STATS 2107
Statistical Modelling and Inference II

Practical 5: Linear models and model selection

Sharon Lee, Matt Ryan

Semester 2 2022

Contents

Simple linear regression

First you need to load some data. Let's go back to the `mpg` dataset.

```
library(tidyverse)
data(mpg)
```

To start, produce a scatterplot of cty on displ.

From the scatterplot, comment on the following:

1) Strength
2) Direction
3) Linear or non-linear

Quiz Questions

1. Comment on the relationship between `cty` and `displ`.

Fitting a linear model

To fit a linear model, we use the command `lm()`. It needs the data and a formula to describe the model.

Recall that the variables to the left of the tilde ~ are the response variables, while the variables to the right of the ~ are the predictors.

Fit the linear model with `cty` as the response variable and `displ` as the predictor:

```
M1 <- lm(cty ~ displ, data = mpg)
summary(M1)
```

```
##
## Call:
## lm(formula = cty ~ displ, data = mpg)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3109 -1.4695 -0.2566  1.1087 14.0064
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  25.9915     0.4821   53.91   <2e-16 ***
## displ        -2.6305     0.1302  -20.20   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.567 on 232 degrees of freedom
## Multiple R-squared:  0.6376, Adjusted R-squared:  0.6361
## F-statistic: 408.2 on 1 and 232 DF,  p-value: < 2.2e-16
```

Quiz Questions

Is there a significant relationship between `cty` and `displ`?

2. State the null and alternative hypothesis for testing the coefficient of `displ`.
3. State the test statistic and associated p-value for the above hypothesis.
4. At 5% significance level, will the null hypothesis be rejected?
5. Is there a statistically significant linear relationship between `dislp` and `cty`.

Obtaining coefficients

We can obtain the coefficients by using the `summary()` command, but this does not return them in a nice form. A better way is to use the `tidy()` command in the `broom` package. This will return all of the information as a data frame making it nice and easy to include the coefficients as a table in Rmarkdown.

```
library(broom)
tidy(M1)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)    26.0      0.482      53.9 3.30e-133
## 2 displ          -2.63     0.130     -20.2 4.74e- 53
```

Get Model summaries

As well as the function `tidy()`, `broom` will also give summaries of the model with `glance()`:

```
glance(M1)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squa~1 sigma stati~2  p.value    df logLik   AIC   BIC devia~3
##       <dbl>        <dbl> <dbl>   <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1     0.638        0.636  2.57    408. 4.74e-53     1  -552. 1109. 1120.   1529.
## # ... with 2 more variables: df.residual <int>, nobs <int>, and abbreviated
## #   variable names 1: adj.r.squared, 2: statistic, 3: deviance
```

This is very useful if you want to compare multiple models - see later.

