot, thus missing some of the central observations, and it also misses the rightmost observations.

To specify polynomial regression models in R there are two ways: we start with the least recommended option.

```
# example for a cubic model
lm(y~x+I(x^2)+I(x^3))
```

The I() function is the `AsIs` function and can be used to specify powers, just as you would expect. This is because within `lm` writing `x^2` instead of `I(x^2)` would be interpreted as "add second order interactions" (something we are not going to consider). So, shortly, do not write the powers without using `I()`.

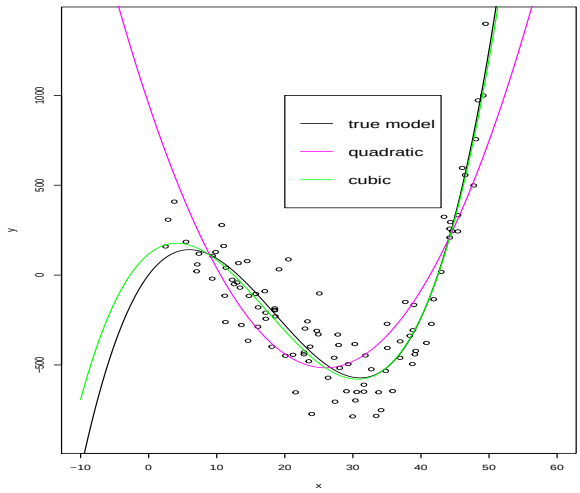


Figure 7: Simulated data, true model and polynomial fits with $p = 2$ and 3.

And here is the second option, which is recommended because it is quite compact to write when p is not small:

```
# example for a cubic model
lm(y~poly(x,3,raw=TRUE))
```

So far so good, nothing really surprising happened. But let's look at higher order polynomials. Figure 8 shows what happens with $p = 5$. Now, here is tricky to decide. Forget for a moment about the black line, i.e. assume we do not know where the truth is. How could we decide what is best, between $p = 3$ and $p = 5$? We could give a look at their R^2: for $p = 3$ we

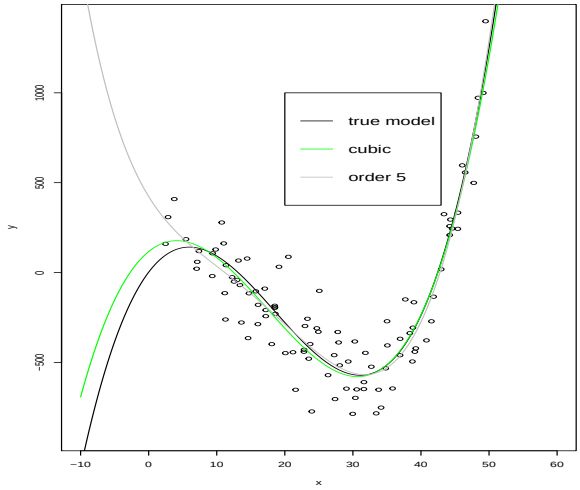


Figure 8: Simulated data, true model and polynomial fits with $p = 3$ and 5.

have $R^2 = 0.865$, which means we are able to explain about 86.5% of the observed variability. With $p = 5$ it is $R^2 = 0.872$, so we explain about 87%. Now, it seems that with $p = 5$ we get a very small increase of 0.7% in quality of the fit. Is it worth having a larger model for such a small increase? In fact, let's not forget that in reality the best model has to be the one with $p = 3$, because the data have been generated by a cubic model. As you can see, deciding on what is the "best model" is often difficult, because in reality we do not know which one is the true model. So here we should act as if we did not know this information.

