

ETC2420/5242: Statistical thinking. Assignment 2

Due 12pm (midday), 20 October, 2021

Instructions

- **Due Date:** ~~15 October, 2021. 12:00pm (Midday)~~ **20 October, 2021. 12:00pm (Midday)** via Moodle¹
- **Weight:** 20%
- **Number of submissions:** 1 per group.

This is a *group assignment*. The groups have been centrally assigned.

- All students should complete question 1
- 4 person ETC2420 students should do question 2
- All ETC5242 students should do question 3
- Questions that you were not supposed to do will not be marked.

Main changes to the marking scheme for this assignment

Please read all instructions carefully, but here are the most important changes.

Messy R output will be penalised. The report should not have extraneous R output (such as messages, warnings, or printing out variables). You will lose marks for this.

Code comments will not be marked as if they are discussions of results. The answers to all questions should contain paragraphs of text separately from the code. Code comments will not be marked. This means that if you do not write text outside of your code (and outside of the code comments) you will receive very few marks.

The handling of group marks will be modified. After the assignment is submitted, students will be asked to complete a short survey outlining the contributions of the group members. If a group member is agreed to have under-performed, they will be contacted and their mark will be reduced. Marks will never increase from this process: everyone in the group is responsible for the entire submitted assignment.

Specific instructions for ETC2420

All students will complete question 1 and four person groups will complete question 2.

The assignment **must** be submitted as a pdf file. All code must be available and neatly formatted. It should be clear that the code submitted produces the analysis in the report.

The report should not have extraneous R output (such as messages, warnings, or printing out variables). You will lose marks for this.

The answers to all questions should contain paragraphs of text separately from the code. **Code comments will not be marked.** This means that if you do not write text outside of your code (and outside of the code comments) you will receive very few marks.

It is *suggested* that you prepare the report reproducibly by using RMarkdown, setting the seed, and generating all of the plots and tables. However, we will accept a report prepared in Word or another word-processing system as long as

¹Extension because I released it a bit later than I wanted and I know there is a pile-up of assessment around now.

1. The code is included in the report as appropriate.
2. All of the code is **also submitted as a separate .R file** that is cleanly structured and, when run, produces all of the output in the assignment.
3. It is clear which code produces which results.

The assignment will be marked out of 40 (30 for 3 people groups).

- 10 marks for the writing and formatting of the report. The report should be written in complete, clear sentences. All answers should be explained fully. All graphs should have meaningful labels and titles. Marks will be removed for extraneous R output.
- 20 marks for Question 1
- 10 marks for Question 2 (four person groups)

Specific instructions for ETC5242

All students will complete questions 1 and 3. (Do not do question 2.)

The assignment **must** be prepared using RMarkdown and submitted as a pdf. All analysis code should be available. Extraneous messages should not be outputted. It is preferred that you knit directly to pdf, but you won't lose marks from converting from pdf *as long as the results are legible*.

The assignment will be marked out of 40

- 10 marks for the writing and formatting of the report. The report should be written in complete, clear sentences. All answers should be explained fully. All graphs should have meaningful labels and titles.
- 20 marks for Question 1
- 10 marks for Question 3

Question 1 (All students)

For this question, students should write *a short report* to answer the questions. This should be written in paragraphs of text. Code comments will not be marked, so you will lose a lot of marks if you do not write paragraphs of text.

The report should contain:

- A brief introduction to the problem.
- An answer to all of the questions. You can use subheadings (**##** and **###** in RMarkdown) to separate the sections if you wish.
- A clear conclusion.
- All code needed to do the analysis (this should follow good coding practice)
- *Only* graphs that are directly referenced in the text. (They must be well constructed and legible.)
- *No extraneous R output*.

