Simple Linear Regression Model

Marco Antonio Florenzano Mollinetti[1]
[1]**University of Tsukuba, Systems Optimization Laboratory**
mollinetti@syou.cs.tsukuba.ac.jp

Before we Begin

- Go to the github repo:
  - https://github.com/Mollinetti/Experiment-Design-R
  - Download the script for this class! (in the 'scripts' folder, class_5.r!)
- Run the snippet at the beginning to load/install the required libraries

Agenda

- Introduction
- Linear Regression
- Confidence intervals of LR
- Assumptions
- Goodness of Fit
- Simple Polynomial Regression
- Statistical Learning with LR (bonus round)

Introduction

- Linear regression is used to predict a quantitative response
- Although simple, still very effective up until today
- Linear model to fit a line to data (although we can increase the flexibility to fit curves)

Introduction

- Variables are now named:
  - Predictors:
    - Independent variables
    - What we want to use to predict
    - Usually called X
  - Response:
    - Dependent variable
    - What we predict
    - Usually called Y

Introduction

- Linear regression is used for two tasks:
  - Inference
  - Prediction

Introduction

- Linear regression is used for two tasks:
  - Inference
    - Understand the way Y is affected by changes in X
    - What is the relationship between each X and Y?
    - Can the relationship between Y and X be summarized using a linear equation?
  - Prediction

Introduction

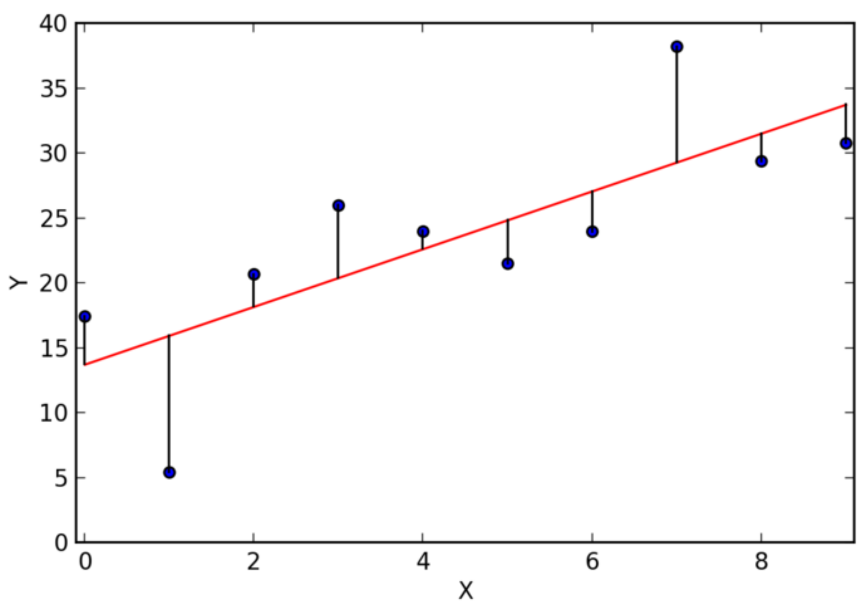
- Linear regression is used for two tasks:
  - Inference
  - Prediction
    - X is readily available
    - Y is not available
    - Build a model to predict Y based on X with minimal error, generating a $\hat{Y}$

Agenda

- Introduction
- Linear Regression
- Confidence intervals of LR
- Assumptions
- Goodness of Fit
- Simple Polynomial Regression
- Statistical Learning with LR (bonus round)

