TMA4315 Generalized Linear Models

Compulsory exercise 1: Linear models for Gaussian data

Deadline: Friday, September 28, 2018 at 16.00

To be handed in on Blackboard.

Students may work on the assignment alone, or in groups of two or three, and are also encouraged to collaborate across groups. Each group needs to hand in an [R Markdown] (http://rmarkdown.rstudio.com/) document with answers to the questions as well as all source code. Hand in the file as an R Markdown file (`.Rmd`) *and* a pdf file (`.pdf`). Make sure that the `.Rmd` compiles as both pdf and html before you hand it in. Include all code you have written in the `mylm.R` file in the end of your file (hint: use `eval = FALSE` in the chunk option) (only include the final version).

You shall deliver a document that answers all questions, using print-outs from `R` when asked for/when it seems necessary. It should include calculations you needed to answer the questions, and enough information so the teaching assistant can see that you have understood the theory (but do not write too much either, and exclude code not directly necessary for the answers). You will often be asked to interpret the parameters. This does not mean that you should just say what the numeric values are or state how you interpret the parameter in general, but rather explain what these values mean for the model in question.

Please write the names of the group members and the group number you have on Blackboard on top of your document!

Guidance by the teaching assistant (Ingeborg) will be given Tuesdays and Thursdays from 10:15-12:00 in room 922 at the 9th floor of Sentralbygg 2. She will be there from the beginning, but leaves if no one shows up. If you know you will be arriving very late, send her an email (ingeborg.hem@ntnu.no) to ensure that she will be in 922.

In this project you will create your own `R`-package to handle linear models for Gaussian data and use this package to analyse a dataset. You can find more information about the package system and classes in `R` at https://cran.r-project.org/doc/contrib/Leisch-CreatingPackages.pdf. The text below describes how the package can be created using R-Studio since it makes it easy to rebuild and reload the package. The description is reasonably platform-independent, and should work on Windows, Mac and Linux. R-Studio is a useful integrated development environment (IDE) for `R`.

Introduction

Follow these steps to get started:

1. Install `R` and R-Studio if necessary (already installed on machines in Nullrommet and Banachrommet) (can be useful to update `R` if you do not have the latest version). Contact orakel@ntnu.no if you run into technical problems with your own computer.
2. Start R-Studio
3. Select *File–New Project*
4. Select create project from *New Directory*
5. Select project type *R Package*
6. Give the package the name *mylm*
7. Select the desired directory in the field *Create project as subdirectory of*
8. Select *Create Project*

This will create the required folder structure and minimum files required for an `R`-package. The project folder `mylm` will be a subfolder of the directory chosen. The file "DESCRIPTION" contains a description of the package and you can change it with your information. The folder "man" will contain documentation for your package and can be ignored for now. The folder of interest is “R”, which contains the source code for your package.

Each time you make a change to the source code of the package, you must compile it (in R-studio this is called build) again and load the library in your `R`-session. In R-Studio this can be done by

- Select *Build–Build and Reload.* Build is a window pane in the upper right part of the standard Rstudio set-up (together with Environment and History).

Save the file https://www.math.ntnu.no/emner/TMA4315/2018h/mylm.R in your folder *mylm/R/ (replace if such a file already exists). Remember that you need to rebuild the package in R-Studio before it will be available in your `R`-session. When you write source code that uses the functions in your package, you will first need to load the package with `library(mylm)`.

The next time you start R-studio you may have to select *File–Open Project* and select the file `mylm.Rproj` in the folder *mylm* to use R-studio to build the package.

Part 1: Explanatory analysis of the dataset (0.5 points)

Before you start working on your `mylm` package you will study the dataset. (So, for this part your `mylm` package is not needed.)

The dataset we will work on is from Canada, and consists of 3987 observations on the following 5 variables:

- `wages`, composite hourly wage rate from all jobs
- `education`, number of years in schooling
- `age`, in years
- `sex`, Male or Female
- `language`, English, French or Other

The dataset is available in the library `car` in `R`, but with missing data. The following command will store the data as a dataframe called `SLID`, and then remove all rows containing missing data.

```r
install.packages("car")
library(car)
data(SLID, package = "carData")
SLID <- SLID[complete.cases(SLID), ]
```

Draw a matrix of diagnostic plots of all the variables and comment briefly on the relationship between some of the variables. Hint: The function `ggpairs` (from the `GGally` library) makes a diagnostic plot matrix using the `ggplot` functionality.

We want to study how the qualitative variable `wages` depends on one or more explanatory variables. Which assumptions do we need make about the data if we want to perform a multiple linear regression analysis?