To aid with the last point, I *strongly* suggest you include the following code as your first R chunk:

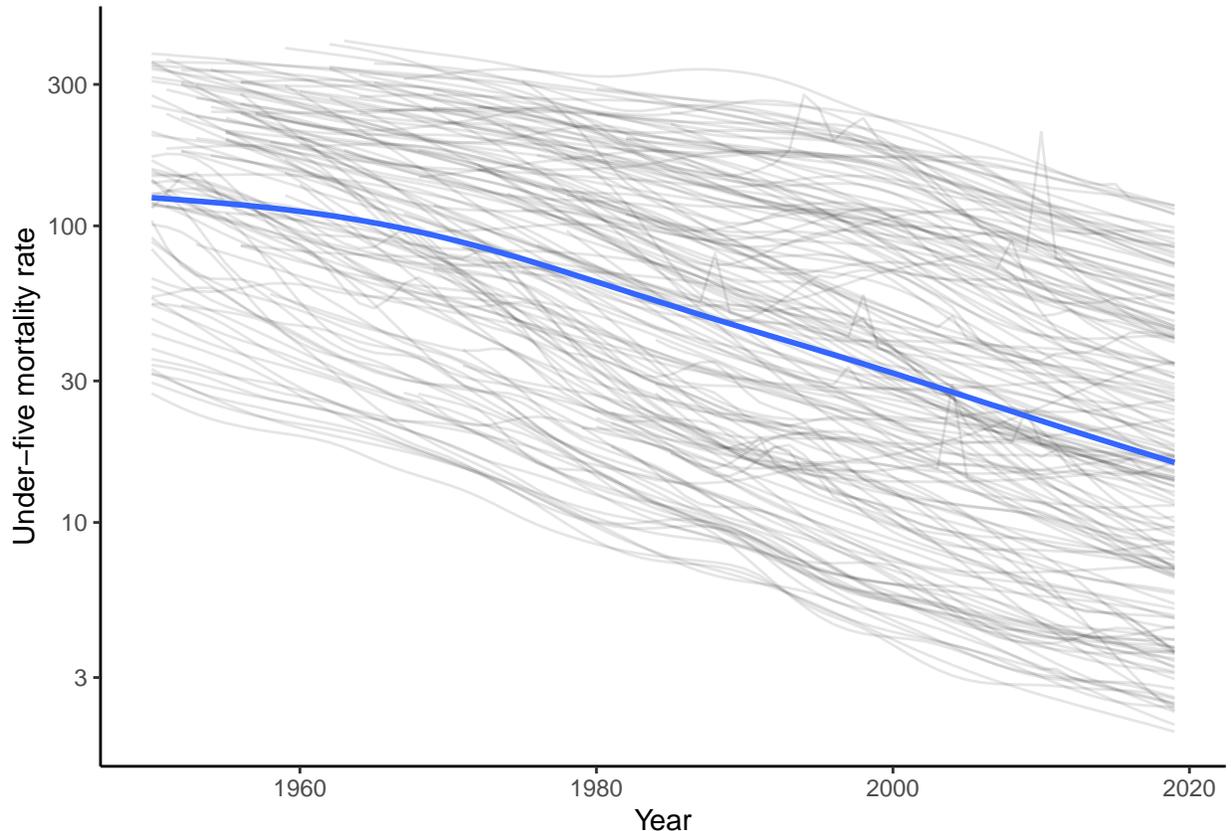
```
```${r,echo = FALSE, include = FALSE}
knitr::opts_chunk$set(echo = TRUE, message=FALSE, warning=FALSE)
library(printr)
options(digits = 2)
```
```

The task: Estimating neonatal mortality

One of this century's global goals has been the reduction of childhood mortality across all countries. There has been enormous effort put into this goal at all levels from the united nations down to local interventions.

While there is still a lot of variation between countries, the rate of childhood mortality (measured as the number of deaths of children under five years old per 1000 live births) has decreased over the decades.

The following plot shows the *Under-five mortality rate* (U5MR, aka the number of children under 5 who die per 1000 live births) estimates published by the UN Inter-agency Group for Child Mortality Estimation (IGME)².



As under-5 mortality has decreased, the *composition* of child mortality has changed over time. In particular a larger proportion of deaths are classed as *neonatal mortality*, which is a death that occurs within one month of birth.

In this question, we are going to look the data published by the IGME³ on neonatal mortality, available as `neonatal_mortality.csv`.

The data contains the following columns:

- `country_name`: The (short) name of the country
- `year`: The year the data was measured
- `region`: The name of one of seven large global regions
- `nmr`: The observed number of neonatal deaths per thousand live births (the neonatal mortality rate). This is measured either using a country's vital registration system (births and deaths register) or using some sort of high-quality survey⁴.
- `u5mr`: The estimated under-five mortality rate.

