Regression Models Course Notes

Xing Su

Contents

Introduction to Regression

- linear regression/linear models → go to procedure to analyze data
- *Francis Galton* invented the term and concepts of regression and correlation
  - he predicted child's height from parents height
- questions that regression can help answer
  - prediction of one thing from another
  - find simple, interpretable, meaningful model to predict the data
  - quantify and investigate variations that are unexplained or unrelated to the predictor → **residual variation**
  - quantify the effects of other factors may have on the outcome
  - assumptions to generalize findings beyond data we have → **statistical inference**
  - **regression to the mean** (see below)

Notation

- regular letters (i.e. X, Y) = generally used to denote **observed** variables
- Greek letters (i.e. μ, σ) = generally used to denote **unknown** variables that we are trying to estimate
- $X_1, X_2, \ldots, X_n$ describes n data points
- $\bar{X}$, $\bar{Y}$ = observed means for random variables X and Y
- $\hat{\beta}_0$, $\hat{\beta}_1$ = estimators for true values of β_0 and β_1

Empirical/Sample Mean

- **empirical mean** is defined as

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

- **centering** the random variable is defined as

$$\tilde{X}_i = X_i - \bar{X}$$

  - mean of $\tilde{X}_i = 0$

Empirical/Sample Standard Deviation & Variance

- **empirical variance** is defined as

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} X_i^2 - n\bar{X}^2\right) \Leftarrow \text{shortcut for calculation}$$

- **empirical standard deviation** is defined as $S = \sqrt{S^2}$
  - average squared distances between the observations and the mean
  - has same units as the data
- **scaling** the random variables is defined as X_i/S
  - standard deviation of $X_i/S = 1$

Normalization

- **normalizing** the data/random variable is defined

$$Z_i = \frac{X_i - \bar{X}}{s}$$

  - empirical mean = 0, empirical standard deviation = 1
  - distribution centered around 0 and data have units = # of standard deviations away from the original mean
    - ***example***: $Z_1 = 2$ means that the data point is 2 standard deviations larger than the original mean
- normalization makes non-comparable data **comparable**

Empirical Covariance & Correlation

- Let (X_i, Y_i) = pairs of data
- **empirical covariance** is defined as

$$Cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y}) = \frac{1}{n-1}\left(\sum_{i=1}^{n} X_i Y_i - n\bar{X}\bar{Y}\right)$$

  - has units of $X\times$ units of Y
- **correlation** is defined as

$$Cor(X,Y) = \frac{Cov(X,Y)}{S_x S_y}$$

  where S_x and S_y are the estimates of standard deviations for the X observations and Y observations, respectively

  - the value is effectively the covariance standardized into a unit-less quantity
  - $Cor(X,Y) = Cor(Y,X)$
  - $-1 \leq Cor(X,Y) \leq 1$
  - $Cor(X,Y) = 1$ and $Cor(X,Y) = -1$ only when the X or Y observations fall perfectly on a positive or negative sloped line, respectively
  - $Cor(X,Y)$ measures the strength of the linear relationship between the X and Y data, with stronger relationships as $Cor(X,Y)$ heads towards -1 or 1
  - $Cor(X,Y) = 0$ implies no linear relationship

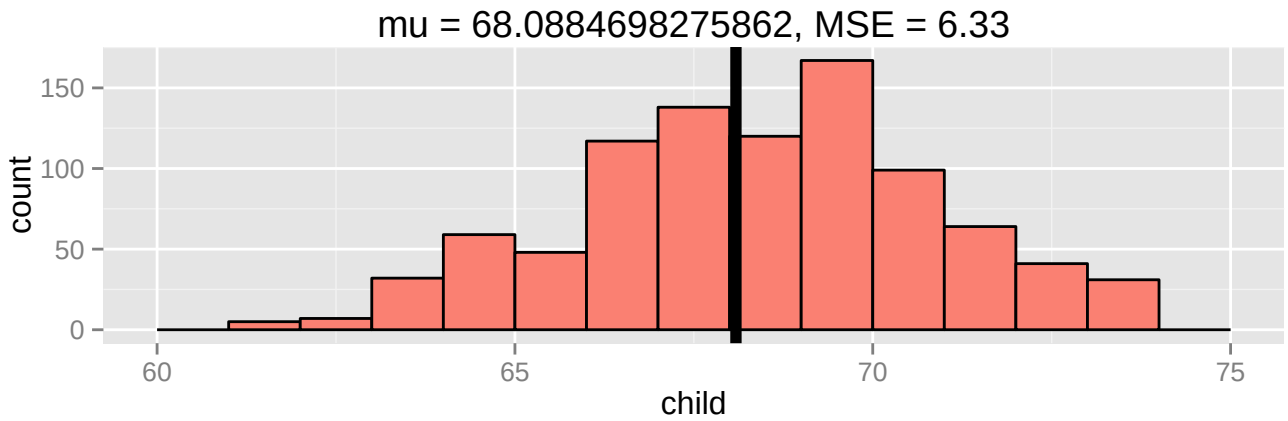
Dalton's Data and Least Squares

- collected data from 1885 in `UsingR` package
- predicting children's heights from parents' height
- observations from the marginal/individual parent/children distributions
- looking only at the children's dataset to find the best predictor
  - "middle" of children's dataset → best predictor
  - "middle" → center of mass → mean of the dataset
    - Let Y_i = height of child i for $i = 1, \ldots, n = 928$, the "middle" = μ such that

$$\sum_{i=1}^{n} (Y_i - \mu)^2$$

    - $\mu = \bar{Y}$ for the above sum to be the smallest → **least squares = empirical mean**
  - ***Note**: `manipulate` function can help to show this*

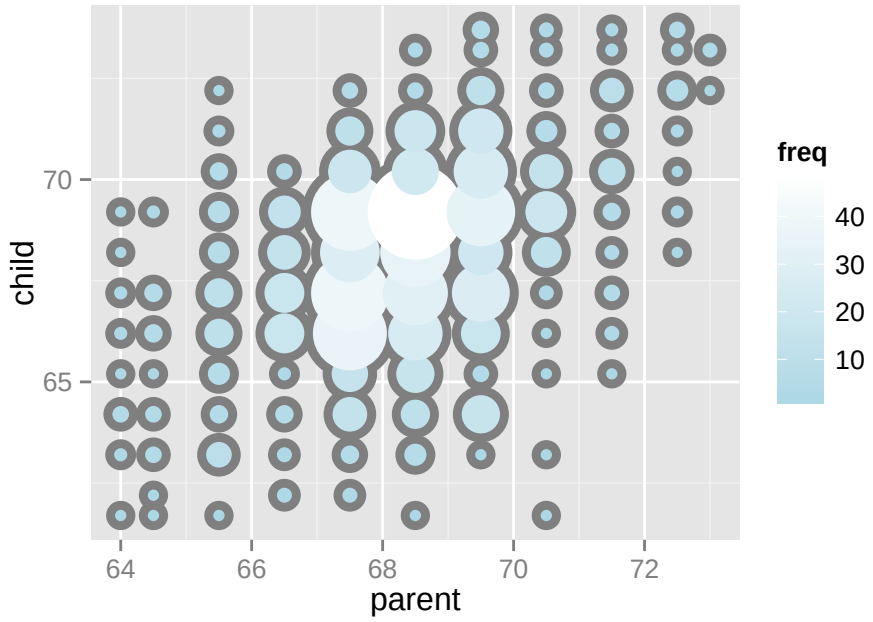
```
# load necessary packages/install if needed
library(ggplot2); library(UsingR); data(galton)
# function to plot the histograms
myHist <- function(mu){
    # calculate the mean squares
    mse <- mean((galton$child - mu)^2)
    # plot histogram
    g <- ggplot(galton, aes(x = child)) + geom_histogram(fill = "salmon",
        colour = "black", binwidth=1)
    # add vertical line marking the center value mu
    g <- g + geom_vline(xintercept = mu, size = 2)
    g <- g + ggtitle(paste("mu = ", mu, ", MSE = ", round(mse, 2), sep = ""))
    g
}
# manipulate allows the user to change the variable mu to see how the mean squares changes
#   library(manipulate); manipulate(myHist(mu), mu = slider(62, 74, step = 0.5))]
# plot the correct graph
myHist(mean(galton$child))
```



- in order to visualize the parent-child height relationship, a scatter plot can be used

- ***Note**: because there are multiple data points for the same parent/child combination, a third dimension (size of point) should be used when constructing the scatter plot*

```
library(dplyr)
# constructs table for different combination of parent-child height
freqData <- as.data.frame(table(galton$child, galton$parent))
names(freqData) <- c("child (in)", "parent (in)", "freq")
# convert to numeric values
freqData$child <- as.numeric(as.character(freqData$child))
freqData$parent <- as.numeric(as.character(freqData$parent))
# filter to only meaningful combinations
g <- ggplot(filter(freqData, freq > 0), aes(x = parent, y = child))
g <- g + scale_size(range = c(2, 20), guide = "none" )
# plot grey circles slightly larger than data as base (achieve an outline effect)
g <- g + geom_point(colour="grey50", aes(size = freq+10, show_guide = FALSE))
# plot the accurate data points
g <- g + geom_point(aes(colour=freq, size = freq))
# change the color gradient from default to lightblue -> $white
g <- g + scale_colour_gradient(low = "lightblue", high="white")
g
```



Derivation for Least Squares = Empirical Mean (Finding the Minimum)

