Take-home exam

Indrek Seppo

2021-01-12

Hi, this is the take-home exam. It will take time (I would expect a whole day, but I might be wrong), but if you are serious about about data analysis or econometrics, it will be worth it. Do not be intimidated by the length of this document – most of it are just hints for you.

This is an **individual task**. I fully expect you to consult your fellow students when you run into trouble, but **I do not want to see copy-pasted versions** of the work. There are a zillion ways to do this, so believe me I will notice when you are sending the same version from multiple people. Just take your time, this should be absolutely doable for each and everyone at your level.

Please document the steps you do in a **R Markdown** or **R Notebook** document and upload the final version (docx, pdf or html) or the .Rnw file.

Simulation study – attenuation bias

Whenever you see a new method you have two ways of undestanding it a) go through the maths b) just create your own world, simulate the data and take a look how well the method copes. I am terrible in matrix algebra, so I usually simulate stuff. Let's use this method for understanding what is a thing called **attenuation bias**.

Attenuation bias is something that is usually ignored, but you are expected to know it if you are going deeper into statistical analysis, and in some cases it might add to your understanding of the problem.

In short – the regression model expects the variables on the right hand side of the equation mark (the independent variables) to be measured precisely, without measurement error (this is not the case for the variable on the left hand side – the one you are "predicting", this can be measured with error without biasing your estimates). If there is a measurement error in the right hand side variables, then the parameter you are estimating (the β_x in regression) will be systematically downwards biased – meaning that on average your estimate will be lower than the real world value you are trying to estimate. You expect your point estimate to be somewhat different than the real parameter anyway – this is why you have the confidence intervals, but you expect it to be on average the same as the real parameter (when 1000 people run the same test or you run it 1000 times then you expect the average of the results to be rather close to the real value), so a systematic bias is a bad thing.

We will now do a quick simulation study: we will generate a new world with known parameters and take a look how good are our methods in recovering these parameters. For this we will take a lot of samples from this world and try to estimate the parameters from these samples a lot of times to see how good

or bad our estimates usually are.

Lets go through an example – imagine that you know that there are positive returns to mathematical abilities in the labour market. Meaning that the better someone understand mathematics, the higher their wage should be[1]. But you cannot measure these mathematical abilities precisely, you measure them with an error. What we are going to show next is that if we measure mathematical abilities with an error, our regression coefficients will on average undervalue the effect of mathematical abilities on wages.

[1] The reality is probably somewhat more complicated. While students grades in mathematics in the highschool are usually one of the few grades that have some predictive power for future earnings, it might not be that studying mathematics helps, it might just be that people with high level of general intelligence tent to have good grade in mathematics and are also able to get higher wages.

Imagine you have data about some people's mathematical abilities and wages and you are trying to estimate how much would it help if you could raise someones mathematical score by 10 points. Let's create a very simple world where the wage of a person is linearly dependent on the persons maths score and then some random noise ε:

$$wage = \beta_0 + \beta_1 * mathsability + \varepsilon$$

As we are creating this world, we can choose the β_0 and β_1 ourselves. So lets assume the relationship is this:

$$wage = 500 + 50 * mathsability + \varepsilon$$

So someone whose mathsability is 0 would probably earn 500 (+/- some random component), and we would expect someone whose mathsability is 10 to earn $500 + 50 * 10 = 1000$.

Now imagine that we can only measure the mathsability with some error – sometimes we overestimate it and sometimes we underestimate it. Lets assume our error in this measurement is normally distributed, has mean 0 (so on average we are correct) and standard deviation of 1 (so 2/3 of the times our measurement is less than 1 points off, 95% of the times our measurement is less than 2 point off).

We are now going to show that if we are using this observed maths score as an independent variable, we will underestimate the effect of mathsability on wages.

Your turn (detailed hints come after the exercise), answer to every question gives you 10 points of the total of 100:

1) Create a data frame with the first column being id of a person (from 1 to 1000)[2] and the second row to be maths score. Assume the math score is normally distributed in our population with mean 15 and standard deviation of 3. To do so, you will need to know a bit about how to generate random data in R. **NB, You will find some detailed hints after the exercise, take a look at them now!**

[2] This is actually redundant – we do not use the id anywhere, but I included the id here anyway for the clarity.

```
##   id mathsability
## 1  1     15.26917
## 2  2     16.97151
## 3  3     15.59475
## 4  4     12.62194
## 5  5     14.45025
## 6  6     12.77786
```

2) Visualise the resulting distribution of the maths score using `geom_histogram` to check how it turned out.

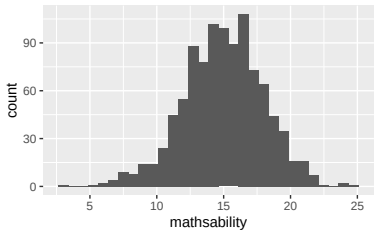


Figure 1: This is what I got.