Part 2: Linear regression with the `mylm` package (4.5 points)

You can use asymptotic expressions for all the tests. This means that instead of the exact t_ν-test you can use the asymptotic z-test, and instead of the exact $F_{r,m}$-test you can use the asymptotic χ^2_m-test (but test statistic scaled with r, details below). The `lm` function in `R` uses the exact (not asymptotic) tests, but later in this course we will use models that need the asymptotic tests (for the binary regression, the Poission regression, ...). That is why you will use them already here. The asymptotic expressions are valid when the number of observations is large.

When you are finished with the whole exercise, your `mylm`-package should be able to

1. calculate the estimates of the regression coefficients, and the estimated covariance matrix of the parameter estimates
2. calculate the standard error and z-test for each estimated coefficient
3. calculate the fitted values and the residuals
4. calculate the residual sum of squares (SSE) and degrees of freedom
5. calculate the analysis of variance table
6. calculate the coefficient of determination (R^2)

Note that you will be creating a class called `mylm` in this exercise. That is why all function names end with ".mylm". You do NOT need to include ".mylm" when using the functions, as `R` detects the class of the object you are giving to the function. This will hopefully become more clear during the exercise, the important thing for now is that you are aware that you are creating a new class, called `mylm`.

If you are not familiar with lists in `R`, http://www.r-tutor.com/r-introduction/list might be to some help before you begin.

Please include the statistical formulas and theory you have used to write the content of the functions in your `mylm` package.

a) (1 points)

Fill-in the missing parts in the `mylm` function in your `mylm` package required to calculate the estimates of the coefficients. Fit a simple linear regression model to the data with `wages` as the response and `education` as predictor to check if the function works.

The commands

```
model1 <- mylm(wages ~ education, data = SLID)
print(model1)
```

and

```
model1b <- lm(wages ~ education, data = SLID)
print(model1b)
```

should give the same coefficient values. The function `mylm` returns an object of class `mylm` (this is already in the code). You will have to implement a `print.mylm` function in your package.
Hint 1: See `?lm` for help.
Hint 2: The functions `print.default`, `cat`, and `format` will be helpful when you are printing results. The two former are printing numbers and text, while the latter can control the number of digits, the spacing between entries and more.
Hint 3: The function `browser` in `R` can be used for debugging.

Note that you do not have to spend much time on how the outputs are printed, as long as you and others that are using your functions can understand the output. Making a pretty output will not substantially increase your statistical learning, or your grade on the exercise.

b) (1.5 points)

Develop the `mylm` function further, so it can calculate the estimated covariance matrix for the parameter estimates.

What are the estimates and standard errors of the intercept and regression coefficient for this model? Test the significance of the regression coefficient using a z-test. What is the interpretation of the parameter estimates?

Fill-in the missing parts in `summary.mylm` such that

```
summary(model1)
```

gives a similar table of tests of significance as `summary` used on an `lm` object. Note that the results will not be the same since `lm` uses a t-test, and you should use a z-test in `mylm`. Hint: To calculate two-sided p-values remember the standard normal distribution is symmetric around 0.

c) (0.5 points)

Implement a plot function for the `mylm` class (called `plot.mylm`) that makes a scatter plot with fitted value on the x-axis and residuals on the y-axis (the raw residuals). This function shall use the `ggplot` functionality from the library `ggplot2`! (Don't make this too complicated.) Comment on the plot.

d) (1 points)

After a scaling, the χ^2-distribution is the limiting distribution of an F-distribution as the denominator degrees of freedom goes to infinity. The normalization is $\chi^2 = \text{(numerator degrees of freedom)} \cdot F$.

- What is the residual sum of squares (SSE) and the degrees of freedom for this model?
- What is total sum of squares (SST) for this model? Test the significance of the regression using a χ^2-test.
- What is the relationship between the χ^2- and z-statistic in simple linear regression? Find the critical value(s) for both tests.

Fill-in the missing parts in the function `anova.mylm` so that it shows the results in a similar way as the `anova` function on an object from `lm`. Note that your values differ from the ANOVA table produced by `lm`, as you use the χ^2-test instead of the F-test used by `lm`. You shall use the same ANOVA type as `lm`, i.e., a sequential ANOVA, also called type I ANOVA. Type II and III also exists, but we do not consider them in this exercise. Print the ANOVA-table and comment briefly.

e) (0.5 points)

What is the "coefficient of determination" (R^2) for this model and how do you interpret this value?
Modify the function `summary.mylm` so that it shows this value.