- Let X_i = **regressor/predictor**, and Y_i = **outcome/result** so we want to minimize the the squares:

$$\sum_{i=1}^{n}(Y_i - \mu)^2$$

- Proof is as follows

$$\sum_{i=1}^{n}(Y_i - \mu)^2 = \sum_{i=1}^{n}(Y_i - \bar{Y} + \bar{Y} - \mu)^2 \Leftarrow \text{added} \pm \bar{Y} \text{which is adding 0 to the original equation}$$

$$(expanding\ the\ terms) = \sum_{i=1}^{n}(Y_i - \bar{Y})^2 + 2\sum_{i=1}^{n}(Y_i - \bar{Y})(\bar{Y} - \mu) + \sum_{i=1}^{n}(\bar{Y} - \mu)^2 \Leftarrow (Y_i - \bar{Y}), (\bar{Y} - \mu) \text{ are the terms}$$

$$(simplifying) = \sum_{i=1}^{n}(Y_i - \bar{Y})^2 + 2(\bar{Y} - \mu)\sum_{i=1}^{n}(Y_i - \bar{Y}) + \sum_{i=1}^{n}(\bar{Y} - \mu)^2 \Leftarrow (\bar{Y} - \mu) \text{ does not depend on } i$$

$$(simplifying) = \sum_{i=1}^{n}(Y_i - \bar{Y})^2 + 2(\bar{Y} - \mu)(\sum_{i=1}^{n}Y_i - n\bar{Y}) + \sum_{i=1}^{n}(\bar{Y} - \mu)^2 \Leftarrow \sum_{i=1}^{n}\bar{Y} \text{ is equivalent to } n\bar{Y}$$

$$(simplifying) = \sum_{i=1}^{n}(Y_i - \bar{Y})^2 + \sum_{i=1}^{n}(\bar{Y} - \mu)^2 \Leftarrow \sum_{i=1}^{n}Y_i - n\bar{Y} = 0 \text{ since } \sum_{i=1}^{n}Y_i = n\bar{Y}$$

$$\sum_{i=1}^{n}(Y_i - \mu)^2 \geq \sum_{i=1}^{n}(Y_i - \bar{Y})^2 \Leftarrow \sum_{i=1}^{n}(\bar{Y} - \mu)^2 \text{ is always} \geq 0 \text{ so we can take it out to form the inequality}$$

  - because of the inequality above, to minimize the sum of the squares $\sum_{i=1}^{n}(Y_i - \mu)^2$, $\bar{Y}$ **must be equal to** μ

- An alternative approach to finding the minimum is taking the ***derivative*** with respect to μ

$$\frac{d(\sum_{i=1}^{n}(Y_i - \mu)^2)}{d\mu} = 0 \Leftarrow \text{setting this equal to 0 to find minimum}$$

$$-2\sum_{i=1}^{n}(Y_i - \mu) = 0 \Leftarrow \text{divide by -2 on both sides and move } \mu \text{ term over to the right}$$

$$\sum_{i=1}^{n}Y_i = \sum_{i=1}^{n}\mu \Leftarrow \text{for the two sums to be equal, all the terms must be equal}$$

$$Y_i = \mu$$

Regression through the Origin

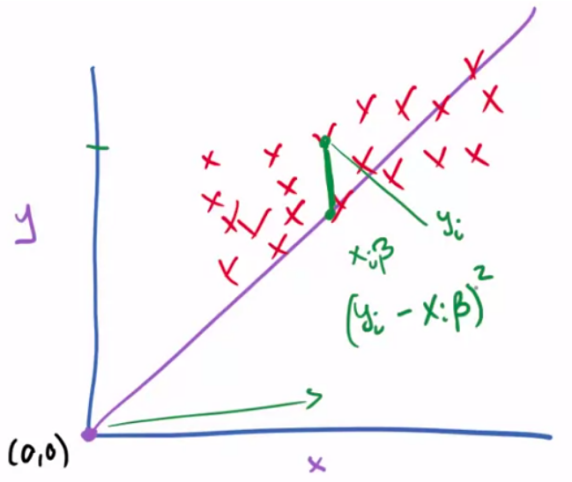
- Let X_i = parents' heights (**regressor**) and Y_i = children's heights (**outcome**)
- find a line with slope β that passes through the origin at (0,0)

$$Y_i = X_i\beta$$

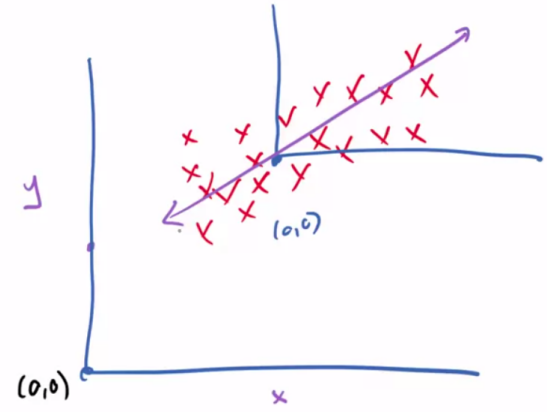
such that it minimizes

$$\sum_{i=1}^{n}(Y_i - X_i\beta)^2$$

- ***Note**: it is generally a bad practice forcing the line through (0, 0)*



- ***Centering Data/Gaussian Elimination***
  - ***Note**: this is **different** from regression through the origin, because it is effectively moving the regression line*
  - subtracting the means from the X_is and Y_is moves the origin (reorienting the axes) to the center of the data set so that a regression line can be constructed
  - ***Note**: the line constructed here has an **equivalent slope** as the result from linear regression with intercept*



Derivation for β

- Let $Y = \beta X$, and $\hat{\beta}$ = estimate of β, the slope of the least square regression line

$$\sum_{i=1}^{n}(Y_i - X_i\beta)^2 = \sum_{i=1}^{n}\Big[(Y_i - X_i\hat{\beta}) + (X_i\hat{\beta} - X_i\beta)\Big]^2 \Leftarrow \text{added } \pm X_i\hat{\beta} \text{ is effectively adding zero}$$

$$(expanding\ the\ terms) = \sum_{i=1}^{n}(Y_i - X_i\hat{\beta})^2 + 2\sum_{i=1}^{n}(Y_i - X_i\hat{\beta})(X_i\hat{\beta} - X_i\beta) + \sum_{i=1}^{n}(X_i\hat{\beta} - X_i\beta)^2$$

$$\sum_{i=1}^{n}(Y_i - X_i\beta)^2 \geq \sum_{i=1}^{n}(Y_i - X_i\hat{\beta})^2 + 2\sum_{i=1}^{n}(Y_i - X_i\hat{\beta})(X_i\hat{\beta} - X_i\beta) \Leftarrow \sum_{i=1}^{n}(X_i\hat{\beta} - X_i\beta)^2 \text{ is always positive}$$

$$(ignoring\ the\ second\ term\ for\ now,\ for\ \hat{\beta}\ to\ be\ the\ minimizer\ of\ the\ squares,\ the\ following\ must\ be\ true)$$

$$\sum_{i=1}^{n}(Y_i - X_i\beta)^2 \geq \sum_{i=1}^{n}(Y_i - X_i\hat{\beta})^2 \Leftarrow \text{every other } \beta \text{ value creates a least square criteria that is } \geq \hat{\beta}$$

$$(this\ means) \Rightarrow 2\sum_{i=1}^{n}(Y_i - X_i\hat{\beta})(X_i\hat{\beta} - X_i\beta) = 0$$

$$(simplifying) \Rightarrow \sum_{i=1}^{n}(Y_i - X_i\hat{\beta})X_i(\hat{\beta} - \beta) = 0 \Leftarrow (\hat{\beta} - \beta) \text{ does not depend on } i$$

$$(simplifying) \Rightarrow \sum_{i=1}^{n}(Y_i - X_i\hat{\beta})X_i = 0$$

$$(solving\ for\ \hat{\beta}) \Rightarrow \hat{\beta} = \frac{\sum_{i=1}^{n} Y_iX_i}{\sum_{i=1}^{n} X_i^2} = \beta$$

- ***example***
  - Let $X_1, X_2, \ldots, X_n = 1$

$$\sum_{i=1}^{n}(Y_i - X_i\beta)^2 = \sum_{i=1}^{n}(Y_i - \beta)^2$$

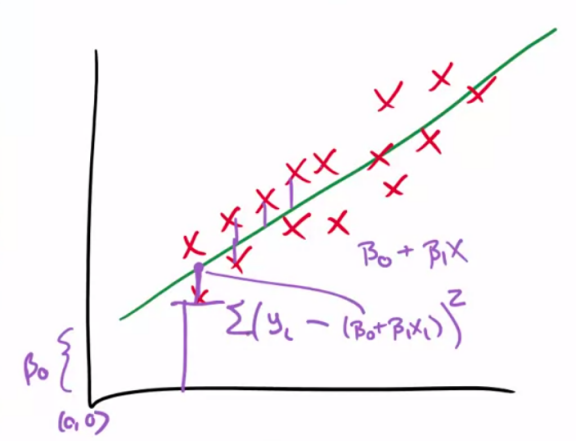
$$\Rightarrow \hat{\beta} = \frac{\sum_{i=1}^{n} Y_iX_i}{\sum_{i=1}^{n} X_i^2} = \frac{\sum_{i=1}^{n} Y_i}{\sum_{i=1}^{n} 1} = \frac{\sum_{i=1}^{n} Y_i}{n} = \bar{Y}$$

  - ***Note****: this is the result from our previous derivation for least squares = empirical mean*

Finding the Best Fit Line (Ordinary Least Squares)

- best fitted line for predictor, X, and outcome, Y is derived from the **least squares**

$$\sum_{i=1}^{n}\{Y_i - (\beta_0 + \beta_1 X_i)\}^2$$



- each of the data point contributes equally to the error between the their locations and the regression line $\rightarrow$ goal of regression is to ***minimize*** this error

Least Squares Model Fit

- model fit $= Y = \beta_0 + \beta_1 X$ through the data pairs (X_i, Y_i) where Y_i as the outcome
  - ***Note**: this is the model that we use to guide our **estimated** best fit (see below)*
- best fit line with estimated slope and intercept (X as predictor, Y as outcome) $\rightarrow$

$$Y = \hat{\beta}_0 + \hat{\beta}_1 X$$

where

$$\hat{\beta}_1 = Cor(Y, X)\frac{Sd(Y)}{Sd(X)} \quad \hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$$

  - [***slope***] $\hat{\beta}_1$ has the units of Y/X
    - $Cor(Y, X)$ = unit-less
    - $Sd(Y)$ = has units of Y
    - $Sd(X)$ = has units of X
  - [***intercept***] $\hat{\beta}_0$ has the units of Y
  - the line passes through the point $(\bar{X}, \bar{Y})$
    - this is evident from equation for β_0 (rearrange equation)
- best fit line with X as outcome and Y as predictor has slope, $\hat{\beta}_1 = Cor(Y, X)Sd(X)/Sd(Y)$.
- slope of best fit line = slope of best fit line through the origin for centered data $(X_i - \bar{X}, Y_i - \bar{Y})$
- slope of best fit line for normalized the data, $\{\frac{X_i - \bar{X}}{Sd(X)}, \frac{Y_i - \bar{Y}}{Sd(Y)}\} = Cor(Y, X)$

