Some linear models

Søren Højsgaard

2024-09-06

Contents

1 Packages needed + some settings

These packages must be installed from CRAN first:

```
library(doBy)
library(broom)
library(ggplot2)
library(patchwork)
library(pander)
library(kableExtra)
library(caracas)
```

Sometimes, we need at version of a package which is not yet on CRAN. One example is the `doBy` package, where we need the development version on github. To fetch this version we need the `remotes` package to be installed:

```r
##install.packages("remotes")
##remotes::install_github("hojsgaard/doBy")
```

2 Linear models in general

For each observable:

- $y = m + e$
- $m = b_1 x_1 + b_2 x_2 + ... + b_p x_p$
- $e \sim N(0, \sigma^2)$ independent

More detailed

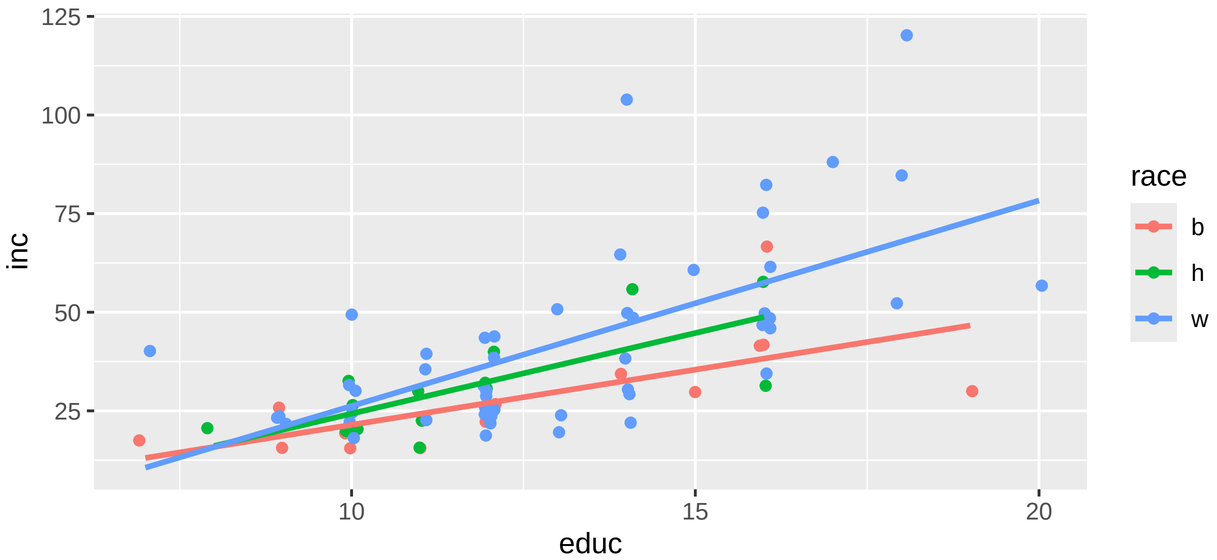
- $y_i = m_i + e_i$
- $m_i = b_1 x_{i1} + b_2 x_{i2} + ... + b_p x_{ip}$
- $e_i \sim N(0, \sigma^2)$ independent

3 The income data

```r
dat <- doBy::income
dat |> head()
```

```
##   inc educ race
## 1  16   10    b
## 2  18    7    b
## 3  26    9    b
## 4  16   11    b
## 5  34   14    b
## 6  22   12    b
```

```r
pl0 <- dat |> ggplot(aes(x=educ, y=inc, color=race)) + geom_jitter(width=0.1)
pl0 + geom_smooth(method="lm", se=FALSE)
```