If we go further and try $p = 10$ and $p = 20$ we obtain $R^2 = 0.876$ and $R^2 = 0.887$. These are relatively negligible increases compared to the previous models, if we take into account how big these models are becoming (recall we only have $n = 100$ observations to fit). However, a naive analyst may prefer $p = 20$ just because of the R^2. This would of course be a generally inappropriate strategy. For example let's give a look at the fit of these latter polynomials. As you can see in Figure 9 weird things happen. There are so many coefficients in the model, too many in fact, that give too much freedom to the model. For example look what happens for abscissas outside of the observational interval, that is for $x < 1$ and $x > 50$: there we have no data and the predictions for $p = 10$ and $p = 20$ react unexpectedly. So many parameters to explain a dynamic that requires less than five parameters gives too much freedom to the models. These models "adapt too much" to the data (look at how the model with $p = 20$ bends to try to closely follow the observations). We say that models with $p = 10$ and 20 **overfit** the data. This means that for $p = 10$ and 20 the models are too ad-hoc for the available data, and would be unable to represent further (future) data, should we have more. "Too ad-hoc" means that **they would not generalize to not-yet observed data**. In other words, it would be a disaster to use such models to predict new observations. Using the the blue line ($p = 20$) to make a prediction when $x < 0$ would be very risky. When $x < 0$ the blue model varies so much that it cannot be contained within the plot window.

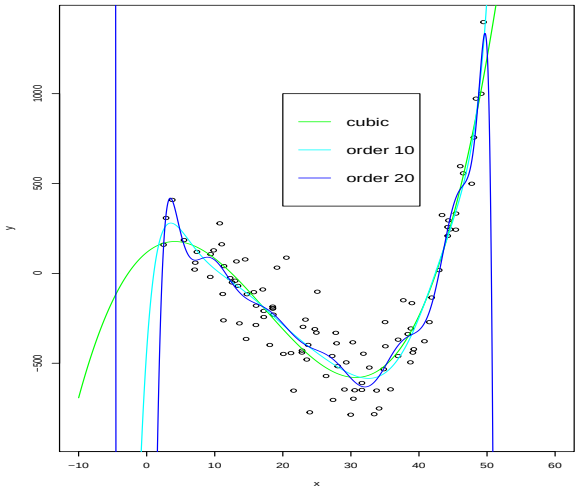


Figure 9: Simulated data, true model and polynomial fits with $p = 10$ and 20.

Instead the cubic model has some ability to "wiggle": it does not try to adapt too much to the current data, so when used to predict new observations it is sort of more robust. If you feel a bit confused, look at Figure 10. A model that underfits has too few parameters and is unable to follow the dynamics in the data: this is the case for the quadratic model ($p = 2$). A model that overfits adapts too much to the data, and might fail to generalize to fit not-yet-available data (bad for predictions). The "just right" model is able to follow the dynamics of the observation, but not too closely, so that it can better generalize to future data. The "just right" model will necessarily have an R^2 that is smaller than the model that overfits, because it gets less close to the data, but we gain in robustness.

The reason why we are interested in the ability of a model to fit also unobserved data (i.e. potential, future data), is that we want to **understand how a phenomenon works**. In full generality, that is **not limitedly to the data that we happen to have at the moment**.

In the end, how do we choose a model that is able to fit without overfit/underfit, and able to generalize to unobserved data? This is the subject of next section.

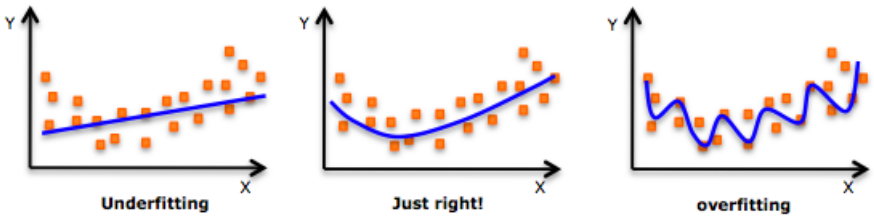


Figure 10: An exemplification of possible fits.

5 Training data and testing data

In this section we reuse the simulated data we created in the previous section. However, as an assignment you will work with real data for increasing fun.