Derivation for β_0 and β_1

- Let $Y = \beta_0 + \beta_1 X$, and $\hat{\beta}_0/\hat{\beta}_1$ = estimates β_0/β_1, the intercept and slope of the least square regression line, respectively

$$\sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_i)^2 = \sum_{i=1}^{n}(Y_i^* - \beta_0)^2 \quad where\ Y_i^* = Y_i - \beta_1 X_i$$

$$solution\ for \sum_{i=1}^{n}(Y_i^* - \beta_0)^2 \Rightarrow \hat{\beta}_0 = \frac{\sum_{i=1}^{n} Y_i^*}{n} = \frac{\sum_{i=1}^{n} Y_i - \beta_1 X_i}{n}$$

$$\Rightarrow \hat{\beta}_0 = \frac{\sum_{i=1}^{n} Y_i}{n} - \beta_1 \frac{\sum_{i=1}^{n} X_i}{n}$$

$$\Rightarrow \hat{\beta}_0 = \bar{Y} - \beta_1 \bar{X}$$

$$\Longrightarrow \sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_i)^2 = \sum_{i=1}^{n}\left[Y_i - (\bar{Y} - \beta_1 \bar{X}) - \beta_1 X_i\right]^2$$

$$= \sum_{i=1}^{n}\left[(Y_i - \bar{Y}) - (X_i - \bar{X})\beta_1\right]^2$$

$$= \sum_{i=1}^{n}\left[\tilde{Y}_i - \tilde{X}_i\beta_1\right]^2 \quad where\ \tilde{Y}_i = Y_i - \bar{Y}, \tilde{X}_i = X_i - \bar{X}$$

$$\Rightarrow \hat{\beta}_1 = \frac{\sum_{i=1}^{n} \tilde{Y}_i \tilde{X}_i}{\sum_{i=1}^{n} \tilde{X}_i^{\,2}} = \frac{(Y_i - \bar{Y})(X_i - \bar{X})}{\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

$$\Rightarrow \hat{\beta}_1 = \frac{(Y_i - \bar{Y})(X_i - \bar{X})/(n-1)}{\sum_{i=1}^{n}(X_i - \bar{X})^2/(n-1)} = \frac{Cov(Y,X)}{Var(X)}$$

$$\Rightarrow \hat{\beta}_1 = Cor(Y,X)\frac{Sd(Y)}{Sd(X)}$$

$$\Rightarrow \hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$$

Examples and R Commands

- $\hat{\beta}_0$ and $\hat{\beta}_1$ can be manually calculated through the above formulas
- `coef(lm(y ~ x))` = R command to run the least square regression model on the data with `y` as the outcome, and `x` as the regressor
  - `coef()` = returns the slope and intercept coefficients of the `lm` results

```r
# outcome
y <- galton$child
# regressor
x <- galton$parent
# slope
beta1 <- cor(y, x) * sd(y) / sd(x)
# intercept
beta0 <- mean(y) - beta1 * mean(x)
# results are the same as using the lm command
results <- rbind("manual" = c(beta0, beta1), "lm(y ~ x)" = coef(lm(y ~ x)))
# set column names
colnames(results) <- c("intercept", "slope")
# print results
results
```

```
##           intercept     slope
## manual     23.94153 0.6462906
## lm(y ~ x)  23.94153 0.6462906
```

- slope of the best fit line = slope of best fit line through the origin for centered data
- `lm(y ~ x - 1)` = forces a regression line to go through the origin (0, 0)

```
# centering y
yc <- y - mean(y)
# centering x
xc <- x - mean(x)
# slope
beta1 <- sum(yc * xc) / sum(xc ^ 2)
# results are the same as using the lm command
results <- rbind("centered data (manual)" = beta1, "lm(y ~ x)" = coef(lm(y ~ x))[2],
    "lm(yc ~ xc - 1)" = coef(lm(yc ~ xc - 1))[1])
# set column names
colnames(results) <- c("slope")
# print results
results
```

```
##                            slope
## centered data (manual) 0.6462906
## lm(y ~ x)              0.6462906
## lm(yc ~ xc - 1)        0.6462906
```

- slope of best fit line for normalized the data = $Cor(Y, X)$

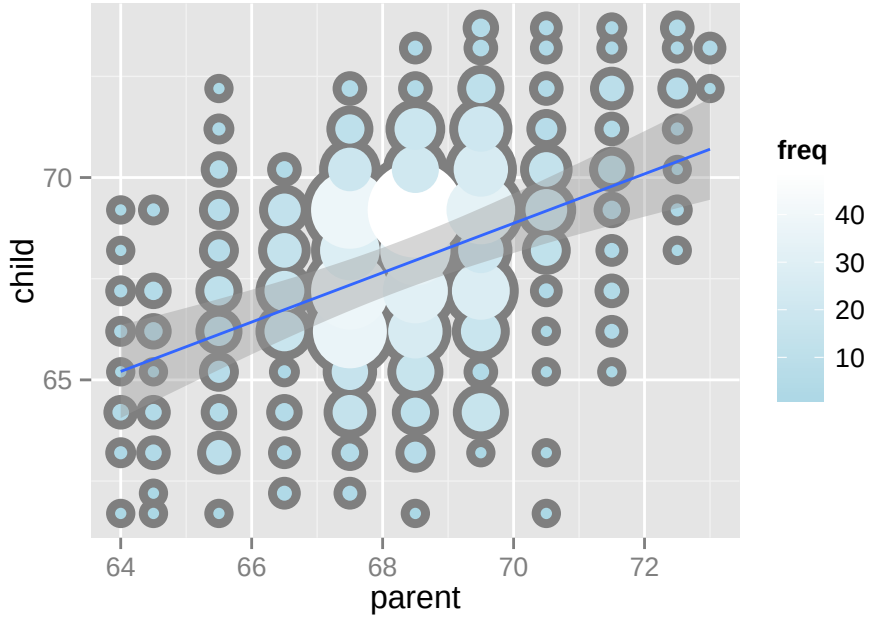
```
# normalize y
yn <- (y - mean(y))/sd(y)
# normalize x
xn <- (x - mean(x))/sd(x)
# compare correlations
results <- rbind("cor(y, x)" = cor(y, x), "cor(yn, xn)" = cor(yn, xn),
    "slope" = coef(lm(yn ~ xn))[2])
# print results
results
```

```
##                    xn
## cor(y, x)   0.4587624
## cor(yn, xn) 0.4587624
## slope       0.4587624
```

- `geom_smooth(method = "lm", formula = y~x)` function in `ggplot2` = adds regression line and confidence interval to graph
  - `formula = y~x` = default for the line (argument can be eliminated if `y~x` produces the line you want)

```
# constructs table for different combination of parent-child height
freqData <- as.data.frame(table(galton$child, galton$parent))
names(freqData) <- c("child (in)", "parent (in)", "freq")
```

```
# convert to numeric values
freqData$child <- as.numeric(as.character(freqData$child))
freqData$parent <- as.numeric(as.character(freqData$parent))
g <- ggplot(filter(freqData, freq > 0), aes(x = parent, y = child))
g <- g + scale_size(range = c(2, 20), guide = "none" )
g <- g + geom_point(colour="grey50", aes(size = freq+10, show_guide = FALSE))
g <- g + geom_point(aes(colour=freq, size = freq))
g <- g + scale_colour_gradient(low = "lightblue", high="white")
g <- g + geom_smooth(method="lm", formula=y~x)
g
```



Regression to the Mean

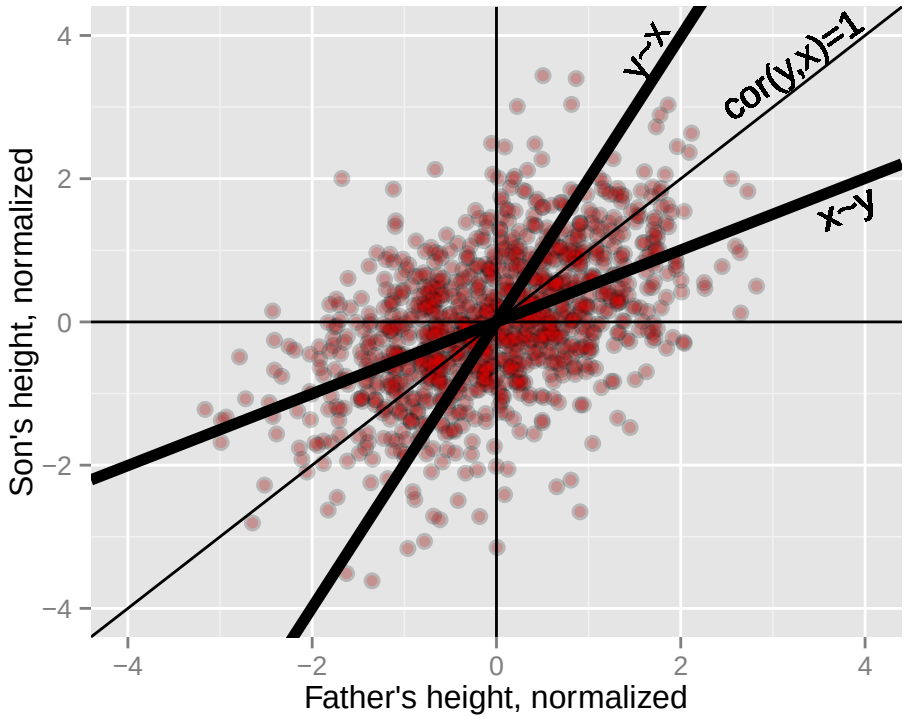
- first investigated by Francis Galton in the paper *"Regression towards mediocrity in hereditary stature" The Journal of the Anthropological Institute of Great Britain and Ireland , Vol. 15, (1886)*
- **regression to the mean** was invented by Fancis Galton to capture the following phenomena
  - children of tall parents tend to be tall, but not as tall as their parents
  - children of short parents tend to be short, but not as short as their parents
  - parents of very short children, tend to be short, but not as short as their child
  - parents of very tall children, tend to be tall, but not as tall as their children
- in thinking of the extremes, the following are true
  - $P(Y < x|X = x)$ gets bigger as x heads to very large values
    - in other words, given that the value of X is already very large (extreme), the chance that the value of Y is as large or larger than that of X is small (unlikely)
  - similarly, $P(Y > x|X = x)$ gets bigger as x heads to very small values
    - in other words, given that the value of X is already very small (extreme), the chance that the value of Y is as small or smaller than that of X is small (unlikely)
- when constructing regression lines between X and Y, the line represents the intrinsic relationship ("mean") between the variables, but does not capture the extremes ("noise")
  - unless $Cor(Y, X) = 1$, the regression line or the intrinsic part of the relationship between variables won't capture all of the variation (some noise exists)