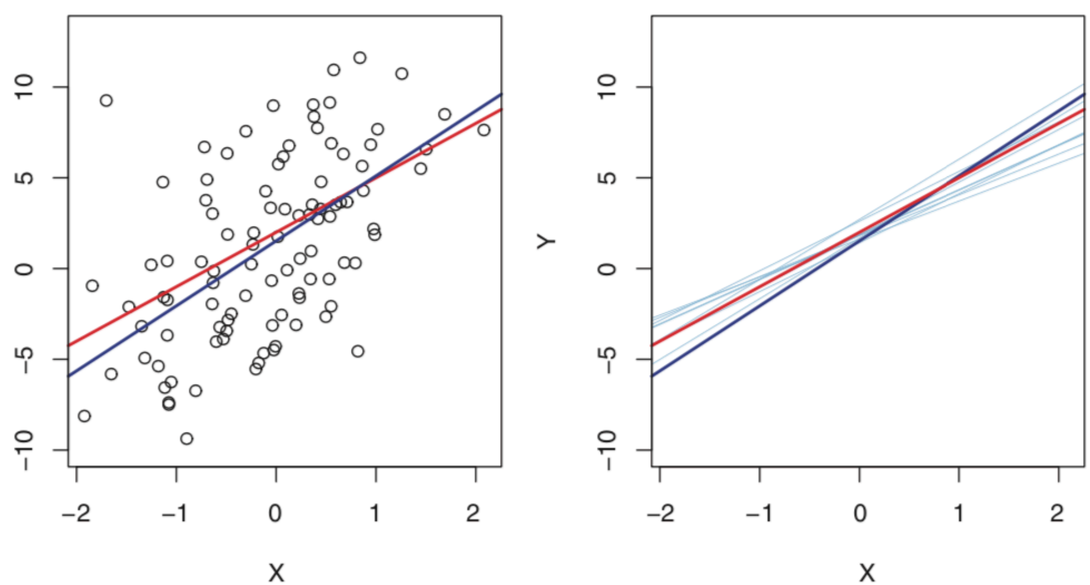
Linear Regression

- Goal: estimate coefficients that best fit the data
- What is the best way to fit a line such that it is the least distance from all data points?



Linear Regression

- Answer: Ordinary Least Squares (OLS)*
- Minimizes the distance of the error: (fitted values – actual values)



Linear Regression

- LR assumes there is approximately a linear relationship between X and Y

$$Y \approx \beta_0 + \beta_1 X_1 + \epsilon$$

Linear Regression

- LR assumes there is approximately a linear relationship between X and Y

model coefficients: what is computed

$$Y \approx \beta_0 + \beta_1 X_1 + \epsilon$$

Predictor and response are always given

Linear Regression

- LR assumes there is approximately a linear relationship between X and Y

Slope: average increase in Y with increase in X

Response

Error: a catch-all for what we miss in the simple model

$$Y \approx \beta_0 + \beta_1 X_1 + \epsilon$$

Intercept: expected value of Y when X is 0

Predictor

Linear Regression

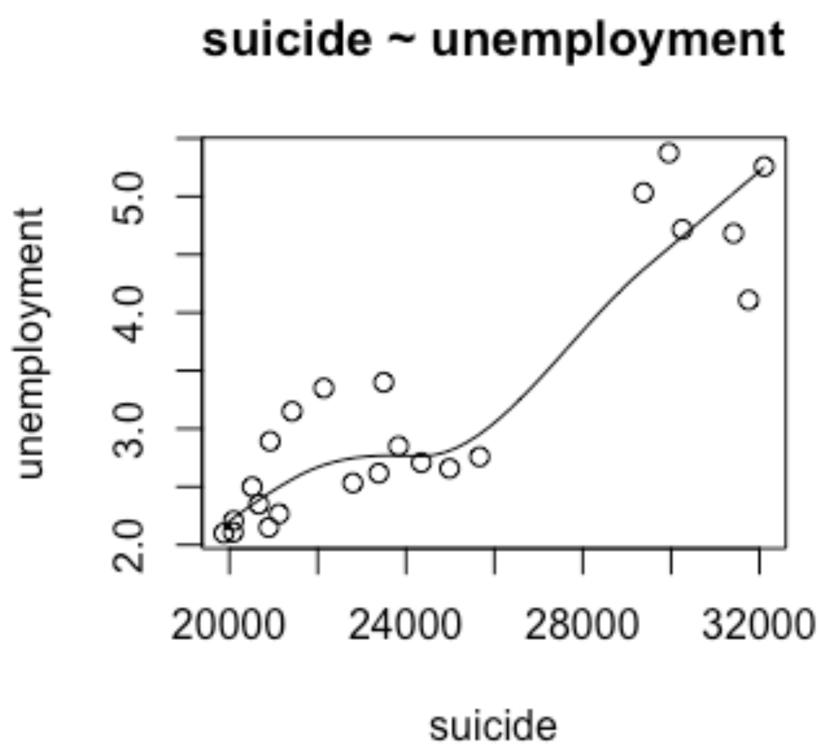
- Let’s use the American annual rate of suicide dataset ‘hwd.txt’
- Predictors: unemployment, realgdp
- Response: suicide rate
- Time is unaccounted