Neonatal mortality rate is a strictly positive variable, so we model it on a transformed scale (otherwise there is very little hope of Gaussian residuals). Alexander and Alkema (2018)⁵ recommend instead modelling the

²The original version of all data used in this question is available at <https://childmortality.org/data>

³Some of the data has been excluded and I have not given all of the columns.

⁴This information is available in the original data.

⁵<https://www.demographic-research.org/volumes/vol38/15/38-15.pdf>

fraction

$$\log\left(\frac{\text{nmr}}{\text{u5mr} - \text{nmr}}\right),$$

which is the (log of the) number of neonatal deaths per 1000 live births divided by the number of non-neonatal deaths per 1000 live births.

Your task is to produce a linear regression model to estimate the average neonatal mortality rate (NMR). Your answer should include discussion (and graphs where appropriate). You should:

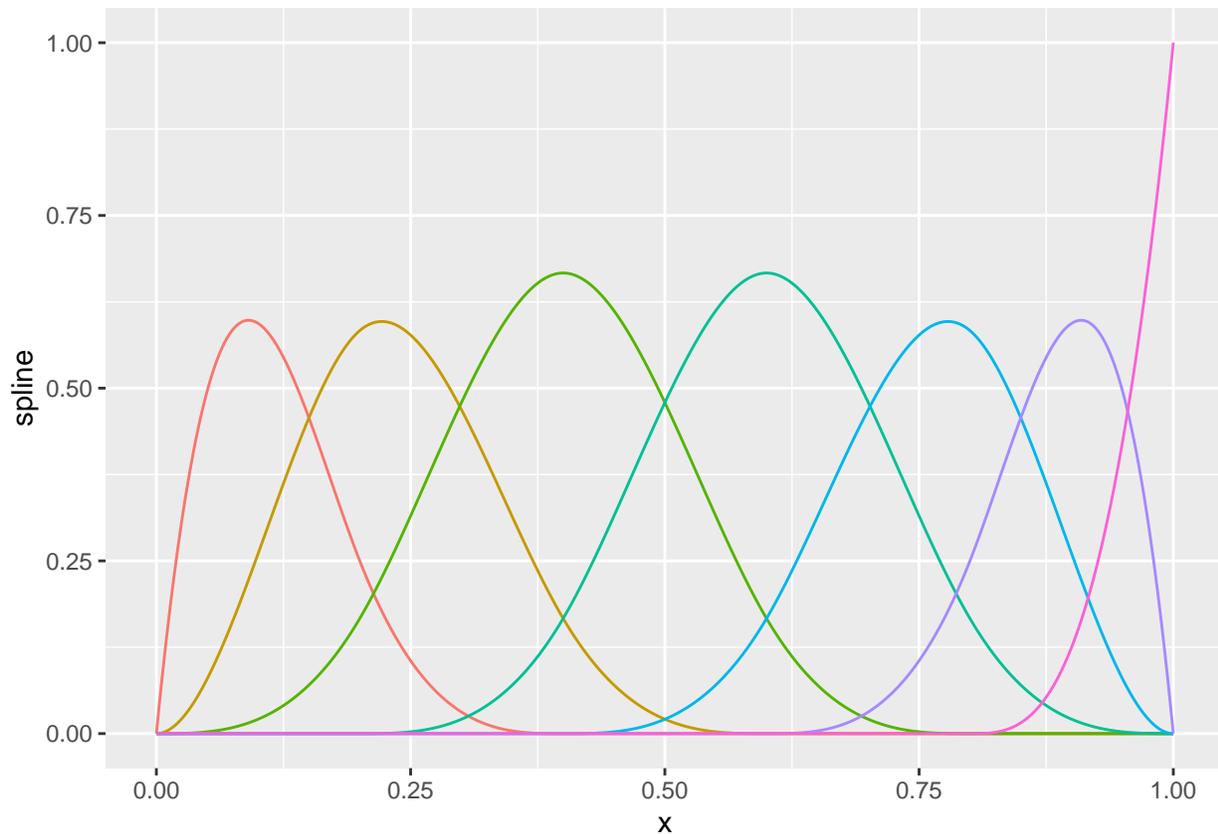
- Explain the choice of variables in your model (you should not use `country_name`, if you use U5MR, you should use it on the log-scale!). In particular you should consider whether an *interaction effect*⁶ should be used.
- Assess your linear model and comment on its fit. This should be done a) for all data simultaneously; b) for data in each region; and c) for data in a maximum of 3 countries that should be chosen to highlight different aspects of the fit diagnostics.
- Estimate the root mean square error and the mean absolute error on a test set. The test set should be produced using the argument `strata = region`.
- Produce a prediction, with prediction intervals⁷, of the NMR on its natural scale (aka not on the log-scale) and plot these a) for all data simultaneously; b) for data in each region; and c) for data in a maximum of 3 countries that show different aspects of the fit.