Dalton's Investigation on Regression to the Mean

- both X, child's heights, and Y, parent's heights, are **normalized** so that they mean of 0 and variance of 1
- regression lines must pass $(\bar{X}, \bar{Y})$ or $(0, 0)$ in this case
- slope of regression line = $Cor(Y, X)$ regardless of which variable is the outcome/regressor (because standard deviations of both variables = 1)
  - ***Note**: however, for both regression lines to be plotted on the same graph, the second line's slope must be $1/Cor(Y, X)$ because the two relationships have flipped axes*

```
# load father.son data
data(father.son)
# normalize son's height
y <- (father.son$sheight - mean(father.son$sheight)) / sd(father.son$sheight)
# normalize father's height
x <- (father.son$fheight - mean(father.son$fheight)) / sd(father.son$fheight)
# calculate correlation
rho <- cor(x, y)
# plot the relationship between the two
g = ggplot(data.frame(x = x, y = y), aes(x = x, y = y))
g = g + geom_point(size = 3, alpha = .2, colour = "black")
g = g + geom_point(size = 2, alpha = .2, colour = "red")
g = g + xlim(-4, 4) + ylim(-4, 4)
# reference line for perfect correlation between
# variables (data points will like on diagonal line)
g = g + geom_abline(position = "identity")
# if there is no correlation between the two variables,
# the data points will lie on horizontal/vertical lines
```

```
g = g + geom_vline(xintercept = 0)
g = g + geom_hline(yintercept = 0)
# plot the actual correlation for both
g = g + geom_abline(intercept = 0, slope = rho, size = 2)
g = g + geom_abline(intercept = 0, slope = 1 / rho, size = 2)
# add appropriate labels
g = g + xlab("Father's height, normalized")
g = g + ylab("Son's height, normalized")
g = g + geom_text(x = 3.8, y = 1.6, label="x~y", angle = 25) +
    geom_text(x = 3.2, y = 3.6, label="cor(y,x)=1", angle = 35) +
    geom_text(x = 1.6, y = 3.8, label="y~x", angle = 60)
g
```



Statistical Linear Regression Models

- goal is use statistics to draw inferences → generalize from data to population through models
- **probabilistic model for linear regression**

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

where ϵ_i represents the sampling errors and is assumed to be iid $N(0, \sigma^2)$

- this model has the following properties
  - $E[Y_i \mid X_i = x_i] = E[\beta_0] + E[\beta_1 x_i] + E[\epsilon_i] = \mu_i = \beta_0 + \beta_1 x_i$
  - $Var(Y_i \mid X_i = x_i) = Var(\beta_0 + \beta_1 x_i) + Var(\epsilon_i) = Var(\epsilon_i) = \sigma^2$
    - $\beta_0 + \beta_1 x_i$ = line = constant/no variance
- it can then be said to have Y_i as independent $N(\mu, \sigma^2)$, where $\mu = \beta_0 + \beta_1 x_i$ <– likelihood equivalent model
  - **likelihood** = given the outcome, what is the probability?
    - in this case, the likelihood is as follows

$$\mathcal{L}(\beta_0, \beta_1, \sigma) = \prod_{i=1}^{n} \left\{ (2\pi\sigma^2)^{-1/2} \exp\left(-\frac{1}{2\sigma^2}(y_i - \mu_i)^2 \right) \right\}$$

      where $\mu_i = \beta_0 + \beta_1 x_i$
    - above is the **probability density function** of n samples from the normal distribution → this is because the regression line is normally distributed due to ϵ_i
  - **maximum likelihood estimator (MLE)** = most likely estimate of the population parameter/probability
    - in this case, the maximum likelihood = -2 minimum natural log (ln, base e) likelihood

$$-2 \log \mathcal{L}(\beta_0, \beta_1, \sigma) = n \log(2\pi\sigma^2) + \frac{1}{\sigma^2} \sum_{i=1}^{n} (y_i - \mu_i)^2$$

      - since everything else is constant, minimizing this function would only depend on $\sum_{i=1}^{n}(y_i - \mu_i)^2$, which from our previous derivations yields $\hat{\mu}_i = \beta_0 + \beta_1 \hat{x}_i$
    - ***maximum likelihood estimate*** = $\mu_i = \beta_0 + \beta_1 x_i$

Interpreting Regression Coefficients

- for the linear regression line

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

MLE for β_0 and β_1 are as follows

$$\hat{\beta}_1 = Cor(Y, X) \frac{Sd(Y)}{Sd(X)} \quad \hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$$

- β_0 = expected value of the outcome/response when the predictor is 0

$$E[Y|X = 0] = \beta_0 + \beta_1 \times 0 = \beta_0$$

  - ***Note**: $X = 0$ may not always be of interest as it may be impossible/outside of data range (i.e blood pressure, height etc.)*

  - it may be useful to move the intercept at times

$$\begin{aligned} Y_i &= \beta_0 + \beta_1 X_i + \epsilon_i \\ &= \beta_0 + a\beta_1 + \beta_1(X_i - a) + \epsilon_i \\ &= \tilde{\beta}_0 + \beta_1(X_i - a) + \epsilon_i \quad where \tilde{\beta}_0 = \beta_0 + a\beta_1 \end{aligned}$$

  - ***Note**: shifting X values by value a changes the intercept, but not the slope*
  - often, a is set to $\bar{X}$ so that the intercept is interpreted as the expected response at the average X value

- β_1 = expected change in outcome/response for a 1 unit change in the predictor

$$E[Y \mid X = x + 1] - E[Y \mid X = x] = \beta_0 + \beta_1(x + 1) - (\beta_0 + \beta_1 x) = \beta_1$$

  - sometimes it is useful to change the units of X

$$\begin{aligned} Y_i &= \beta_0 + \beta_1 X_i + \epsilon_i \\ &= \beta_0 + \frac{\beta_1}{a}(X_i a) + \epsilon_i \\ &= \beta_0 + \tilde{\beta}_1(X_i a) + \epsilon_i \end{aligned}$$

  - multiplication of X by a factor a results in dividing the coefficient by a factor of a
  - ***example***:
    * X = height in m
    * Y = weight in kg
    * β_1 has units of kg/m
    * converting X to $cm \implies$ multiplying X by $100\frac{cm}{m}$
    * this mean β_1 has to be divided by $100\frac{cm}{m}$ for the correct units.

$$X\ m \times 100\frac{cm}{m} = (100\ X)cm \quad \text{and} \quad \beta_1 \frac{kg}{m} \times \frac{1}{100}\frac{m}{cm} = \left(\frac{\beta_1}{100}\right)\frac{kg}{cm}$$

- 95% confidence intervals for the coefficients can be constructed from the coefficients themselves and their standard errors (from `summary(lm)`)
  - use the resulting intervals to evaluate the significance of the results

Use Regression Coefficients for Prediction

- for observed values of the predictor, $X_1, X_2, \ldots, X_n$, the prediction of the outcome/response is as follows

$$\hat{\mu}_i = \hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X$$

  where μ_i describes a point on the regression line

Example and R Commands

- `diamond` dataset from `UsingR` package
  - diamond prices in Singapore Dollars, diamond weight in carats (standard measure of diamond mass, 0.2g)
- `lm(price ~ I(carat - mean(carat)), data=diamond)` = mean centered linear regression
  - ***Note**: arithmetic operations must be enclosed in `I()` to work*

- `predict(fitModel, newdata=data.frame(carat=c(0, 1, 2)))` = returns predicted outcome from the given model (linear in our case) at the provided points within the `newdata` data frame
  - if `newdata` is unspecified (argument omitted), then `predict` function will return predicted values for all values of the predictor (x variable, carat in this case)
    - ***Note**: `newdata` has to be a dataframe, and the values you would like to predict (x variable, `carat` in this case) has to be specified, or the system won't know what to do with the provided values*
- `summary(fitModel)` = prints detailed summary of linear model
- ***example***

```
# standard linear regression for price vs carat
fit <- lm(price ~ carat, data = diamond)
# intercept and slope
coef(fit)
```

```
## (Intercept)       carat
##   -259.6259   3721.0249
```

```
# mean-centered regression
fit2 <- lm(price ~ I(carat - mean(carat)), data = diamond)
# intercept and slope
coef(fit2)
```

```
##            (Intercept) I(carat - mean(carat))
##              500.0833             3721.0249
```

```
# regression with more granular scale (1/10th carat)
fit3 <- lm(price ~ I(carat * 10), data = diamond)
# intercept and slope
coef(fit3)
```

```
##   (Intercept) I(carat * 10)
##     -259.6259      372.1025
```

```
# predictions for 3 values
newx <- c(0.16, 0.27, 0.34)
# manual calculations
coef(fit)[1] + coef(fit)[2] * newx
```

```
## [1]  335.7381  745.0508 1005.5225
```

```
# prediction using the predict function
predict(fit, newdata = data.frame(carat = newx))
```