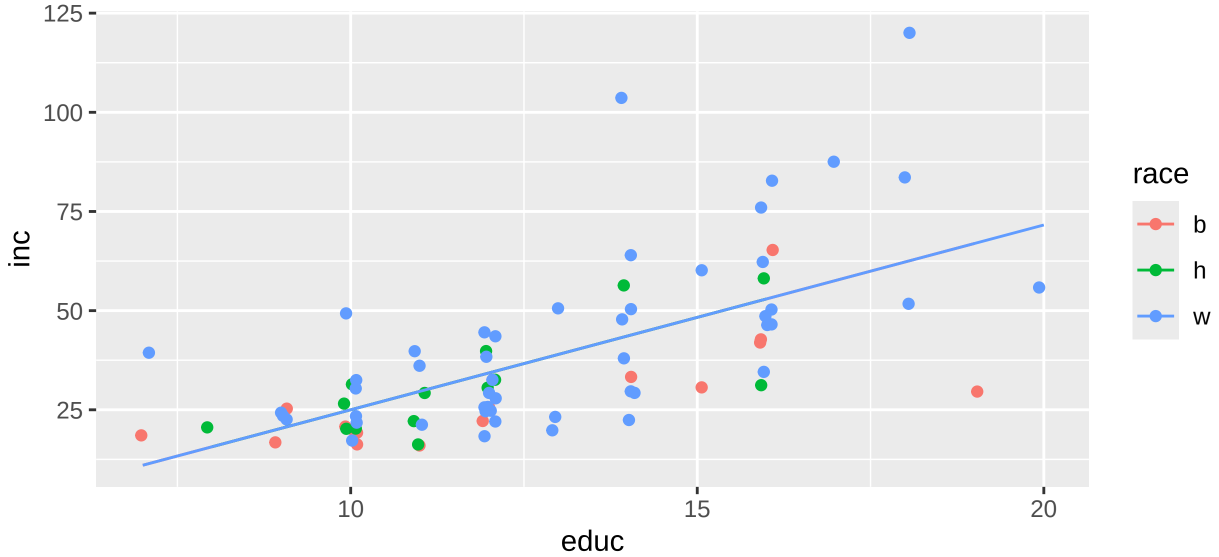


4 Four different models

4.1 Model 1 - Simple linear regression

Income grows linearly with years of education; no effect of ethnicity (Simple linear regression)

```r
mm1 <- lm(inc ~ educ, data=dat)
pl0 + geom_line(aes(y=fitted(mm1)))
```



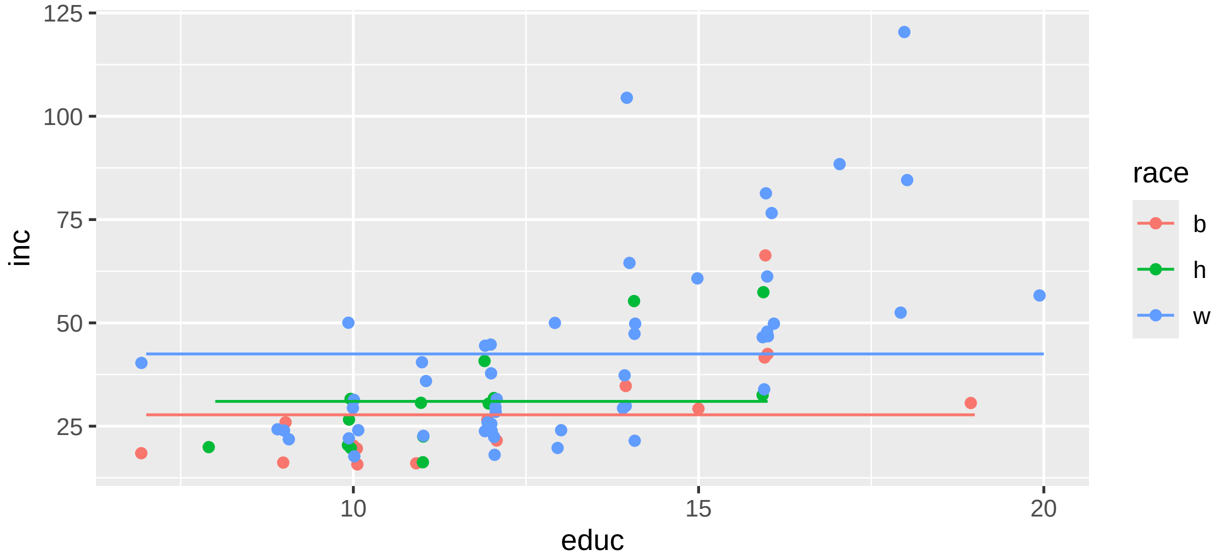

```r
mm1 |> tidy()
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   -21.6      8.08      -2.67 9.20e- 3
## 2 educ            4.66     0.622      7.50 8.85e-11
```

4.2 Model 2 - one-way ANOVA

Income is constant across all levels of education within ethnic groups

```r
mm2 <- lm(inc ~ race, data=dat)
pl0 + geom_line(aes(y=fitted(mm2)))
```




```r
mm2 |> tidy()
```