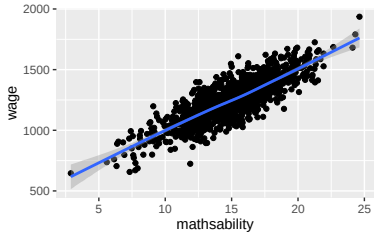
3) Create a new variable `otherfactors` into the dataframe – this would be a cumulative effect of all the other factors. If there are many small factors, which are independent from each other then the sum of them will have an approximately normal distribution, so lets create this as a random variable from normal distribution with mean 0 and standard deviation of 100. Think about it for a moment – we are saying that other factors (like how good the person is in geography or what is her favourite movie) are adding something with mean 0 and standard deviation of 100 to the wages. With normal distribution around 2/3 of the values are inside +/- 1 standard deviations from the mean, so what we are saying is that the other variables (besides the mathsability) will affect the wage by less than 100€ in around 2/3 of the cases and by less than 200€ in around 95% of the cases[3].

[3] This would be a peculiar world – in reality the maths ability definitely helps, but the other variables are much more important.

4) Now we can create the wages for everyone. It depends on `mathsability` and `otherfactors`, so create the new variable into the data frame so that:

$$wage = 500 + 50 * mathsability + other factors$$

5) Visualize the data, adding all the data points and a smoother (`geom_smooth()` – you should have enough data that there will be no overfitting, so even regular `geom_smooth()` should give a rather linear line.)



6) Now assume that we are not measuring the mathsability precisely. Lets call the observed ability `obsscore`, and assume that on our measurement error is normally distributed with mean 0 and standard deviation of 1. Create a variable for measurement error (call it however you like) and then a variable named `obsscore` (this is the score that we can observe, the observed mathematical ability) which is `mathsability + measurement error`[4].

[4] Think about it for a moment as well – we are saying that the measurement error is a normally distributed variable with standard deviation of 1. This means again that in around 2/3 of the cases our mistake in measuring the maths ability is less than 1 points (whatever the points mean here), and in 95% of the cases it is less than 2 points.

You will get a different result thank to randomness, but the head of my data frame looks like this:

id	mathsability	otherfactors	wage	measurementerror	obsscore
1	15.26917	7.89870	1271.357	-0.2393073	15.02986
2	16.97151	96.30441	1444.880	-0.1622497	16.80926
3	15.59475	-98.79047	1180.947	-3.3659748	12.22877

7) Run two regressions:

a) regression predicting `wage` using `mathsability`
b) regression predicting `wage` using `obsscore`

extract the point estimate of the parameter `mathsability` from the first regression and of `obsability` from the second regression. If you want to do the bonus exercise #11 extract the standard errors for this parameter as well. How? You'll see in the hints below.

I got the point estimate for the first regression to be 50.6, with a standard error of 1.1 (remember, your correct value is probably between point estimate +/- 1.96*standard error)[5]

8) You have now done it once and got a result. What if we do it 1000 times? Would we on average get the right result? How much would our results differ from the correct result (we know the correct result here, as we have created our own world). Repeat all the steps aside from creating graphs for 1000 times, saving the point estimates of the parameters to a data frame. Again, I will give you hints of how to do it below.

9) What is the mean point estimate of the mathsability? What is the mean point estimate of the obsability? You know that one additional point in mathematical ability should be associated with 50 more euros – you created this world like this. I got 50.05 for ability (this is the perfectly measured property – you will get something similar, rather close to the real one) and 45.06 for score (this is the measurement with measurement error, again you will get something similar, definitely less than 50).

10) Visualise the distribution of these point estimates that you got using either `geom_density()` or `geom_histogram()` (it does not have to look exactly like what I have done here – it can be easier, as long as it shows the difference).

This is what I got – you should have slightly different figure, but I would bet that the result is the same – in vast majority of the cases you would end up with too low estimate for the effect of maths ability when you are using a noisy measurement of it (remember, our measurement error had zero mean, meaning that sometimes we overestimate it, sometimes underestimate it, but we do not consistently over- or underestimate it, on average we estimate it correctly). On the other hand, remember that our measurement error was pretty

[5] Actually – in 95% of the cases your confidence intervals will contain the correct value. There are a lot of smart people out there who would tell you that it is incorrect to say that there is 95% probability that the correct value is inside the confidence intervals (telling you that the correct value is not a random variable, so it does not have a probability distribution). While they are right, don't worry about it – if you run this 100 times, then (assuming the model is correct, what it never is) in 95% of the times the confidence intervals will contain the correct value, and in the everyday language one would in this case say that there is 95% probability that the correct value is inside the condifence intervals. If the last sentences made any sense to you I recommend you to go and read

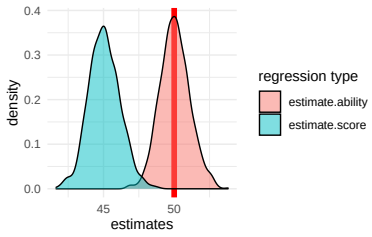


Figure 2: Densities of the point estimates (correct value indicated by red line)

large – in most of the real world cases even if you have a measurement error it is probably not so big.