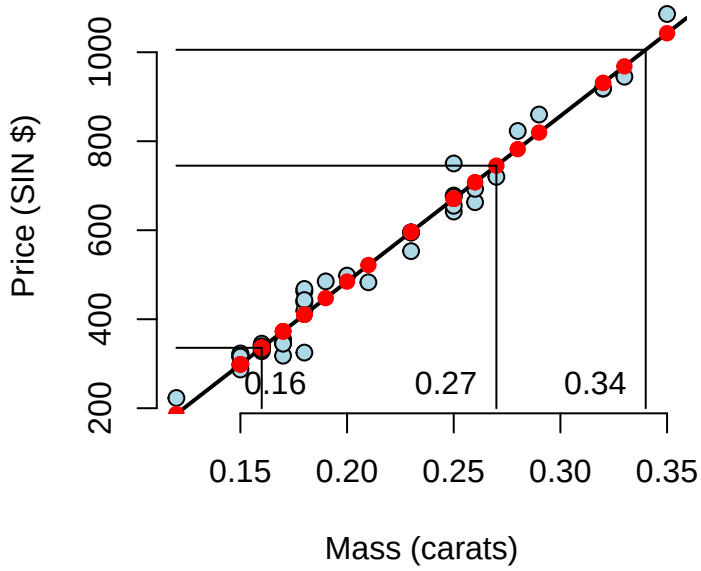
```
##         1         2         3
##  335.7381  745.0508 1005.5225
```

- **interpretation**
  - we expect 3721.02 (SIN) dollar increase in price for ***every carat increase*** in mass of diamond

  - or 372.1 (SIN) dollar increase in price for ***every 1/10 carat*** increase in mass of diamond

- **prediction**
  - for 0.16, 0.27, and 0.34 carats, we predict the prices to be 335.74, 745.05, 1005.52 (SIN) dollars

```
# plot the data points
plot(diamond$carat, diamond$price, xlab = "Mass (carats)", ylab = "Price (SIN $)",
     bg = "lightblue", col = "black", cex = 1.1, pch = 21,frame = FALSE)
# plot linear regression line
abline(fit, lwd = 2)
# plot predictions for every value of carat (in red)
points(diamond$carat, predict(fit), pch = 19, col = "red")
# add guidelines for predictions for 0.16, 0.27, and 0.34
lines(c(0.16, 0.16, 0.12), c(200, coef(fit)[1] + coef(fit)[2] * 0.16,
      coef(fit)[1] + coef(fit)[2] * 0.16))
lines(c(0.27, 0.27, 0.12), c(200, coef(fit)[1] + coef(fit)[2] * 0.27,
      coef(fit)[1] + coef(fit)[2] * 0.27))
lines(c(0.34, 0.34, 0.12), c(200, coef(fit)[1] + coef(fit)[2] * 0.34,
      coef(fit)[1] + coef(fit)[2] * 0.34))
# add text labels
text(newx+c(0.03, 0, 0), rep(250, 3), labels = newx, pos = 2)
```

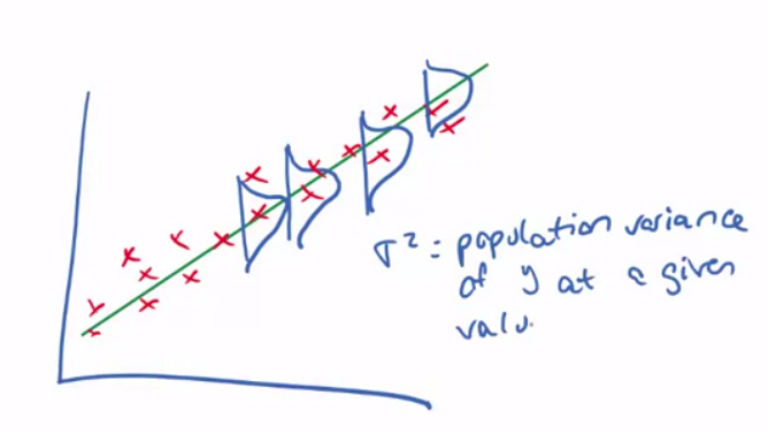


Derivation for Maximum Likelihood Estimator

- ***Note**: this derivation is for the maximum likelihood estimator of the mean, μ, of a normal distribution as it is the basis of the linear regression model*
- **linear regression model**

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

  follows a normal distribution because $\epsilon_i \sim N(0, \sigma^2)$
- for the above model, $E[Y_i] = \mu_i = \beta_0 + \beta_1 X_i$ and $Var(Y_i) = \sigma^2$



- the **probability density function (pdf)** for an outcome x from the normal distribution is defined as

$$f(x \mid \mu, \sigma^2) = (2\pi\sigma^2)^{-1/2} \exp\left(-\frac{1}{2\sigma^2}(y_i - \mu_i)^2\right)$$

- the corresponding pdf for n iid normal random outcomes $x_1, \ldots, x_n$ is defined as

$$f(x_1, \ldots, x_n \mid \mu, \sigma^2) = \prod_{i=1}^{n} (2\pi\sigma^2)^{-1/2} \exp\left(-\frac{1}{2\sigma^2}(y_i - \mu_i)^2\right)$$

  which is also known as the **likelihood function**, denoted in this case as $\mathcal{L}(\mu, \sigma)$
- to find the **maximum likelihood estimator (MLE)** of the mean, μ, we take the derivative of the likelihood $\mathcal{L}$ with respect to $\mu \rightarrow \frac{\partial \mathcal{L}}{\partial \mu}$
- since derivatives of products are quite complex to compute, taking the log (base e) makes the calculation much simpler
  - log properties:
    - $\log(AB) = \log(A) + \log(B)$
    - $\log(A^B) = B \log(A)$
  - because log is always increasing and **monotonic**, or preserves order, finding the maximum MLE = finding th maximum of log transformation of MLE
- -2 log of **likelihood function**

$$\log(\mathcal{L}(\mu, \sigma)) = \sum_{i=1}^{n} -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(y_i - \mu_i)^2 \Leftarrow \text{multiply -2 on both sides}$$

$$-2\log(\mathcal{L}(\mu, \sigma)) = \sum_{i=1}^{n} \log(2\pi\sigma^2) + \frac{1}{\sigma^2}(y_i - \mu_i)^2 \Leftarrow \sigma^2 \text{ does not depend on } i$$

$$-2\log(\mathcal{L}(\mu, \sigma)) = n\log(2\pi\sigma^2) + \frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - \mu_i)^2$$

- minimizing -2 log likelihood is ***computationally equivalent*** as maximizing log likelihood

$$\frac{\partial \log(\mathcal{L}(\mu,\sigma))}{\partial \mu} = \frac{1}{\sigma^2}\frac{\partial \sum_{i=1}^{n}(y_i-\mu_i)^2}{\partial \mu} = 0 \Leftarrow \text{setting this equal to 0 to find minimum}$$

$$\Rightarrow -\frac{2}{\sigma^2}\sum_{i=1}^{n}(y_i-\mu_i) = 0 \Leftarrow \text{divide by } -2/\sigma^2 \text{ on both sides}$$

$$\sum_{i=1}^{n}(y_i-\mu_i) = 0 \Leftarrow \text{move } \mu \text{ term over to the right}$$

$$\sum_{i=1}^{n} y_i = \sum_{i=1}^{n}\mu_i \Leftarrow \text{for the two sums to be equal, all the terms must be equal}$$

$$y_i = \mu_i$$

- in the case of our linear regression, $\mu_i = \beta_0 + \beta_1 X_i$ so

$$\frac{\partial \mathcal{L}(\mu,\sigma))}{\partial \mu} = \frac{\partial \mathcal{L}(\beta_0,\beta_1,\sigma))}{\partial \mu}$$

$$\text{MLE} \Rightarrow Y_i = \mu_i$$

$$\mu_i = \beta_0 + \beta_1 X_i$$

Residuals

- Residual, e_i = difference between the observed and predicted outcome

$$e_i = Y_i - \hat{Y}_i$$

  - Or, vertical distance between observed data point and regression line
  - Least squares minimizes $\sum_{i=1}^{n} e_i^2$
- e_i can be interpreted as estimates of the regression error, ϵ_i
- e_i can also be interpreted as the outcome (Y) with the linear association of the predictor (X) removed
  - or, "Y adjusted for X"
- $E[e_i] = 0 \rightarrow$ this is because the mean of the residuals is expected to be 0 (assumed Gaussian distribution)
  - the Gaussian distribution assumption also implies that the error is **NOT** correlated with any predictors
  - `mean(fitModel$residuals)` = returns mean of residuals $\rightarrow$ should equal to 0
  - `cov(fit$residuals, predictors)` = returns the covariance (measures correlation) of residuals and predictors $\rightarrow$ should also equal to 0
- $\sum_{i=1}^{n} e_i = 0$ (if an intercept is included) and $\sum_{i=1}^{n} e_i X_i = 0$ (if a regressor variable X_i is included)
- for standard linear regression model
  - positive residuals = above the line
  - negative residuals = below
- residuals/residual plots can highlight poor model fit

Estimating Residual Variation

- residual variation measures how well the regression line fit the data points
- **MLE of variance**, σ^2, of the linear model = $\frac{1}{n}\sum_{i=1}^{n} e_i^2$ or the ***average squared residual/mean squared error***
  - the square root of the estimate, σ, = **root mean squared error (RMSE)**
- however, a more common approach is to use

$$\hat{\sigma}^2 = \frac{1}{n-2}\sum_{i=1}^{n} e_i^2$$

  - $n-2$ is used instead of n to make the estimator ***unbiased*** $\rightarrow E[\hat{\sigma}^2] = \sigma^2$
  - ***Note**: the -2 is accounting for the degrees of freedom for intercept and slope, which had to be estimated*
- `deviance(fitModel)` = calculates sum of the squared error/residual for the linear model/residual variation
- `summary(fitModel)$sigma` = returns the residual variation of a fit model or the **unbiased RMSE**
  - `summary(fitModel)` = creates a list of different parameters of the fit model

```
# get data
y <- diamond$price; x <- diamond$carat; n <- length(y)
# linear fit
fit <- lm(y ~ x)
# calculate residual variation through summary and manual
rbind("from summary" = summary(fit)$sigma, "manual" =sqrt(sum(resid(fit)^2) / (n - 2)))
```

```
##                   [,1]
## from summary 31.84052
## manual       31.84052
```

Total Variation, R^2, and Derivations

- **total variation** = ***residual variation*** (variation after removing predictor) + ***systematic/regression variation*** (variation explained by regression model)

$$\sum_{i=1}^{n}(Y_i - \bar{Y})^2 = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2$$