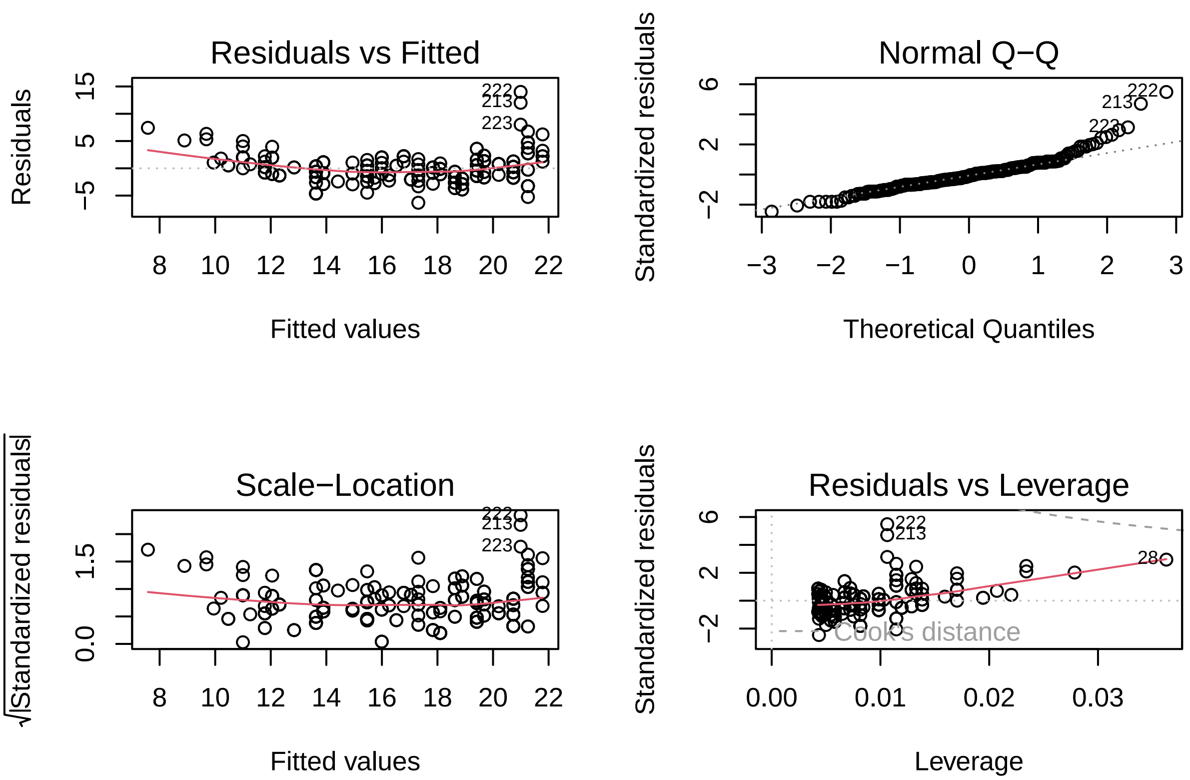
Assumption checking

Recall the following assumptions for linear regression.

1) Linearity
2) Normality
3) Constant variance
4) Independence

We can obtain plots to help assess these with the `plot()` function when applied to a linear model. We can get the usual four with

```
tmp <- par(mfrow = c(2,2))
plot(M1)
```




```
par(tmp)
```

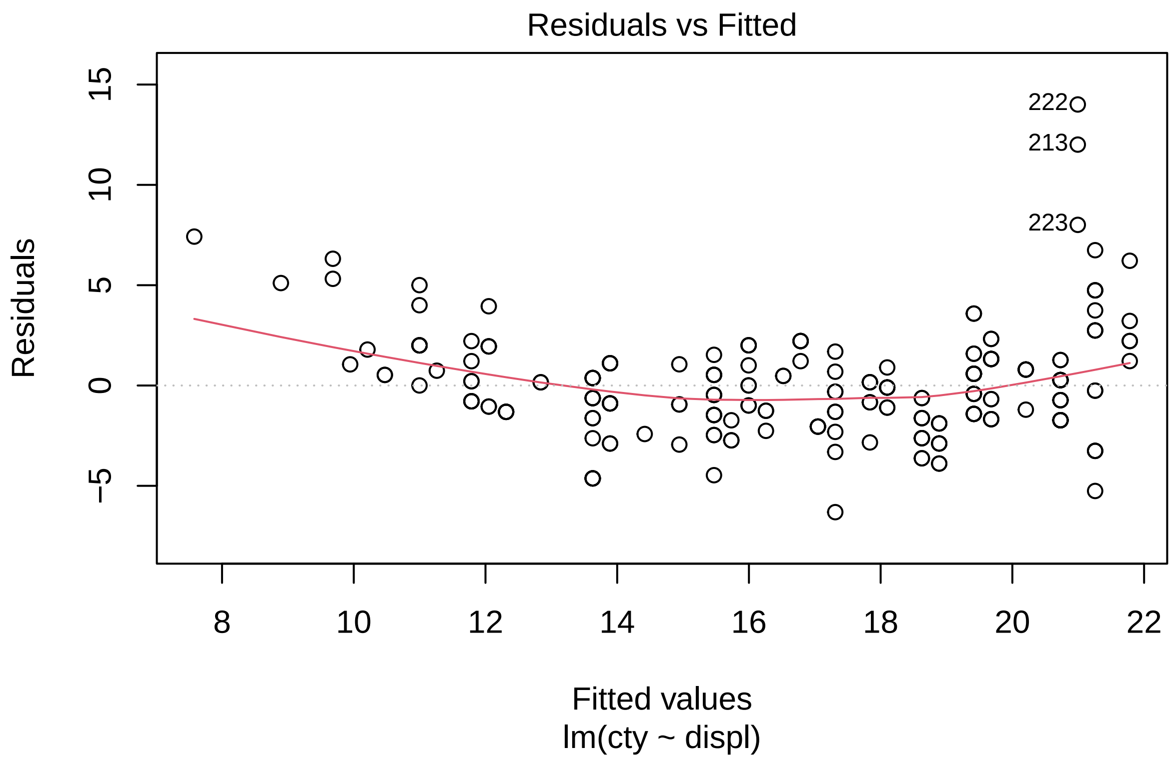
This will return four plots:

1) Residual vs fitted
2) Normal QQ plot
3) Scale-Location plot
4) Residual vs Leverage plot

Notice the use of `par` to get 2-by-2 plots.

You can also get a single plot using `which`:

```
plot(M1, which = 1)
```



There are six plots all up that you can choose from.

If you want to obtain the residuals and fitted values, then you can use the following commands:

```
residuals(M1)
fitted(M1)
```

Quiz Questions

6. Which plot can be used to assess the following?
   a) linearity
   b) normality
   c) constant variance
7. Comment on whether the assumptions were satisfied for the fitted model.

Prediction

Prediction is performed with the `predict()` function. If it is called with no new data, then it will predict for the observations in the data frame.

```
predict(M1)[1:10]
```

```
##          1         2         3         4         5         6         7         8
```

```
## 21.25660 21.25660 20.73050 20.73050 18.62612 18.62612 17.83697 21.25660
##        9       10
## 21.25660 20.73050
```

If you would like to predict for new values, then you must create a new data frame with the values you are interested in:

```
newdata <- tibble(
  displ = 3:6
)
newdata
```

```
## # A tibble: 4 x 1
##   displ
##   <int>
## 1     3
## 2     4
## 3     5
## 4     6
```

```
newdata$pred <- predict(M1, newdata = newdata)
newdata
```

```
## # A tibble: 4 x 2
##   displ  pred
##   <int> <dbl>
## 1     3  18.1
## 2     4  15.5
## 3     5  12.8
## 4     6  10.2
```

Categorical variables in a linear model

If you want to use categorical variables, then the `lm()` is still the one to use. The only change is if you would like to check if the variable has a significant effect on the response, then you should first perform a one-way ANOVA:

```
M2  <- lm(cty ~ drv, data = mpg)
anova(M2)
```

```
## Analysis of Variance Table
##
## Response: cty
##            Df Sum Sq Mean Sq F value    Pr(>F)
## drv         2 1878.8  939.41  92.676 < 2.2e-16 ***
## Residuals 231 2341.5   10.14
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Again `tidy()` will put this into a data frame for you.

```
tidy(anova(M2))
```

```
## # A tibble: 2 x 6
##   term         df sumsq meansq statistic   p.value
##   <chr>     <int> <dbl>  <dbl>     <dbl>     <dbl>
## 1 drv           2 1879.  939.       92.7  2.82e-30
## 2 Residuals   231 2342.   10.1      NA    NA
```

Quiz Questions

8. Is there a statistically significant relationship between `cty` and `drv`?

Multiple linear regression

We can put the two variables in together:

```
M3  <- lm(cty ~ displ + drv, data = mpg)
```

Have a go with the following commands and work out what the formulae notation does:

```
M4  <- lm(cty ~ displ + drv + displ:drv, data = mpg)
M5  <- lm(cty ~ displ * drv, data = mpg)
M5  <- lm(cty ~ displ * drv * cyl, data = mpg)
M6  <- lm(cty ~ (displ + drv + cyl)^2, data = mpg)
M7  <- lm(cty ~ displ * (drv + cyl), data = mpg)
```

Multiple models (extension for your own time)

How do you fit multiple models?

Well, you could perfect the art of copy and paste, but you can also use the `map()` functions from the `purrr` package. The following codes show how to fit separate models with the predictors: `drv`, `displ`, and `cyl`. The first code gives a data frame of the coefficients, while the second one gives the summary statistics for each model. Have a look and see if you can work out what they are doing.