We discussed the importance for a model of being able to predict *unseen observations*. Hypothetically, if someone could give us m pairs of previously **unseen** observations, that is **new observations** that are unused when fitting the model

$$\begin{bmatrix} y_1^{new} & x_1^{new} \\ y_2^{new} & x_2^{new} \\ \vdots & \vdots \\ y_m^{new} & x_m^{new} \end{bmatrix}$$

then a strategy to evaluate the predictive power of a polynomial having degree p would be:

1. obtain the $\hat{\beta}_0, ..., \hat{\beta}_p$ by fitting the usual n pairs of **seen** (previously available) observations $\{y_i, x_i\}_{i=1}^n$, then
2. use the $\hat{\beta}_0, ..., \hat{\beta}_p$ and the "unseen covariates" $(x_1^{new}, ..., x_m^{new})$ to predict m responses using

$$\hat{y}_i(p) = \hat{\beta}_0 + \hat{\beta}_1 x_{i1}^{new} + \hat{\beta}_2 (x_{i2}^{new})^2 + \cdots + \hat{\beta}_p (x_{ip}^{new})^p, \qquad i = 1, ..., m \tag{2}$$

   then
3. evaluate the quality of these predictions on the unseen y_i^{new} using

$$pMSE(p) = \frac{1}{m} \sum_{i=1}^{m} (y_i^{new} - \hat{y}_i(p))^2.$$

The latter is the *prediction mean-squared-error* (pMSE) for a model having degree p. We declare as "best model" the one having a degree that produces the smallest $pMSE(p)$. Which means that we have to create an algorithm that computes $pMSE(p)$ for a desired range of possible $p = 1, 2, 3, ...$

Because of how the pMSE index has been constructed, it effectively tests the predictive ability of a model. In fact we check our predictions $\hat{y}_i(p)$ against measurements y_i^{new} where the latter *have not contributed to the estimation of the* $\hat{\beta}_j$, thus providing a sanity check for our model. If we were doing the sanity check using the same observations that have been employed to obtain the parameter estimates, well...that would not be a sanity check. It would be wrong, because we know the $\hat{\beta}_0, \cdots, \beta_p$ should fit about ok the available data, but we want to know if they fit well also the new data!

Ok all the above sounds interesting. But how to do this in practice? How to obtain new/unseen data, if all we got is the dataset at hand?

Here is how the concepts of **training and testing datasets** come into place. We basically split the data we have into two subsets. To keep some consistency of notation with what we wrote above, say that our dataset has N observations (up to now we used the letter n, just think that this is now N):

- the first subset is made up of n observations *randomly extracted* from the N we have, *without replacement*. This constitutes the **training data**.
- The remaining $N - n$ observations are the **testing data**.

Example of how to produce sampling without replacement. Suppose you have a matrix D having $N = 100$ rows and ten columns. We want to sample $n = 60$ integers uniformly from $\{1, 2, ..., N\}$. The samples represent indeces of rows of matrix D.

```
# create a matrix D with dimensions 100 x 10 having values from a standard Gaussian
D <- matrix(rnorm(1000), ncol=10)  # creates 1000 draws, then distribute them across 10 columns.
# So D has 100 rows and 10 columns.
indeces <- sample.int(100,60)  # sample 60 integers from the set {1,2,...,100}
```

The above gives a hint of how to proceed to create the training and testing datasets.

```
training <- D[indeces,]   # the training data
testing  <- D[-indeces,]  # the testing data
```

Notice, using `sample.int` we have randomly extracted 60 integers in $\{1, 2, ..., 100\}$ without replacement, and stored those into `indeces`. Then since we want those integers to represent rows of D, we plug them into `D[indeces,]`. This way we select all columns of D and all of its rows having index in `indeces`. If D was our dataset, then `D[indeces,]` would constitute our training data, a matrix of 60×10 entries. Then testing data are the remaining rows, which we can select using `-indeces`. The `-` (minus symbol) in front of `indeces` means "consider all rows, except the ones in `indeces`". Therefore `D[-indeces]` constitute testing data, a matrix of 40×10 entries.

