DATA SOCIETY®

Week 3 Day 2 - Introduction to Linear Regression

One should look for what is and not what he thinks should be.
-Albert Einstein.

Module completion checklist

Objective	Complete
Summarize why linear regression and how it can be helpful in a business setting	
Evaluate data using basic statistics such as summary statistics,covariance, correlation	
Summarize the basis of linear regression, identify what type of learning method it is	
Prepare the data set to run a linear regression model	
Recognize what to look for in the data (NAs, variance)	
Implement and run the linear regression model on the given data	
Evaluate the linear model and analyze summary metrics	
Validate the linear regression model by reviewing assumptions	
Summarize how to implement multiple regression and analyze output	
Evaluate the final model and look for heteroscadacity and coliniarity	

Directory settings

In order to maximize the efficiency of your workflow, you may want to encode your directory structure into `variables`

Let the `main_dir` be the variable corresponding to your `hhs-r-2020` folder

```
# Set `main_dir` to the location of your `hhs-r-2020` folder (for Mac/Linux).
main_dir = "~/Desktop/hhs-r-2020"
# Set `main_dir` to the location of your `hhs-r-2020` folder (for Windows).
main_dir = "C:/Users/[username]/Desktop/hhs-r-2020"
# Make `data_dir` from the `main_dir` and remainder of the path to data directory.
data_dir = paste0(main_dir, "/data")
# Make `plots_dir` from the `main_dir` and remainder of the path to plots directory.
plot_dir = paste0(main_dir, "/plots")
# Set directory to data_dir.
setwd(data_dir)
```

Installing packages

We always start off with installing/loading the needed libraries for the day
For the packages we have not yet used, we will define them before using them
For now, we are just installing and loading each package

```r
# install.packages("caret")
library(caret)
# install.packages("car")
library(car)
# install.packages("corrplot")
library(corrplot)
# install.packages("AppliedPredictiveModeling")
library(AppliedPredictiveModeling)
```

What questions can we answer?

Some questions that linear regression will help answer are:

- How much does a *single factor* account for a final outcome?
- How do *several variables together* account for a final outcome?
- What set of variables together will best explain the final outcome?

Motivation

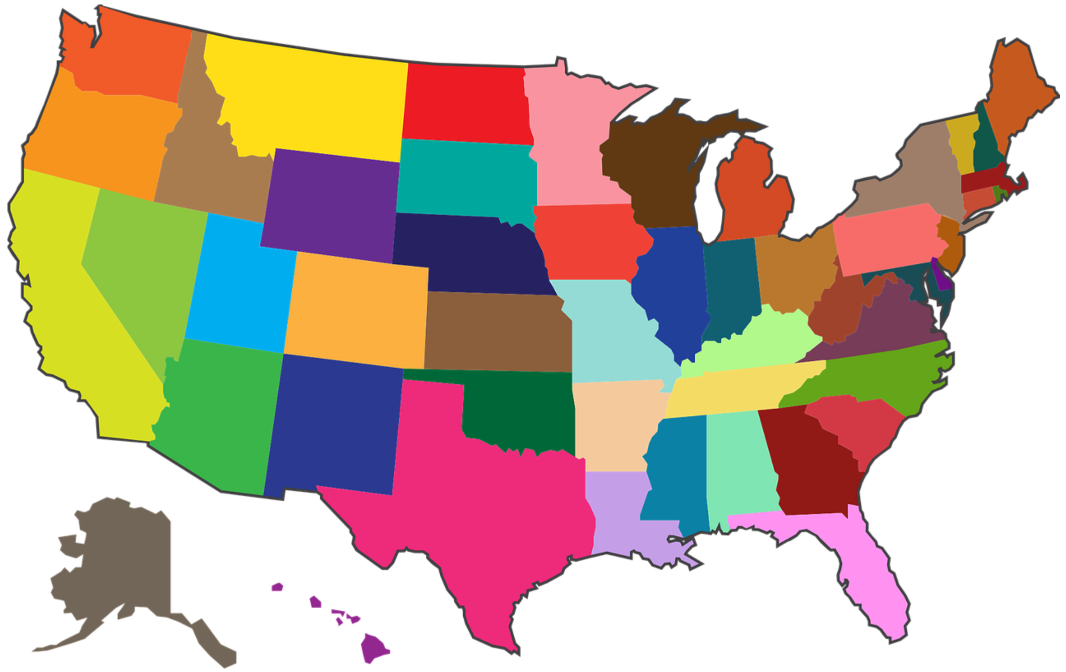
Why would managers want to know how factors affect a final outcome?

- To provide better quality products based on adjusting the factors
- To anticipate maintenance expenditures and decrease costs by understanding each aspect of the final product

Datasets in R: state.x77 data

We are going to make use of some datasets that come pre-loaded with R. These datasets have some data related to the 50 US states and include:

```
state.x77
state.abb
state.area
state.name
state.region
state.center
state.division
```



Datasets in R: state.x77 data

```r
# This dataset is of type `matrix`. We don't want to modify the original dataset,
# so let's set this dataset to a variable, so that we can manipulate it freely.
state_data = as.data.frame(state.x77)

# The dataset contains 50 rows (i.e. 50 states) and 8 columns.
# It's easy to check the dimensions of any object in R with a simple `dim` function.
dim(state_data)
```

```
[1] 50  8
```

```r
# Since matrix is a 2-dimensional object we get a vector with 2 entries:
# 1. The first one corresponds to the number of rows
dim(state_data)[1]
```

```
[1] 50
```

```r
# 2. The second tells us how many columns we have
dim(state_data)[2]
```

```
[1] 8
```

Datasets in R: state.x77 data

For the purpose of single variable regression, we are going to subset the dataset
We will keep `Income` and `HS Grad` in the new dataset

```
state_data_sub = state_data[,c(2,6)]
```

Summary statistics

Summary statistics help us understand our data better

Summary statistics we will review are:

Summary statistic	Definition
Min	the minimum number in the vector
1st quartile	the median of the lower half of the dataset/vector
Median	the point separating the higher half of the dataset/vector
Mean	the central value or average of the dataset/vector
3rd quartile	the median of the higher half of the dataset/vector
Max	the maximum number in the vector

Summary statistics: income & HS graduation rate

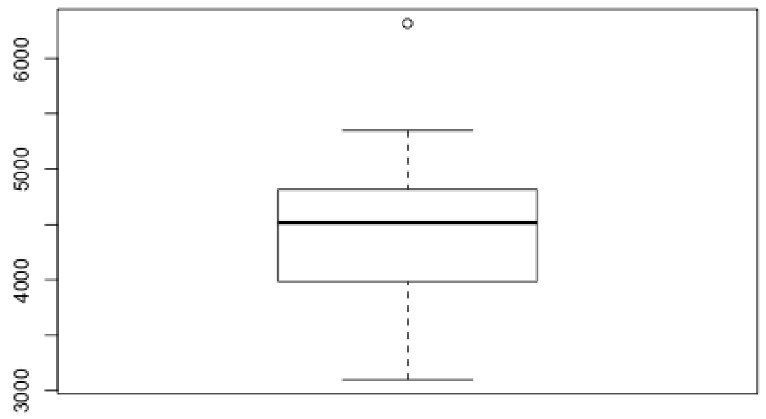
Numerical summary: Income

```r
# State data subset - Income
summary(state_data_sub$Income)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   3098    3993    4519    4436    4814    6315 
```

Graphical summary: Income

```r
# Univariate plot: box-and-whisker plot.
boxplot(state_data_sub$Income)
```



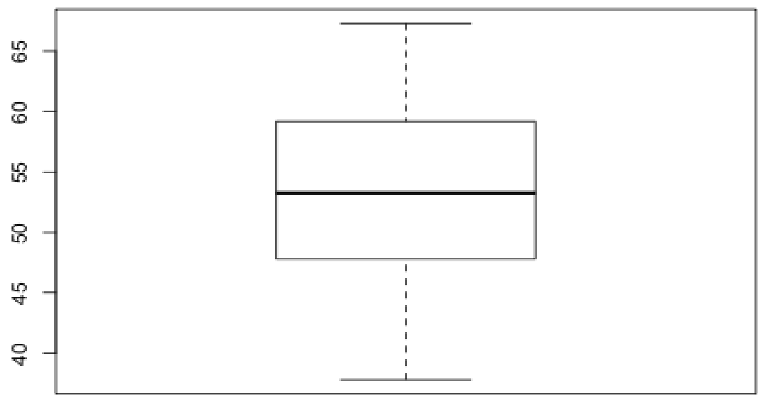
Numerical summary: HS Grad Rate

```r
# State data subset - HS Grad
summary(state_data_sub$`HS Grad`)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  37.80   48.05   53.25   53.11   59.15   67.30 
```

Graphical summary: HS Grad Rate

```r
# Univariate plot: box-and-whisker plot.
boxplot(state_data_sub$`HS Grad`)
```



Summary statistics: covariance

$$Cov(X,Y) = \frac{\sum_{i=1}^{N}(x_i - mean_x)(y_i - mean_y)}{(N-1)}$$

Covariance of two variables (x,y) is a measure of **association between x and y**

- It is **positive** if y **increases with increasing** x
- It is **negative** if y **decreases with increasing** x
- It is **zero** if there is **no linear tendency** for y to change with x

Covariance is very **sensitive** to the scale of the two variables, which makes comparing covariance across variables **very difficult**

I'm too smart and too sensitive to live in a world like ours.

Summary statistics: covariance in R

We can easily calculate covariance in `R`

```
cov(state_data_sub$Income,state_data_sub$`HS Grad`)
```

```
[1] 3076.769
```

The value of the covariance can be very large, simply because the data we are using (income) has a large value

Thankfully, we can use correlation to determine relationships instead - `scaled covariance`

Summary statistics: correlation is scaled cov

We can use **correlation**:

$$r_{xy} = \frac{Cov(X,Y)}{\sqrt{var(X)var(Y)}}$$

This formula controls for the scale of the variables and guarantees that **correlation is always between -1 and +1**

We can easily calculate correlation in `R`

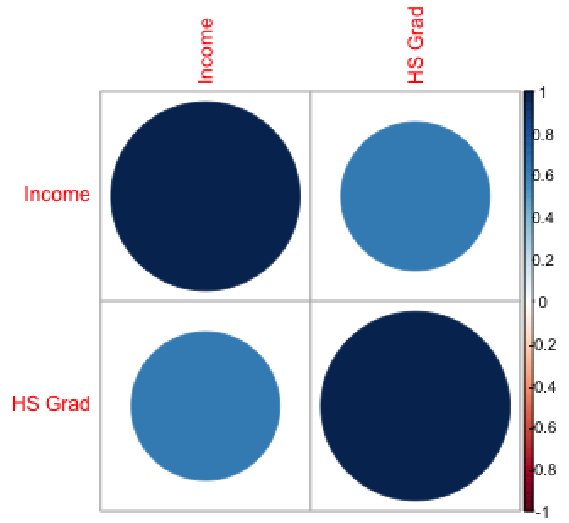
```
cor(state_data_sub$Income,state_data_sub$`HS
Grad`)
```

```
[1] 0.6199323
```

```
# And we can save a correlation matrix
state_cor = cor(state_data_sub)
```

We can also build correlation plots, like we reviewed in an earlier lesson

```
# Create correlation plot.
corrplot(state_cor)
```



Linear regression: measuring relationships

A linear relationship between 2 variables is:

$$y = mx + b$$

$$m = \frac{Change..in..y}{Change..in..x} = slope$$

`y`: variable 1 - **dependent** variable, what you want to predict - target variable
`x`: variable 2 - **independent** variable, what you want to use to predict - predictor
`m`: rate of change (**slope**)
`b`: value of y when x = 0

Linear regression is a **supervised learning method**

- We have at least one predictor (x)
- We have one target variable (y)

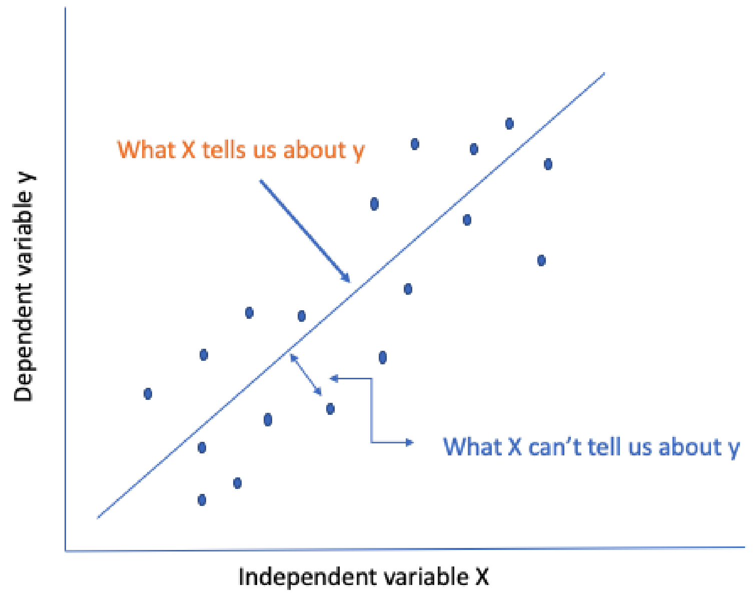
Linear regression: methodology

Simple linear regression uses the least squares line fit algorithm

This method is based on finding the best fitting line for a set of data points

The result of simple linear regression is a best fit line, or rather the 2 parameters that specify the line ($y = mx + b$):

- **Slope** (`m`)
- **Intercept** (`b`)



Linear regression: expected outcome

Given the line of best fit, we can calculate the residuals between the values of the response variable that we predicted vs the actual values that we have

Given the residuals we can calculate different metrics that will allow us not only to verify the model's assumptions, but also measure the performance of the algorithm

Knowledge Check 1

Exercise 1

Module completion checklist

Objective	Complete
Summarize why we would want to use linear regression and how it can help in a business setting	✔
Evaluate data using basic statistics such as summary statistics,covariance, correlation	✔
Summarize the basis of linear regression, identify what type of learning method it is	✔
Prepare the data set to run a linear regression model	
Recognize what to look for in the data, NAs, variance	
Implement and run the linear regression model on the given data	
Evaluate the linear model and analyze summary metrics	
Validate the linear regression model by looking at it's assumptions	
Summarize how to implement multiple regression and analyze output	
Evaluate the final model and look for heteroscadacity and colinearity	

Linear regression: state data

We will be using the state data that we subsetted for simple linear regression

```
# Look at the data we will be working with
head(state_data_sub)
```

```
           Income HS Grad
Alabama      3624    41.3
Alaska       6315    66.7
Arizona      4530    58.1
Arkansas     3378    39.9
California   5114    62.6
Colorado     4884    63.9
```

We will be predicting `HS Grad`, this is our target variable (y)

We will using `Income` to predict `HS Grad`, this is our predictor (x)