```
mpg %>%
  select(drv, displ, cyl) %>%
  map(~lm(cty ~ .x, data = mpg)) %>%
  map_df(tidy, .id = "predictor")
```

```
## # A tibble: 7 x 6
##   predictor term        estimate std.error statistic   p.value
##   <chr>     <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 drv       (Intercept)   14.3       0.314    45.7   1.19e-117
## 2 drv       .xf            5.64      0.440    12.8   9.88e- 29
## 3 drv       .xr           -0.250     0.710    -0.352 7.25e-  1
## 4 displ     (Intercept)   26.0       0.482    53.9   3.30e-133
## 5 displ     .x            -2.63      0.130   -20.2   4.74e- 53
## 6 cyl       (Intercept)   29.4       0.627    46.9   2.50e-120
## 7 cyl       .x            -2.13      0.103   -20.7   1.07e- 54
```

```
mpg %>%
  select(drv, displ, cyl) %>%
  map(~lm(cty ~ .x, data = mpg)) %>%
  map_df(glance, .id = "predictor")
```

```
## # A tibble: 3 x 13
##   predictor r.squared adj.r.sq~1 sigma stati~2  p.value    df logLik   AIC   BIC
##   <chr>         <dbl>      <dbl> <dbl>   <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1 drv           0.445      0.440  3.18    92.7 2.82e-30     2  -602. 1211. 1225.
## 2 displ         0.638      0.636  2.57   408.  4.74e-53     1  -552. 1109. 1120.
## 3 cyl           0.649      0.648  2.53   429.  1.07e-54     1  -548. 1102. 1112.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>, and
## #   abbreviated variable names 1: adj.r.squared, 2: statistic
```

Model selection

Backward selection

Next, we will set up the full model and the smallest model. As there is curvature in the dataset, we will use the log transform of `cty`.

```
full  <- log(cty) ~ (cyl + displ + drv)^2
smallest  <- log(cty) ~ 1
```

Quiz Questions

9. What is the model formula for the smallest model?

Now we fit the full model:

```
backwards.lm  <- lm(full, data = mpg)
```

Now we check if a term can be removed from the full model:

```
drop1(backwards.lm, test = "F")
```

```
## Single term deletions
##
## Model:
## log(cty) ~ (cyl + displ + drv)^2
##           Df Sum of Sq    RSS     AIC F value  Pr(>F)
## <none>                 3.0498 -995.62
## cyl:displ  1  0.078322 3.1281 -991.68  5.7526 0.01728 *
## cyl:drv    2  0.067623 3.1174 -994.49  2.4834 0.08576 .
## displ:drv  2  0.026184 3.0760 -997.62  0.9616 0.38387
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The first argument of `drop1()` is the linear model and the second argument, `test`, is where we define the test statistic to be used.

Now we remove the term with the largest P-value:

```
backwards.lm  <- update(backwards.lm, .~. - displ:drv)
```

Continue the process until you get the smallest possible model without removing significant terms.

Quiz Questions

10. What is the final model?

Forward selection

This time we start with the smallest model:

```
forwards.lm  <- lm(smallest, data = mpg)
```

Check if we can add a term to the smallest model. We need to give `add1()` command the possible terms to consider by giving the full model as the `scope`.

```
add1(forwards.lm, scope = full, test = "F")
```

```
## Single term additions
##
## Model:
## log(cty) ~ 1
```

```
##        Df Sum of Sq     RSS     AIC F value    Pr(>F)
## <none>              14.4654 -649.35
## cyl     1   10.0063  4.4592 -922.72 520.603 < 2.2e-16 ***
## displ   1    9.7125  4.7529 -907.80 474.089 < 2.2e-16 ***
## drv     2    6.6599  7.8055 -789.72  98.549 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

and then add it. In this case we add `cyl` as it has the largest F value.

```
forwards.lm  <- update(forwards.lm,.~. + cyl)
```

Continue adding terms until no significant terms are missing.

Quiz Questions

11. What is the final model?

Stepwise selection

Start with the smallest model.

```
step.lm  <- lm(smallest, data = mpg)
```

Check if can add a term to the smallest model.

