Tutorial 2b: regression

Tian Zheng

September 20, 2016

In this example, we are going to use the dataset `state.x77` that comes with standard `R` installation. It is a data set about the 50 states of united states.

Type `help(state.x77)` in your console window to read more about this data set.

```
library(datasets)
statedata=as.data.frame(state.x77)
```

There are 8 variables in this data set. The "individuals" of this data set are the 50 states.

```
Population: population estimate as of July 1, 1975

Income: per capita income (1974)

Illiteracy:illiteracy (1970, percent of population)

Life Exp: life expectancy in years (1969-71)

Murder: murder and non-negligent manslaughter rate per 100,000 population (1976)

HS Grad: percent high-school graduates (1970)

Frost: mean number of days with minimum temperature below freezing (1931-1960) in capital or large city

Area: land area in square miles
```

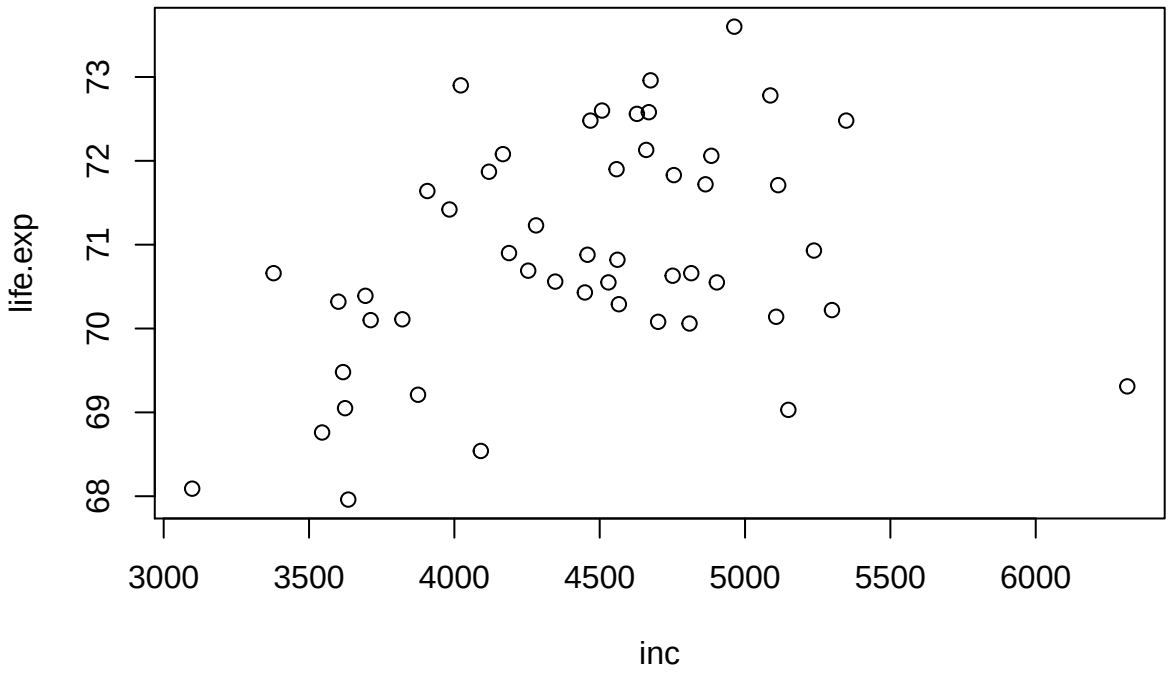
First we can modify the column names (the variable names) for easier use. The function `colnames` can be used to retrieve the colume names or assign new names.

```
colnames(statedata)=c("popu", "inc", "illit", "life.exp", "murder", "hs.grad", "frost", "area")
```

Make a scatterplot.

For this example, let's look at the association between life expectancy (`life.exp`) and income (`inc`). The scatterplot below shows a positive association between these two variables.

```
plot(life.exp~inc, data=statedata)
```



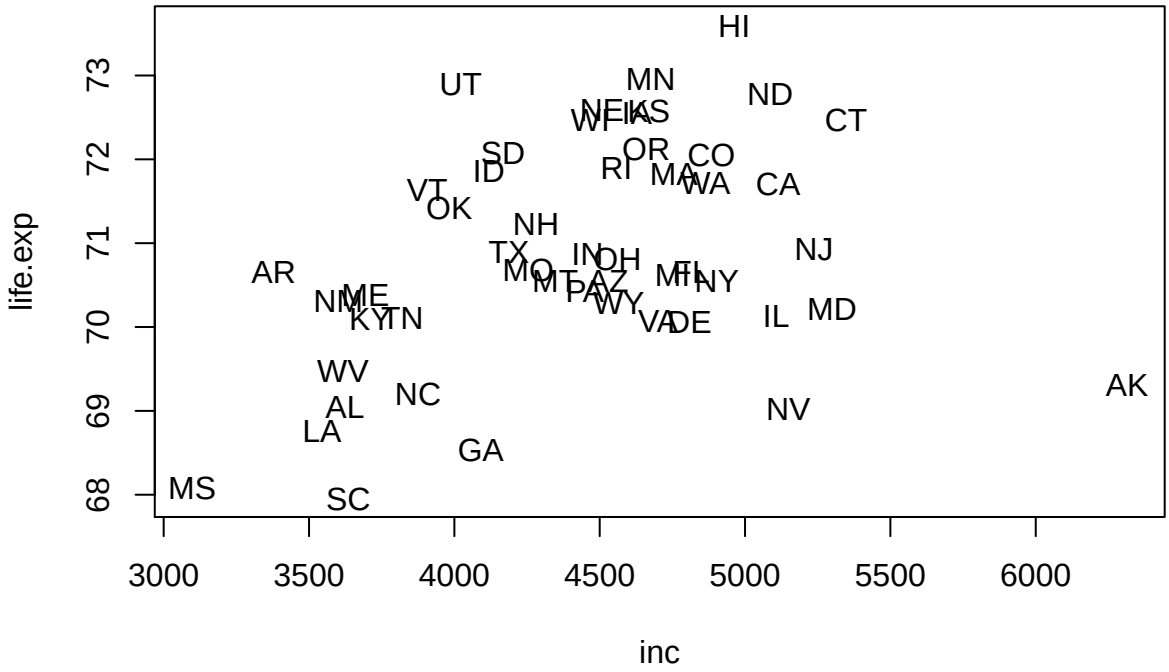
We can compute the correlation between these two variables. The value of the correlation indicates a weak positive linear association.

```
cor(statedata[,"life.exp"], statedata[,"inc"])
```

```
## [1] 0.3402553
```