Linear Regression

- Question: which predictor better describes the relationship between predictor and response?
- Pretend we can only choose one!
- Build a linear model using one predictor

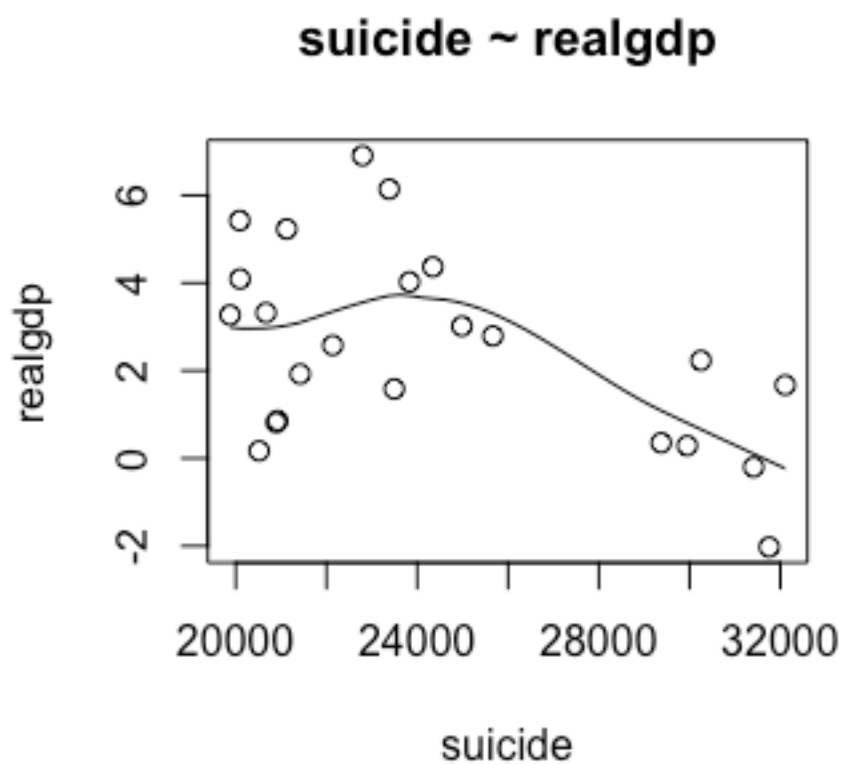
Linear Regression

- First, let's check the representation of our data



Linear Regression

- First, let's check the representation of our data



Linear Regression

- Can we see any trends before even calculating the correlation?
- Just to confirm, let's check the correlation matrix

```
> cor(hwd[-1])
               suicide unemployment    realgdp
suicide      1.0000000    0.9007385 -0.5121380
unemployment 0.9007385    1.0000000 -0.5829779
realgdp     -0.5121380   -0.5829779  1.0000000
```

Linear Regression

- Fitting a simple regression model in R is easy:

```
#t-value: prob  that its not 0
fit1 <- lm(suicide ~ unemployment, data = hwd)
#summary of the regression
summary(fit1)
```

```
fit2 <- lm(suicide ~ realgdp, data = hwd)
```

Linear Regression

- Fitting a simple regression model in R is easy:

Linear regression function call

predictor

```r
#t-value: prob  that its not 0
fit1 <- lm(suicide ~ unemployment, data = hwd)
#summary of the regression
summary(fit1)
```

response

~ like anova

Data frame reference

```r
fit2 <- lm(suicide ~ realgdp, data = hwd)
```

Linear Regression

- After building the linear model, we need to:
  - Calculate the confidence interval
  - Verify how good it fits the data
  - Verify if its in accordance to the assumption of LR
- If it fits poorly or fails in any assumption, trying another model is recommended

Agenda

- Introduction
- Linear Regression
- Confidence intervals of LR
- Assumptions
- Goodness of Fit
- Simple Polynomial Regression
- Statistical Learning with LR (bonus round)

Confidence Intervals

- Using the standard error statistic computed from the fitted linear model we can compute confidence intervals
- Quantify the uncertainty of the predictor and the response

$$SE = \sigma_{\bar{x}} \approx \frac{s}{\sqrt{n}}$$

Confidence Intervals

- Our example:

```
> confint(fit1)
                 2.5 %    97.5 %
(Intercept)  112.361671 128.78658
unemployment   8.505942  13.36643
```