In order to improve your model fit, you should also consider the non-linear effect of one variable. For this assignment, we will use B-Splines, which are a better method than polynomial regression for this problem.

A B-Spline is a smooth function that is only non-zero for a small part of the domain that are designed to add together to make smooth functions. The following graph plots 7 b-spline basis functions.

⁶An interaction effect between a (possibly continuous) covariate x and a discrete covariate u can be fit using `lm(y ~ x * u)`. It will model the effect of x , u , and then a separate slope of x for *each* level of u . See this chapter (hyperlink) in Statistical Thinking from the 21st Century

⁷CLT intervals are fine.

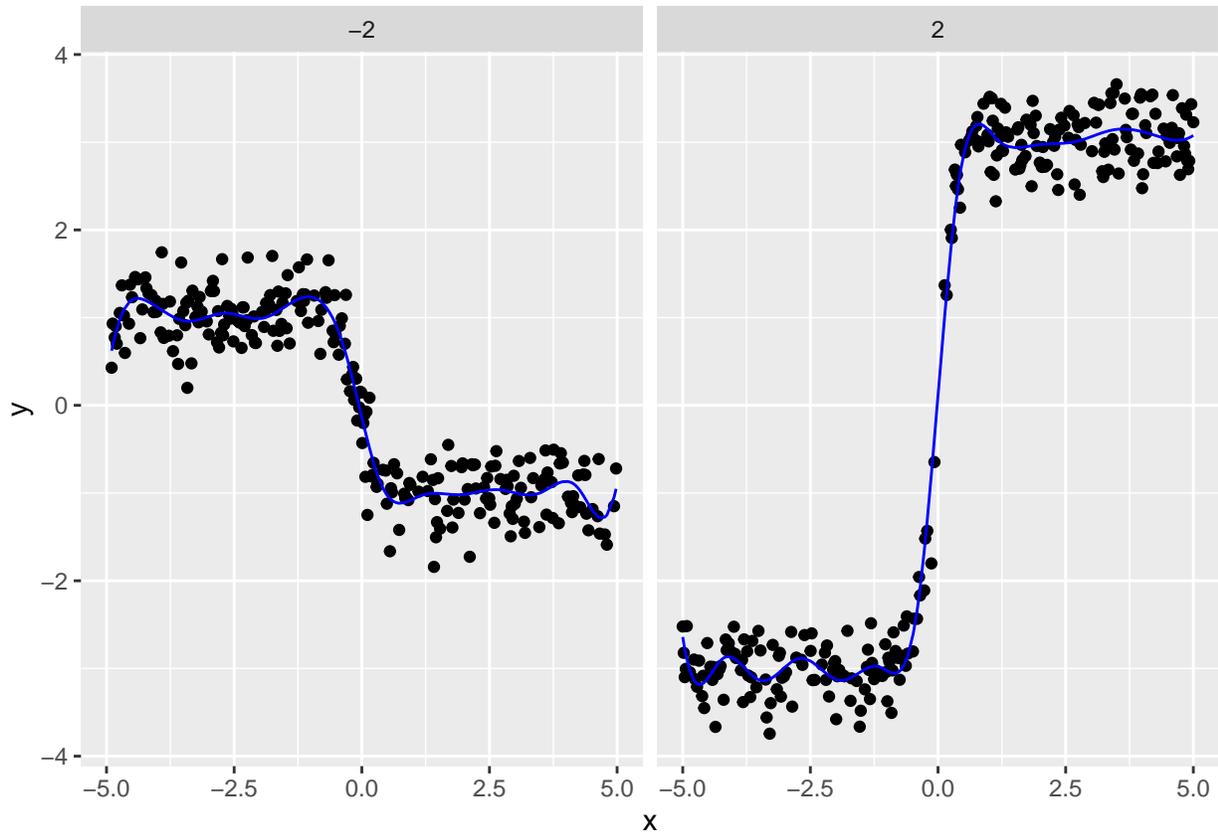


B-splines can be added to a regression using the `spline::bs()` function. This function can be used in an `lm` formula like an ordinary variable in your tibble. An illustrative example, which fits an interaction model and uses 15 basis functions (`df = 15`), is below.

```
library(splines)
library(broom)
library(tidyverse)
dat <- tibble(x = seq(-5,5, length.out = 500),
              z = sample(c(-2,2), size = 500, replace = TRUE),
              y = (1 + z) * tanh(3 * x) + rnorm(500, sd = 0.3)) %>%
  mutate(z = factor(z))

fit <- lm(y ~ bs(x, df = 15) * z, data = dat)

fit %>% augment(data = dat) %>% ggplot(aes(x,y)) +
  geom_point() +
  geom_line(aes(x, .fitted), colour = "blue") +
  facet_wrap(~z)
```



Modify the linear model you produced to incorporate an appropriate non-linear effect. You should do the following:

- Explain your choice of model, using appropriate visualisations to support your choice.
- Use cross-validation to select an appropriate number of basis functions for `bs()`
- For your final model, assess your linear model and comment on its fit. This should be done a) for all data simultaneously; b) for data in each region; and c) for data in a maximum of 3 countries that should be chosen to highlight different aspects of the fit diagnostics.
- Estimate the root mean square error and the mean absolute error using the same test set as before.
- Produce a prediction, with prediction intervals⁸, of the NMR on its natural scale (aka not on the log-scale) and plot these a) for all data simultaneously; b) for data in each region; and c) for data in a maximum of 3 countries that show different aspects of the fit.
- Write a paragraph or two describing the differences between the two models and explaining which you think is a more appropriate model of the data.

Question 2 (ETC2420 students only, four person groups only)