Linear regression: EDA

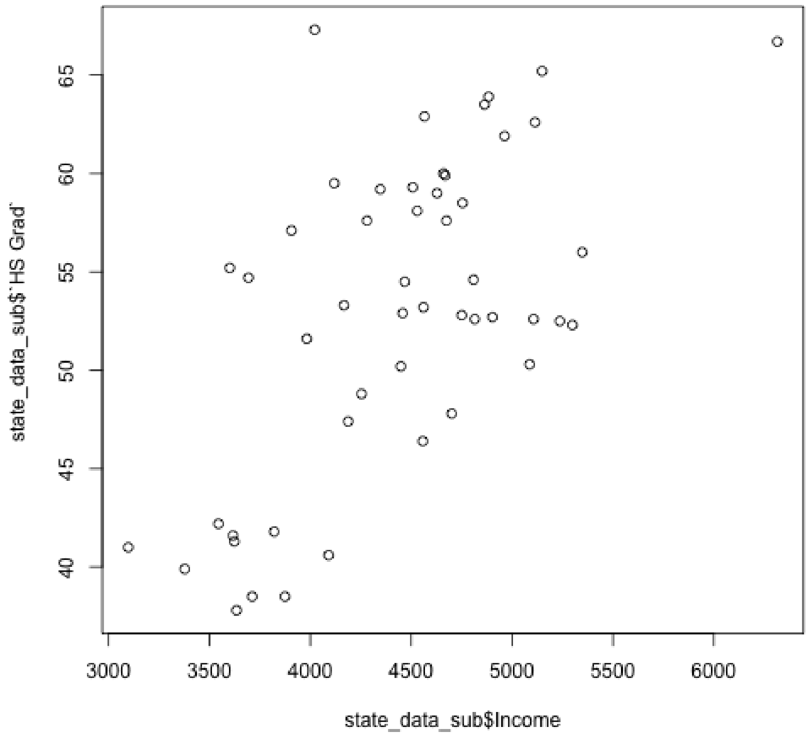
We subset our state dataset earlier, we are now going to get it ready to run a linear regression
First, let's conduct simple EDA to understand our data better

- Scatterplot of the two variables, Income and HS Grad
- Histogram of each variable, Income and HS Grad

EDA: scatterplot

- We will use `plot` to build both the scatterplot and histogram
- In this module, we want to predict HS Grad rate based on income
- In this visualization, we are looking at the interaction of the two variables

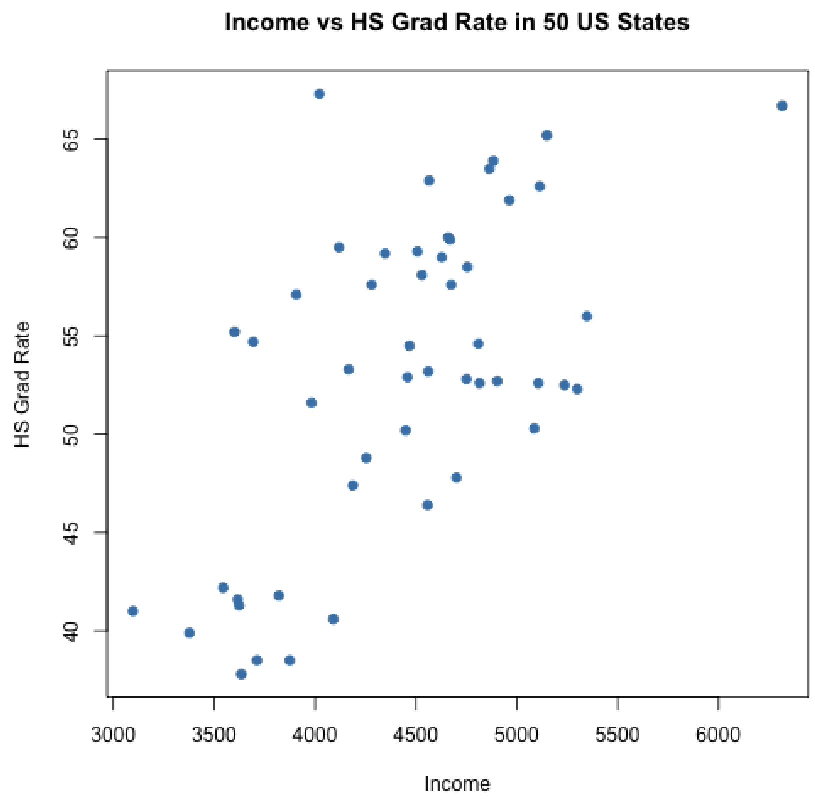
```
# Make scatterplot.
plot(state_data_sub$Income,     #<- x-axis
     state_data_sub$`HS Grad`) #<- y-axis
```



EDA: improved scatterplot

Remember we made our simple `plot` a little fancier?
Let's do that again, add `col`, `pch`, `xlab`, `ylab` and `main` arguments

```
plot(state_data_sub$Income,
     state_data_sub$`HS Grad`,
     col = "steelblue",
     pch = 19,
     xlab = "Income",
     ylab = "HS Grad Rate",
     main = "Income vs HS Grad Rate in 50 US
States")
```

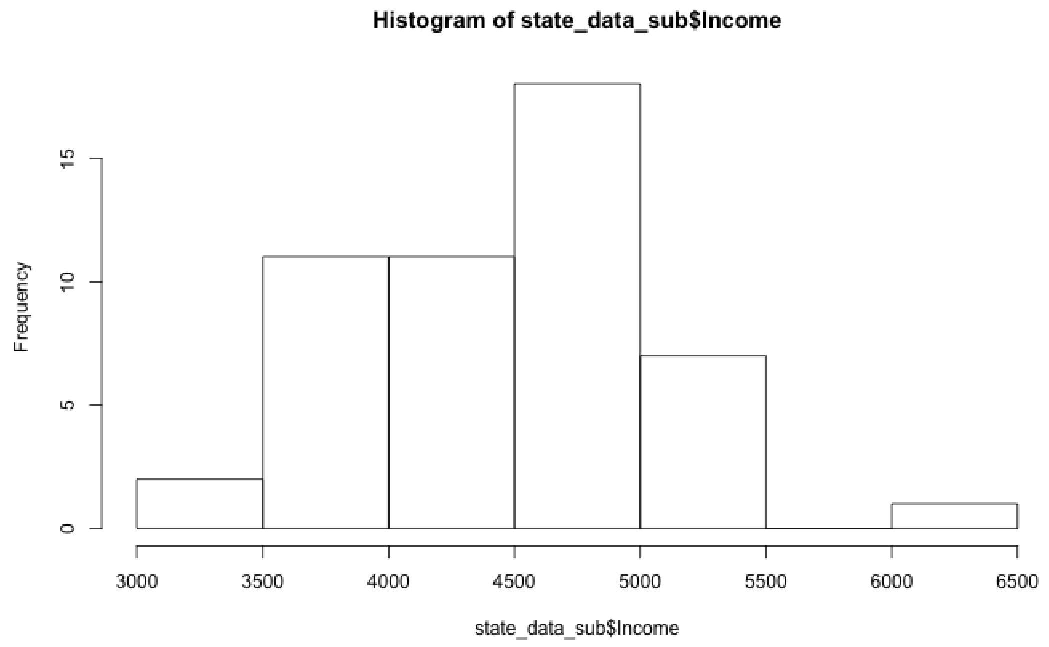


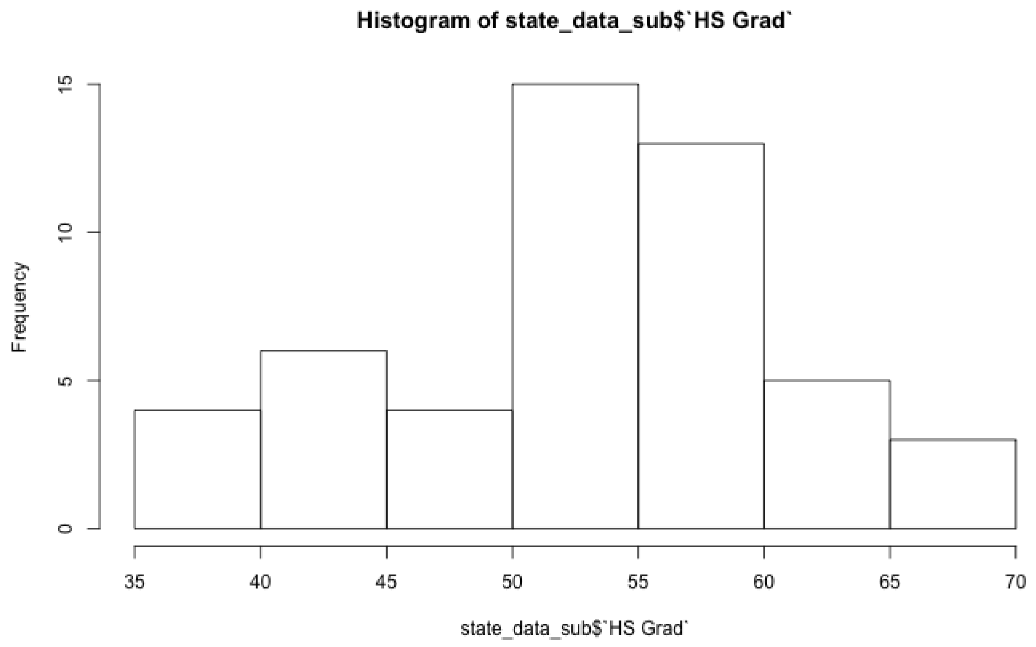
EDA: histogram

We will use `plot` to now build a histogram

In this visualization, we are looking at the distribution of each variable

```
par(mfrow = c(1,2))            #<- show two plots in one row
hist(state_data_sub$Income)    #<- histogram of income
hist(state_data_sub$`HS Grad`) #<- histogram of hs grad
```