Mean of the suicide rate

With each increase in unemployment, the suicide will increase around this much

Agenda

- Introduction
- Linear Regression
- Confidence intervals of LR
- Assumptions
- Goodness of Fit
- Simple Polynomial Regression
- Statistical Learning with LR (bonus round)

Assumptions

- The linear regression model must follow these assumptions:
  1. Normality
  2. Linearity of data
  3. Non-correlation of errors
  4. Homoscedascity
  5. Outliers without leverage
  6. Collinearity (Multiple case)

Assumptions

- Normality assumption
  - Qq-plot of studentized residuals
  - Histogram of studentized residuals
  - Density plot of predictors/response

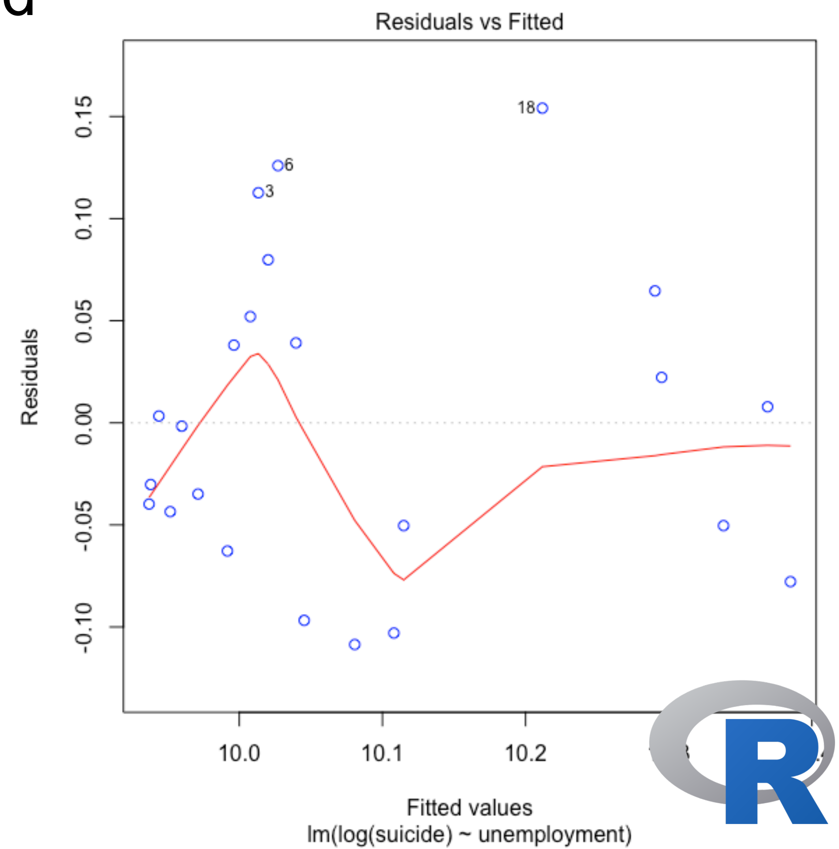
```
#qqplot of the fitted model
qqPlot(fit1, main="QQ Plot") ##qq plot of studentized residuals


#histogram of the studentized residuals
hist(studres(fit1), freq=FALSE,
     main="Distribution of Studentized Residuals")
xfit<-seq(min(studres(fit1)),max(studres(fit1)),length=40)
yfit<-dnorm(xfit)
lines(xfit, yfit)
```

Assumptions

- Linearity of data (Linear model, linear data)
  - QQ-plot of residuals against fitted
  - Components + residuals plot

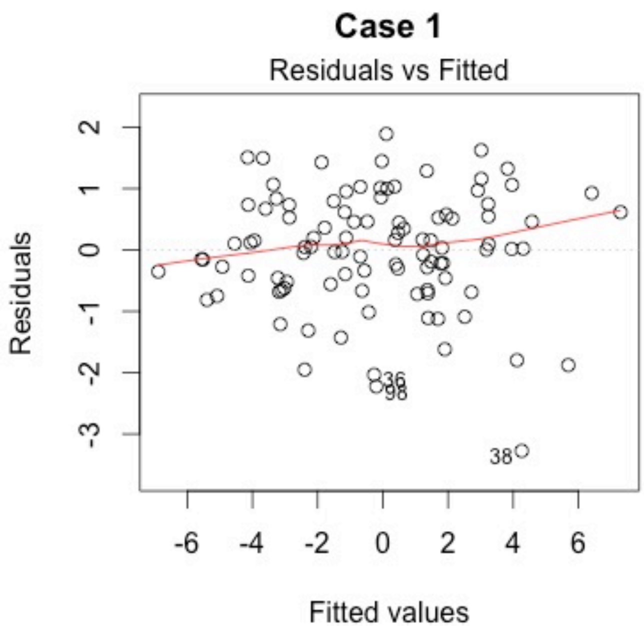
```
#residual  against fitted values plot
plot(fit1, which=1, col = c('blue'))
```

```
# component + residual plot
crPlots(fit1)
```

