Regression estimation

Peter Lugtig

10 oktober, 2021

For the next exercise, we will need the following libraries

```
require(sampling)
require(survey)
require(dplyr)
require(ggplot2)
```

Introduction

For the purpose of this exercise, we are again using the example we also used for ratio estimation. At Utrecht University, there are about 1000 machines on campus, some of them used more often than others. We want to estimate how much coffee each machine makes on average. Instead of using a stratied and/or cluster design, a previous exercise showed that ratio estimation can really improve on those designs in terms of improving precision.

A disadvantage in the coffee example was however that the ratio of energy use/number of coffees produced per machine was not completely constant: when a machine produced close to 0 coffees, there was still some energy use, leading to a small bias in the ratio estimator. In this exercise, we will try on improve on the ratio estimator, and introduce model-based estimation as a completely new way to do inference. Lets first generate the cofffee population dataset again.

```
set.seed(11)
coffees <- round(rpois(1000,350)+rnorm(100,0,sd=150))
coffees[coffees<0] <- 0
# and energy use
energy <- 0.072*coffees+rnorm(n=1000,mean=0,sd=1)
energy[energy<0] <- 0
coffeedata <- as.data.frame(cbind(coffees,energy))
names(coffeedata) <- c("n","energy")
```

And we will draw a SRS sample of size=100 for the purpose of later comparing our results to. We will again focus on estimating the total number of coffees drunk at the UU.

```
set.seed(11)
coffeedata$srs <- srswor(100,nrow(coffeedata))
coffee <- subset(coffeedata, srs==1)

# specify population size and survey design object
coffee$fpc <- 1000
srsdesign <-svydesign(ids=~1,fpc=~fpc,data=coffee)
```

```
svymean(~n, design=srsdesign) # se =15415
# or the confidence interval
confint(svymean(~n, design=srsdesign)) #[312237;372663]
```

We will also use the ratio estimator we designed in the previous exercise using the srs sample.

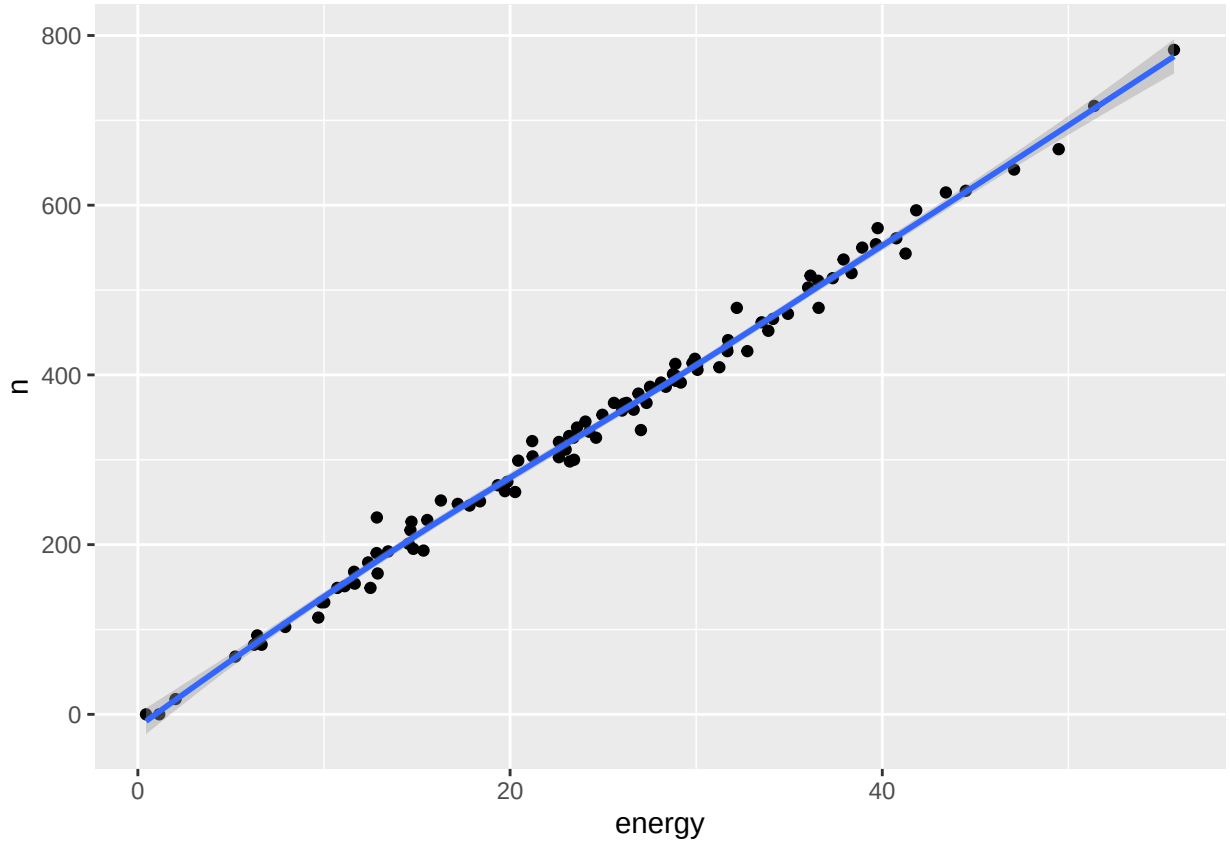
```
# create a new dataset, without n as dependent variable
coffeedata2 <-coffeedata
# I am deleting the true coffee here for the machines not in the survey
coffeedata2$n[coffeedata2$srs!=1] <- NA

# estimate the ratio
ratio <- svyratio(~n,~energy, design=srsdesign)
# what is this ratio?
print(ratio) # with 1 KwH 13.81 coffees are made on average

#below, we take the sum(energy-values * ratio estimator)
predict(ratio, total=mean(coffeedata2$energy))

# and store the results as an object
meancoffeeratio <- unlist(predict(ratio, total=mean(coffeedata2$n)))[[1]]
meancoffeeratio - mean(coffeedata$n) # -2.103, there seems to be a bit of bias
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Question 1:

The plot you see just above this question shows the relation between energy use and the number of coffees (n). The ratio estimator is doing the same thing as fitting a regression model with just one predictor, but with the intercept being 0 (the origin) Lets fit a regression model that does just that. Run the code below, and compare your results to the ratio estimator. What do you find?