Automatic variable selection is an important problem in linear regression. In this question, we will look at (*backwards*) *stepwise regression*. This procedure is as follows:

1. Fit the linear model with *all* variables and estimate the root mean square error using cross-validation.
2. For each variable in the model fit the regression model without that variable and estimate the root mean-square error of the resulting model using cross-validation
3. Remove the variable that results in the *smallest reduction* in RMSE.
4. Repeat steps 2. and 3. until all of the variables are gone.
5. Choose the model with the lowest cross-validated RMSE.

⁸CLT intervals are fine.

Your tasks are as follows:

1. Write a function that performs backwards step-wise regression on a model. It should have the signature

```
bsr <- function(data, full_formula, threshold = 0.9) {}
```

2. Simulate 100 data points from a model with 4 covariates with non-zero regression coefficients and 10 additional covariates that have zero regression coefficients. This can be done with the following code.

```
p <- 14
n <- 100
data <- matrix(rnorm(n * p), nrow = n, ncol = p) %>%
  as_tibble() %>% # columns will be called V1, ..., V14
  mutate(y = -0.2 + V1 + 2 * V2 - 3 * V3 + V4 + rnorm(n, sd = 0.4))
```

3. Run your `bsr()` function on the simulated data and comment on the results.

You will probably need to use the `update.formula()` function to remove variables from the regression. Here is an example that removes V1 and then V2.

```
all_vars <- paste0("V", 1:5)
formula_all <- reformulate(all_vars, response = "y")
formula_all

## y ~ V1 + V2 + V3 + V4 + V5
remove <- "V1"
change_formula <- paste0("~ . -", remove) %>% as.formula()
formula_reduced <- update.formula(formula_all, change_formula, data = data)
formula_reduced

## y ~ V2 + V3 + V4 + V5
remove <- "V2"
change_formula <- paste("~ . -", remove) %>% as.formula()
formula_reduced <- update.formula(formula_reduced, change_formula)
formula_reduced

## y ~ V3 + V4 + V5
```

Question 3 (ETC5242 students only)

Automatic variable selection is an important problem in linear regression. In this question, we will look at variable selection using the LASSO. The LASSO is a penalised regression method that can be fit using the following `tidymodels` specification:

```
library(tidymodels)
spec <- linear_reg(penalty = tune(), mixture = 1) %>%
  set_engine("glmnet")
```

The LASSO has the very special property that it will estimate some coefficients to exactly zero, effectively excluding them from the model. If you have enough data and your covariates aren't too related, it will select only the relevant observations.