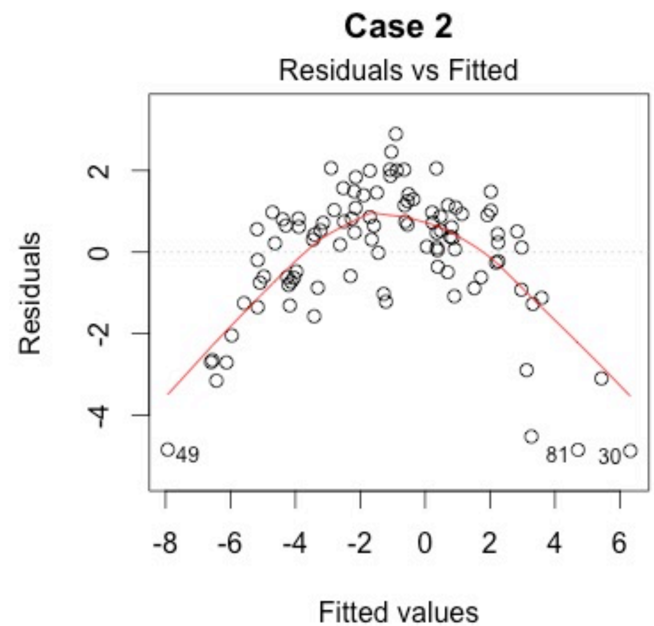


Assumptions

- Linearity of data



Good case



Bad case

Assumptions

- Non-correlation of errors
  - Durbin-Watson test
  - Rejecting H_0 does not invalidate the model. However, care must be taken!!!

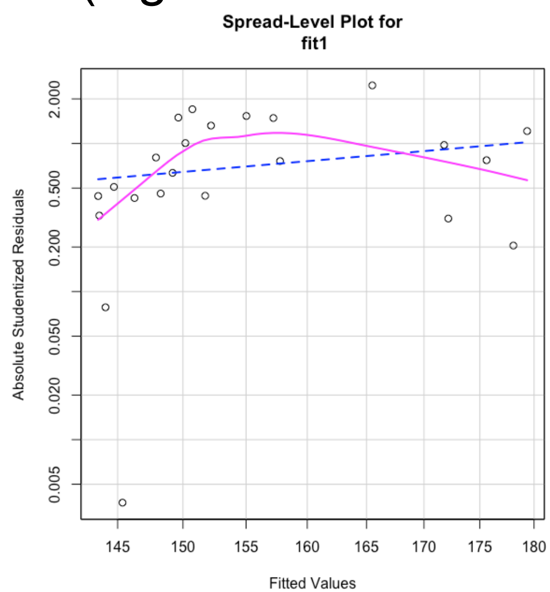
```
#TESTING FOR INDEPENDENCE (AUTOCORRELATED ERRORS)
durbinWatsonTest(fit1) #positive autocorrelation
```

H_0: variables are not autocorrelated

Assumptions

- Homoscedascity (homogeneity in the variance of errors)
  - Non-constant error variance test
  - Breusch-pagan test
  - Spread level plots (log of the absolute studentized residuals vs. log of fitted values)