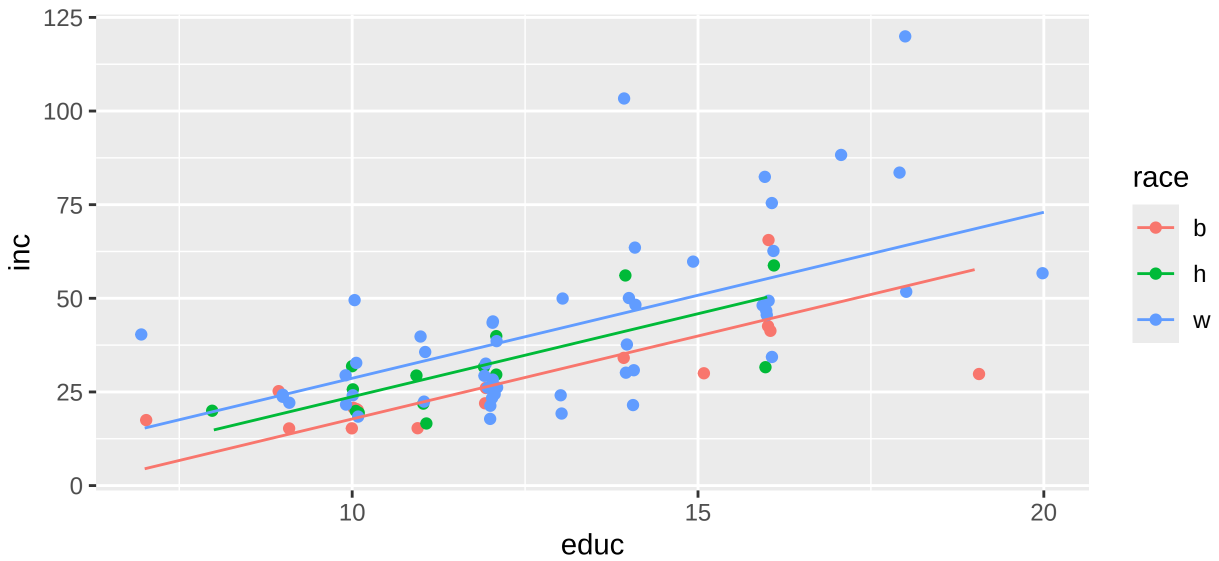
```
## # A tibble: 3 x 5
##   term        estimate std.error statistic     p.value
##   <chr>          <dbl>     <dbl>     <dbl>       <dbl>
## 1 (Intercept)    27.8       4.97      5.59 0.000000337
```

```
## 2 raceh           3.25      7.27      0.447 0.656
## 3 racew          14.7       5.71      2.58  0.0118
```

4.3 Model 3 - ANCOVA

Income grows linearly with years of education but with offset depending on ethnicity

```
mm3 <- lm(inc ~ race + educ, data=dat)
pl0 + geom_line(aes(y=fitted(mm3)))
```



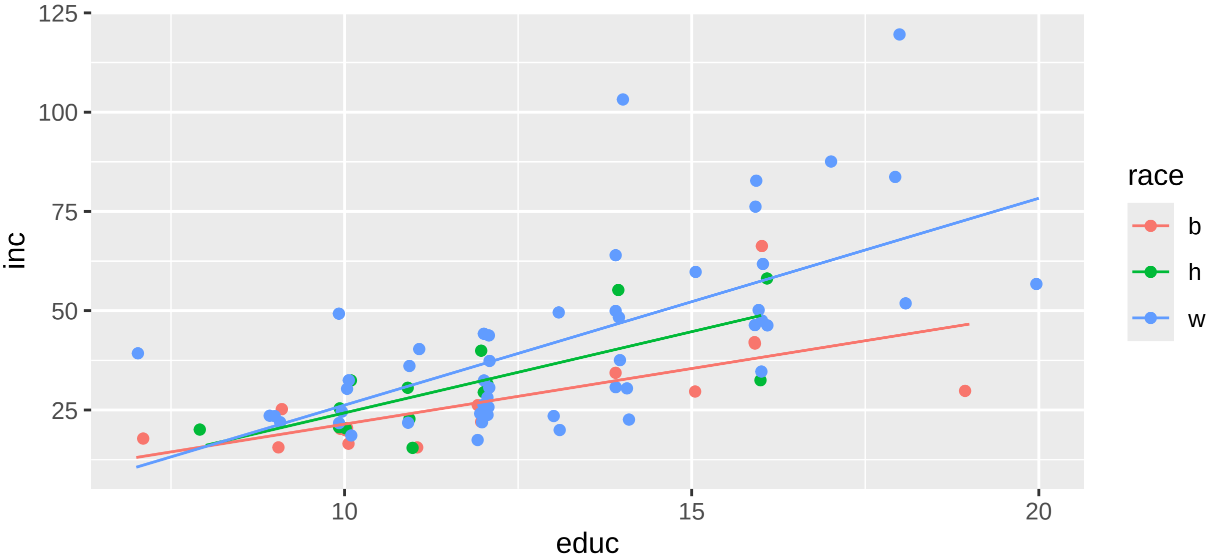

```
mm3 |> tidy()
```

```
## # A tibble: 4 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   -26.5      8.51      -3.12 2.57e- 3
## 2 raceh           5.94     5.67       1.05 2.98e- 1
## 3 racew          10.9      4.47       2.43 1.74e- 2
## 4 educ            4.43     0.619      7.16 4.42e-10
```