Part 3: Multiple linear regression (3 points)

a) (1 points)

Make any changes necessary to make the function `mylm` and `print.mylm` work also for multiple linear regression. Fit a linear regression model to the data with `wages` as the response with `education` and `age` as predictors.
You can regress a variable `y` on `x1` and `x2` from the data frame `myframe` with

```
model2 <- mylm(y ~ x1 + x2, data = myframe)
```

b) (1 points)

What are the estimates and standard errors of the intercepts and regression coefficients for this model? Test the significance of the coefficients using a z-test. What is the interpretation of the parameters?
Make any changes necessary to make the function `summary.mylm` work also for multiple linear regression.

c) (1 points)

The estimated effect of `education` and of `age` on `wages` can be modelled in two simple linear regressions (one model with only `age` as covariate and one with only `eduacation`) or a multiple linear regression (both covariates in the model). Why (and when) does the parameter estimates found (the two simple and the one multiple) differ? Explain/show how you can use `mylm` to find these values.

Part 4: Testing the `mylm` package (2 points)

Use dummy variable coding only for this part.

Now your `mylm` package should be able to do regression on any linear model. Confirm this by fitting models with `wages` as response, and the following predictors:

- `sex, age, language` and `education^2` (see the function `I()` to include functions of variables in the formula. This gives the same results as if you transform the data before fitting the models, but is simpler. To use `education^2` directly only gives the linear variable, which is not what we want now here!)
- `language` and `age` with an interaction term between the predictors
- `education` and no intercept (you can remove the intercept by adding `-1` to the formula. Note that R^2 is not defined when the intercept is removed from the model, so your R^2 will probably differ from that of `lm`. Ignore this, and use other diagnostics to evaluate the model.)

For each of the three models: Use your `mylm`-package to fit the models and include some diagnostics. Write a few sentences about the interpretation and significance of the parameters, and mention one small change that could make the model better.

Analysis of variance (ANOVA) (0 points, optional)

This part is optional!

Use your `mylm` package to solve this problem.

Until now, you have not been asked to think about how the model matrix is created. The default in both `lm` and your `mylm` is using dummy variable coding (treatment) for the categorical variables. In this part, you will use both dummy variable and effect coding (sum to zero). The following code specifies how the model matrix is created for `mylm`, and should work for you:

```r
# sum to zero, need to be specified
model3a <- mylm(wages ~ sex + language, data = SLID,
                contrasts = list(language = "contr.sum",
                                 sex = "contr.sum"))

# treatment (default, does not need specification,
# so the next two lines should give the same results)
model3b <- mylm(wages ~ sex + language, data = SLID)
model3b <- mylm(wages ~ sex + language, data = SLID,
                contrasts = list(language = "contr.treatment",
                                 sex = "contr.treatment"))
```

If you want to see how the model matrix looks like in the two cases, you must add it to the returned elements from `mylm`.

a) (0 points)

Fit a linear regression model to the data with `wages` as the response with `sex` and `language` as covariates. What are the estimate and standard error of the intercept and effects of factors for this model? Test the significance of the factors using a z-test. What is the interpretation of the parameters?

Do this for **both** effect and dummy variable coding (so you shall interpret all parameters for both types of coding), and comment briefly on the main differences (e.g., do the parameter estimates or the fitted values change?). Notice how the model matrix changes in the two cases.

b) (0 points)

You might have to make changes to the `anova.mylm` function in your package to handle multiple predictors (depending on how you implemented part `anova.mylm`). You should still use sequential ANOVA, as before.

Do a sequential anova of the models from 4a), with **both** effect and dummy variable coding. Comment on the results, and explain why they are as they are.
Why are the commands

```r
anova(mylm(wages ~ sex + language, data = SLID))
```

and

```r
anova(mylm(wages ~ language + sex, data = SLID))
```

giving different results? What does this tell us about the data? (Hint: https://mcfromnz.wordpress.com/2011/03/02/anova-type-iiiiii-ss-explained/)

c) (0 points)

For both effect coding and dummy variable coding, fit a model with `wages` as response, and `sex` and `age` as covariates. Also add an interaction term (hint: to add interactions between variables `x1` and `x2`, write `x1*x2` in the formula). What is now the interpretation of the parameters, and especially of the interaction term? Answer for both types of coding.

You shall now test whether the regression coefficients have a common slope or not, for **both** types of coding. Plot the regression lines for `wages` as a function of `age` for the different values of `sex` in the same plot, and perform the test using the plot (use `ggplot`, but only "simple" functions as `geom_point` and `geom_line` are allowed to use, `geom_smooth` is not ok as it uses `lm`).
Describe the graph. What is the conclusion? How can the test be performed without graphs? Does that give the same conclusion?
(Hint: `geom_abline` will not work alone, use `geom_line` instead. You can plot the regression lines for effect coding and dummy variable coding in the same plot, but they should be distinguishable, e.g., effect can be lines and dummy can be dots.)