We can now use these concepts for model selection by looping across the p degrees of a polynomial, and **for each degree we fit the training data, obtain parameter estimates based on training data, predict responses using the testing covariates and compare predictions with the testing responses**, as outlined in steps (1)-(2)-(3) at the beginning of this section.

For extra clarity, algorithm 1 shows how we act using a `for` loop for the hypothetical case where we want to screen polynomials up to order 5.

5.1 Back to the simulated data

We now look at the artificial data considered in section 4. There we had $N = 100$ observations. Here we consider training data of size $n = N/2$. We check polynomials from order $p = 2$ up to 10 by running algorithm 1 and obtain Figure 11. The smallest pMSE value corresponds to $p = 4$ where $pMSE(4) = 19184.95$ (notice, if you write your own code you may obtain different results due to different randomly sampled training/testing data). So we didn't obtain that $p = 3$ (the true model) has the smallest pMSE! Why is that? Unfortunately there are several elements that come into play:

- is N too small to be able to identify the correct model?
- Or, is n not large enough to obtain reliable parameter estimates, compared to the size of N?

Algorithm 1 computation of pMSE on training and testing data

For a given dataset, extract randomly a training dataset D^{train} of size n, so that the testing dataset D^{test} has size $N - n$.
Call x^{train} and y^{train} respectively the x and y values from D^{train}, then call x^{test} and y^{test} the x and y values from D^{test}
for $p = 1$ to 5 **do** `# suppose we screen up to order 5`
Fit y^{train} using a polynomial of order p based on x^{train}, and obtain parameter estimates;
Make predictions based on x^{test} covariates, as in equation (2), that is:

$$\hat{y}_i(p) = \hat{\beta}_0 + \hat{\beta}_1 x^{test} + \cdots + \hat{\beta}_p (x^{test})^p$$

Compute and store $pMSE(p)$:

$$pMSE(p) = \frac{1}{N-n} \sum_{i=1}^{N-n} (y_i^{test} - \hat{y}_i(p))^2.$$

end for
Output $pMSE(1), ..., pMSE(5)$

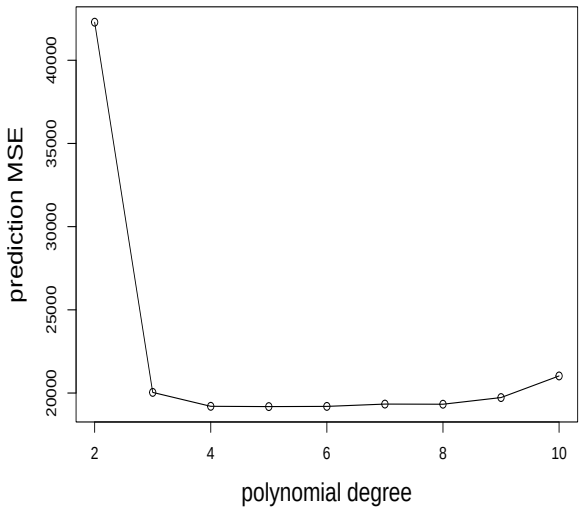


Figure 11: pMSE vs p when 50% of the data is used for the training set and the total size of data is $N = 100$.

- does it depend on the specific random elements extracted to construct training and testing data?
- are the data very noisy (large σ), so that the true "signal" $f(x)$ is difficult to capture?

Here we address the second point: perhaps having $n = N/2$ does not allow polynomials to be fit on enough data. In fact, it is typically best to choose $n > N/2$ (usually 70% or 80% of the total data size N) so to learn the fitted parameters better (because these are fit on more observations). We now split data so that 80% is allocated for the training data and only 20% for testing, and obtain Figure 12. Also here $p = 3$ is not the best.

What is the real issue here? **It seems that the size of the original dataset, $N = 100$, is not big enough to let us identify the correct model.** Several models are plausible enough. If you try from to simulate $N = 500$ data points anew, just as we did in section 4, you will consistently select $p = 3$ as best model, both for the case when n is 50% and 80% of N.