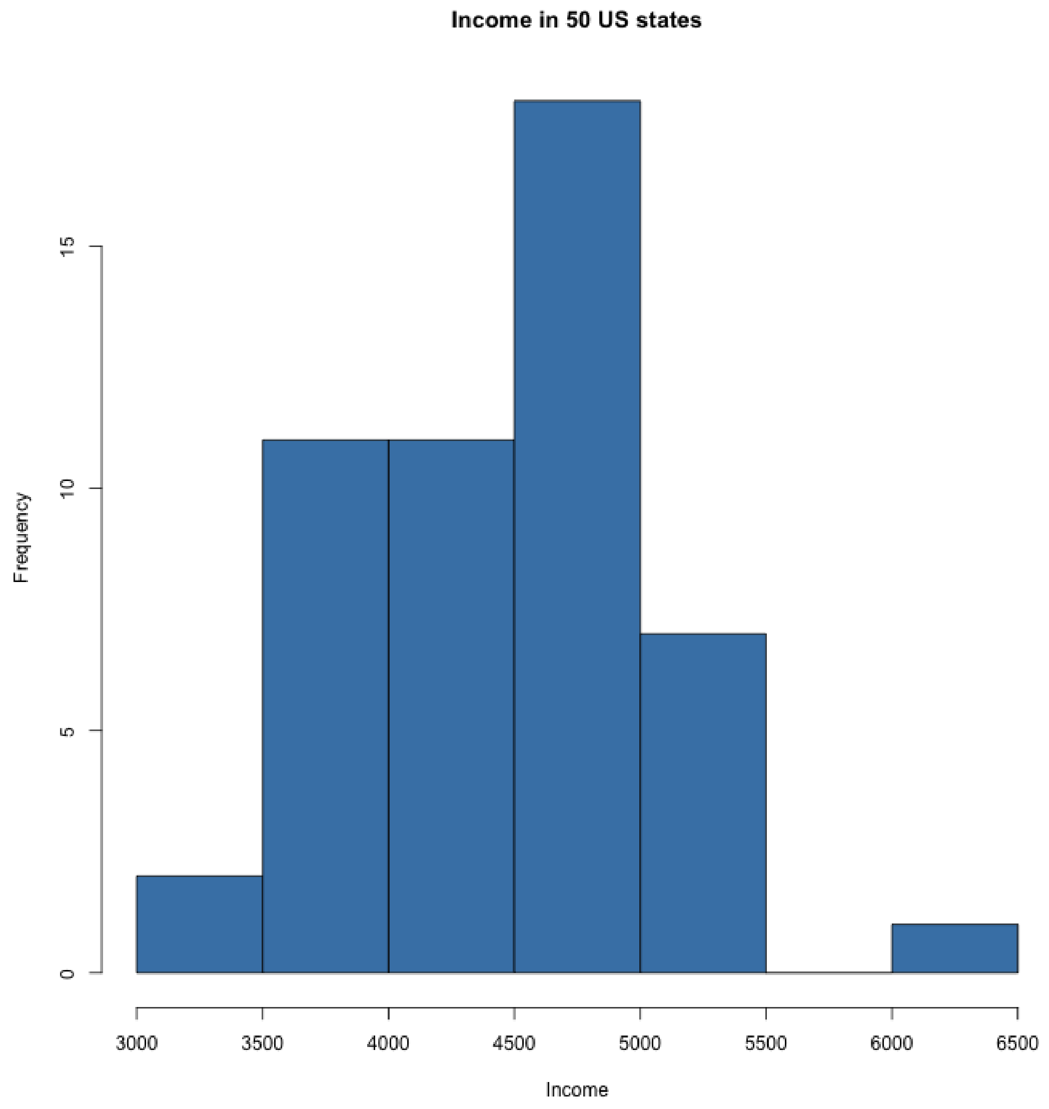
EDA: improved histogram

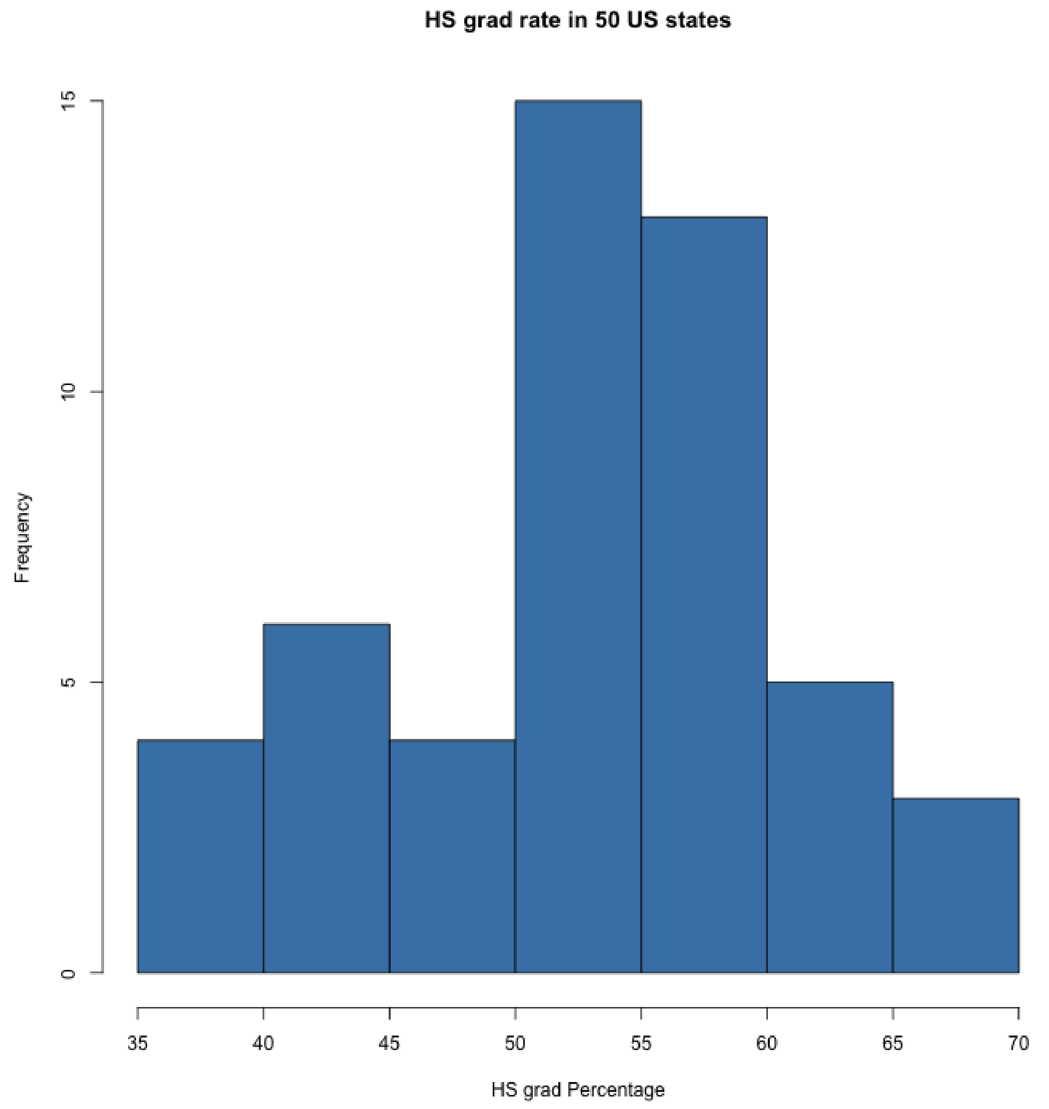
Remember we made our simple `hist` a little fancier?

Let's do that again

```
par(mfrow = c(1,2))
hist(state_data_sub$Income,
     col = "steelblue",             #<- add color
     xlab = "Income",               #<- name the x-axis
     main = "Income in 50 US states") #<- name the plot
hist(state_data_sub$`HS Grad`,
     col = "steelblue",                         #<- add color
     xlab = "HS grad Percentage",               #<- name the x-axis
     main = "HS grad rate in 50 US states")     #<- name the plot
```

EDA: improved histogram





Linear regression: data cleaning

So far, we have reviewed what we need to do to get our data ready for a model
This is where we start incorporating it
In linear regression, before actually running the regression, we look for:

- **NAs**
- Since simple linear regression relies on numerical computations, NAs can produce non-numeric results
- **Near zero variance (NZV)**
- Zero variance variables are variables with a single value
- They are not predictive as the predictor variable's value is always the same

Data cleaning: NAs

First, let us look for NAs in our dataset
Remember earlier on we built a function to impute NAs with the mean of the vector?
We are going to build a modified version of that function, this will just tell us if there are NAs or not

```
whichNA = function(dataset){
  for(i in 1:ncol(dataset)){
    is_na = is.na(dataset[, i])
    if(any(is_na)){
      na_ids = which(is_na)
      message = paste0(
            colnames(dataset)[i],
            "they are",
            na_ids)
      print(message)
    }
    else(print(paste0(colnames(dataset)[i]," has
no NAs")))
  }
}
```

Data cleaning: use NA function

Now lets use this function on our dataset

```
whichNA(state_data_sub)
```

```
[1] "Income has no NAs"
[1] "HS Grad has no NAs"
```

We see that there aren't NAs in the dataset

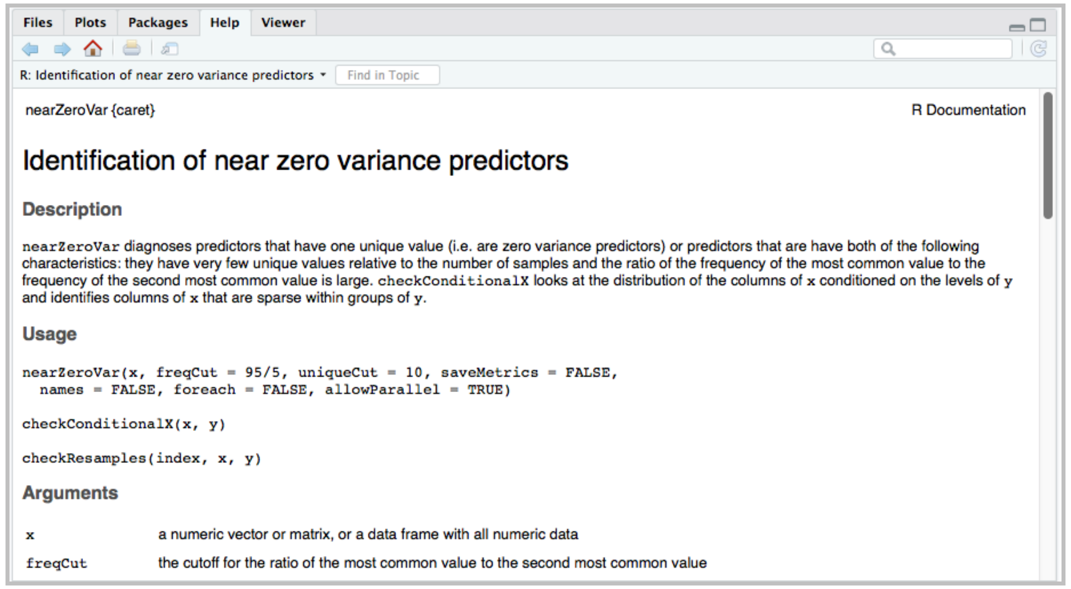
Now we move forward to look for zero variance

Data cleaning: near zero variance

We are concerned with variables that have zero variance because they will not be helpful in prediction

We can use a function `nearZeroVar` from the `caret` package, which we loaded earlier today

```
?caret::nearZeroVar
```



nearZeroVar {caret} R Documentation

Identification of near zero variance predictors

Description

`nearZeroVar` diagnoses predictors that have one unique value (i.e. are zero variance predictors) or predictors that are have both of the following characteristics: they have very few unique values relative to the number of samples and the ratio of the frequency of the most common value to the frequency of the second most common value is large. `checkConditionalX` looks at the distribution of the columns of `x` conditioned on the levels of `y` and identifies columns of `x` that are sparse within groups of `y`.

Usage

```
nearZeroVar(x, freqCut = 95/5, uniqueCut = 10, saveMetrics = FALSE,
  names = FALSE, foreach = FALSE, allowParallel = TRUE)

checkConditionalX(x, y)

checkResamples(index, x, y)
```

Arguments

`x`	a numeric vector or matrix, or a data frame with all numeric data
`freqCut`	the cutoff for the ratio of the most common value to the second most common value

Data cleaning: testing for near zero variance

Now lets run `nearZeroVar` on the dataset

```
nearZeroVar(state_data_sub,
            saveMetrics = TRUE)   #<- we set saveMetrics = TRUE
```

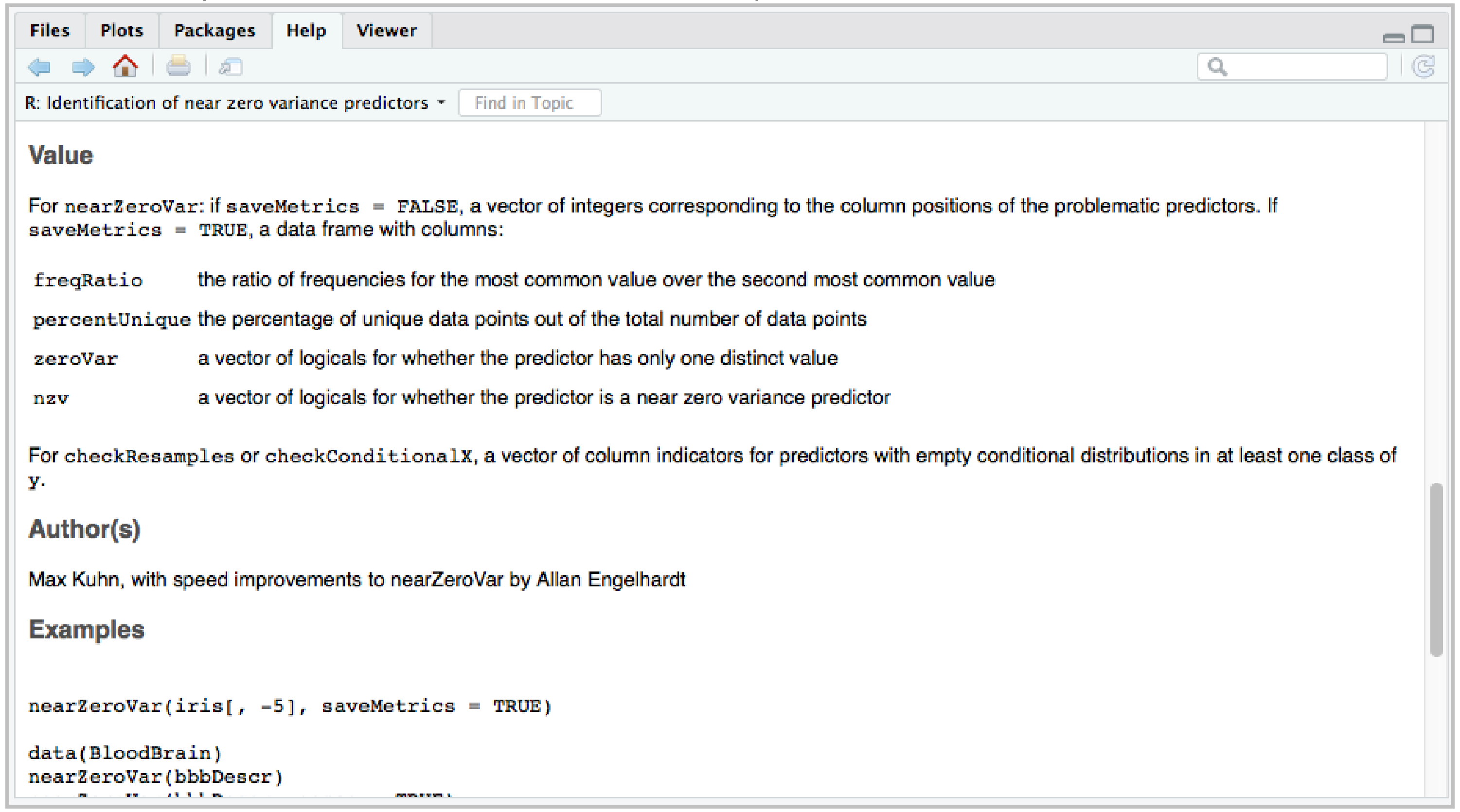
```
        freqRatio percentUnique zeroVar   nzv
Income          1           100   FALSE FALSE
HS Grad         1            94   FALSE FALSE
```

We can confirm that we do not have an issue with either variable having zero or near zero variance

- `freRatio`: the ratio of frequencies for the most common value over the second most common value
- `percentUnique`: the percentage of unique data points out of the total number of data points
- `zeroVar`: a vector of logicals for whether the predictor has **only one distinct value**
- `nzv`: a vector of logicals for whether the predictor is a **near zero variance predictor**

Data cleaning: testing for near zero variance

Here is a quick look at what the output means:



Value

For `nearZeroVar`: if `saveMetrics = FALSE`, a vector of integers corresponding to the column positions of the problematic predictors. If `saveMetrics = TRUE`, a data frame with columns:

`freqRatio`	the ratio of frequencies for the most common value over the second most common value
`percentUnique`	the percentage of unique data points out of the total number of data points
`zeroVar`	a vector of logicals for whether the predictor has only one distinct value
`nzv`	a vector of logicals for whether the predictor is a near zero variance predictor

For `checkResamples` or `checkConditionalX`, a vector of column indicators for predictors with empty conditional distributions in at least one class of `y`.

Author(s)

Max Kuhn, with speed improvements to nearZeroVar by Allan Engelhardt

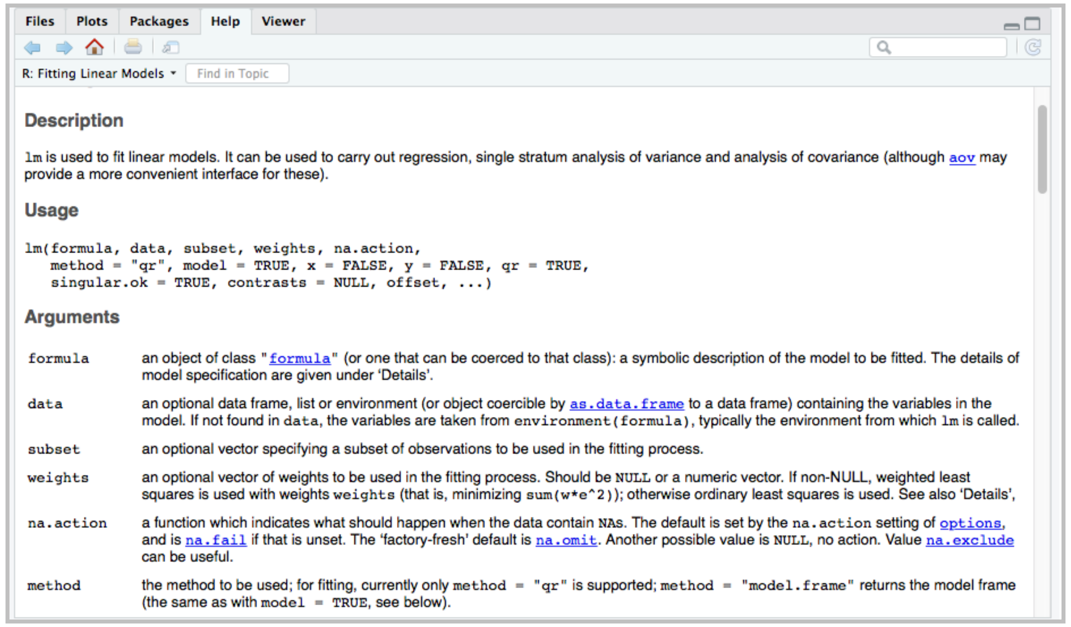
Examples

```
nearZeroVar(iris[, -5], saveMetrics = TRUE)

data(BloodBrain)
nearZeroVar(bbbDescr)
```

Implement linear regression: lm

Let's look at the formula we will use to run linear regression

```
?lm

lm(formula, #<- model to be fitted
   data,    #<- dataset
   subset,  #<- optional vector subseting
training data
   ...)
```



Description

`lm` is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although aov may provide a more convenient interface for these).