- R^2 = percent of total variability that is explained by the regression model

$$\begin{aligned} R^2 &= \frac{\text{regression variation}}{\text{total variation}} = \frac{\sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} \\ &= 1 - \frac{\text{residual variation}}{\text{total variation}} = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} \end{aligned}$$

- $0 \leq R^2 \leq 1$
- R^2 = sample correlation squared
  - `cor(outcome, predictor` = calculates the correlation between predictor and outcome → the same as calculating R^2
- R^2 can be a ***misleading*** summary of model fit
  - deleting data → inflate R^2
  - adding terms to a regression model → always increases R^2
  - `example(anscombe)` demonstrates the fallacy of R^2 through the following
    - basically same mean and variance of X and Y
    - identical correlations (hence same R^2)
    - same linear regression relationship
- **relationship between R^2 and r**

Correlation between X and Y $\Rightarrow r = Cor(Y, X)$

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

$$recall \Rightarrow (\hat{Y}_i - \bar{Y}) = \hat{\beta}_1(X_i - \bar{X})$$

$$(substituting(\hat{Y}_i - \bar{Y})) = \hat{\beta}_1^2 \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

$$recall \Rightarrow \hat{\beta}_1 = Cor(Y, X)\frac{Sd(Y)}{Sd(X)}$$

$$(substituting\hat{\beta}_1) = Cor(Y, X)^2 \frac{Var(Y)}{Var(X)} \times \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

$$recall Var(Y) = \sum_{i=1}^{n}(Y_i - \bar{Y})^2 and Var(X) = \sum_{i=1}^{n}(X_i - \bar{X})^2$$

$$(simplifying) \Rightarrow R^2 = Cor(Y, X)^2$$

$$\text{Or } R^2 = r^2$$

- ***total variation derivation***

$$\text{First, we know that } \bar{Y} = \hat{\beta}_0 + \hat{\beta}_1 \bar{X}$$
$$(transforming) \Rightarrow \hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$$

$$\text{We also know that } \hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$$

$$\text{Next, the residual } = (Y_i - \hat{Y}_i) = Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i$$
$$(substituting\ \hat{\beta}_0) = Y_i - (\bar{Y} - \hat{\beta}_1 \bar{X}) - \hat{\beta}_1 X_i$$
$$(transforming) \Rightarrow (Y_i - \hat{Y}_i) = (Y_i - \bar{Y}) - \hat{\beta}_1 (X_i - \bar{X})$$

$$\text{Next, the regression difference } = (\hat{Y}_i - \bar{Y}) = \hat{\beta}_0 - \hat{\beta}_1 X_i - \bar{Y}$$
$$(substituting\ \hat{\beta}_0) = \bar{Y} - \hat{\beta}_1 \bar{X} - \hat{\beta}_1 X_i - \bar{Y}$$
$$(transforming) \Rightarrow (\hat{Y}_i - \bar{Y}) = \hat{\beta}_1 (X_i - \bar{X})$$

$$\text{Total Variation} = \sum_{i=1}^{n} (Y_i - \bar{Y})^2 = \sum_{i=1}^{n} (Y_i - \hat{Y}_i + \hat{Y}_i - \bar{Y})^2 \Leftarrow (adding\ \pm \hat{Y}_i)$$
$$(expanding) = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 + 2 \sum_{i=1}^{n} (Y_i - \hat{Y}_i)(\hat{Y}_i - \bar{Y}) + \sum_{i=1}^{n} (\hat{Y}_i - \bar{Y})^2$$

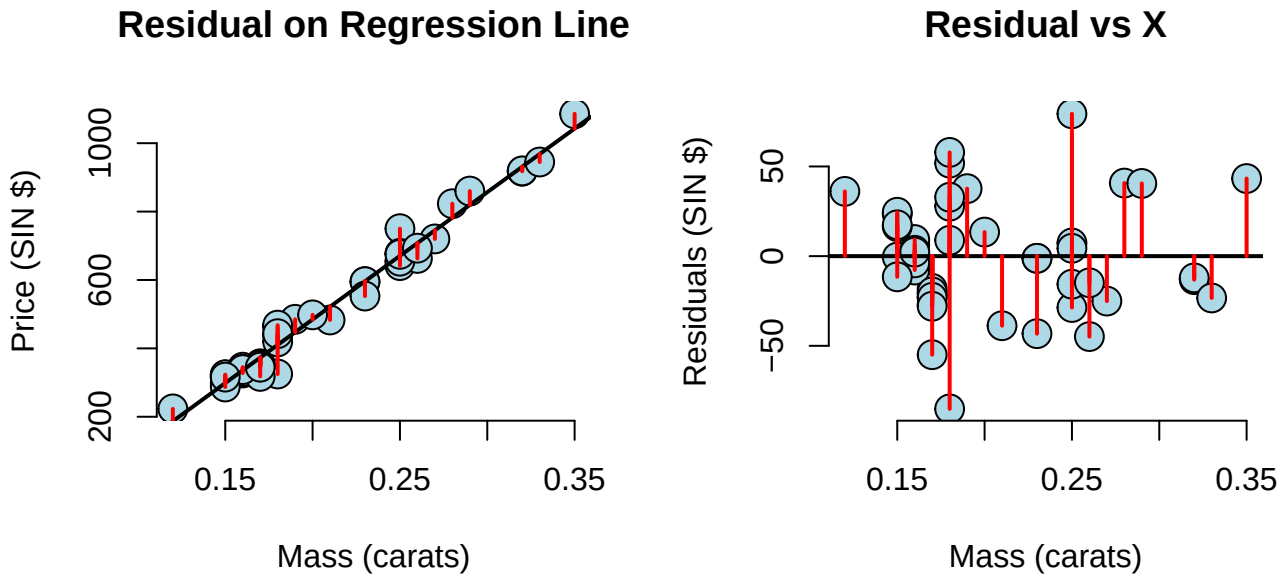
$$\text{Looking at } \sum_{i=1}^{n} (Y_i - \hat{Y}_i)(\hat{Y}_i - \bar{Y})$$
$$(substituting\ (Y_i - \hat{Y}_i), (\hat{Y}_i - \bar{Y})) = \sum_{i=1}^{n} \left[(Y_i - \bar{Y}) - \hat{\beta}_1 (X_i - \bar{X}) \right] \left[\hat{\beta}_1 (X_i - \bar{X}) \right]$$
$$(expanding) = \hat{\beta}_1 \sum_{i=1}^{n} (Y_i - \bar{Y})(X_i - \bar{X}) - \hat{\beta}_1^2 \sum_{i=1}^{n} (X_i - \bar{X})^2$$
$$(substituting\ Y_i, \bar{Y}) \Rightarrow (Y_i - \bar{Y}) = (\hat{\beta}_0 + \hat{\beta}_1 X_i) - (\hat{\beta}_0 + \hat{\beta}_1 \bar{X})$$
$$(simplifying) \Rightarrow (Y_i - \bar{Y}) = \hat{\beta}_1 (X_i - \bar{X})$$
$$(substituting\ (Y_i - \bar{Y})) = \hat{\beta}_1^2 \sum_{i=1}^{n} (X_i - \bar{X})^2 - \hat{\beta}_1^2 \sum_{i=1}^{n} (X_i - \bar{X})^2$$
$$\Rightarrow \sum_{i=1}^{n} (Y_i - \hat{Y}_i)(\hat{Y}_i - \bar{Y}) = 0$$

$$\text{Going back to} \sum_{i=1}^{n} (Y_i - \bar{Y})^2 = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 + 2 \sum_{i=1}^{n} (Y_i - \hat{Y}_i)(\hat{Y}_i - \bar{Y}) + \sum_{i=1}^{n} (\hat{Y}_i - \bar{Y})^2$$
$$(since\ second\ term = 0) \Rightarrow \sum_{i=1}^{n} (Y_i - \bar{Y})^2 = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 + \sum_{i=1}^{n} (\hat{Y}_i - \bar{Y})^2$$

Example and R Commands

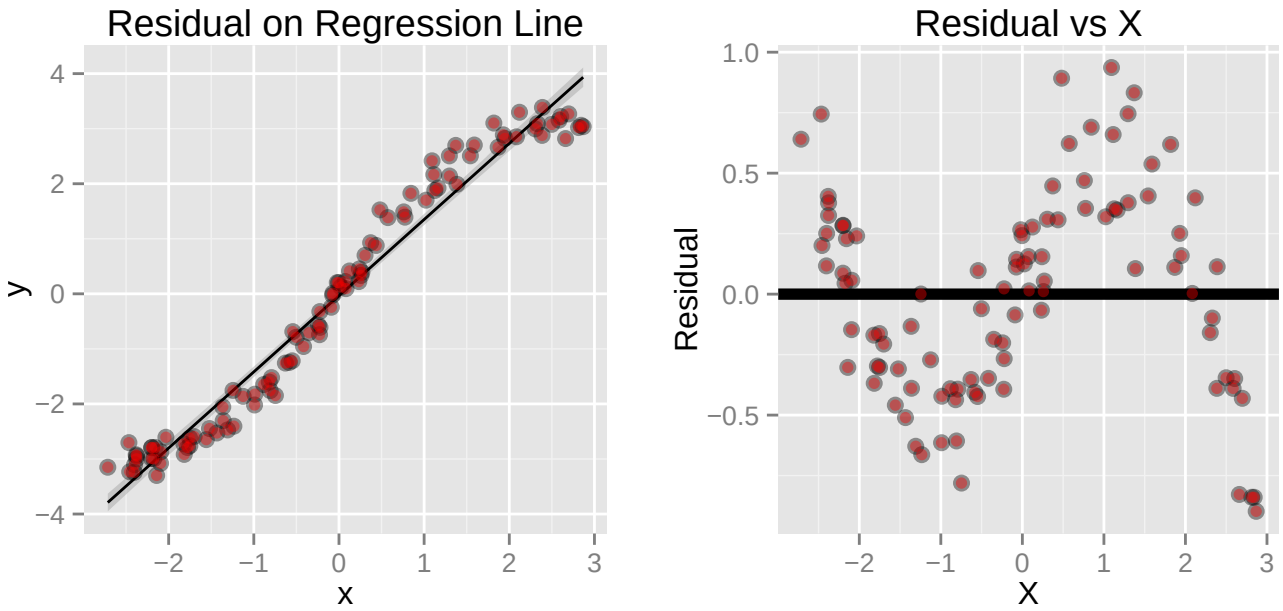
- `resid(fitModel)` or `fitModel$residuals` = extracts model residuals from the fit model (lm in our case) → list of residual values for every value of X
- `summary(fitModel)$r.squared` = return R^2 value of the regression model

```r
# load multiplot function
source("multiplot.R")
# get data
y <- diamond$price; x <- diamond$carat; n <- length(y)
# linear regression
fit <- lm(y ~ x)
# calculate residual
e <- resid(fit)
# calculate predicted values
yhat <- predict(fit)
# create 1 x 2 panel plot
par(mfrow=c(1, 2))
# plot residuals on regression line
plot(x, y, xlab = "Mass (carats)", ylab = "Price (SIN $)", bg = "lightblue",
    col = "black", cex = 2, pch = 21,frame = FALSE,main = "Residual on Regression Line")
# draw linear regression line
abline(fit, lwd = 2)
# draw red lines from data points to regression line
for (i in 1 : n){lines(c(x[i], x[i]), c(y[i], yhat[i]), col = "red" , lwd = 2)}
# plot residual vs x
plot(x, e, xlab = "Mass (carats)", ylab = "Residuals (SIN $)", bg = "lightblue",
    col = "black", cex = 2, pch = 21,frame = FALSE,main = "Residual vs X")
# draw horizontal line
abline(h = 0, lwd = 2)
# draw red lines from residual to x axis
for (i in 1 : n){lines(c(x[i], x[i]), c(e[i], 0), col = "red" , lwd = 2)}
```