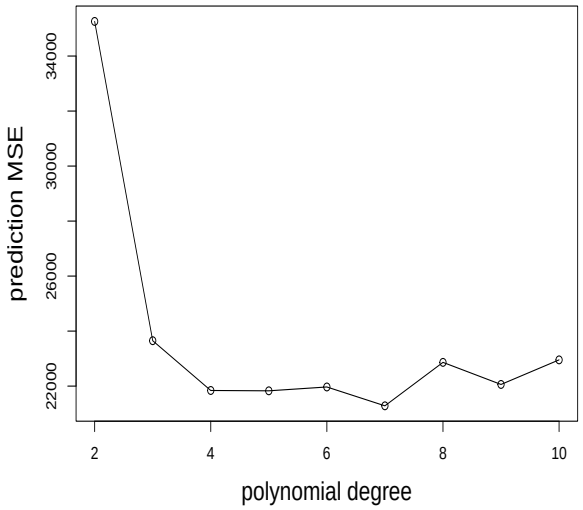


Figure 12: pMSE vs p when 80% of the data is used for the training set and the total size of data is $N = 100$.

How to connect circles with lines similarly to figures 11-12? You first plot the circles, and then you add the lines as in

```
plot(c(1,2,3),c(7,11,13))  # first do this
lines(c(1,2,3),c(7,11,13)) # and THEN connect everything (the other way around won't work)
```

Two final considerations: unlike R^2, pMSE does not necessarily increase with p. In fact pMSE should "punish" unnecessarily large models, which is good. Also, when values of pMSE are very similar, as in Figure 12 for $p > 2$, don't just pick the model with the smallest pMSE, instead apply Occam's razor, and among models with a similar pMSE prefer the one with the smallest number of parameters.

Then: by "how much" our favourite/picked model is better (or slightly worse) than others at predicting? The pMSE is defined on the square of the scale of the response variable y, which is unintuitive to interpret. However $\sqrt{pMSE}$ is defined on the same scale as the response. So considering plots of $\sqrt{pMSE}$ can sometimes tell you something more. Example: if y represents expended calories after 1 hr of physical exercise, then if say $\sqrt{pMSE(3)} = 30$ and $\sqrt{pMSE(4)} = 10$ it means that the prediction error we get with $p = 4$ is reduced by 20 units (calories) compared to p=3. Whether a 20 calories difference is "very little" or "considerable" is up to you and the specific research question. In any case, reasoning in terms of squared calories (ie comparing $pMSE(3) = 900$ with $pMSE(4) = 100$) it would not be as intuitive.

6 Variable transformations

In the previous lecture notes we mentioned that we want residuals $e_i = y_i - \hat{y}_i$ to be about symmetric around 0. There has been research around transforming the response variable Y in such a way that residuals are approximately symmetric, and having them randomly scattered around 0 implies that the fitted line nicely goes through the data, ie the regression model is appropriate.

A first attempt was the so-called Box-Cox transformation (by George Box and David Cox), where they consider a variable $y > 0$ and attempt to transform it by introducing a parameter

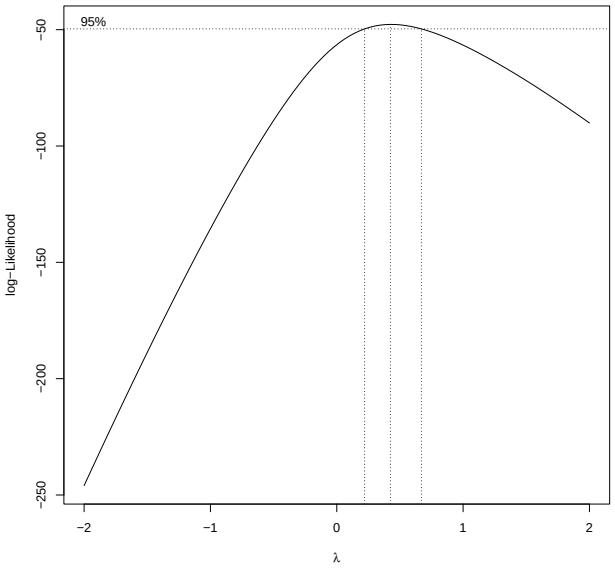


Figure 13: Loglikelihood vs λ

$\lambda > 0$ that needs to be estimated somehow. So they have the following transformation