Usage

```
lm(formula, data, subset, weights, na.action,
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
   singular.ok = TRUE, contrasts = NULL, offset, ...)
```

Arguments

`formula`	an object of class "`formula`" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `lm` is called.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If non-NULL, weighted least squares is used with weights `weights` (that is, minimizing `sum(w*e^2)`); otherwise ordinary least squares is used. See also 'Details',
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The 'factory-fresh' default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.
`method`	the method to be used; for fitting, currently only `method = "qr"` is supported; `method = "model.frame"` returns the model frame (the same as with `model = TRUE`, see below).

Implement: build a linear model

We use `lm` and set our target variable (y) and our predictor (x)

```
# Build linear model.
lin_model = lm(`HS Grad` ~ Income,
               data = state_data_sub)

# Inspect the output of the `lm` function.
lin_model
```

```

Call:
lm(formula = `HS Grad` ~ Income, data = state_data_sub)

Coefficients:
(Intercept)       Income
  16.961557     0.008149
```

Implement: review lm output

The output includes two things:

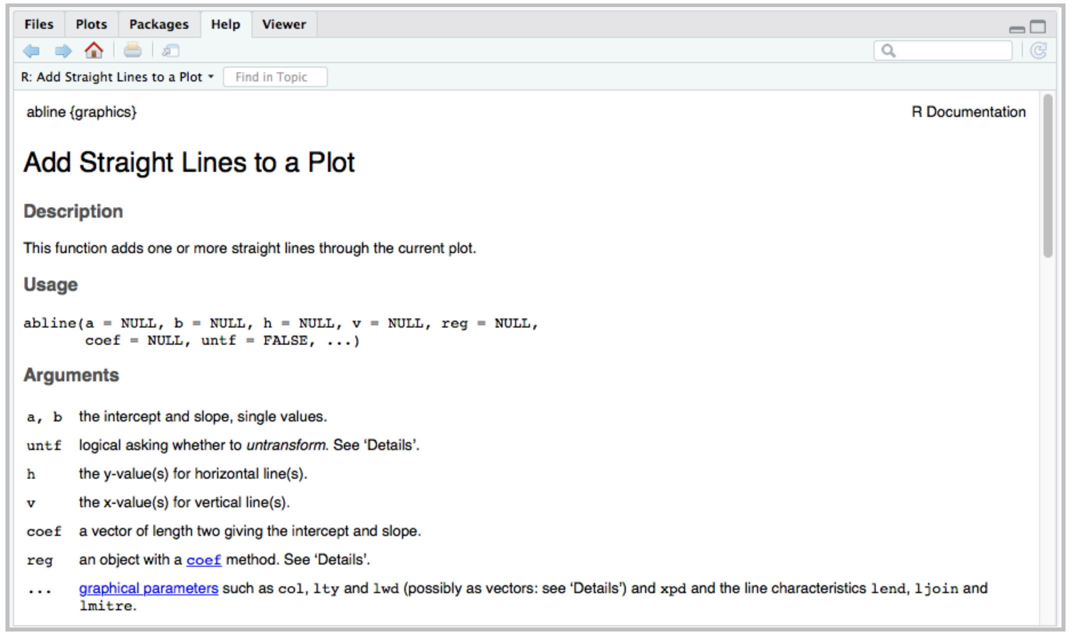
1. Original command call we gave the function
2. Coefficients of the least squares line fit
    i. The first one is the **intercept** (i.e. the value on the y-axis where the line crosses it)
    ii. The second one is the **slope** of the fitted line (the rate of change of the independent variable)

Given the slope and the y-intercept we can easily reconstruct the line and plot it
In R you can plot any line given its slope and intercept, by using the `abline` function

Implement: abline

We've already created a scatter plot for the two variables
We can reuse it and just add the `abline` to it

```
?abline
```



abline {graphics} R Documentation

Add Straight Lines to a Plot

Description

This function adds one or more straight lines through the current plot.

Usage

```
abline(a = NULL, b = NULL, h = NULL, v = NULL, reg = NULL,
       coef = NULL, untf = FALSE, ...)
```

Arguments

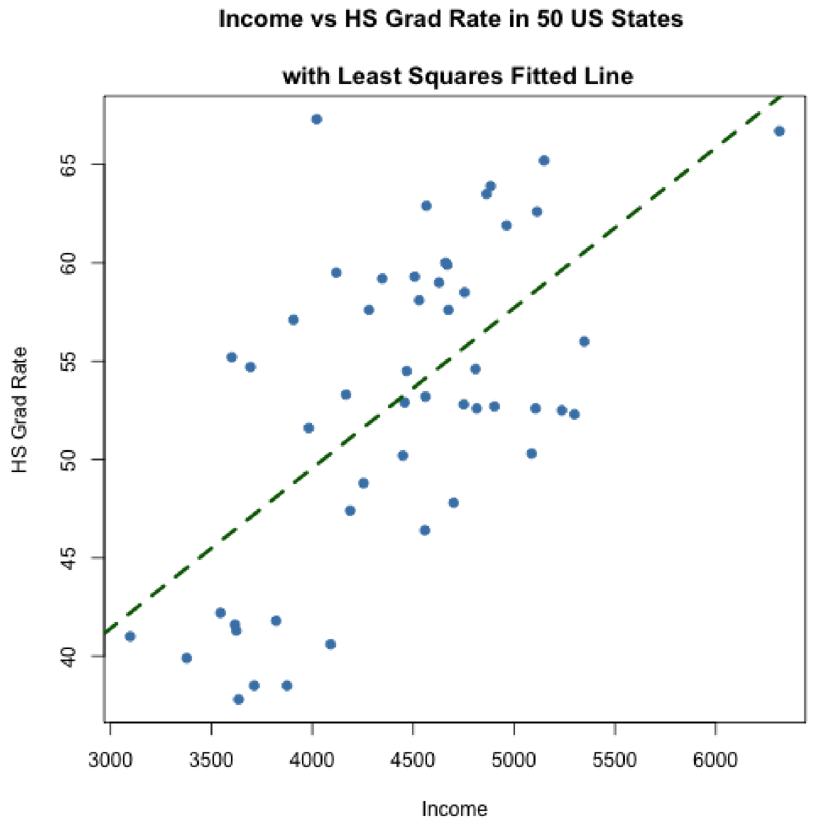
`a, b`	the intercept and slope, single values.
`untf`	logical asking whether to *untransform*. See 'Details'.
`h`	the y-value(s) for horizontal line(s).
`v`	the x-value(s) for vertical line(s).
`coef`	a vector of length two giving the intercept and slope.
`reg`	an object with a `coef` method. See 'Details'.
`...`	graphical parameters such as `col`, `lty` and `lwd` (possibly as vectors: see 'Details') and `xpd` and the line characteristics `lend`, `ljoin` and `lmitre`.

Implement: adding abline to scatterplot

Let's use the scatterplot we built for EDA and add the new line of best fit

```r
# Plotting the data with fitted Least Squares Line.
plot(state_data_sub$Income,
     state_data_sub$`HS Grad`,
     col = "steelblue",
     pch = 19,
     xlab = "Income",
     ylab = "HS Grad Rate",
     main = "Income vs HS Grad Rate in 50 US States
             \n with Least Squares Fitted Line") #<- add a new line to title with `\n`
abline(lin_model$coefficients[1],    #<- intercept from the model
       lin_model$coefficients[2],    #<- slope from the model
       col = "darkgreen",           #<- color of the line
       lwd = 3,                      #<- line width
       lty = 2)                      #<- line type
```

Implement: plot the line of best fit



Knowledge Check 2

Exercise 2

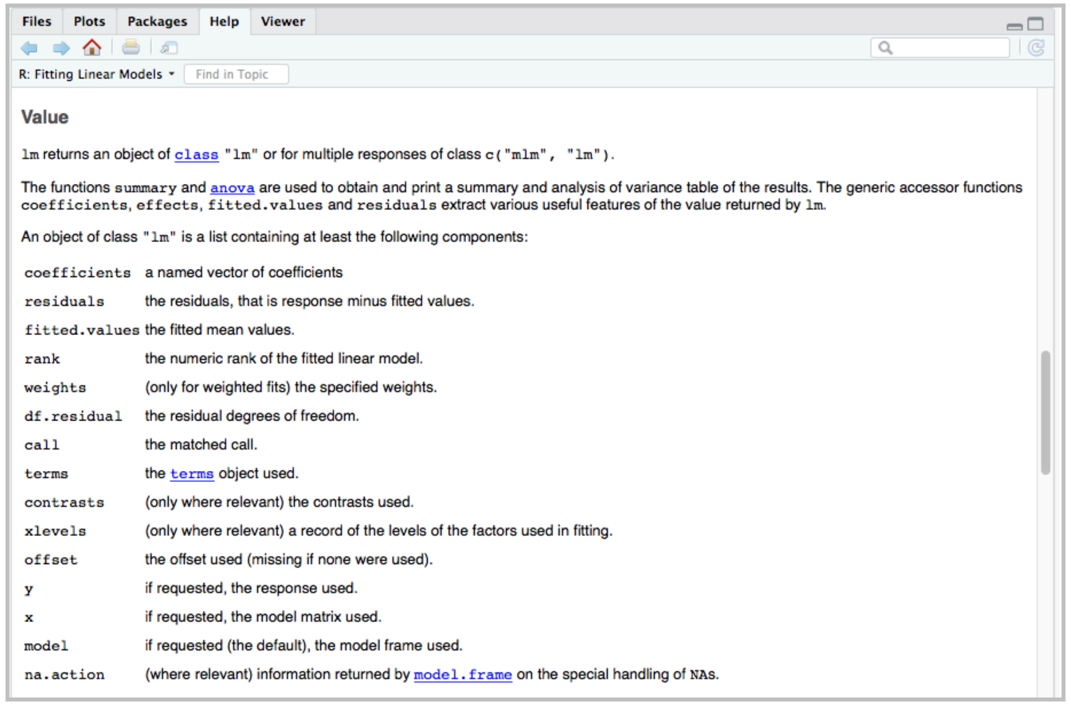
Module completion checklist

Objective	Complete
Summarize why we would want to use linear regression and how it can help in a business setting	✔
Evaluate data using basic statistics such as summary statistics,covariance, correlation	✔
Summarize the basis of linear regression, identify what type of learning method it is	✔
Prepare the data set to run a linear regression model	✔
Recognize what to look for in the data, NAs, variance	✔
Implement and run the linear regression model on the given data	✔
Evaluate the linear model and analyze summary metrics	
Validate the linear regression model by looking at it's assumptions	
Summarize how to implement multiple regression and analyze output	
Evaluate the final model and look for heteroscadacity and colinearity	

Evaluate linear model: summary

Now that we have run the linear regression and plotted out output we want to evaluate our results

We will do this by using `summary(lm)`



Value

`lm` returns an object of class "lm" or for multiple responses of class c("mlm", "lm").

The functions `summary` and anova are used to obtain and print a summary and analysis of variance table of the results. The generic accessor functions `coefficients`, `effects`, `fitted.values` and `residuals` extract various useful features of the value returned by `lm`.

An object of class "lm" is a list containing at least the following components:

`coefficients`	a named vector of coefficients
`residuals`	the residuals, that is response minus fitted values.
`fitted.values`	the fitted mean values.
`rank`	the numeric rank of the fitted linear model.
`weights`	(only for weighted fits) the specified weights.
`df.residual`	the residual degrees of freedom.
`call`	the matched call.
`terms`	the terms object used.
`contrasts`	(only where relevant) the contrasts used.
`xlevels`	(only where relevant) a record of the levels of the factors used in fitting.
`offset`	the offset used (missing if none were used).
`y`	if requested, the response used.
`x`	if requested, the model matrix used.
`model`	if requested (the default), the model frame used.
`na.action`	(where relevant) information returned by model.frame on the special handling of NAs.

Evaluate: using summary

```
# Interpreting the summary of linear model
summary(lin_model)
```

```

Call:
lm(formula = `HS Grad` ~ Income, data = state_data_sub)

Residuals:
    Min      1Q  Median      3Q     Max 
-10.038  -4.774  -1.067   5.022  17.564 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 16.961557   6.665384   2.545   0.0142 *  
Income       0.008149   0.001489   5.474 1.58e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.403 on 48 degrees of freedom
Multiple R-squared:  0.3843,	Adjusted R-squared:  0.3715 
F-statistic: 29.96 on 1 and 48 DF,  p-value: 1.579e-06
```

Evaluate: call

```
summary(lin_model)$call
```

```
lm(formula = `HS Grad` ~ Income, data =
state_data_sub)
```

The first output you see is `Call`

This is the formula that you gave the `lm` function

Check and make sure that your target variable and predictors are correct

Evaluate: residuals

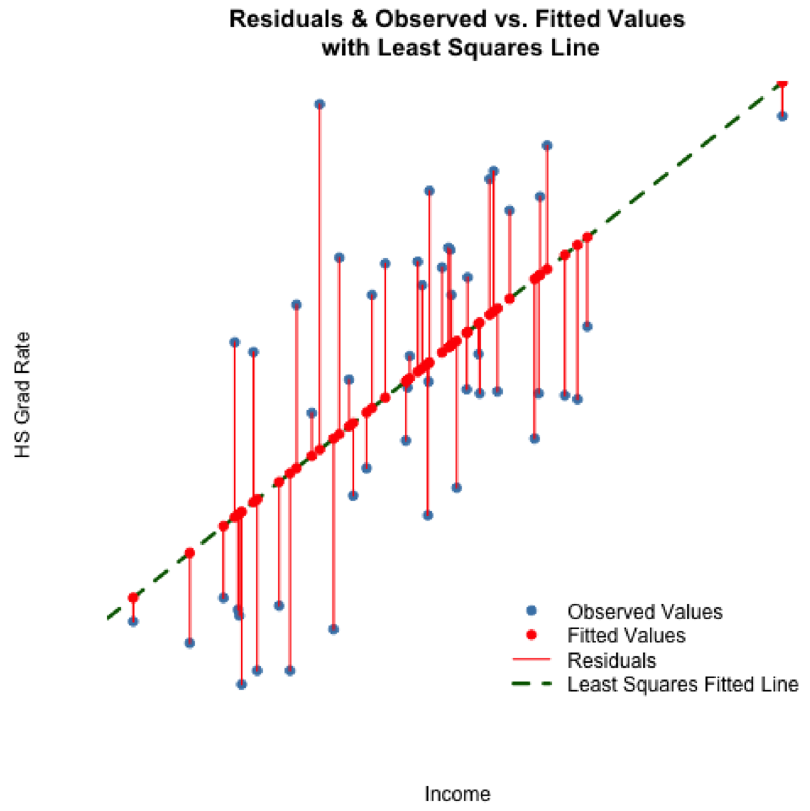
Residuals are a very important part of the evaluation process
Remember the definition of residuals:

- Residual = distance from the actual data points (values of y) to the average expected value of y for every value of x

```
summary(lin_model$residuals)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
-10.038  -4.774  -1.067   0.000   5.022  17.564 
```

summary(lin_model$residuals) shows a 5-statistic summary of model residuals



Evaluate: coefficients estimate

The coefficient in simple linear regression, is represented by two unknown constants that represent the intercept and slope terms

The first of four components is:

- **Estimate**

```
summary(lin_model)$coefficient
```

```
                    Estimate   Std. Error   t value
Pr(>|t|)
(Intercept) 16.961557393 6.665384287 2.544723
1.420708e-02
Income        0.008148799 0.001488701 5.473763
1.578773e-06
```

In simple linear regression, `coefficients` are two **unknown constants that represent the *intercept* and *slope*** terms, which we have covered

The coefficient ***Estimate***:

- Intercept: the expected value of x, in our example `HS Grad` rate, for y to equal 0, e.g. Someone to not have an `Income`
- Slope: the effect x has on y, in our example the effect `HS Grad` rate has on `Income`

Evaluate: coefficients standard error

The second of four components is:

- Std. Error

```
summary(lin_model)$coefficient
```

```
                Estimate  Std. Error  t value
Pr(>|t|)
(Intercept) 16.961557393 6.665384287 2.544723
1.420708e-02
Income       0.008148799 0.001488701 5.473763
1.578773e-06
```

The coefficient ***Standard Error***:

- Each corresponding value measures the average amount that the coefficient estimates vary from the actual average value of our response variable
- Ideally, we want a lower number relative to its coefficients

Evaluate: coefficient t-value

The third component to the coefficient is:

- t value

```
summary(lin_model)$coefficient
```

```
                 Estimate  Std. Error  t value
Pr(>|t|)
(Intercept) 16.961557393 6.665384287 2.544723
1.420708e-02
Income       0.008148799 0.001488701 5.473763
1.578773e-06
```

The coefficient ***t-value***:

- Measure of **how many standard deviations our coefficient estimate is far away from 0**
- The null hypothesis for the t-test is that the coefficient is 0 standard deviations away from 0
- We want to be able to reject the null and say that the coefficient is far away from 0, this allows us to declare there is a relationship between the target and the predictor
- t-values are also then used to compute the p-values

Evaluate: coefficients p-value

The last component to the coefficient is:

- Pr(>|t|)

```
summary(lin_model)$coefficient
```

```
                 Estimate  Std. Error  t value
Pr(>|t|)
(Intercept) 16.961557393 6.665384287 2.544723
1.420708e-02
Income       0.008148799 0.001488701 5.473763
1.578773e-06
```

The coefficient ***Pr(>|t|)***

- This represents the **p-value** or the **probability of observing any value equal or larger than *t***
- A small p-value indicates it is unlikely the relationship between x and y is based on chance, we want a small p-value
- Typically a p-value of under 5% is a good cut-off point, but this all matters on the confidence interval that has been set

Evaluate: residual standard error

Residual standard error measures the quality of a linear regression fit

Below are the two components of residual standard error

```
# Residual standard error
summary(lin_model)$sigma
```

```
[1] 6.403336
```

```
# Degrees of freedom (total rows - number of
variables)
summary(lin_model)$df
```

```
[1]  2 48  2
```

In theory, every linear model is assumed to contain an error term `E`

The residual standard error is the **average amount that the target variable will deviate from the true regression line**

The residual standard error should not be 0, this means its a perfect model and most probably overfit

If the residual standard error can't be shown then the model probably does not have any predictive ability

Evaluate: r-squared

R-squared and adjusted r-squared are the next two components to discuss

```
# R-squared
summary(lin_model)$r.squared
```

```
[1] 0.3843161
```

R-squared is a statistic that provides a **measure of how well the model is fitting the actual data**

It is the measure of the linear relationship between the predictor and the target

It always lies between 0 and 1

The way to interpret an r-squared would be

- An r-squared of .38 means that around 38% of the variance found in the target variable can be explained by the predictor

It is hard to say what a "good r-squared" is

It depends heavily on the subject matter and application

Evaluate: Adjusted r-squared

Adjusted r-squared is r-squared that accounts for multiple predictors

```
# Adjusted r-squared
summary(lin_model)$adj.r.squared
```

```
[1] 0.3714893
```

Adjused r-squared is the same measure as r-squared

In multiple regression, r-squared will constantly increase with added variables

Adjusted r-squared will adjust for additional variables

Evaluate: f-statistic

The **f-statistic** is a good indicator of whether there is a relationship between our predictor and the target variable

```
summary(lin_model)$fstatistic
```

```
   value     numdf     dendf 
29.96208  1.00000 48.00000 
```

The further the F-statistic is from 1 the better it is ***
F-statistic in regression is used to assess the overall performance of the model
It is usually used in combination with the p-value
A low p-value (usually below 5%) will indicate we can reject the null hypothesis of the F-test which says that the regression model with 0 predictors is equal to the current regression model

Final evaluation of summary metrics

We have reviewed each component, now let's evaluate them all together and decide whether or not to move on to the next step of evaluating this simple linear regression model

```
summary(lin_model)
```

```
Call:
lm(formula = `HS Grad` ~ Income, data =
state_data_sub)

Residuals:
    Min      1Q  Median      3Q     Max
-10.038  -4.774  -1.067   5.022  17.564

Coefficients:
             Estimate Std. Error t value
Pr(>|t|)
(Intercept) 16.961557   6.665384   2.545
0.0142 *
Income       0.008149   0.001489   5.474 1.58e-
06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05
'.' 0.1 ' ' 1

Residual standard error: 6.403 on 48 degrees of
freedom
Multiple R-squared:  0.3843,    Adjusted R-
squared:  0.3715
F-statistic: 29.96 on 1 and 48 DF,  p-value:
1.579e-06
```

Final evaluation: part 1

```
Call:
lm(formula = `HS Grad` ~ Income, data =
state_data_sub)

Residuals:
    Min       1Q   Median       3Q      Max
-10.038   -4.774  -1.067    5.022  17.564

Coefficients:
             Estimate Std. Error t value
Pr(>|t|)
(Intercept) 16.961557   6.665384   2.545
0.0142 *
Income       0.008149   0.001489   5.474 1.58e-
06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05
'.' 0.1 ' ' 1

Residual standard error: 6.403 on 48 degrees of
freedom
Multiple R-squared:  0.3843,    Adjusted R-
squared:  0.3715
F-statistic: 29.96 on 1 and 48 DF,  p-value:
1.579e-06
```

Residuals: this is the one metric we dive much deeper into when looking at assumptions

Coefficients: We look over all four components of the two coefficients and determine that the intercept (`HS Grad`) and `Income` are significant and should be kept in the model

Residual standard error: the residual standard error is not 0 and is showing, so our model is not perfect but is able to predict

Final evaluation: part 2

```
Call:
lm(formula = `HS Grad` ~ Income, data =
state_data_sub)

Residuals:
    Min       1Q  Median       3Q      Max
-10.038  -4.774  -1.067   5.022  17.564

Coefficients:
             Estimate Std. Error t value
Pr(>|t|)
(Intercept) 16.961557   6.665384   2.545
0.0142 *
Income       0.008149   0.001489   5.474 1.58e-
06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05
'.' 0.1 ' ' 1

Residual standard error: 6.403 on 48 degrees of
freedom
Multiple R-squared:  0.3843,    Adjusted R-
squared:  0.3715
F-statistic: 29.96 on 1 and 48 DF,  p-value:
1.579e-06
```

Multiple r-squared: the r-squared of .38 means that around 38% of the variance found in `HS Grad` can be explained by `Income`

Adjusted r-squared: Adjusted r-squared does not matter as much here since we are dealing with single linear regression and it is most helpful once multiple variables are added into the model

F-statistic: we see a significant p-value of .000001579 so we are able to reject the null hypotheses that the intercept only model equal the predictor and intercept model and say that this model does have predictive power

Assumptions of linear regression

We ran and analyzed our model: good fit based off the `summary` function

We now double check that the model meets the assumptions of linear regression

Simple linear regression assumptions must be tested for, otherwise this method is not appropriate for the given dataset

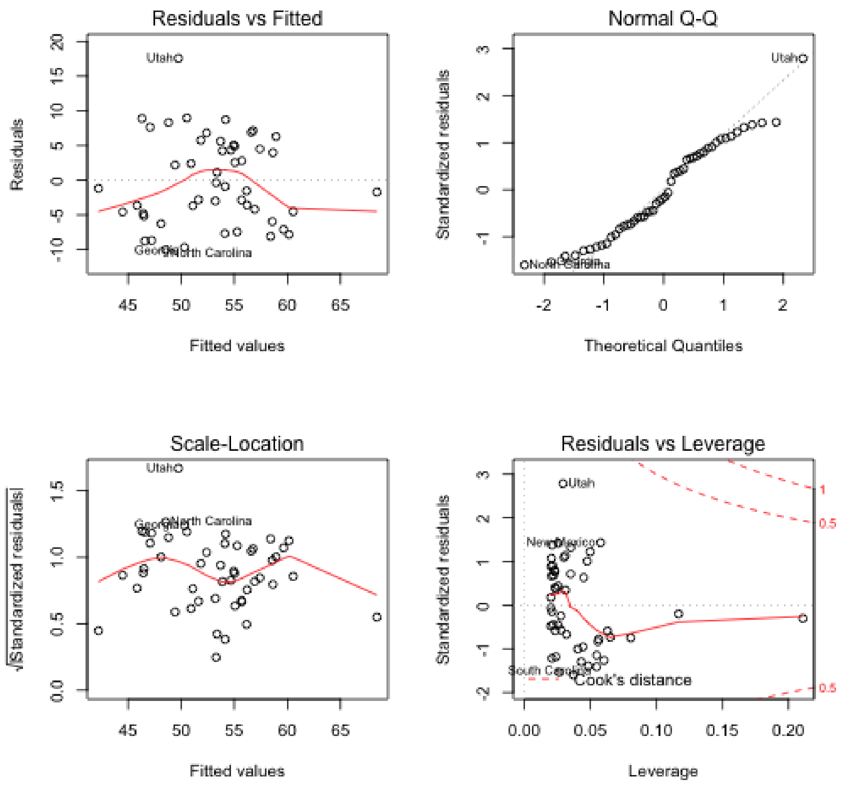
They can be abbreviated into the acronym **'LINE'**

- The relationship between the predictor and response variable must be **L**inear
- The residuals must be **I**ndependent
- The residuals must be **N**ormally distributed
- The residuals must have **E**qual variance

We will now review how to use the residuals to see how well or poorly linear regression fits the data

Assumptions: plot

We can use the simple `plot()` function on the `lm` object to spit out four plots:



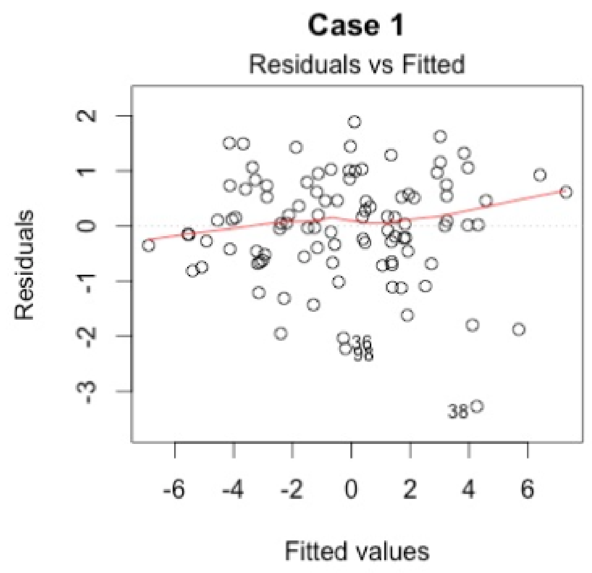
Linear

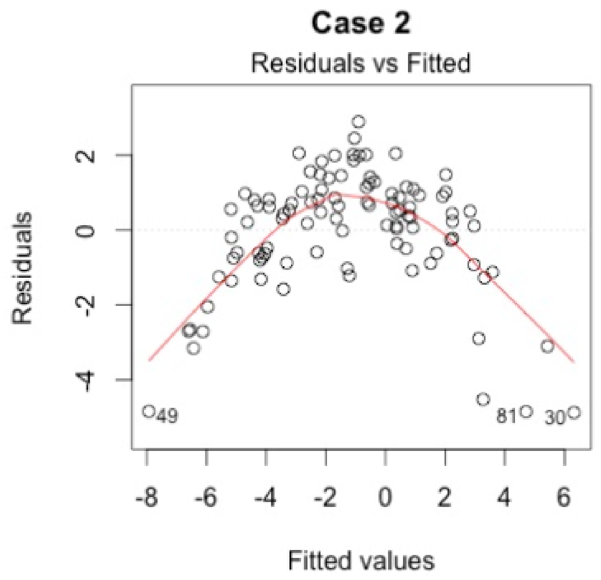
The first assumption is:

The relationship between the predictor and response variables must be [L]inear

Meets assumption

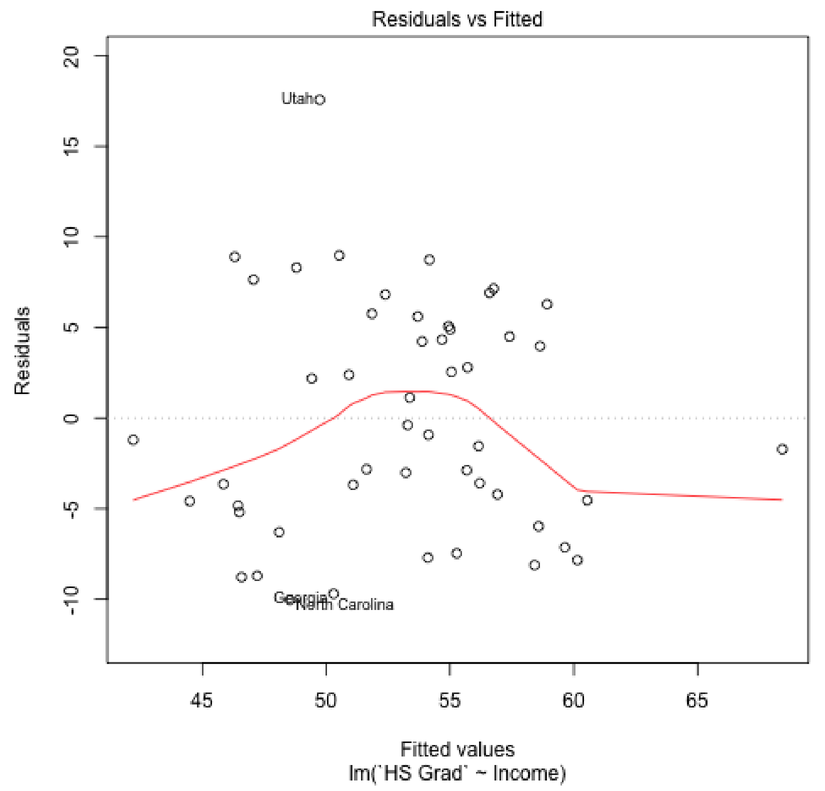
Does not meet assumption





Meets assumpion: Linear satisfied

Our simple linear regression model **Meets assumption**
It does not meet the assumption perfectly but is not extremely non-linear