```
# first we fit a regression model without an intercept (the 0 is therefore included)
# note we still tell R to estimate this model under the srs sampling design
out <- svyglm(n~ 0 + energy,design=srsdesign)
print(out$coefficients) # show the regression coefficient. (very similar to our ratio estimator)
# comparison
print(ratio)
# the reason for the small difference between the ratio and regression coefficient?
# We have forced the regression line through the origin now!
# but the difference really is tiny,
# and we can probably live with the small bias in the ratio estimator.
```

Question 2:

In regression estimation, we cannot estimate the mean directly. We first need to predict the individual values in the population, using our regression coefficient ("out"), and the auxiliary variable we use, and then use the predicted values to estimate the quantity we want to estimate. Run the code below to estimate the mean number of coffees using the regression model we just fitted. Do we find a difference as compared to the ratio estimator?

```
# I am first just creating a new dataset, where I can add the predicted values later
predicted <- coffeedata2
# now add a predicted n (coffee) to the new dataset
# that multiplies the values on ' energy'  with the regression coefficent
predicted$predregr <- predict(out,newdata=predicted, total=mean(coffeedata$n))
mean(predicted$predregr)
# or by hand
mean(coffeedata$energy)*out$coefficients

# difference with the ratio estimator:
mean(predicted$predregr) - meancoffeeratio #last object here was the mean using ratio-estimation
```

The means are roughly the same! As they should be, because we fit basicly the same model.

#What about the standard error from our regression estimator?

If we want to get the SE for the regression-based estimator, this is a bit more tricky. First, we have to fit the regression model with a normal lm() function, not the 'svyglm' function. Now, you are probably slightly puzzled about NOT using svyglm, because we are now ignoring the whole sampling design! Indeed, I will show you in a minute why. Lets first show how to get the standard error. Run the code below:

```
# first we fit a normal lm function, all the rest is copied from the earlier regression model.
out2 <- lm(n~ 0 + energy,data=coffee)
print(out2$coefficients) # same regression as with svyglm
# and we are adding a new column to our data with the newly predicted values
predicted$predregr2 <- predict(out2,newdata=predicted, total=mean(coffeedata$n))
mean(predicted$predregr2) #341.493 = same
```

```r
# the predict() function includes a handy way to calculate
# standard errors following the "Delta"  method:

# here: Taylor series expansion (see also Fundamentals course),
# that you can get by adding "se=T" and then ask for the residual
# the reason for this is again,
# we are using *the individual* predicted values
predict(out2,newdata=predicted, total=sum(coffeedata$energy),se=T)$residual.scale
# se= 14.33, which is much larger than the ratio estimate
```

Question 3:

Compare the standard errors from the ratio estimator, and regression estimator. Write these down below:

Standard error ratio-estimator:
Standard error regression-estimator:

Question 4:

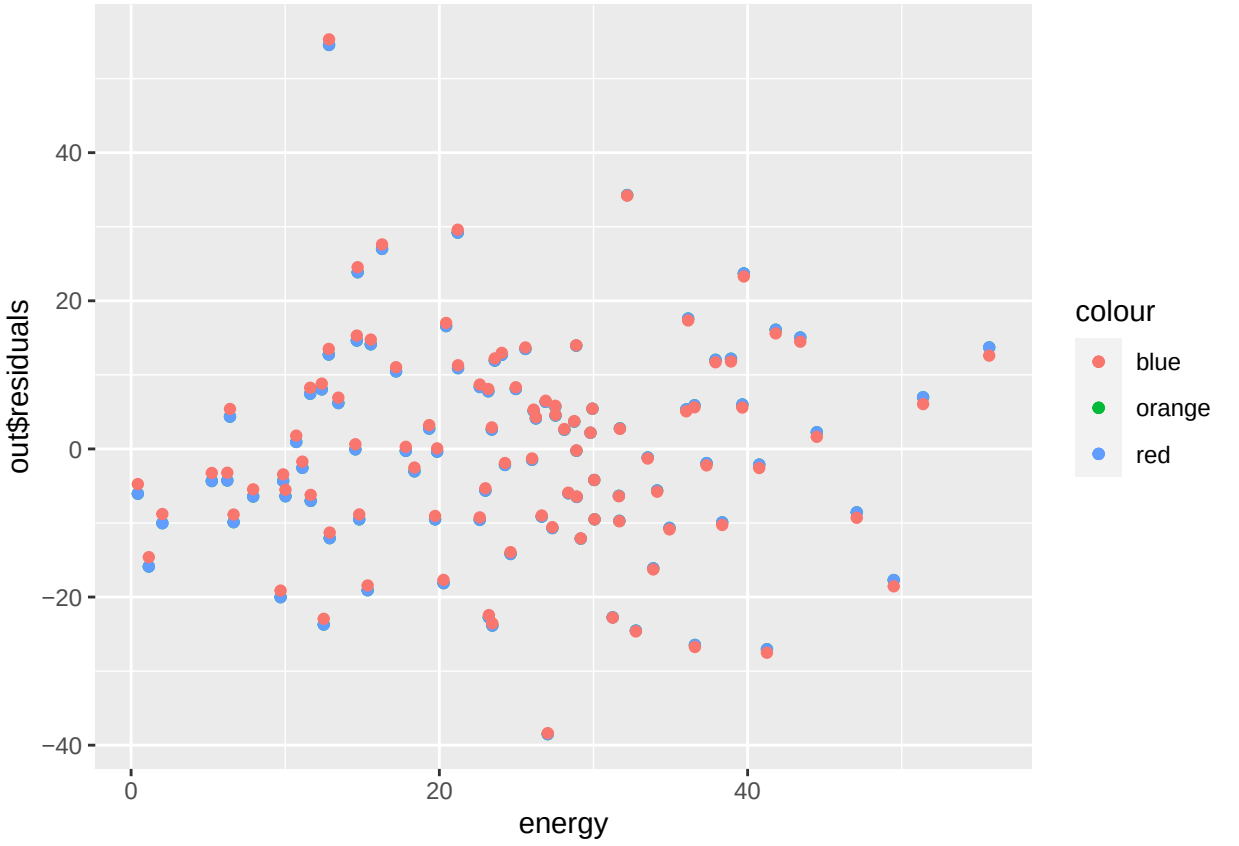
We saw earlier that the ratio estimator was a bit biased, because we assume the regression line goes through the origin. Or in other words, that the ratio between the two variables in the ratio-estimator is constant. When we switch to a regression model, we can however also add an intercept to the model.

Run the regression model from question 1 again, but now add an intercept to the model. What do you find when you compare the two models?

##Question 5:

The standard error in regression-based models do not represent *sampling* uncertainty anymore, but rather represent *uncertainty about the regression line*. Although the standard errors of the SRS estimate, and the regression-based estimate seem similar in size, they have completely different meanings now! We can plot the residuals from the different models we have estimated so far, to understand better what the standard error means. Run the code below. Can you explain from looking at the plot what the size of the standard error is?