$$y(\lambda) = \begin{cases} \frac{y^\lambda - 1}{\lambda}, & \text{if } \lambda \neq 0 \\ \log(y), & \text{if } \lambda = 0, \end{cases}$$

and in a regression context, $y(\lambda)$ would become the new response variable. Studying the construction of this transformation is not central for this course[4]. In one sentence, λ gets estimated from data (together with the β's) using a maximum likelihood approach. Now, the newly obtained $y(\lambda)$ when plotted against x may (or may not!) resemble a more linear relationship and result in residuals more evenly scattered around the zero. I added "may not" as the transformation will not help in also looking into transformations of the covariate(s). So the Box-Cox method will not do anything regarding possibly searching for transformations of x, if such a transformation is also needed. It is up to the researcher to see if x also need to be transformed, and otherwise it may be that transforming only y will not produce any improvement.

You may use

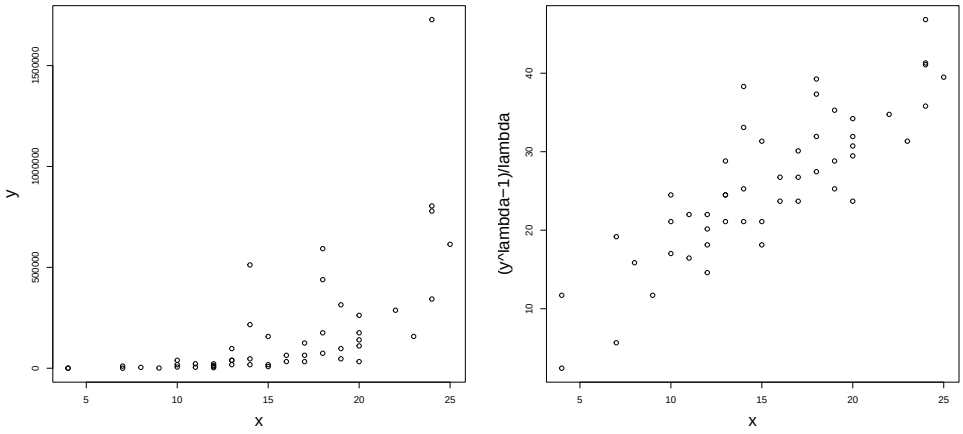
```
library(MASS)
bc <- boxcox(mymodel)  # mymodel is defined via lm()
```

where the above would automatically produce a plot like Figure 13. There, you notice that the loglikelihood peaks when λ is around 0.5. To find the λ value that maximizes the loglikelihood (and hence maximizes the likelihood) you can do the following

```
bc    # you may type it to see what it contains.
      # bc$x contains lambda values, bc$y it's loglikelihood values
id_max <- which(bc$y==max(bc$y))  # find the index of the best lambda
lambda <- bc$x[id_max] # fetch the best lambda value
```

which would return the maximizer in Figure 13, namely $\hat{\lambda}$.

[4]Optional: the interested student can give a look at `https://www.ime.usp.br/~abe/lista/pdfm9cJKUmFZp.pdf`



(a) Original variables (b) After box-cox transformation

Figure 14: An example where Box-Cox is used.