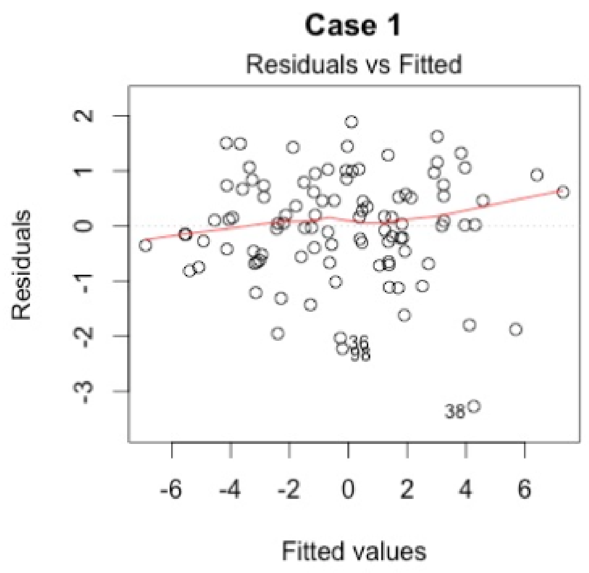


Independent residuals

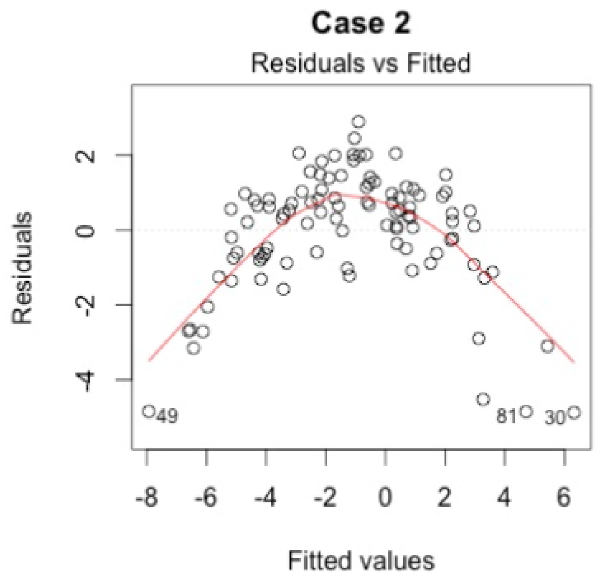
The second assumption is:

The residuals must be [I]ndependent

Meets assumption



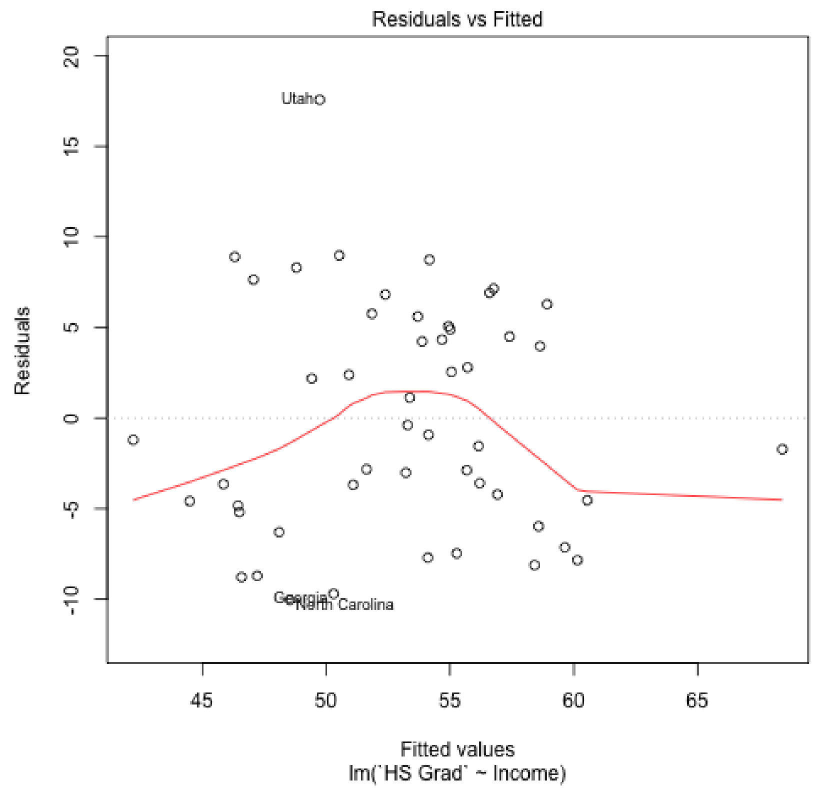
Does not meet assumption



Meets assumption: Independence satisfied

Our simple linear regression model **Meets assumption**

It looks like the residuals are randomly scattered, therefore illustrating independence



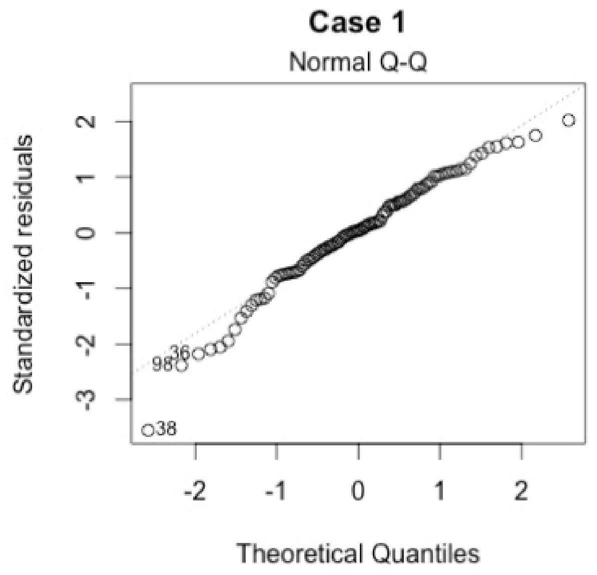
Normally distributed residuals

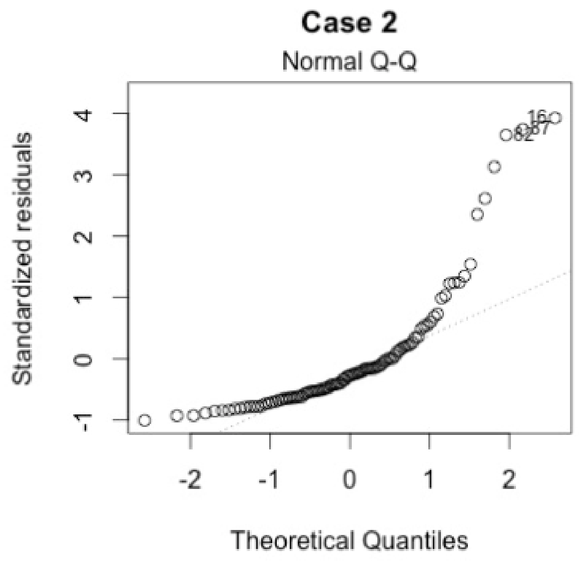
The third assumption is:

The residuals must be [N]ormally ditributed

Meets assumption

Does not meet assumption

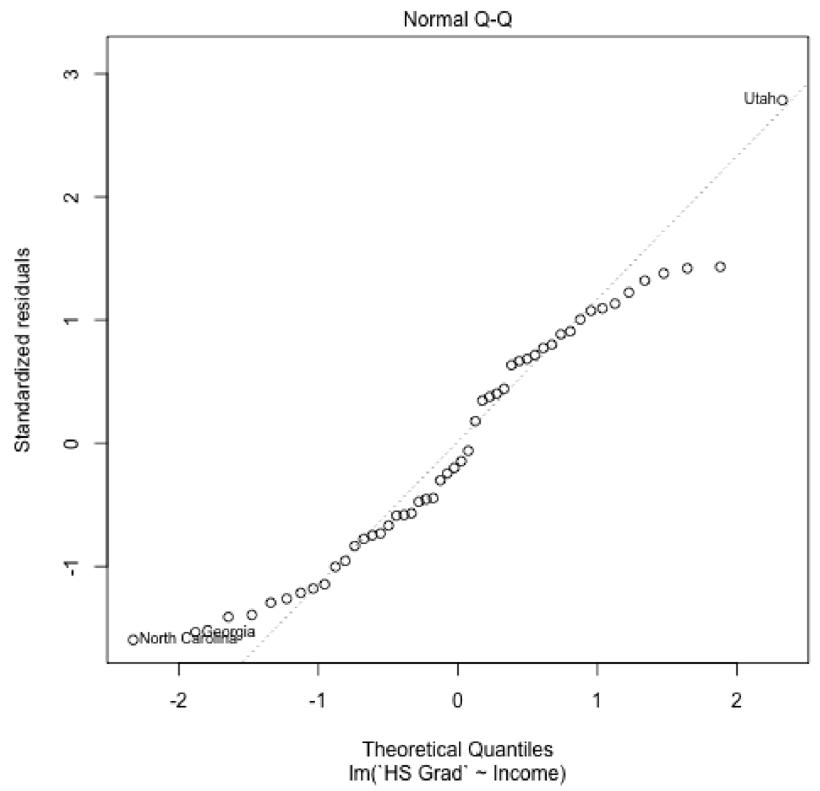




Meets assumption: Normality satisfied

Our simple linear regression model **Meets assumption**

It looks like the residuals follow the straight line quite well

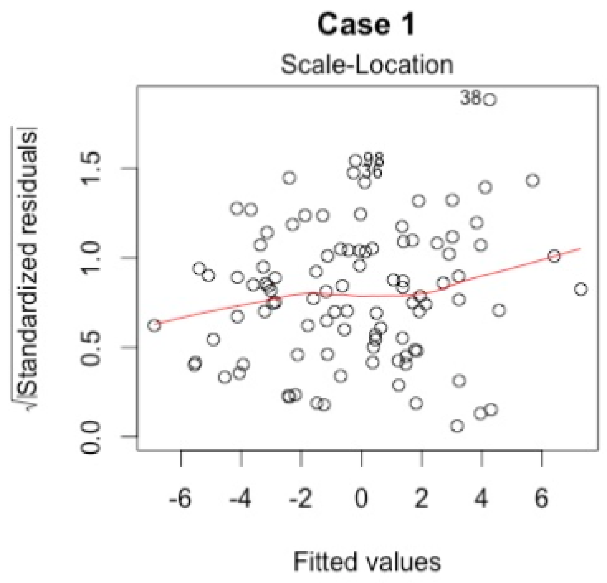


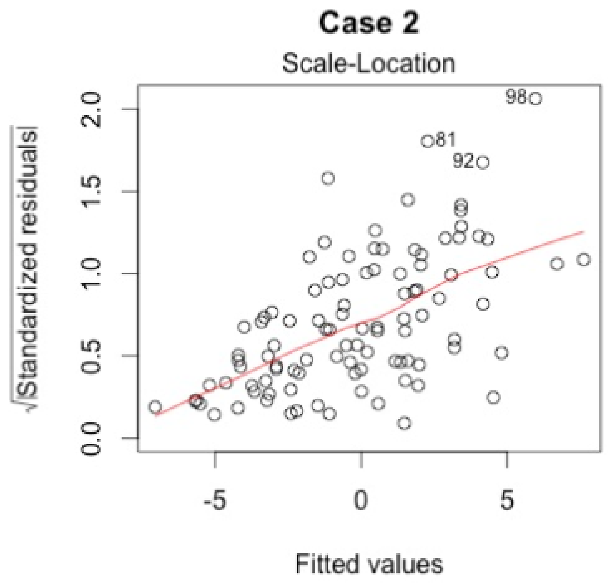
Equal residual variance

The fourth assumption is:

The residuals must have [E]qual variance

Meets assumption **Does not meet assumption**



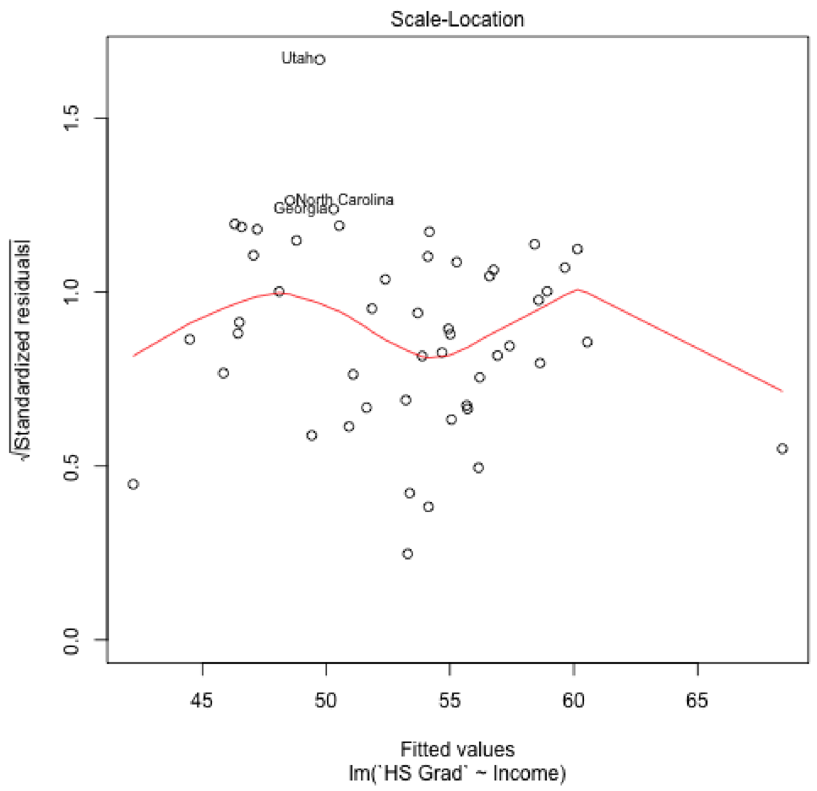


Meets assumption: Equal residual variance satisfied

Our simple linear regression model **Meets assumption**

It looks like the residuals are spread equally along the line

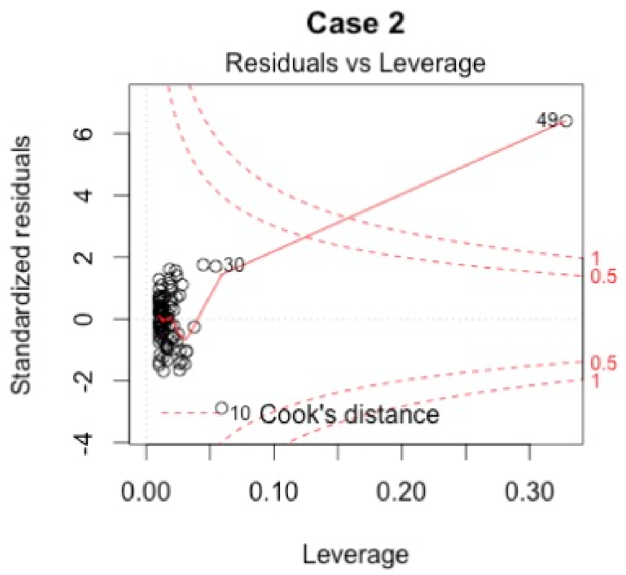
This symbolizes equal variance, also commonly referred to as homoscedasticity