For example, the following code simulates some data and fits the full linear model, the true model (with no spurious covariate), and the LASSO estimate (I skipped the tuning but you will need to tune to find a good value of the penalty parameter).

```
set.seed(123)
p <- 14
```

```

n <- 1000
data <- matrix(rnorm(n * p), nrow = n, ncol = p) %>%
  as_tibble() %>% # columns will be called V1, ..., V14
  mutate(y = V1 + 2 * V2 - 3 * V3 + V4 + rnorm(n, sd = 0.4))

rec <- recipe(y ~ ., data = data)

spec_lm <- linear_reg() %>%
  set_engine("lm")

spec_lasso <- linear_reg(penalty = 0.07, mixture = 1) %>%
  set_engine("glmnet")

wf <- workflow() %>%
  add_recipe(rec)

## True model
rec_true <- recipe(y ~ V1 + V2 + V3 + V4, data = data)

fit_true <- wf %>%
  add_model(spec_lm) %>%
  update_recipe(rec_true) %>%
  fit(data = data) %>%
  tidy() %>%
  rename(true = estimate) %>%
  select(term, true) %>%
  bind_rows(tibble(term = paste0("V",5:14), true = 0))

## Full model
fit_full <- wf %>%
  add_model(spec_lm) %>%
  fit(data = data) %>%
  tidy() %>%
  rename(full = estimate) %>%
  select(full)

## Fit LASSO
fit_lasso <- wf %>%
  add_model(spec_lasso) %>%
  fit(data = data) %>%
  tidy() %>%
  rename(lasso = estimate) %>%
  select(lasso)

## Make a table that compares everything
fit_true %>% bind_cols(fit_full, fit_lasso)

```

| term | true | full | lasso |
|-------------|-------|-------|-------|
| (Intercept) | 0.01 | 0.01 | 0.01 |
| V1 | 1.01 | 1.01 | 0.95 |
| V2 | 2.01 | 2.01 | 1.94 |
| V3 | -3.01 | -3.01 | -2.94 |

| term | true | full | lasso |
|------|------|-------|-------|
| V4 | 1.01 | 1.01 | 0.94 |
| V5 | 0.00 | 0.03 | 0.00 |
| V6 | 0.00 | -0.01 | 0.00 |
| V7 | 0.00 | 0.00 | 0.00 |
| V8 | 0.00 | -0.01 | 0.00 |
| V9 | 0.00 | -0.01 | 0.00 |
| V10 | 0.00 | 0.00 | 0.00 |
| V11 | 0.00 | 0.00 | 0.00 |
| V12 | 0.00 | 0.00 | 0.00 |
| V13 | 0.00 | 0.01 | 0.00 |
| V14 | 0.00 | 0.00 | 0.00 |

You can see that in this case the LASSO correctly selected the first four variables and produced very similar estimates.

A challenge with the LASSO is it is very difficult to find analytical formulas for confidence intervals. For example, there is no easy equivalent of a CLT-style confidence interval. So in this question we are going to examine the bootstrap for solving this task.

Your task is as follows:

- Write a function that uses a residual bootstrap to compute an 80% confidence interval for the LASSO estimate for each regression parameters. You should tune the penalty parameter *each time* the LASSO is run! Do not use any pre-made bootstrap functions in R to complete this task.
- Write a function that checks the *coverage* of each of these intervals.
- Comment on whether or not the (modified) residual bootstrap achieves the nominal coverage

The residual bootstrap is as follows:

1. Fit the LASSO to the data to estimate the parameters $\hat{\beta}$
2. Compute the residuals $e_i = y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \dots - \hat{\beta}_p x_{ip}$
3. Normalise the residuals⁹ $\epsilon_i = e_i - \bar{e}$
4. Re-sample the residuals with replacement to get $\epsilon_i^*, i = 1, \dots, n$
5. Make bootstrap data $y_i^* = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \dots + \hat{\beta}_p x_{ip} + \epsilon_i^*$
6. Fit the LASSO to the bootstrap data to obtain $\hat{\beta}^*$
7. Compute the bias $\delta^* = \hat{\beta} - \hat{\beta}^*$
8. Make the bootstrap confidence intervals as usual.

⁹This step wasn't done in the lecture code. You also don't need to worry about adjusting for leverage: there is no good equivalent to that here!