Importantly, the two vertical dotted lines represent a 95% confidence interval for λ, and in conclusion **if the confidence interval includes the zero, then you log-transform the original** y and in `lm` you will fit $E(\log(y)) = \beta_0 + \beta_1 x$. **If the confidence interval for** λ **does not include the zero, your new response variable will be** $y(\lambda) = (y^{\hat{\lambda}} - 1)/\hat{\lambda}$ and in `lm` you will fit $E(y(\lambda)) = \beta_0 + \beta_1 x$. In Figure 13 the 95% confidence interval does not include the zero, and therefore we consider as response to fit $(y^{\hat{\lambda}} - 1)/\hat{\lambda}$. Therefore you could do the following:

```
plot(x,(y^lambda-1)/lambda)   # to check how transformed data look
# and then fit
m <- lm((y^lambda-1)/lambda) ~ x)
```

An example of the effect of applying the Box-Cox transformation is in Figure 14.

Warning: Box-Cox transformations cannot do magic, especially if the x variable should also be transformed! If x should enter as, say, $z = e^x$ instead, then `boxcox(y~x)` won't help, but `boxcox(y~z)` would. Moreoever, once the transformation is applied, the practical interpretation of the estimated β's becomes unclear and unintuitive. This is a price to pay unfortunately.

Exercise 1 (=1.5 point): you find this on page 3.

Exercise 2 (=4.5 points): `sleeptab` data again.

(i) (2 points) Before fitting the models, remove from the dataset the three extreme observations you identified in exercise 1 from assignment A1 (tip: there are several ways to do exclude some data, and you can do it as you prefer, for example you do not necessarily need to *edit* the original dataset, see for inspiration `method 2` in line 58 of `demo_cars.R`).

Consider polynomial regression for non-transformed data, that is response `brwt` and covariates given by powers of `bwt`, from order $p = 1$ and up to order 5. Compute the pMSE corresponding to each polynomial (code everything using a `for` loop where p ranges from 1 to 5), where the training data has size `n=floor(0.8*N)` (this is a function rounding to the closest integer from below), then plot the resulting $\sqrt{pMSE}$ vs p. *[**Note**: just because it will be easier to grade, please place `set.seed(321)` before sampling training and testing data. So we get the same results.]*

(ii) (0.5 points) Following up on (i), we wish to identify the "best model" in terms of $\sqrt{pMSE}$, while remembering to take into account the Occam's principle. Give your reasoning (ie motivate) regarding which one you consider to be a sufficiently good model.

(iii) (0.5 points) Plots like those in (i) may change when randomly sampling different units when constructing training and testing data. Therefore try the following (again using the subset of data without the three extreme observations). Put the whole procedure you coded in (i) inside another `for` loop, where you execute model selection for 500 times, each time sampling different training and testing data. Collect into a matrix all pMSE values for each degree and for each of the 500 runs. For each value of p plot the median of the $\sqrt{pMSE}$ values across the 500 runs, then for each p add on the same plot the lines connecting the 0.1 and 0.9 quantiles *(if you cannot see the lines, probably you have to change the size of the y-axis. You can place the option `ylim=c(a,b)` inside `plot()`, where `a` and `b` are suitable axis limits).* By looking at the plot, does your conclusion change about the best model? Discuss.

*[**Note**: again for consistency, please place `set.seed(321)` just before the most external `for` loop.]*
Tip: if `pmse` is the 500×5 matrix of pMSEs, in order to add the required lines you can use
`lines(seq(1,5),apply(pmse, 2, quantile, alpha))`
where `alpha` is the quantile level.

(iv) (1.5 points) **Here reconsider the full data of 62 observations** (reload the data from the given sleepdata.dat file to make sure). (a) Find the optimal λ for the Box-Cox transformation applied to the model with response *brwt* and covariate *bwt*, report the corresponding plot of the loglikelihood vs λ, and then show the plot of the transformed response vs *bwt*. This is supposed to look quite unsatisfactory. (b) However we know from previous exercises that log(*bwt*) could be a better covariate, as the bwt spans different orders of magnitudes. Do the same as in (a) starting from a model with the untransformed brwt but this time with covariate log(*bwt*). Did you get something familiar? (c) Now let's try something different: we want to use lifespan (or a transformation thereof) as covariate and brwt (or a trasnformation thereof) as response. Produce a sensible model by building on what you learned in (a)-(b).