## Regression Modeling

1. **Intro**

Lm(dependent vairable~independent variable,data)

1. **Residuals**

Distance b/w actual data(children’s heights) and estimates given by regression line(errors in estimates)

Fit<-Lm(child~parent,galton)

Summary(fit)

Mean(fit$residuals)

Cov(fit$residuals,galton$parent)

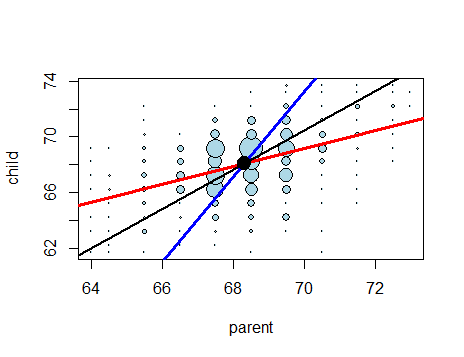
var(data)=var(estimate)+var(residuals) : shows that variance of estimate is always less that data

1. **Least Squares Estimation**

Regression line is line thru data which has “least” squared error.

Slope of regression line=correlation \* ratio of sd(outcomes to predictors)

To normalize data, subtract its mean and divide by sd.



Code for the same:

#plot the original Galton data points with larger dots for more freq pts

y <- galton$child

x <- galton$parent

freqData <- as.data.frame(table(galton$child, galton$parent))

names(freqData) <- c("child", "parent", "freq")

plot(as.numeric(as.vector(freqData$parent)),

as.numeric(as.vector(freqData$child)),

pch = 21, col = "black", bg = "lightblue",

cex = .07 \* freqData$freq, xlab = "parent", ylab = "child")

#original regression line, children as outcome, parents as predictor

abline(mean(y) - mean(x) \* cor(y, x) \* sd(y) / sd(x), #intercept

sd(y) / sd(x) \* cor(y, x), #slope

lwd = 3, col = "red")

#new regression line, parents as outcome, children as predictor

abline(mean(y) - mean(x) \* sd(y) / sd(x) / cor(y, x), #intercept

sd(y) / cor(y, x) / sd(x), #slope

lwd = 3, col = "blue")

#assume correlation is 1 so slope is ratio of std deviations

abline(mean(y) - mean(x) \* sd(y) / sd(x), #intercept

sd(y) / sd(x), #slope

lwd = 2)

points(mean(x), mean(y), cex = 2, pch = 19) #big point of intersection

1. **Residual Variation**

Variation after removing predictor.

Total Variation = Residual Variation + Regression Variation

mu<-mean(galton$child)

sTot <- sum((galton$child-mu)^2)

sRes <- deviance(fit) --Residual Variation, where fit<-lm(child~parent,galton)

sRes/sTot --Total Variation

1-sRes/sTot --Regression Variation

summary(fit)$r.squared --Regression Variation

cor(galton$parent,galton$child)^2 --Sample Corelation squared--same as regression variation

1. **Multivariable Regression**

When we perform a regression in one variable, such as lm(child ~ parent, galton), we get two coefficients, a slope and an intercept. The intercept is really the coefficient of a special regressor which has the same value, 1, at every sample. The function, lm, includes this regressor by default.

The regression in one variable given by lm(child ~ parent, galton) really involves two regressors, the variable, parent, and a regressor of all ones.

The regression line given by lm(child ~ parent, galton) goes through the point x=mean(parent), y=mean(child). We also showed that if we subtract the mean from each variable, the regression line goes through the origin, x=0, y=0, hence its intercept is zero. Thus, by subtracting the means, we eliminate one of the two regressors, the constant, leaving just one, parent. The coefficient of the remaining regressor is the slope.(It is called Gaussian Elimination).

The mean of a variable is equal to its regression against the constant, 1.

Ex: lm(child~1,galton) is same as mean(child)

The general technique is to pick one predictor and to replace all other variables by the residuals of their regressions against that one.

Ex: Volume ~ Girth – 1, where Volume is other variable, Girth is predictor

Code:

# Regress the given variable on the given predictor,

# suppressing the intercept, and return the residual.

regressOneOnOne <- function(predictor, other, dataframe){

# Point A. Create a formula such as Girth ~ Height -1

formula <- paste0(other, " ~ ", predictor, " - 1")

# Use the formula in a regression and return the residual.

resid(lm(formula, dataframe))

}

# Eliminate the specified predictor from the dataframe by

# regressing all other variables on that predictor

# and returning a data frame containing the residuals

# of those regressions.

eliminate <- function(predictor, dataframe){

# Find the names of all columns except the predictor.

others <- setdiff(names(dataframe), predictor)

# Calculate the residuals of each when regressed against the given predictor

temp <- sapply(others, function(other)regressOneOnOne(predictor, other, dataframe))

# sapply returns a matrix of residuals; convert to a data frame and return.

as.data.frame(temp)

}

1. **Multivar examples**

all <- lm(Fertility ~ ., swiss)

to refit the model with different reference group:

relevel(dataframe.variable,value)

ex: to set the reference level as C: spray2 <- relevel(InsectSprays$spray,"C")