- non-linear data/patterns can be more easily revealed through residual plots

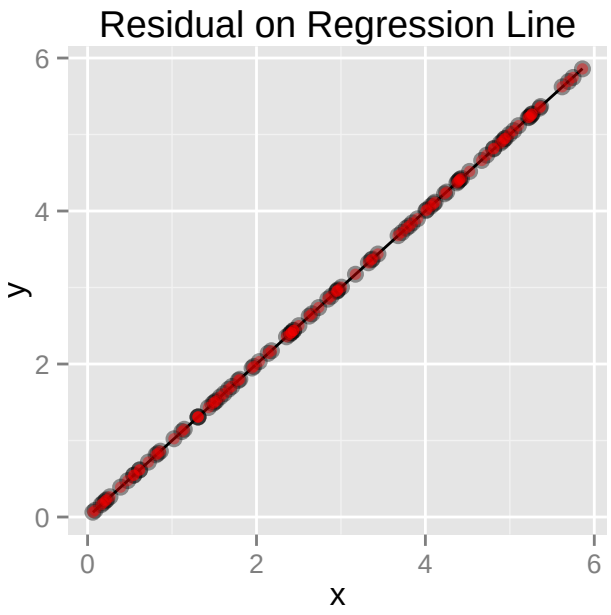
```r
# create sin wave pattern
x <- runif(100, -3, 3); y <- x + sin(x) + rnorm(100, sd = .2);
# plot data + regression
g <- ggplot(data.frame(x = x, y = y), aes(x = x, y = y)) +
    geom_smooth(method = "lm", colour = "black") +
    geom_point(size = 3, colour = "black", alpha = 0.4) +
    geom_point(size = 2, colour = "red", alpha = 0.4)+
 ggtitle("Residual on Regression Line")
# plot residuals
f <- ggplot(data.frame(x = x, y = resid(lm(y ~ x))), aes(x = x, y = y))+
    geom_hline(yintercept = 0, size = 2)+
    geom_point(size = 3, colour = "black", alpha = 0.4)+
    geom_point(size = 2, colour = "red", alpha = 0.4)+
    xlab("X") + ylab("Residual")+ ggtitle("Residual vs X")
multiplot(g, f, cols = 2)
```

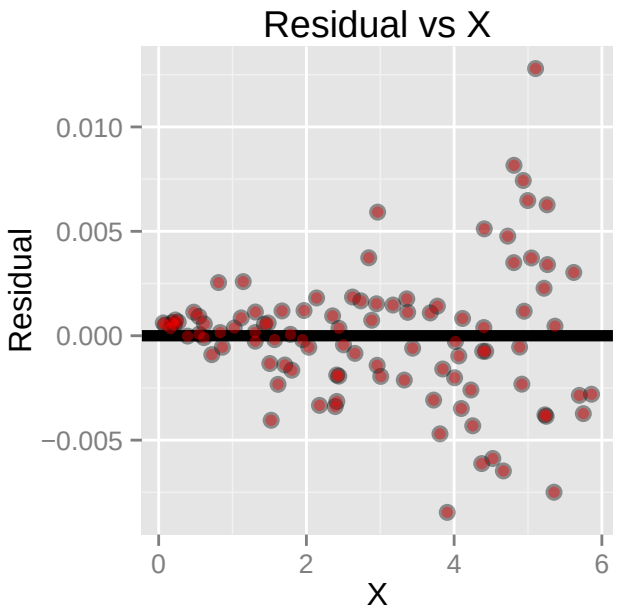


- **heteroskedasticity** = heteroskedastic model's variance is not constant and is a function of x

```r
# create heteroskedastic data
x <- runif(100, 0, 6); y <- x + rnorm(100,  mean = 0, sd = .001 * x)
# plot data + regression
g <- ggplot(data.frame(x = x, y = y), aes(x = x, y = y))+
 geom_smooth(method = "lm", colour = "black")+
 geom_point(size = 3, colour = "black", alpha = 0.4)+
 geom_point(size = 2, colour = "red", alpha = 0.4) +
 ggtitle("Residual on Regression Line")
# plot residuals
f <- ggplot(data.frame(x = x, y = resid(lm(y ~ x))), aes(x = x, y = y))+
           geom_hline(yintercept = 0, size = 2) +
           geom_point(size = 3, colour = "black", alpha = 0.4)+
           geom_point(size = 2, colour = "red", alpha = 0.4)+
           xlab("X") + ylab("Residual") + ggtitle("Residual vs X")
```

```
# combine two plots
multiplot(g, f, cols = 2)
```





Inference in Regression

- statistics used for hypothesis tests and confidence intervals have the following attributes

$$\frac{\hat{\theta} - \theta}{\hat{\sigma}_{\hat{\theta}}} \sim N(0,1)$$

  - it follows a finite sample Student's ***T distribution*** and is ***normally distributed*** if the sample has IID components built in (i.e. ϵ_i)
  - used to test $H_0 : \theta = \theta_0$ vs. $H_a : \theta >, <, \neq \theta_0$.
  - confidence interval for $\theta = \hat{\theta} \pm Q_{1-\alpha/2}\hat{\sigma}_{\hat{\theta}}$, where $Q_{1-\alpha/2}$ = relevant quantile from normal(for large samples)/T distribution(small samples, n-1 degrees of freedom)

Intervals/Tests for Coefficients

- standard errors for coefficients

$$\begin{aligned}
Var(\hat{\beta}_1) &= Var\left(\frac{\sum_{i=1}^n (Y_i - \bar{Y})(X_i - \bar{X})}{((X_i - \bar{X})^2)}\right) \\
(expanding) &= Var\left(\frac{\sum_{i=1}^n Y_i(X_i - \bar{X}) - \bar{Y}\sum_{i=1}^n (X_i - \bar{X})}{((X_i - \bar{X})^2)}\right) \\
Since \sum_{i=1}^n X_i - \bar{X} &= 0 \\
(simplifying) &= \frac{\sum_{i=1}^n Y_i(X_i - \bar{X})}{(\sum_{i=1}^n (X_i - \bar{X})^2)^2} \Leftarrow \text{denominator taken out of } Var \\
(Var(Y_i) = \sigma^2) &= \frac{\sigma^2 \sum_{i=1}^n (X_i - \bar{X})^2}{(\sum_{i=1}^n (X_i - \bar{X})^2)^2} \\
\sigma^2_{\hat{\beta}_1} = Var(\hat{\beta}_1) &= \frac{\sigma^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \\
\Rightarrow \sigma_{\hat{\beta}_1} &= \frac{\sigma}{\sum_{i=1}^n X_i - \bar{X}}
\end{aligned}$$

by the same derivation $\Rightarrow$

$$\begin{aligned}
\sigma^2_{\hat{\beta}_0} = Var(\hat{\beta}_0) &= \left(\frac{1}{n} + \frac{\bar{X}^2}{\sum_{i=1}^n (X_i - \bar{X})^2}\right)\sigma^2 \\
\Rightarrow \sigma_{\hat{\beta}_0} &= \sigma\sqrt{\frac{1}{n} + \frac{\bar{X}^2}{\sum_{i=1}^n (X_i - \bar{X})^2}}
\end{aligned}$$

- σ is unknown but it's estimate is as follows

$$\hat{\sigma}^2 = \frac{1}{n-2}\sum_{i=1}^n e_i^2$$

- under IID Gaussian errors (assumed in linear regression with term ϵ_0), statistics for $\hat{\beta}_0$ and $\hat{\beta}_1$ are as follows

$$\frac{\hat{\beta}_j - \beta_j}{\hat{\sigma}_{\hat{\beta}_j}} \quad where\ j = 0, 1$$

  - these statistics follow t **distribution** with $n-2$ degrees of freedom for small n and normal distribution for large n