Attenuation bias is **not important** when you are only trying to predict the outcome from the inputs. If you have the observed maths score of someone, then the best prediction for her wage is exactly what you get from the regression, no matter if this score is measured with error or not (it is measured with error next time as well, and this is the best prediction for the variable that is measured with error). It is important when you are trying to understand how much would the wage increase if you are able to raise the maths ability by one unit. You will not see corrections or discussions of this in most of economics papers, but you can expect a question about this in conferences when you are clearly having some measurement problems. You would usually answer, that yes, of course I am underesimating the effect a bit, in reality it is probably even bigger.

11) (bonus exercise, 5 points bonus) I don't want to push it – it has been a long exercise all along, but if you dare then calculate how often was the real mathscore (50) inside the confidence intervals for either of the regressions? You can create confidence intervals by adding and substracting 1.94 standard errors from the point estimate.

What I get, is that the true value is inside the confidence intervals in 1000 runs in 94.8% of the cases – this is close to 95% which I expected (as this is how the confidence intervals are defined – so that if we had a quazillion of runs, the true value would be inside them in 95% of the cases). And only 0.7% of cases if we use the score as a proxy for mathematical ability – this is expected as well, we know that these estimates are systematically biased.

12) The Holidays bonus (priceless)

I don't know if you have read The Martian from Andy Weir. Do not start it during the exam session, you will not put it down before the final page (and it is a rather different experience compared to the movie, if you have seen it). But the guy has a nice short story online, called The Egg: http://www.galactanet.com/oneoff/theegg_mod.html. I believe this will be the best short story you will read today.

Hints

1) Create a data frame with the first column being id of a person (from 1 to 1000) and the second row to be maths score. This math score should come from a normal distribution with mean 15 and standard deviation of 3.

You need to recall how data frames were created:

```
mydataframe <- data.frame(variable1=c(1,2,3), variable2=c(3:5))
```

Notice how I use `c(3:5)` here – this creates all the integers from 3 to 5. It will be a bit easier than to write all the integers from 1 to 1000 by hand.

You can add new variables to a dataframe just by saying:

```
mydataframe$newvariable <- somevector
```

You need to know how to create random variables – to draw some variables from a distribution.

R has many kinds of distributions built in. So imagine you want to flip a coin, but you do not have a coin available. This would have previously ruined your day, but luckily you now know R. You just have to know that this is a binomial distribution – a coin can either have a value of 1 or 0, and you are done:

```
rbinom(n=1, size=1, prob=0.5)
```

```
## [1] 0
```

Lets look at the help file:

```
?rbinom
```

`r` in front of `binom` in `rbinom()` means that you are asking R to generate a single random value from a binomial distribution, size=1 means that you only toss the coin once, and prob=0.5 means that there is 50% of chance for 0 and 50% of chance for 1. So if you want 10 coin flips:

```
rbinom(n=10, size=1,prob=0.5)
```

```
##  [1] 0 0 1 1 0 0 0 1 0 1
```

Run it again:

```
rbinom(n=10, size=1, prob=0.5)
```

```
##  [1] 0 1 1 0 1 1 1 0 1 1
```

You most probably had different values than the last time. This is because the values are randomly generated. Now, R is all about reproducible research. So what if you want random numbers, but so that they will stay the same next time? You have to set a seed, this will always generate the same values (I have set the seed to 1 but you can have any other number there):

```
set.seed(1)
rbinom(n=10, size=1, prob=0.5)
```

```
##  [1] 0 0 1 1 0 1 1 1 1 0
```

NB! Be cautious of where you set the seed. I would recommend to put it somewhere in the beginning of the code and only set it once here (definitely do not set it inside the for-cycle you have later - every single run will then have the same "random" numbers). The following code will now always have the same results, but if you change the order of the code or add something, the results will change.

There are other distributions. If you want to take a random pick from a normal distribution with mean=100 and standard deviation of 10, you would write:

```
rnorm(n=1, mean=100, sd=10)
```

```
## [1] 91.79532
```

What if you want 1000 of them? Just set the parameter `n` to be 1000.

With this you should be able to complete the first part of the exercise.