4.4 Model 4 - ANCOVA with interaction

Income grows linearly with years of education but with offset and slope depending on ethnicity

```
mm4 <- lm(inc ~ race * educ, data=dat)
pl0 + geom_line(aes(y=fitted(mm4)))
```




```
mm4 |> tidy()
```

```
## # A tibble: 6 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)    -6.54      15.0    -0.436  0.664
## 2 raceh         -10.1       26.5    -0.380  0.705
## 3 racew         -19.3       18.3    -1.06   0.294
## 4 educ            2.80       1.18    2.37   0.0205
## 5 raceh:educ      1.29       2.19    0.588  0.558
## 6 racew:educ      2.41       1.42    1.70   0.0933
```

5 Model fits

A summary of how well the models fit the data is given by the residual standard deviation

$$\sqrt{\frac{1}{n-p}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

and by the coefficient of determination R^2 which is the squared correlation between the observed and fitted values.

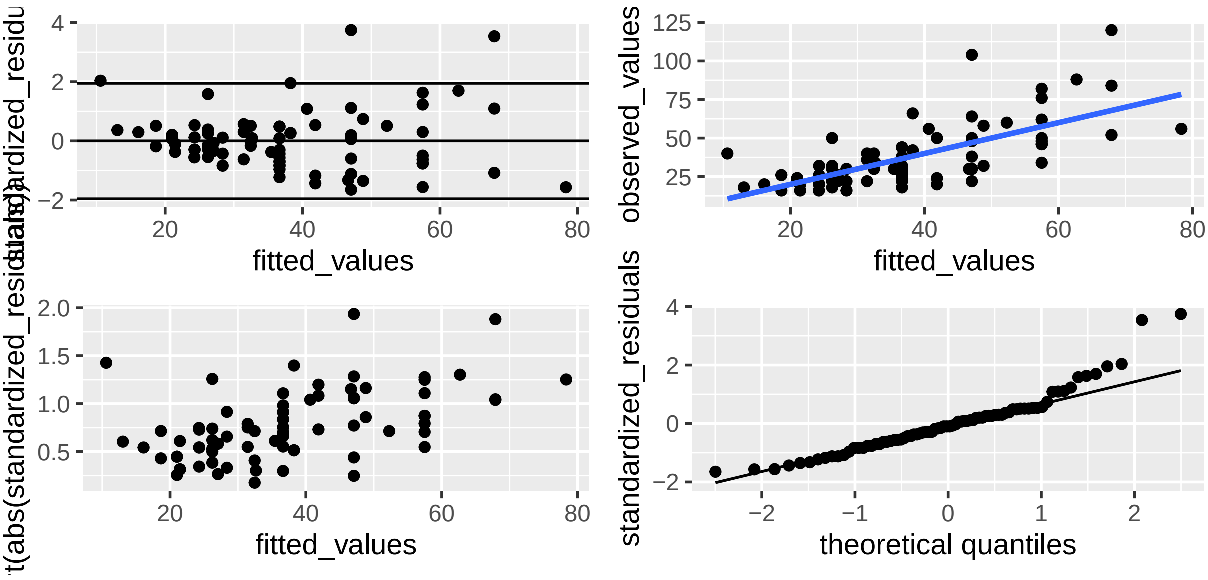
$$cor(y, \hat{y})^2$$

```
mm4 |> glance()
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic       p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>         <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     0.482         0.448  15.4      13.8       1.62e-9     5  -329.  672.  689.
## # i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

6 Diagnostic plots

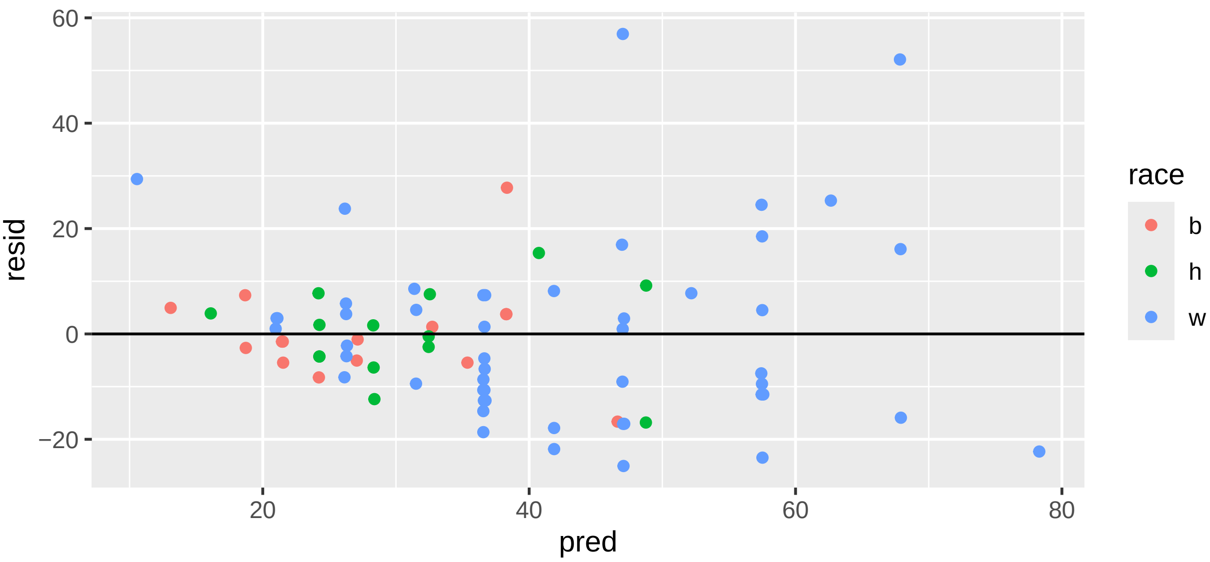
```
mm4 |> doBy::plot_lm()
```