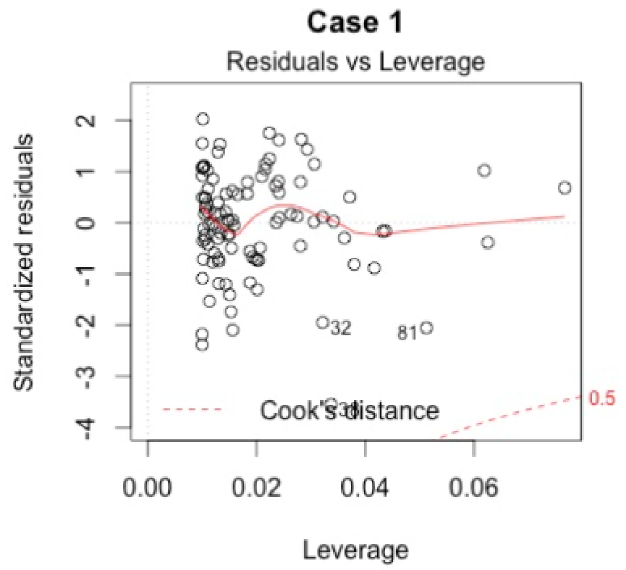
Influential cases: residuals vs. leverage

We just validated our four assumptions
The fourth plot of the `plot(lm)` looks for outliers
We look at the residuals vs. leverage plots to find influential cases

Influential cases

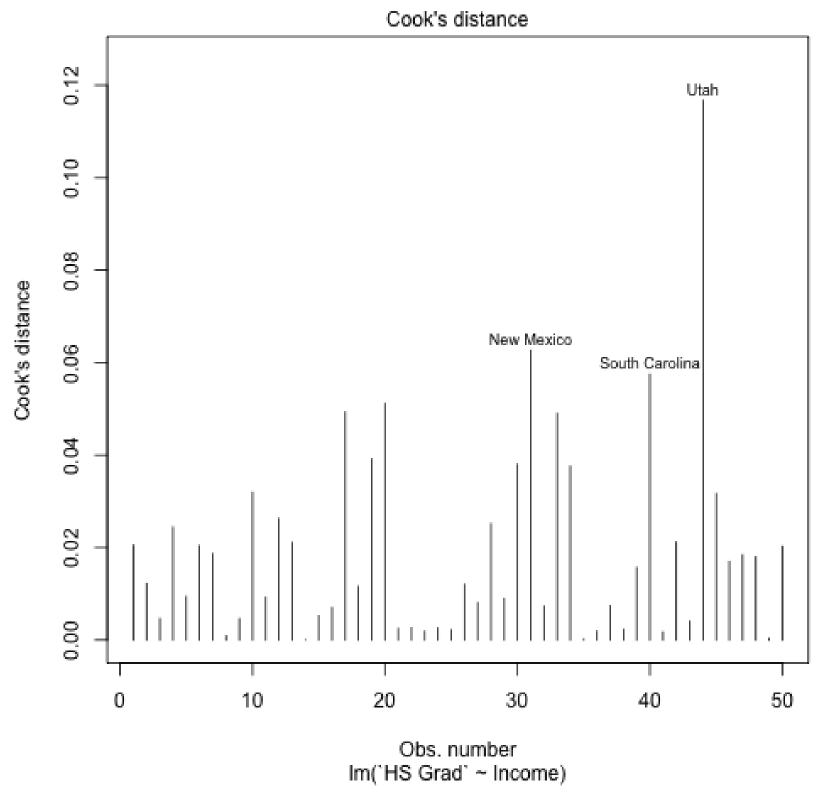


No influential cases



Influential cases: no highlighted outliers

Our residuals are spread out more than the 'non-influential' case
However, none of them are past cook's distance of labeling them as outliers
We note that this is something to look for and move forward



Importance of checking assumptions and outliers

We can see if it makes sense to use a linear approach

- There may be more in the data
- We can use a more complex model

This is the final step in the go/no go test for linear regression

We have analyzed and validated our simple linear regression model

Now we can move to `multiple regression`

Knowledge Check 3

Exercise 3

Module completion checklist

Objective	Complete
Summarize why we would want to use linear regression and how it can help in a business setting	✔
Evaluate data using basic statistics such as summary statistics,covariance, correlation	✔
Summarize the basis of linear regression, identify what type of learning method it is	✔
Prepare the data set to run a linear regression model	✔
Recognize what to look for in the data, NAs, variance	✔
Implement and run the linear regression model on the given data	✔
Evaluate the linear model and analyze summary metrics	✔
Validate the linear regression model by looking at it's assumptions	✔
Summarize how to implement multiple regression and analyze output	
Evaluate the final model and look for heteroscadacity and colinearity	

Multiple regression

The concept of multiple linear regression is really not that different from simple linear regression

Instead of a single predictor variable we just have `at least two predictor variables`

We still have one target variable that is continuous

Multiple regression: methodology

Mostly, multiple linear regression follows the same parameters of simple linear regression

Stays the same as single regression

We still follow the slope intercept model from single linear regression - $y = mx + b$

We still have one coefficient for intercept (b)

Summary function of `lm` measures the same thing

We still follow LINE for assumptions of linear regression

Introduced for multiple regression

Instead of finding 1 coefficient for slope, we will find multiple

The number of coefficients for slope depends directly on how many predictors we use

One additional assumption for multiple regression, the predictors are not correlated

Multiple regression: improving performance

Remember the model we built for single variable regression, looking at the relationship between HS Grad and Income

We concluded that this relationship was linear

We also saw from the r2 value, that only around 38% of the variance was explained

Let's try and improve this model by adding in one more variable to predict HS Grad rate

Multiple regression: adding a variable

```
# Let's look at the original state data
head(state_data)
```

```
           Population Income Illiteracy Life Exp Murder HS Grad Frost
Alabama          3615   3624        2.1    69.05   15.1    41.3    20
Alaska            365   6315        1.5    69.31   11.3    66.7   152
Arizona          2212   4530        1.8    70.55    7.8    58.1    15
Arkansas         2110   3378        1.9    70.66   10.1    39.9    65
California      21198   5114        1.1    71.71   10.3    62.6    20
Colorado         2541   4884        0.7    72.06    6.8    63.9   166
             Area
Alabama     50708
Alaska     566432
Arizona    113417
Arkansas    51945
California 156361
Colorado   103766
```

```
# We are going to add the variable Life Exp to our subset
# and run a multiple regression
state_data_sub$`Life Exp` = state_data$`Life Exp`
```

Data Cleaning: NAs

Remember we have to clean our data before running the model

We've already looked at the presence of NAs in the `Income` and `HS Grad` columns

Let's look at the `Life Exp` column now

We can either run the function we wrote earlier, or just simply check for NAs within the `Life Exp` variable

```
# Check which entry(s) in in `Life Exp` column are NAs.
which(is.na(state_data_sub$`Life Exp`) == TRUE)
```

```
integer(0)
```

We don't have any NAs, so we can proceed to checking for zero/near zero variance

Data Cleaning: near zero variance

Check if `Life Exp` has near-zero or zero variance and save the metrics

```
nearZeroVar(state_data_sub$`Life Exp`,
            saveMetrics = TRUE)
```

```
  freqRatio percentUnique zeroVar   nzv
1         1            94   FALSE FALSE
```

We see we are good to move forward, no NAs and there is variance

Build a multiple regression model

Setting up multiple regression in R is very similar to the way you'd set up simple linear regression
The only difference would be the presence of several (at least two) predictor variables in the model

```
# The formula for `n` predictor variables now looks like this:
# response_var ~ predictor_var_1 + predictor_var_2 + ... + predictor_var_n

# Build a multiple linear regression model using Income and `Life Exp` as
# predictor variables and `HS Grad` as the response variable.
mult_lin_model = lm(`HS Grad`~ Income + `Life Exp`, #<- formula with 2 predictor & 1 response variables
                    data = state_data_sub)                 #<- dataset

# Check the model's coefficients.
mult_lin_model
```

```

Call:
lm(formula = `HS Grad` ~ Income + `Life Exp`, data = state_data_sub)

Coefficients:
(Intercept)       Income   `Life Exp`
 -1.538e+02    6.271e-03    2.526e+00
```

Evaluate multiple regression: model performance

Remember walking through all the `summary(lm)` metrics?

We are now going to look at the summary of the multiple regression model and evaluate the model

```
summary(mult_lin_model)
```

```

Call:
lm(formula = `HS Grad` ~ Income + `Life Exp`, data = state_data_sub)

Residuals:
     Min       1Q   Median       3Q      Max 
-11.6953  -2.8984  -0.8514   4.2628  12.2900 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.538e+02  4.316e+01  -3.563 0.000854 ***
Income       6.271e-03  1.383e-03   4.536 3.97e-05 ***
`Life Exp`   2.526e+00  6.329e-01   3.992 0.000228 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.592 on 47 degrees of freedom
Multiple R-squared:  0.5402,	Adjusted R-squared:  0.5206 
F-statistic: 27.61 on 2 and 47 DF,  p-value: 1.175e-08
```

Evaluate: model performance part 1

```
summary(mult_lin_model)
```

```

Call:
lm(formula = `HS Grad` ~ Income + `Life Exp`, data =
state_data_sub)

Residuals:
     Min       1Q   Median       3Q      Max
-11.6953  -2.8984  -0.8514   4.2628  12.2900

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.538e+02  4.316e+01  -3.563 0.000854 ***
Income       6.271e-03  1.383e-03   4.536 3.97e-05 ***
`Life Exp`   2.526e+00  6.329e-01   3.992 0.000228 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
' 1

Residual standard error: 5.592 on 47 degrees of freedom
Multiple R-squared:  0.5402,    Adjusted R-squared:  0.5206
F-statistic: 27.61 on 2 and 47 DF,  p-value: 1.175e-08
```

Residuals: once again, this is the one metric we dive much deeper into when looking at assumptions

Coefficients: We look over all four components of the three coefficients and determine that the intercept (`HS Grad`), `Income` and `Life Exp` are significant and should be kept in the model

Residual standard error: the residual standard error is not 0 and is showing, so our model is not perfect but is able to predict

Evaluate: model performance part 2

```
summary(mult_lin_model)
```

```
Call:
lm(formula = `HS Grad` ~ Income + `Life Exp`, data =
state_data_sub)

Residuals:
     Min       1Q   Median       3Q      Max
-11.6953  -2.8984  -0.8514   4.2628  12.2900

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.538e+02  4.316e+01  -3.563 0.000854
***
Income       6.271e-03  1.383e-03   4.536 3.97e-05
***
`Life Exp`   2.526e+00  6.329e-01   3.992 0.000228
***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
0.1 ' ' 1

Residual standard error: 5.592 on 47 degrees of
freedom
Multiple R-squared:  0.5402,    Adjusted R-squared:
0.5206
F-statistic: 27.61 on 2 and 47 DF,  p-value: 1.175e-
08
```

Multiple r-squared: the .54 means that around 54% of the variance found in `HS Grad` can be explained by `Income` and `Life Exp` - our model has **improved!!**

Adjusted r-squared: the adjusted-r squared is lower than the r-squared, but not by too much. Therefore we can conclude the additional variable has improved the model, for now

F-statistic: p-value of 0.00000001175 is **significant** so we are able to reject the null hypotheses and say that this model does have predictive power

Multiple regression: assumptions

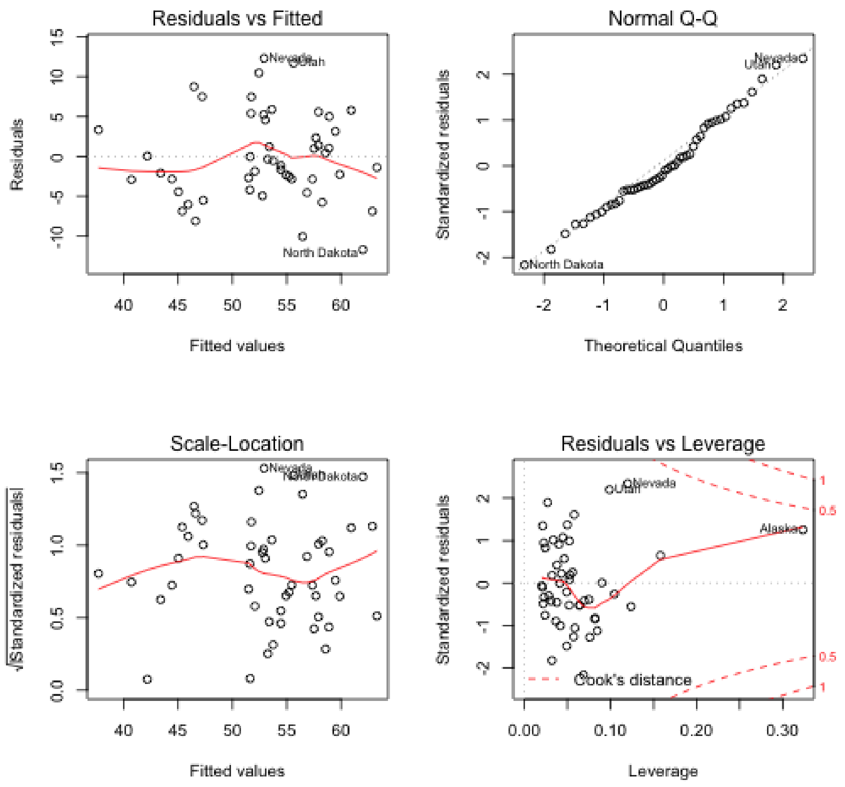
Remember **LINE** from earlier today?
You should by now!