There is one observation that is far away from the rest of the points. We would like to know which state is corresponding to that point. Let's add state abbreviations to the plot.

```
plot(life.exp~inc, data=statedata, type="n")
text(life.exp~inc, data=statedata, state.abb)
```



Fit a simple linear regression

Now let's fit the following linear regression model between Y (`life.exp`) and X (`inc`)

$$Y = \beta_0 + \beta_1 X + \varepsilon.$$

Here β_0 and β_1 are the regression coefficients of the model and ε represents independent random errors.

Fitting a linear regression is to derive

$$\hat{Y} = b_0 + b_1 X.$$

Here $\hat{Y}$ is a fitted prediction for the observed life expectancies. The difference $Y - \hat{Y}$ is the prediction error, or residual.

b_0 and b_1 are estimates for the regression coefficients. They are identified via the least square regression method that minimizes

$$\sum_{i=1}^{n} (Y - \hat{Y})^2,$$

i.e., the sum of squared prediction errors.

```
model1=lm(life.exp~inc, data=statedata)
model1
```

```
##
## Call:
## lm(formula = life.exp ~ inc, data = statedata)
##
## Coefficients:
## (Intercept)          inc
##   6.758e+01    7.433e-04
```

In the above output, the intercept is b_0 and the coeffient under the column (`inc`) is b_1–the slope. The slope is estimated to be 7.4×10^{-4}. The maganitude of this value does not mean that the effect of income on life expectance is very small. This maganitude is decided by the maganitude of the X variable and Y variable.

Sampling uncertainty in regression analysis

Consider a population of 50 states and we identify the true regression line in this population. Here the function `abline` add a straight line to an existing plot.

```
plot(life.exp~inc, data=statedata,
      xlab="Life Expectancy", ylab="Income")
abline(model1)
```

Now we can consider 4 random samples. We use a `for` loop to run an identical regression analysis on 4 randomly selected samples.

Within the loop, we will implement the following steps for each repetition.

- Step 1: randomly select 10 states using the `sample` function of `R`.
- Step 2: run least square regression on the selected states only
- Step 3: compute a 95% confidence band for the true regression line in the population using the sample.

[**Note**] The states are randomly selected in the following. Therefore, every time you run this file, you will produce different random samples that will give different estimated least regression lines.

```
par(mfrow=c(2,2)) # create a panel of four plotting areas

for(i in 1:4){
  ## Plot the population
  plot(life.exp~inc, data=statedata,
       xlab="Life Expectancy", ylab="Income",
       title=paste("Random sample", format(i)),
       ylim=c(min(life.exp), max(life.exp)+0.3))
  abline(model1)
  if(i==1){
    legend(3030, 74.2,
           pch=c(NA, NA, NA, 1, 16),
           lty=c(1, 1, 2, NA, NA),
           col=c(1, 2, 2, 1, 2),
           c("population truth", "sample estimate",
             "sample confidence band",
             "states", "sampled"),
           cex=0.7,
           bty="n"
           )
  }
  ## Select the sample
```

```
  selected.states=sample(1:50, 10)
  points(statedata[selected.states,"inc"],
         statedata[selected.states,"life.exp"], pch=16, col=2)
  ## Fit a regression line using the sample
  model.sel = lm(life.exp~inc, data=statedata[selected.states,])
  abline(model.sel, col=2)
  ## Make a confidence band.
  #### first calculate the width of the band, W.
  ww=sqrt(2*qf(0.95, 2, nrow(statedata)-2))
  #### generate plotting X values.
  plot.x<-data.frame(inc=seq(3000, 7000, 1))
  #### se.fit=T is an option to save
  #### the standard error of the fitted values.
  plot.fit<-predict(model.sel, plot.x,
                    level=0.95, interval="confidence",
                    se.fit=T)

  #### lines is a function to add connected lines
  #### to an existing plot.
  lines(plot.x$inc, plot.fit$fit[,1]+ww*plot.fit$se.fit,
        col=2, lty=2)
  lines(plot.x$inc, plot.fit$fit[,1]-ww*plot.fit$se.fit,
        col=2, lty=2)
}
```

```
## Warning in plot.window(...): "title" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "title" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in box(...): "title" is not a graphical parameter

## Warning in title(...): "title" is not a graphical parameter

## Warning in plot.window(...): "title" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "title" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in box(...): "title" is not a graphical parameter

## Warning in title(...): "title" is not a graphical parameter

## Warning in plot.window(...): "title" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "title" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in box(...): "title" is not a graphical parameter

## Warning in title(...): "title" is not a graphical parameter

## Warning in plot.window(...): "title" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "title" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in box(...): "title" is not a graphical parameter

## Warning in title(...): "title" is not a graphical parameter
```