H_0: Homoscedascity is observed

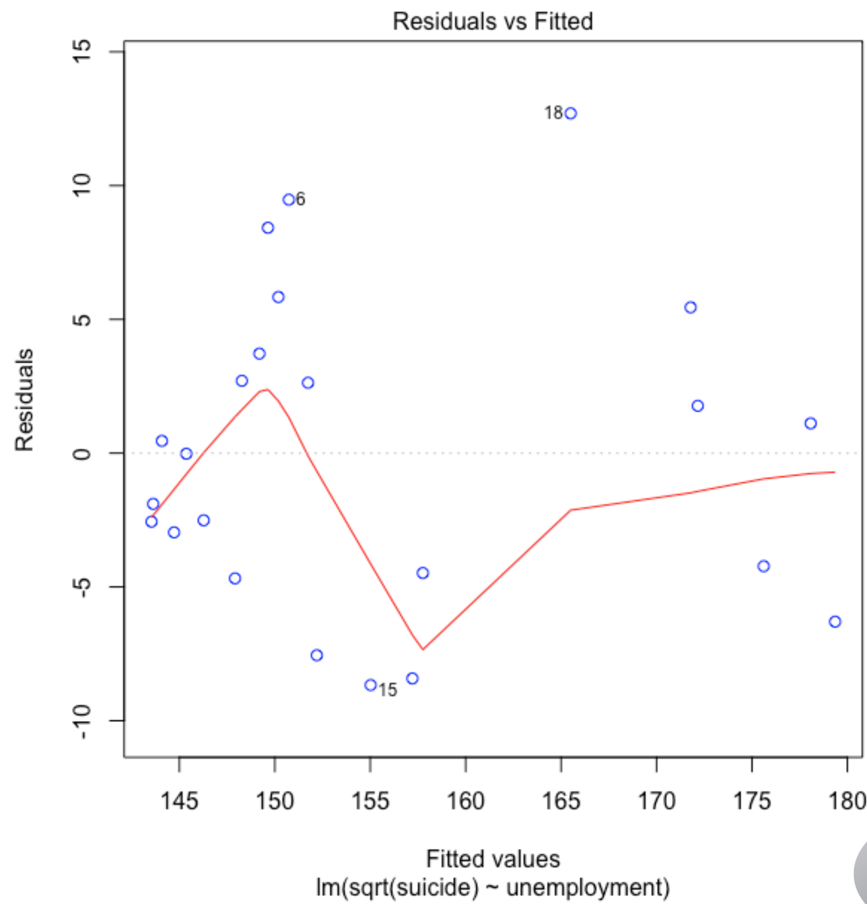


The pink line is the variance, notice it does not follow a trend

Assumptions

- Homoscedascity

No pattern = good



Assumptions

- Outliers with low leverage
  - Outliers
    - measurements far from the fitted model
    - Can be a measurement error
    - Can be remove if it has low leverage
  - Leverage
    - degree of impact on the fitted model
    - High leverage: high impact on the model

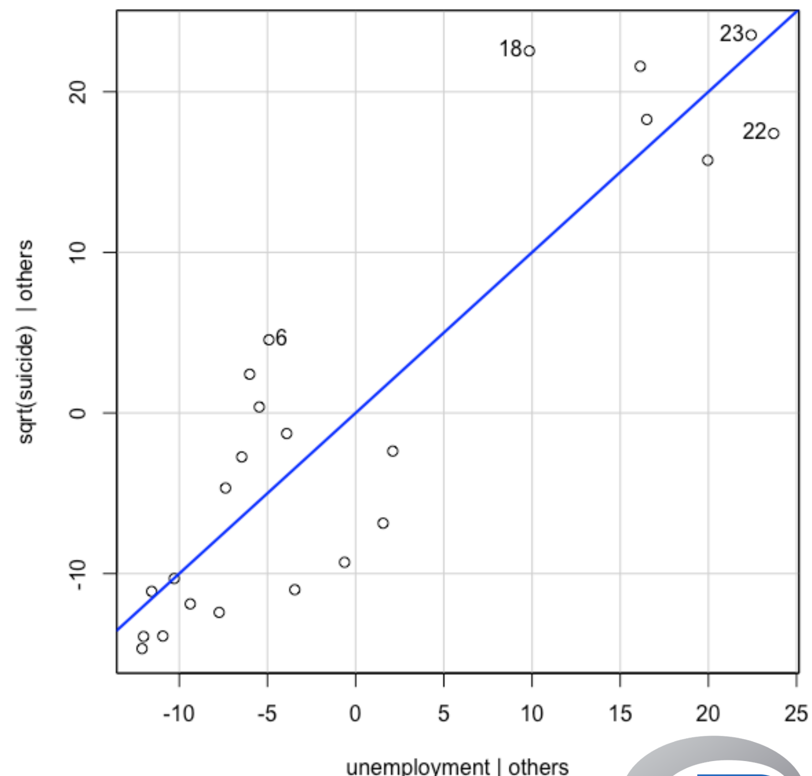
Assumptions

- Outliers with low leverage
  - Residual plot
  - Studentized residuals plot
  - Leverage plot
  - Bonferroni p-value of the most extreme observation