- The relationship between the predictor and response variable must be **L**inear
- The residuals must be **I**ndependent
- The residuals must be **N**ormally distributed
- The residuals must have **E**qual variance

We will now review how to use the residuals to see how well or poorly the multiple linear regression fits the data

Assumptions of multiple regression: plot

We can use the simple `plot()` function on the `lm` object to spit out four plots:



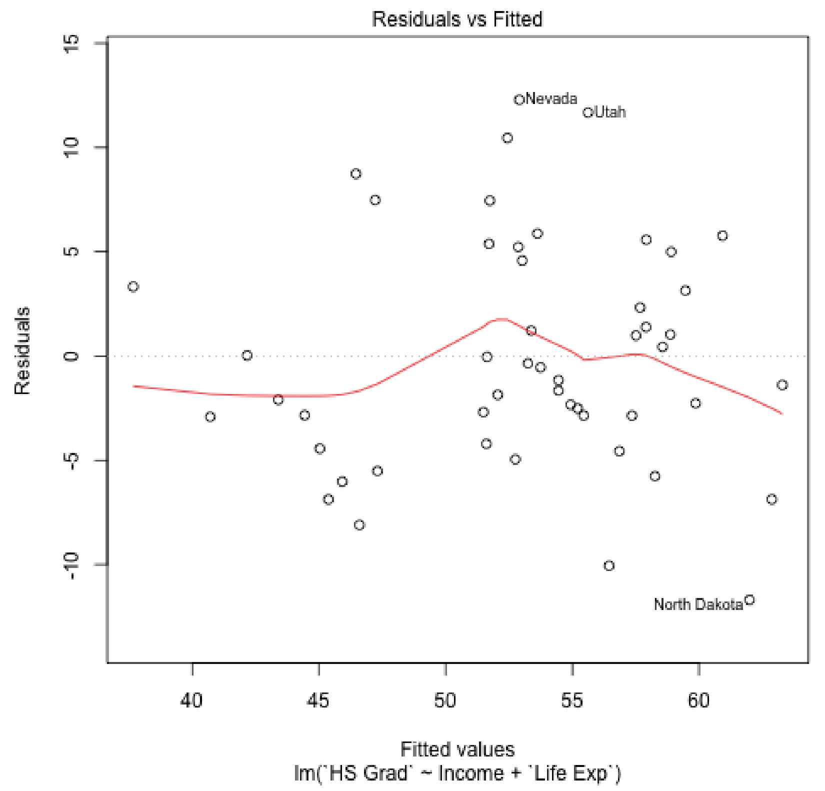
Assumption: linear

Our multiple linear regression model

Meets assumption

It does not meet the assumption perfectly but is not extremely non-linear

It looks a little more linear than our simple linear model

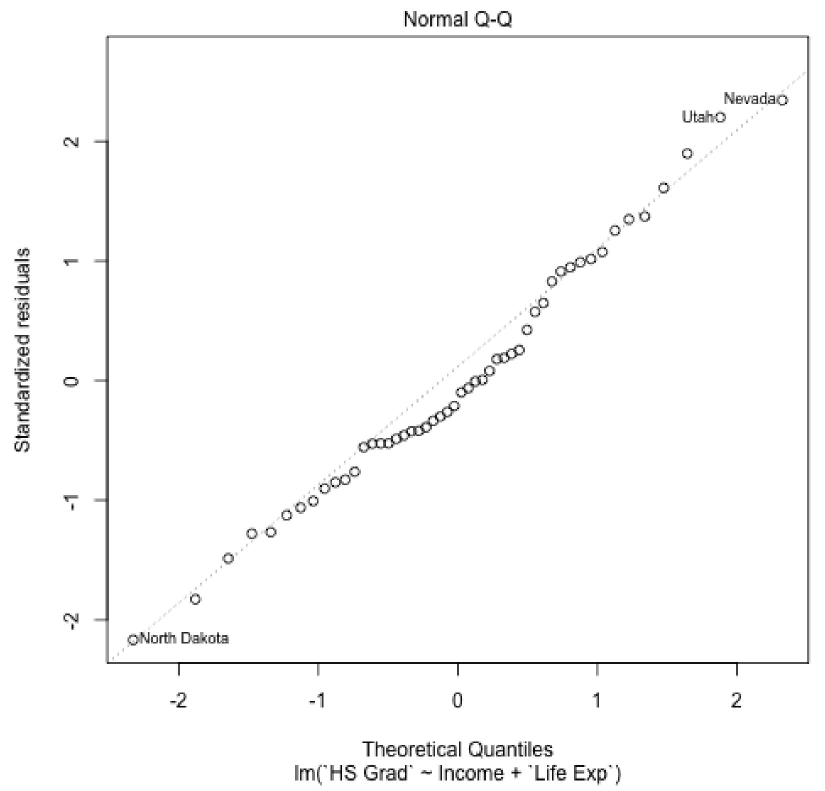


Assumption: normality

Our multiple linear regression model

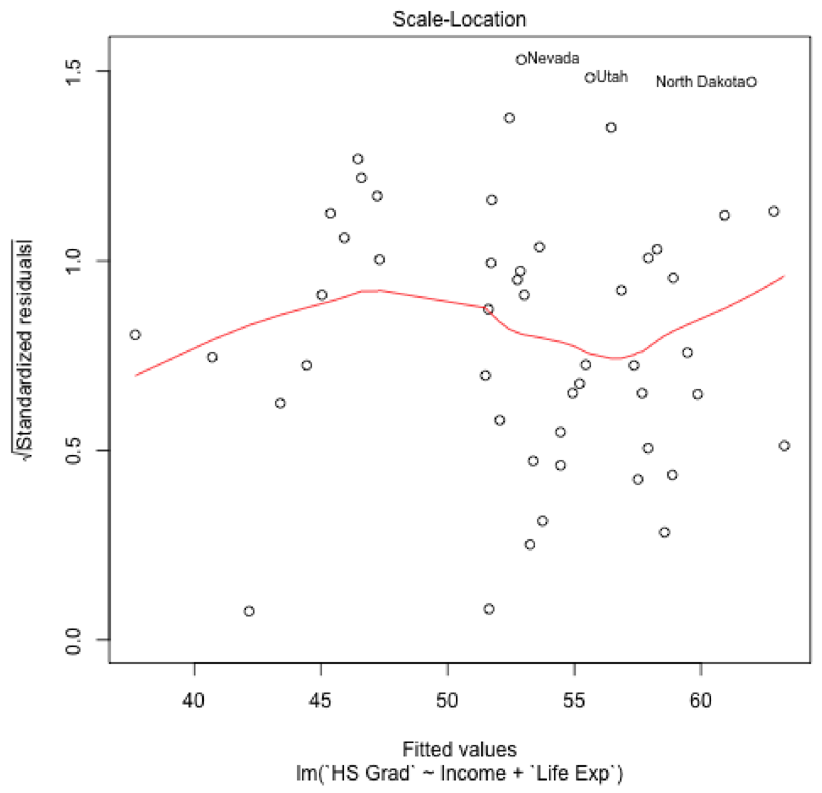
Meets assumption

It looks like the residuals follow the straight line quite well, better than the simple linear model



Assumption: equal variance

Our multiple linear regression model
Meets assumption
It looks like the residuals are spread equally along the line
Once again, this symbolizes equal variance, also commonly referred to as homoscedasticity



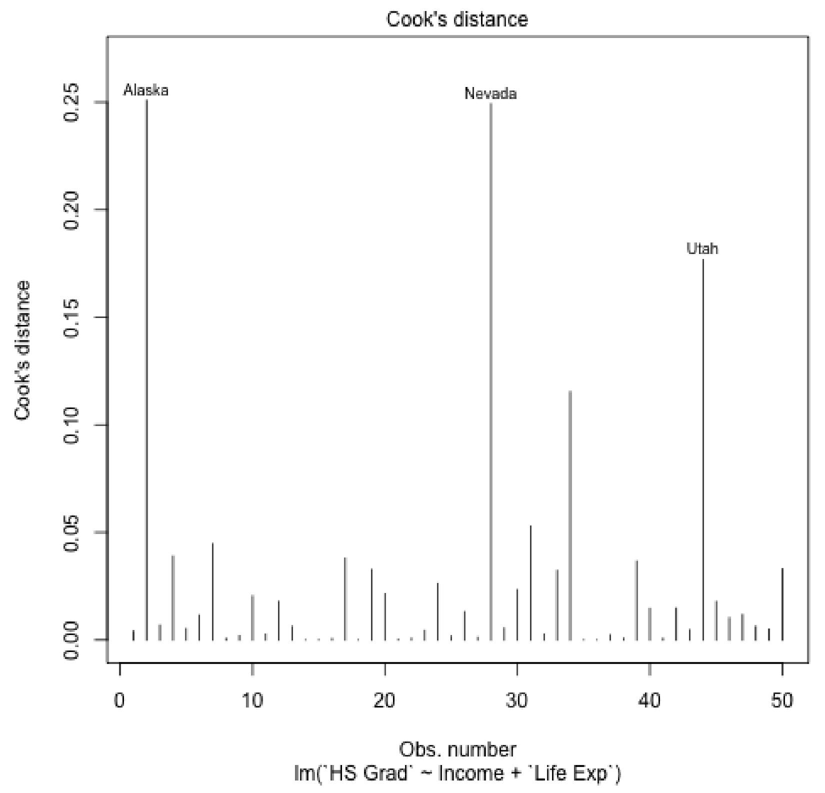
Influential: residuals vs. leverage

Once again, our residuals are spread out more than the 'non-influential' case
However, none of them are past Cook's Distance of labeling them as outliers
We note that this is something to look for and move forward

To remove the outliers:

```
influential = c("Alaska", "Nevada", "Utah")
state_data_sub_no_outlier = state_data_sub[!
(row.names(state_data_sub) %in% influential), ]
```

`Rule of Thumb`: investigate any point over 4/n, where n is the number of observations.



One last thing...

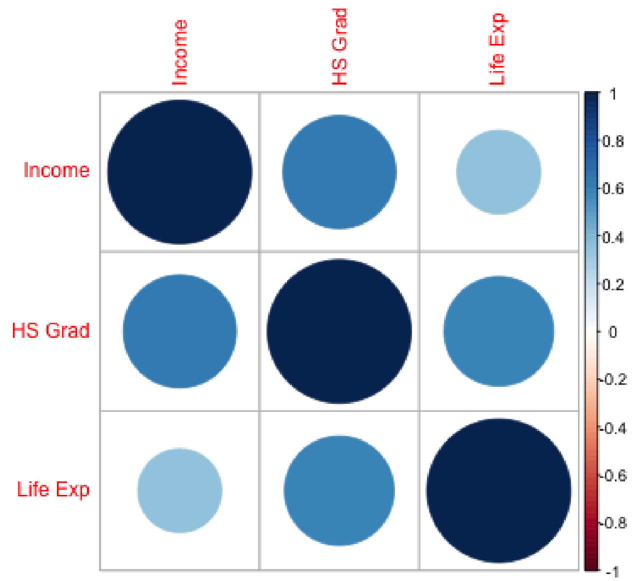
In multiple regression, we MUST check for this last item before validating our model

This is that our predictor variables are uncorrelated

We already looked at the correlation of the variables

Let's look at this

```
cor_matrix = cor(state_data_sub)
corrplot(cor_matrix)
```



Correlated when predicting the target

We can see there is correlation between variables

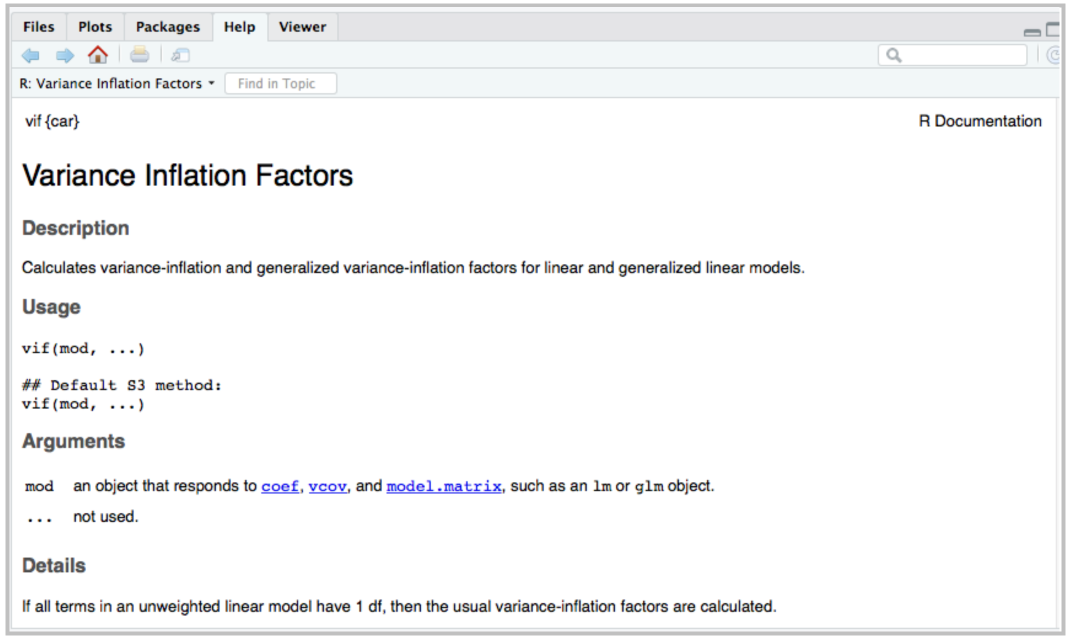
It is unclear if the predictor variables are correlated to each other when predicting the target

For this, we now look at the variance inflation factor (VIF)

Variance inflation factor (VIF)

VIF is obtained using the r-squared value of the regression of that variable against all other explanatory variables
A VIF is calculated for each predictor, those with high values are removed
The definition of 'high' is somewhat arbitrary but usually between 5-10. We will be defining it as 10
We can use a function from the `car` package which we loaded earlier today

```
?vif

vif(mod,...) #<- mod is an object such as lm or
glm
```



vif {car}   R Documentation

Variance Inflation Factors

Description

Calculates variance-inflation and generalized variance-inflation factors for linear and generalized linear models.

Usage

```
vif(mod, ...)

## Default S3 method:
vif(mod, ...)
```

Arguments

mod	an object that responds to `coef`, `vcov`, and `model.matrix`, such as an `lm` or `glm` object.
...	not used.

Details

If all terms in an unweighted linear model have 1 df, then the usual variance-inflation factors are calculated.

Variance inflation factor: testing the model

Let's see if our model is at risk for colinearity

```
vif(mult_lin_model)
```

```
    Income `Life Exp` 
  1.130932   1.130932 
```

Looks good!

VIF is under 10 for both variables

Module completion checklist

Objective	Complete
Summarize why we would want to use linear regression and how it can help in a business setting	✔
Evaluate data using basic statistics such as summary statistics,covariance, correlation	✔
Summarize the basis of linear regression, identify what type of learning method it is	✔
Prepare the data set to run a linear regression model	✔
Recognize what to look for in the data, NAs, variance	✔
Implement and run the linear regression model on the given data	✔
Evaluate the linear model and analyze summary metrics	✔
Validate the linear regression model by looking at it's assumptions	✔
Summarize how to implement multiple regression and analyze output	✔
Evaluate the final model and look for heteroscadacity and colinearity	✔

Knowledge Check 4

Exercise 4

This completes our module

Congratulations!