```
dat <- modelr::add_residuals(dat, mm4)    ## observed - fitted
dat <- modelr::add_predictions(dat, mm4) ## Estimated mean m
dat |> head()
```

```
##   inc educ race resid pred
## 1  16   10    b -5.45 21.5
## 2  18    7    b  4.94 13.1
## 3  26    9    b  7.35 18.7
## 4  16   11    b -8.25 24.3
## 5  34   14    b  1.35 32.6
## 6  22   12    b -5.05 27.1
```

```
dat |> ggplot(aes(pred, resid, color=race)) + geom_jitter(width=0.1) +
  geom_hline(yintercept=0)
```

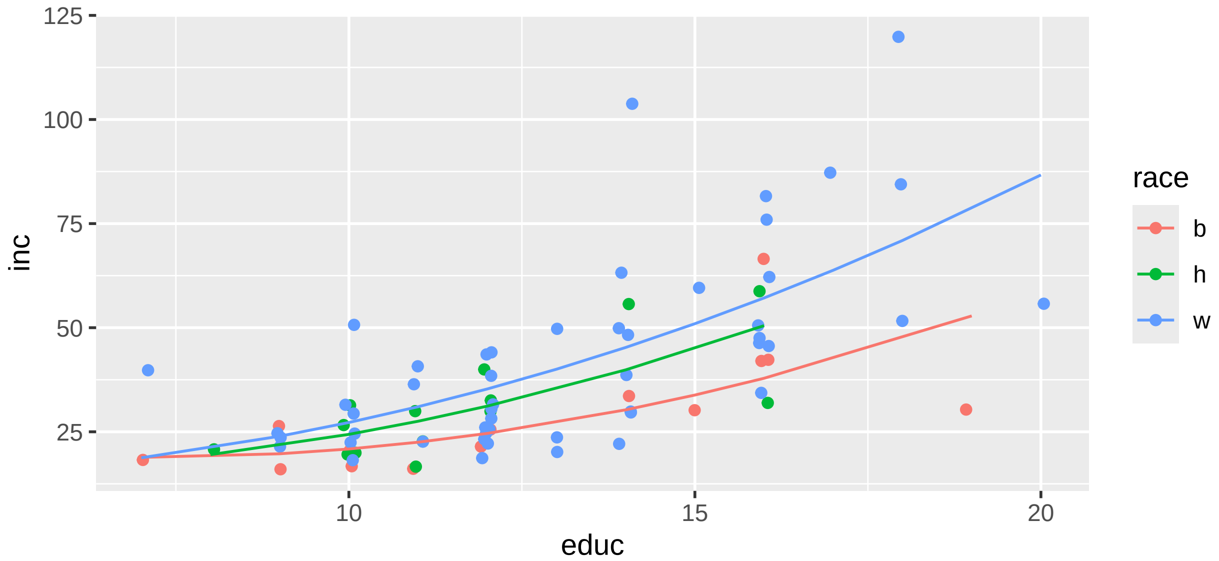


7 Other linear models

7.1 Polynomial regression

Perhaps the relationship between income and education is not linear but quadratic (not too relevant here, but could be in other cases).