H_0: No outliers with low leverage

Assumptions

- Outliers with low leverage



Agenda

- Introduction
- Linear Regression
- Confidence intervals of LR
- Assumptions
- Goodness of Fit
- Simple Polynomial Regression
- Statistical Learning with LR (bonus round)

Goodness of fit

- How to measure how well the linear model fit your data?
- Depends on required task
  - Inference: R-squared, RSE, BIC, AIC
  - Prediction: R-squared, K-fold cross validation
- Bad value of indicators: model is not proper

Goodness of fit

R-squared

- Square of the correlation of the response and the variable
- Values close to 1 explain the variance
- Increases with the variable size
- Not reliable

Goodness of fit

RSE (Root squared error)

- Another measure of predictor against response
- Inversely proportional to the RSS (sum squared residuals)

$$RSE = \sqrt{\frac{1}{n-p-1}\underbrace{\sum_{i=1}^{m}(y_i - \hat{y})^2}_{\boxed{\text{RSS}}}}$$

Goodness of fit

AIC and BIC (Akaike/Bayes information criterion)

- R-adjusted and C_p
- Rewards goodness of fit but includes penalty that as an increasing function of the number of estimated parameters.

- Introduction
- Linear Regression
- Confidence intervals of LR
- Assumptions
- Goodness of Fit
- Simple Polynomial Regression
- Statistical Learning with LR (bonus round)

Simple polynomial regression

- Let's say that using the unemployment predictor to the linear model did not describe the data properly
- And also, let's say that the data has shown to be a bit nonlinear
- Can we alter the linear model to fit nonlinear data?

Simple polynomial regression

- Perhaps a higher order polynomial model using the same predictor may better explain the data
- The new linear model will be as follows:

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \cdots + \beta_n X_1^n$$

Simple polynomial regression

- Which order best fits the data? Trial and error
- When analyzing real data, we usually know little about the shape of the data, so care must be taken
- May contribute to overfitting (the bane of ML)

Overfit: Failure to generalize the fitted model to test data

Simple polynomial regression

- In R, a simple polynomial regression is straightforward:

```
polyfit1 <- lm(suicide ~ poly(unemployment,3), data = hwd)
```

Simple polynomial regression

- In R, a simple polynomial regression is straightforward:

Linear regression function call

predictor

```
polyfit1 <- lm(suicide ~ poly(unemployment,3), data = hwd)
```

response

Polynomial regression of third degree

Poly function is preferred because it does not generate correlated variables

Simple polynomial regression

- Like simple regression, we check the same:
  - Assumptions
  - Goodness of fit test
  - Prediction results (if applicable)

Agenda

- Introduction
- Linear Regression
- Confidence intervals of LR
- Assumptions
- Goodness of Fit
- Simple Polynomial Regression
- Statistical Learning with LR (bonus round)

Statistical Learning with LR

- ”Hello world” of machine learning
- Simplest tool for statistical learning
- Fit a model to predict unknown data
- Training and Testing

Statistical Learning with LR

- Data is now split into two sets:
  - Training (80%)
    - Model is fitted using this portion of the data
  - Testing (20%)
    - The fitted model predicts values with this portion

Statistical Learning with LR

- Underfitting:
  - Performs poorly in the training phase
  - Performs poorly in the testing phase
  - High Bias
  - Better model must be chosen

Statistical Learning with LR

- Overfitting:
  - Performs great in the training phase
  - Performs poorly in the testing phase
  - High Variance
  - Model failed to generalize the fitted model to unforeseen data

Statistical Learning with LR

- We now split the data into training and test

Random rows of the data set

80% of the dataset

```
trainingRowIndex <- sample(1:nrow(hwd), 0.8*nrow(hwd))  # row indices for training data

trainingData <- hwd[trainingRowIndex, ]  # model training data

testData  <- hwd[-trainingRowIndex, ]   # test data
```

Logic indexing

Statistical Learning with LR

- Building the model and doing the prediction

Same as simple regression

Don't forget to use the training data

```
# Build the model on training data -
lmTrainMod <- lm(suicide ~ unemployment, data=trainingData)  # build the model
```