2) Visualise the resulting distribution of the maths score using `geom_histogram` to check how it turned out.

Recall how a typical ggplot plot was created:

```
ggplot(data=somedata, aes(x=somevariable, y=othervariable))+
  geom_something()+
  geom_otherthing()
```

You have just created a dataset in the previous point and you will need to use `geom_histogram()`. This only requires one aesthetics to be set: `x`. You want this `x` to be the mathematical ability you just created into this dataset with the last command.

3) Create a new variable `otherfactors` – this would be a cumulative effect of all the other factors. Create this drawing random values from normal distribution with mean 0 and standard deviation of 100.

Take a look at the first hint and you should be able to create it. By now you should have something like (the values will be different, as you a drawing them from random number generator):

id	mathsability	otherfactors
1	7.829026	44.58671
2	17.856055	141.99177
3	20.891792	-166.11902

4) Now we can create the wages for everyone. It depends on `mathsability` and `otherfactors`, so create a new variable `wage` into the data frame so that

$$wage = 500 + 50 * mathsability + otherfactors$$

You should be able to do it. Just remember to address the variables correctly: `dataframe$variablename`. And remember that in R you have to use the multiplication operator `*`.

5)

Look at the hint #2

6)

You just have to create another random variable – look at hint #1

7) Regressions and extracting values from regression objects

If you want to evaluate a regression like this:

$$height = \beta_0 + \beta_1 * weight + \beta_2 * sex + \varepsilon$$

You would do it like this in R:

```
regr1 <- lm(data=mydataframe, height ~ weight+sex)
```

To look at the results:

```
summary(regr1)
```

There are multiple ways to access the parameters of the regression model. I would recommend using a function called `tidy()` from a package named `broom`:

```
library(broom)
regrtable <- tidy(regr1)
```

And now you can access the parameter of interest. For example:

```
beta <- regrtable$estimate[2]
std.err <- regrtable$std.error[2]
```

You do not need it here, but in case you are working with multiple models and do not want to keep track of the order of the terms, you could do something like:

```
beta.ability <- regrtable$estimate[regrtable$term=="mathsability"]
```

8) How to do it 1000 times?

First create a data frame where you will store the results. Lets call it `results`:

```
results <- data.frame(estimate.ability=rep(NA, 1000), std.err.ability= rep(NA, 1000),
                      estimate.score=rep(NA, 1000), std.err.score=rep(NA, 1000))
```

This created a dataframe with 1000 rows and four columns. All the values are set to `NA`.

Now you need to set up a for cycle. In R you would do it like this:

```
for (i in c(1:1000)){
  cat(i)
}
```

```
## 12345678910111213141516171819202122232425262728293031323334353637383940414243444546474849505152535455
```

Instead of `cat(i)`, which will just print the value of the current `i` to the console, you need to copy what you have done before (apart from graphs) – you first create a new data.frame, generating new values for maths ability, other factors, finding wage, generating new measurement errors, finding the observed score. NB! Do not write set.seed() inside the for cycle, you would get exactly the same answer for every row!

Then you run the regression and extract the values for regression coefficients and standard errors. After which you add them to the dataframe you had created previously (assuming `beta.ability` is the coefficient you have extracted):

```
results$estimate.ability[i] <- beta.ability
```

If you now run the for-cycle you should be done.

9)

Just take the average of the estimate.ability and estimate.score from the dataframe you created.

10)

You can just add two layers of `geom_histogram()`, one for each variable (remember to set the alpha of the last one to be something like 0.5, to be able to see through it), or you can do as I did – use `pivot_longer()` from package `tidyr` to create a name-value pair, so that key would be the name of the estimate and the value would be the value, then add `geom_density()` grouping it with fill parameter which would be the key-variable (containing the name of the estimate).

11) The bonus question (5 points bonus)

You would need to add some more variables in the `results` dataframe – I'll call them `low.estimate.ability` for the lower bound of confidence interval of `estimate.ability`, `high.estimate.ability` for the higher bound and the same for `estimate.score` variable. To get the lower bound you would need to subtract 1.96 standard errors from the point estimate. To get the higher bound you would need to add 1.96 standard errors to the point estimate. Then create two new variables in the dataset using a logical condition that 50 is higher than the lower bound and lower than the higher bound. The resulting vector will be a vector of TRUE's and FALSE's, but you can find an average of this with `mean()` – and this is the percentage of the cases when the true value was inside the confidence intervals.

12) The other bonus question (priceless)

Just enjoy :) You might even find the story in your native language there.

Upload the resulting file (doc or pdf or html) to Moodle. If you run into any trouble, feel free to contact any of the teaching assistants or me (you might get answers more quickly from the teaching assistants though).

Thank you for taking the course!

Indrek, Sheriff, Pearl, Leyla and Danyl