```r
mm5 <- lm(inc ~ race * educ + I(educ^2), data=dat)
pl0 + geom_line(aes(y=fitted(mm5)))
```

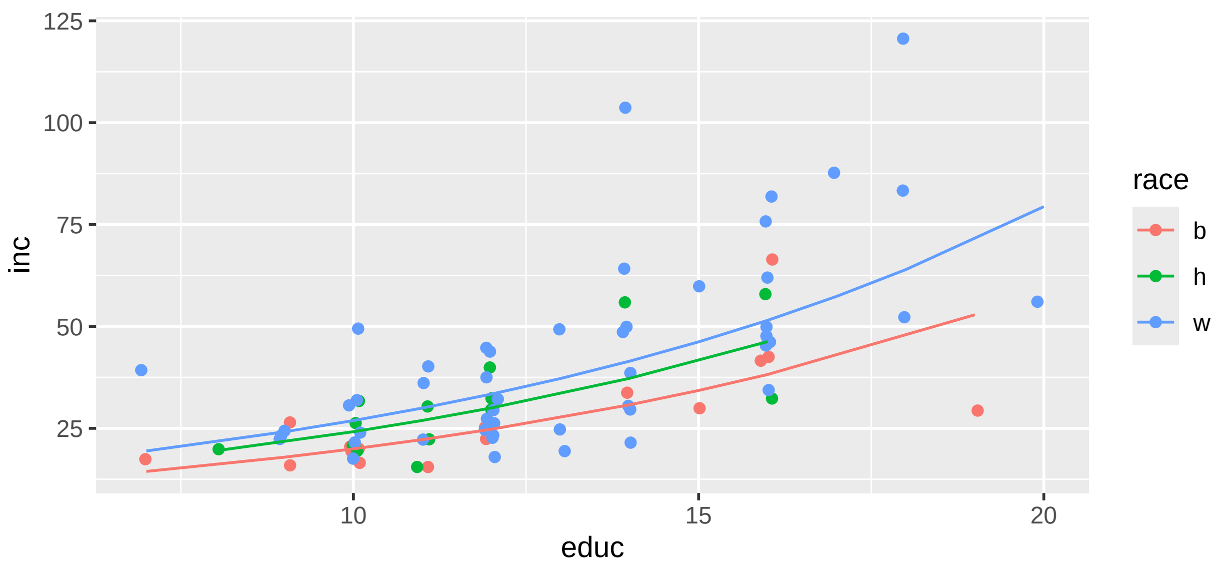



```r
mm5 |> tidy()
```

```
## # A tibble: 7 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   31.0      32.9       0.941   0.350
## 2 raceh        -11.7      26.4      -0.443   0.659
## 3 racew        -15.2      18.5      -0.820   0.415
## 4 educ          -3.41      4.99     -0.682   0.497
## 5 I(educ^2)      0.240     0.188     1.28    0.205
## 6 raceh:educ     1.52      2.19      0.695   0.490
## 7 racew:educ     2.15      1.43      1.51    0.135
```

7.2 Logarithmic regression

```r
mm6 <- lm(log(inc) ~ race + educ, data=dat)
pl0 + geom_line(aes(y=exp(fitted(mm6))))
```



```
mm6 |> tidy()
```

```
## # A tibble: 4 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    1.91     0.182      10.5  1.96e-16
## 2 raceh          0.192    0.121       1.58 1.19e- 1
## 3 racew          0.299    0.0957      3.12 2.57e- 3
## 4 educ           0.108    0.0133      8.16 5.33e-12
```

8 A digression into R - lists and functions

The first few models above are collected onto a lists

```
ml <- list(mm1, mm2, mm3)
```

An element (here a model) can be extracted from the list using `[[`

```
ml[[1]]
```

```
##
## Call:
## lm(formula = inc ~ educ, data = dat)
##
## Coefficients:
## (Intercept)         educ
##      -21.59         4.66
```

A list can have named components (often a good idea):

```
ml <- list(model1=mm1, model2=mm2, model3=mm3)
```

In this case, an element can also be extracted as

```
ml$model1
```

```
##
## Call:
## lm(formula = inc ~ educ, data = dat)
##
## Coefficients:
## (Intercept)         educ
##      -21.59         4.66
```

```
ml[["model1"]]
```

```
##
## Call:
## lm(formula = inc ~ educ, data = dat)
##
## Coefficients:
## (Intercept)         educ
##      -21.59         4.66
```

An element can be added to a list with

```
ml$model4 <- mm4
## ml[[4]] <- mm4 ## alternative
```

Actually, under the hood, a dataframe is a list. We can extract the first two columns of the `income` data with