```
add1(step.lm, scope = full, test = "F")
```

```
## Single term additions
##
## Model:
## log(cty) ~ 1
##        Df Sum of Sq     RSS     AIC F value    Pr(>F)
## <none>              14.4654 -649.35
## cyl     1   10.0063  4.4592 -922.72 520.603 < 2.2e-16 ***
## displ   1    9.7125  4.7529 -907.80 474.089 < 2.2e-16 ***
## drv     2    6.6599  7.8055 -789.72  98.549 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Add in a term with largest F value. (We will use a cutoff for adding of 0.1)

```
step.lm  <- update(step.lm, .~. + cyl)
```

Check if can remove any term, *i.e*, is any term not significant. For the drop stage we use a cutoff of 0.05.

```
drop1(step.lm, test = "F")
```

```
## Single term deletions
##
## Model:
## log(cty) ~ cyl
##        Df Sum of Sq     RSS     AIC F value    Pr(>F)
## <none>               4.4592 -922.72
## cyl     1    10.006 14.4654 -649.35   520.6 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Continue adding and dropping until you get the smallest possible model with all significant terms.

Quiz Questions

12. What is the final model?

Automatic

Backward selection can be done automatically using the `step()` command based on AIC.

```
backward.auto.lm  <- lm(full, data = mpg)
backward.auto.lm  <- step(backward.auto.lm, direction = "backward")
```

```
## Start:  AIC=-995.62
## log(cty) ~ (cyl + displ + drv)^2
##
##             Df Sum of Sq    RSS     AIC
## - displ:drv  2  0.026184 3.0760 -997.62
## <none>                   3.0498 -995.62
## - cyl:drv    2  0.067623 3.1174 -994.49
## - cyl:displ  1  0.078322 3.1281 -991.68
##
## Step:  AIC=-997.62
## log(cty) ~ cyl + displ + drv + cyl:displ + cyl:drv
##
##             Df Sum of Sq    RSS     AIC
## - cyl:drv    2  0.042337 3.1183 -998.42
## <none>                   3.0760 -997.62
## - cyl:displ  1  0.116886 3.1929 -990.89
##
## Step:  AIC=-998.42
## log(cty) ~ cyl + displ + drv + cyl:displ
##
##             Df Sum of Sq    RSS     AIC
## <none>                   3.1183 -998.42
## - cyl:displ  1   0.09117 3.2095 -993.67
## - drv        2   0.92688 4.0452 -941.52
```

Check whether the final model here is the same as backward selection above for this data.

Quiz Questions

13. a) Perform forward selection using the `step()` command. Save your final model to `forward.auto.lm`.
    b) Perform stepwise selection using the `step()` command. Save your final model to `step.auto.lm`.

Putting it all together

Now lets summarise the model results obtained in this practical. In particular, we are going to list the estimated coefficients for each of the final models.

```
list(forwards = forwards.lm,
     backwards = backwards.lm,
     step = step.lm,
     forward_auto = forward.auto.lm,
     backward_auto = backward.auto.lm,
     step_auto = step.auto.lm) %>%
  map_df(broom::tidy, .id = "Model") %>%
  select(Model, term, estimate) %>%
  pivot_wider(names_from = Model, values_from = estimate)
```

Cross validation

A code template for performing cross-validation for a desired number of k-folds is given below.

Challenge

1. Complete the code below to compute the MSE for each cross-validation run.
2. Write the code to calculate CV error across all cross-validation runs.
3. Perform 10-fold cross validation for the following models:

- `log(cyl) ~ cyl + drv + displ + cyl:displ`
- `log(cty) ~ cyl + displ + drv + cyl:displ + displ:drv + cyl:drv`

4. Compare the CV error for the two models.

```
library(modelr)
#' Get CV error
#'
#' @param model_formula = formula of linear model
#' @param data = data frame
#' @param k = number of folds
#'
#' @return CV error
cv_lm  <- function(model_formula, data, k = 5){
  # Split the data, this will create the cross validation folds.
  data_cv  <- crossv_kfold(data, k = k)
  # Fit the model to each training set
  models  <- map(data_cv$train, ~lm(model_formula, data = .))
  # Get prediction error - gasp in awe - a function in a function.
  get_pred  <- function(model, test_data){
    data  <- as.data.frame(test_data)
    # Add predictions is a nice function from modelr, go suss the help file!
    pred  <- add_predictions(data, model)
    return(pred)
  }
  # map2 is like the map function, but accepts 2 inputs instead of 1
  pred  <- map2_df(models, data_cv$test, get_pred, .id = "Run")

  # Get MSE
  # INSERT code to calculate MSE for pred below:
  #MSE  <- ...

  # Get CV
  # INSERT code to calculate CV error below:
  # CV  <- ...
  return(CV)
}
```