```r
# plot the residuals
ggplot(predicted, aes(x=energy))+
  geom_point(data=out, aes(y=out$residuals, colour="orange"))+ # no intercept
  geom_point(data=out2, aes(y=out2$residuals, colour="red"))+ # + intercept
  geom_point(data=out3, aes(y=out3$residuals, colour="blue"))
```




```
# the square root of the squared deviations from the regression line is about 14
```

Imagine there is more information on the sampling frame. Apart from energy consumption, there are now two additional predictors:
- age of the machine (in years) - brand (A or B)

I will also now create a new sample, no longer consisting of a random sample out of the population, but a selective sample of the 100 machines with the highest enery consumption only. This is one of the major advantages of model-based estimation. We do not strictly need a random sample for inference. This is also why we run just a "lm()" function, rather than specifying the "svydesign" object first.

We will after this try to improve the model-based inference by adding more covariates and working from a biased sample.

```
coffeedata$age <- out3$residual/4  + 10
coffeedata$brand <- round((out$residual+rnorm(1000,mean=45,sd=8))/28)
coffeedata$brand[coffeedata$brand==4] <- 0
coffeedata$brand <- as.factor(coffeedata$brand)
levels(coffeedata$brand) <- c("maas","bravilor","nescafe","douwe egberts")

# I am also adding a new dataset where we can store predicted values
predicted <- coffeedata
predicted$n <- NULL

# selecting the sample: arrange by energy (lo:hi), take the last 100 observations (slice_tail) and dele
biasedsample <- coffeedata %>%
```

```
  arrange(energy)%>%
  slice_tail(n=100)%>%
  mutate(srs=NULL)
```

Question 7:

In regression-based inference we need to minimize the standard error of the model to get the best estimator. We can then utake the mean values of X, and use the regression line to predict the mean in Y. In other words, inference will work under the condition that: - we have a a good regression model (low se) - we need to have good information on the level of the population on the covariates. In order to understand this last point: it was nice that we had the variable "energy use" for every machine in our dataset, but it would have sufficed had we only had the mean level of energy for all the machines.

To summarize, we need to have some sample data (no longer a randomly sampled one!), where we can estimate a good regression model, for which we then only need the assumption that the coefficients that we find in our sample also hold in our population. Then, we take the mean values in the population for the covariates used in the regression model, and then multiply:

*our regression model coefficients * population means on covariates in model*

Following this method of inference, first compute the mean values for "brand", "age" and "energy in the population

Mean(brand): Mean(age): Mean(energy):

In a second step, try to estimate from the "biasedsample". Use similar syntax as in question 4, but now use the new sample, and find out whether the new covariates "brand" and "age" improve the model. Do they?

Question 8

Using your model estimated above again, did the fact that we now use a very biased sample (only the machines consuming a lot of energy), lead to a very different result. Why not?

```
# we apparently were able to estimate the regression coefficient correctly for the population, even tho
# we were using a biased sample

# Model-based estimation thus has very different assumptions:
# 1. your model needs to have good explanatory power (to get a precise estimator).
# 2. your model needs to be corectly specified (to avoid bias)
# 3. the model in your sample needs to work in the same way as it does in the population

# and for it to work, you need to have some information about the populations (means, proportions)
# for the variables that you use as covariates in your model
```

– End of document –