```
income2 <- income[, 1:2]
income2 |> head()
```

```
##   inc educ
## 1  16   10
## 2  18    7
## 3  26    9
## 4  16   11
## 5  34   14
## 6  22   12
```

We frequently what to do something on each element of a list. Here we can use `lapply` and `sapply` which will apply a function to each element of the list. For example

```
lapply(income2, mean)
```

```
## $inc
## [1] 37.5
##
## $educ
## [1] 12.7
```

```
sapply(income2, mean)
```

```
##  inc educ
## 37.5 12.7
```

In a more complicated setting, suppose we want to calculate the mean of a squared variable. To do this we can create a custom function

```
mean_squared <- function(x) mean(x^2) ## short
mean_squared <- function(x){          ## readable
  return(mean(x^2))
}
mean_squared(income2$inc)
```

```
## [1] 1830
```

```
mean_squared(income2$educ)
```

```
## [1] 169
```

We can use this function in `lapply` and `sapply` as well

```
sapply(income2, mean_squared)
```

```
##  inc educ
## 1830  169
```

and we can even do so on the fly:

```
sapply(income2, function(x){
  return(mean(x^2))
})
```

```
##  inc educ
## 1830  169
```

9 Model evaluation

One approach is to calculate the root mean squared error (RMSE) for each model.

$$RMSE <- \sqrt{\frac{1}{n-p}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

```r
sapply(ml, function(x) modelr::rmse(x, dat))
```

```
## model1 model2 model3 model4
##   15.7   19.5   15.1   14.8
```

Suggests that Model 4 is the best model.

However, there is a problem with this approach as it can be misleading. We evaluate the model on the same data the model was fitted on. We should evaluate the model on new data. We can do this by splitting the data into a training and a test set.

```r
set.seed(2024)
i <- sample(nrow(dat), size=0.8*nrow(dat))
i |> head()
```

```
## [1] 66 37 45 60 17 32
```

```r
train <- dat[i,]
test <- dat[-i,]
```

We can refit a model to a training dataset and evaluate it on a test dataset

```r
mm1_train <- lm(inc ~ educ, data=train) ## or
mm1_train <- update(mm1, data=train)
modelr::rmse(mm1_train, test)
```

```
## [1] 12.9
```

```r
modelr::rmse(mm1_train, train)
```

```
## [1] 16.3
```

We can do this for all models:

```r
ml_train <- sapply(ml, function(x) update(x, data=train))
sapply(ml_train, function(x) modelr::rmse(x, train))
```

```
## model1 model2 model3 model4
##   16.3   20.0   15.5   15.1
```

```r
sapply(ml_train, function(x) modelr::rmse(x, test))
```

```
## model1 model2 model3 model4
##   12.9   18.0   13.5   14.2
```

On the training data, Model 4 appears to be the best model. However, on the test data, Model 3 is the best model. This is a common situation.

We can create another partiotioning of data:

```r
set.seed(2027)
i <- sample(nrow(dat), size=0.8*nrow(dat))
i |> head()
```

```
## [1] 72 71 56 15 59 22
```

```r
train <- dat[i,]
test <- dat[-i,]
```

```r
ml_train <- sapply(ml, function(x) update(x, data=train))
sapply(ml_train, function(x) modelr::rmse(x, train))
```

```
## model1 model2 model3 model4
##   16.3   19.8   15.4   14.8
```

```r
sapply(ml_train, function(x) modelr::rmse(x, test))
```

```
## model1 model2 model3 model4
##   12.7   18.3   13.9   15.7
```

This time Model 4 is the best model on the training data but Model 1 is the best model on the test data.

We return to this issue later in the course under the name cross validation.

10 Looking into the models - optional*

We illustrate in detail what the fitted values really are for the four models.

We consider only a small subset of data to make it easier to see what is going on.

```r
dat0
```

```
##     inc educ race   resid pred
## 1    16   10    b  -5.453 21.5
## 2    18    7    b   4.944 13.1
## 3    26    9    b   7.346 18.7
## 4    16   11    b  -8.251 24.3
## 17   32   16    h -16.815 48.8
## 18   16   11    h -12.372 28.4
## 19   20   10    h  -4.283 24.3
## 20   58   16    h   9.185 48.8
## 31   30   14    w -17.064 47.1
## 32   48   14    w   0.936 47.1
## 33   40    7    w  29.402 10.6
## 34   84   18    w  16.098 67.9
```

10.1 Model 1 - Simple linear regression