Predict function call

Arguments: model, test data

```
#predicting the model on the train data
suicidePred <- predict(lmTrainMod, testData)  # predict distance
```

Statistical Learning with LR

- Goodness of fit can be calculated by the BIC and/or AIC
- Accuracy is also assessed

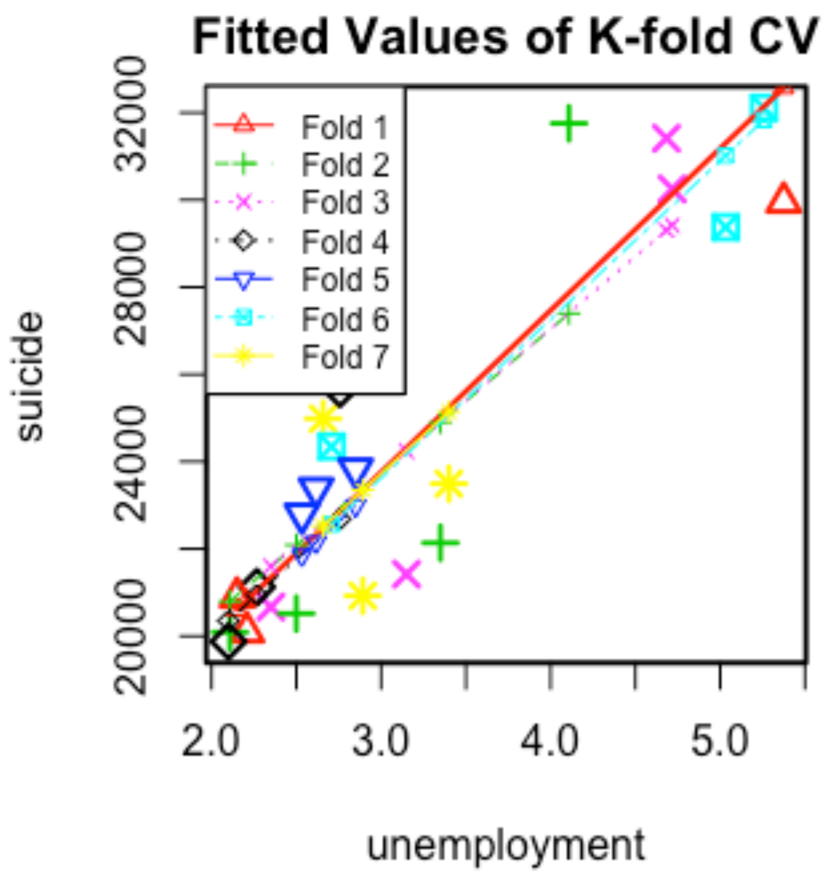
```
#estimated AIC (usually doesnt reflect the goodness of fit in this case)
AIC(lmTrainMod)

#lets calculate the prediction accuracy and error rates
actuals_preds <- data.frame(cbind(actuals=testData$suicide, predicteds=suicidePred))
correlation_accuracy <- cor(actuals_preds)
correlation_accuracy
head(actuals_preds)

#CALCULATION OF THE ERROR RATES
min_max_accuracy <- mean(apply(actuals_preds, 1, min) / apply(actuals_preds, 1, max))
min_max_accuracy

#Mean Absolute percentage error
mape <- mean(abs((actuals_preds$predicteds - actuals_preds$actuals))/actuals_preds$actuals)
mape
```

Statistical Learning with LR

- We will check the goodness of the fit empirically
- K-Fold cross validation
- Get a random portion of the data, test against the model k times

Statistical Learning with LR

```
#K-fold Cross Validation
cvResults <- suppressWarnings(CVlm(data  = hwd, form.lm=suicide ~ unemployment, m=7,
dots=FALSE, seed=1, legend.pos="topleft",  printit=FALSE, main="Fitted Values of K-fold CV")) # performs the CV
```



Next Episode

- Dummy variables (qualitative predictors)
- Multiple Linear Regression
- Generalized Linear model (GLM)
- Regression when the response is qualitative: Logistic Regression
- How to choose the best model among all possible choices?

Simple Linear Regression Model

Marco Antonio Florenzano Mollinetti[1]
[1]**University of Tsukuba, Systems Optimization Laboratory**
mollinetti@syou.cs.tsukuba.ac.jp