```r
coef(model)
```

```
## (Intercept)        educ
##      -14.16        4.01
```

$$m = Xb = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \\ 1 & x_5 \\ 1 & x_6 \\ 1 & x_7 \\ 1 & x_8 \\ 1 & x_9 \\ 1 & x_{10} \\ 1 & x_{11} \\ 1 & x_{12} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} b_1 + b_2 x_1 \\ b_1 + b_2 x_2 \\ b_1 + b_2 x_3 \\ b_1 + b_2 x_4 \\ b_1 + b_2 x_5 \\ b_1 + b_2 x_6 \\ b_1 + b_2 x_7 \\ b_1 + b_2 x_8 \\ b_1 + b_2 x_9 \\ b_1 + b_2 x_{10} \\ b_1 + b_2 x_{11} \\ b_1 + b_2 x_{12} \end{bmatrix}$$

10.2 Model 2 - one-way ANOVA

```r
coef(model)
```

```
## (Intercept)       raceh       racew
##        19.0        12.5        31.5
```

$$
m = Xb = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_1 \\ b_1 \\ b_1 \\ b_1 + b_2 \\ b_1 + b_2 \\ b_1 + b_2 \\ b_1 + b_2 \\ b_1 + b_3 \\ b_1 + b_3 \\ b_1 + b_3 \\ b_1 + b_3 \end{bmatrix}
$$

10.3 Model 3 - ANCOVA

```r
coef(model)
```

```
## (Intercept)       raceh       racew        educ
##      -12.39       -1.07       17.93        3.39
```

$$
m = Xb = \begin{bmatrix} 1 & 0 & 0 & x_1 \\ 1 & 0 & 0 & x_2 \\ 1 & 0 & 0 & x_3 \\ 1 & 0 & 0 & x_4 \\ 1 & 1 & 0 & x_5 \\ 1 & 1 & 0 & x_6 \\ 1 & 1 & 0 & x_7 \\ 1 & 1 & 0 & x_8 \\ 1 & 0 & 1 & x_9 \\ 1 & 0 & 1 & x_{10} \\ 1 & 0 & 1 & x_{11} \\ 1 & 0 & 1 & x_{12} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} b_1 + b_4 x_1 \\ b_1 + b_4 x_2 \\ b_1 + b_4 x_3 \\ b_1 + b_4 x_4 \\ b_1 + b_2 + b_4 x_5 \\ b_1 + b_2 + b_4 x_6 \\ b_1 + b_2 + b_4 x_7 \\ b_1 + b_2 + b_4 x_8 \\ b_1 + b_3 + b_4 x_9 \\ b_1 + b_3 + b_4 x_{10} \\ b_1 + b_3 + b_4 x_{11} \\ b_1 + b_3 + b_4 x_{12} \end{bmatrix}
$$

10.4 Model 4 - ANCOVA with interaction

```r
coef(model)
```

```
## (Intercept)       raceh       racew        educ  raceh:educ  racew:educ
##       26.40      -58.03      -19.71       -0.80        5.56        4.11
```

$$
m = Xb = \begin{bmatrix} 1 & 0 & 0 & x_1 & 0 & 0 \\ 1 & 0 & 0 & x_2 & 0 & 0 \\ 1 & 0 & 0 & x_3 & 0 & 0 \\ 1 & 0 & 0 & x_4 & 0 & 0 \\ 1 & 1 & 0 & x_5 & x_5 & 0 \\ 1 & 1 & 0 & x_6 & x_6 & 0 \\ 1 & 1 & 0 & x_7 & x_7 & 0 \\ 1 & 1 & 0 & x_8 & x_8 & 0 \\ 1 & 0 & 1 & x_9 & 0 & x_9 \\ 1 & 0 & 1 & x_{10} & 0 & x_{10} \\ 1 & 0 & 1 & x_{11} & 0 & x_{11} \\ 1 & 0 & 1 & x_{12} & 0 & x_{12} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix} = \begin{bmatrix} b_1 + b_4 x_1 \\ b_1 + b_4 x_2 \\ b_1 + b_4 x_3 \\ b_1 + b_4 x_4 \\ b_1 + b_2 + b_4 x_5 + b_5 x_5 \\ b_1 + b_2 + b_4 x_6 + b_5 x_6 \\ b_1 + b_2 + b_4 x_7 + b_5 x_7 \\ b_1 + b_2 + b_4 x_8 + b_5 x_8 \\ b_1 + b_3 + b_4 x_9 + b_6 x_9 \\ b_1 + b_3 + b_4 x_{10} + b_6 x_{10} \\ b_1 + b_3 + b_4 x_{11} + b_6 x_{11} \\ b_1 + b_3 + b_4 x_{12} + b_6 x_{12} \end{bmatrix}
$$