- `summary(fitModel)$coefficients` = returns table summarizing the estimate, standar error, t value and p value for the coefficients β_0 and β_1
  - the below example reproduces the table `summary(fitModel)$coefficients` produces
- ***Note**: the variability in the slope, or $Var(\hat{\beta}_1)$, is the largest when the predictor values are spread into two cluster that are far apart from each other*
  - when modeling linear relationships, it is generally good practice to have many data points that evenly cover the entire range of data, increasing the denominator $\sum_{i=1}^{n}(X_i - \bar{X})^2$
  - this is so that variance of slope is minimized and we can be more confident about the relationship
- ***example***

```
# getting data
y <- diamond$price; x <- diamond$carat; n <- length(y)
# calculate beta1
beta1 <- cor(y, x) * sd(y) / sd(x)
# calculate beta0
beta0 <- mean(y) - beta1 * mean(x)
# Gaussian regression error
e <- y - beta0 - beta1 * x
# unbiased estimate for variance
sigma <- sqrt(sum(e^2) / (n-2))
# (X_i - X Bar)
ssx <- sum((x - mean(x))^2)
# calculate standard errors
seBeta0 <- (1 / n + mean(x) ^ 2 / ssx) ^ .5 * sigma
seBeta1 <- sigma / sqrt(ssx)
# testing for H0: beta0 = 0 and beta0 = 0
tBeta0 <- beta0 / seBeta0; tBeta1 <- beta1 / seBeta1
# calculating p-values for Ha: beta0 != 0 and beta0 != 0 (two sided)
pBeta0 <- 2 * pt(abs(tBeta0), df = n - 2, lower.tail = FALSE)
pBeta1 <- 2 * pt(abs(tBeta1), df = n - 2, lower.tail = FALSE)
# store results into table
coefTable <- rbind(c(beta0, seBeta0, tBeta0, pBeta0), c(beta1, seBeta1, tBeta1, pBeta1))
colnames(coefTable) <- c("Estimate", "Std. Error", "t value", "P(>|t|)")
rownames(coefTable) <- c("(Intercept)", "x")
# print table
coefTable
```

```
##              Estimate Std. Error   t value      P(>|t|)
## (Intercept) -259.6259   17.31886 -14.99094 2.523271e-19
## x           3721.0249   81.78588  45.49715 6.751260e-40
```

```
# regression model and the generated table from lm (identical to above)
fit <- lm(y ~ x); summary(fit)$coefficients
```

```
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -259.6259   17.31886 -14.99094 2.523271e-19
## x           3721.0249   81.78588  45.49715 6.751260e-40
```

```r
# store results in matrix
sumCoef <- summary(fit)$coefficients
# print out confidence interval for beta0
sumCoef[1,1] + c(-1, 1) * qt(.975, df = fit$df) * sumCoef[1, 2]
```

```
## [1] -294.4870 -224.7649
```

```r
# print out confidence interval for beta1 in 1/10 units
(sumCoef[2,1] + c(-1, 1) * qt(.975, df = fit$df) * sumCoef[2, 2]) / 10
```

```
## [1] 355.6398 388.5651
```

- **interpretation**: With 95% confidence, we estimate that a 0.1 carat increase in diamond size results in a 355.6 to 388.6 increase in price in (Singapore) dollars.

Prediction Interval

- estimated prediction, $\hat{y}_0$, at point x_0 is

$$\hat{y}_0 = \hat{\beta}_0 + \hat{\beta}_1 x_0$$

- we can construct two prediction intervals
  1. interval for **the line** at x_0

$$\begin{aligned} \text{Interval}: & \hat{y}_0 \pm t_{n-2,1-\alpha/2} \times SE_{line} \\ \text{where } & \hat{y}_0 = \hat{\beta}_0 + \hat{\beta}_1 x_0 \\ \text{and } & SE_{line} = \hat{\sigma}\sqrt{\frac{1}{n} + \frac{(x_0 - \bar{X})^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}} \end{aligned}$$

     - interval has ***varying width***
     - interval is narrow as we are quite confident in the regression line
     - as n increases, the interval becomes ***narrower***, which makes sense because as more data is collected, we are able to get a better regression line
       - ***Note**: if we knew β_0 and β_1, this interval would have zero width*
  2. interval for the **predicted value**, $\hat{y}_0$, at x_0

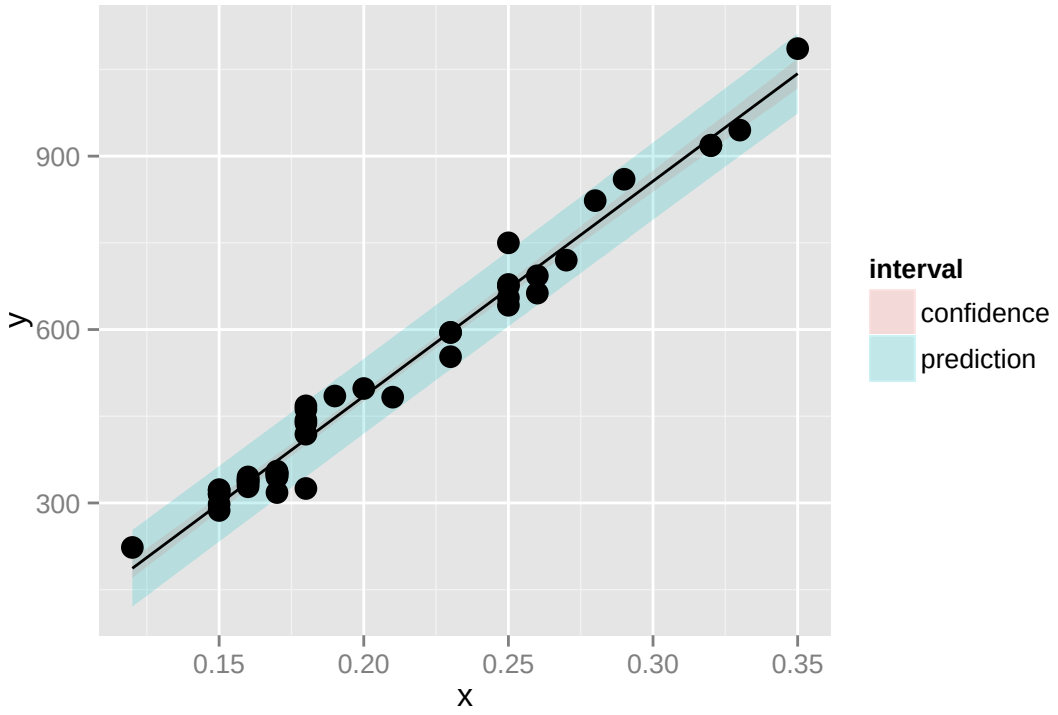
$$\begin{aligned} \text{Interval}: & \hat{y}_0 \pm t_{n-2,1-\alpha/2} \times SE_{\hat{y}_0} \\ \text{where } & \hat{y}_0 = \hat{\beta}_0 + \hat{\beta}_1 x_0 \\ \text{and } & SE_{\hat{y}_0} = \hat{\sigma}\sqrt{1 + \frac{1}{n} + \frac{(x_0 - \bar{X})^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}} \end{aligned}$$

     - interval has ***varying width***
     - the 1 part in the $SE_{\hat{y}_0}$ formula represents the inherent variability in the data
       - no matter how good of a regression line we get, we still can not get rid of the variability in the data
       - ***Note**: even if we know β_0 and β_1, the interval would still have width due to data variance*
- `predict(fitModel, data, interval = ("confidence"))` = returns a 3-column matrix with data for `fit` (regression line), `lwr` (lower bound of interval), and `upr` (upper bound of interval)

  - `interval = ("confidence")` = returns interval for the line
  - `interval = ("prediction")` = returns interval for the prediction
  - `data` = must be a new data frame with the values you would like to predict

- ***example (ggplot2)***

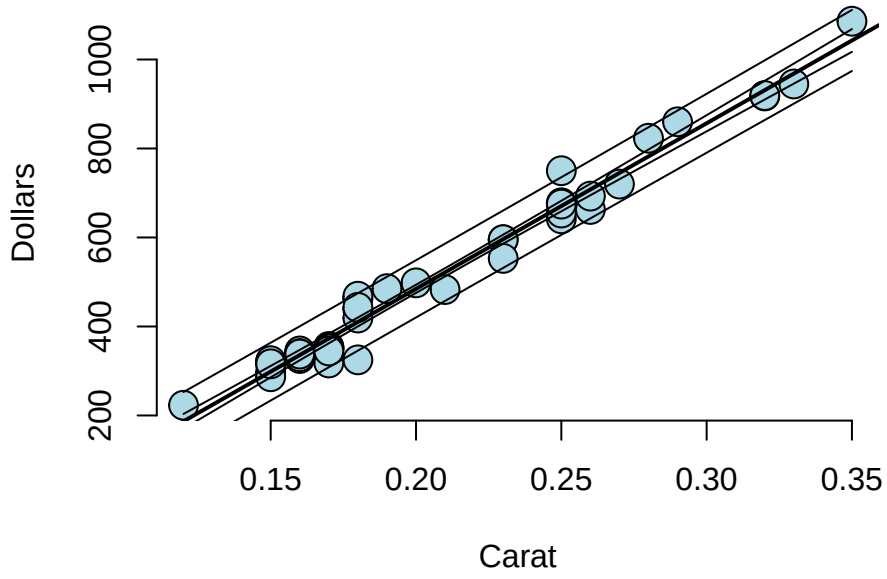
```
# create a sequence of values that we want to predict at
newx = data.frame(x = seq(min(x), max(x), length = 100))
# calculate values for both intervals
p1 = data.frame(predict(fit, newdata= newx,interval = ("confidence")))
p2 = data.frame(predict(fit, newdata = newx,interval = ("prediction")))
# add column for interval labels
p1$interval = "confidence"; p2$interval = "prediction"
# add column for the x values we want to predict
p1$x = newx$x; p2$x = newx$x
# combine the two dataframes
dat = rbind(p1, p2)
# change the name of the first column to y
names(dat)[1] = "y"
# plot the data
g <- ggplot(dat, aes(x = x, y = y))
g <- g + geom_ribbon(aes(ymin = lwr, ymax = upr, fill = interval), alpha = 0.2)
g <- g + geom_line()
g + geom_point(data = data.frame(x = x, y=y), aes(x = x, y = y), size = 4)
```



- ***example (base)***

```
# plot the x and y values
plot(x,y,frame=FALSE,xlab="Carat",ylab="Dollars",pch=21,col="black",bg="lightblue",cex=2)
# add the fit line
```

```r
abline(fit,lwd=2)
# create sequence of x values that we want to predict at
xVals<-seq(min(x),max(x),by=.01)
# calculate the predicted y values
yVals<-beta0+beta1*xVals
# calculate the standard errors for the interval for the line
se1<-sigma*sqrt(1/n+(xVals-mean(x))^2/ssx)
# calculate the standard errors for the interval for the predicted values
se2<-sigma*sqrt(1+1/n+(xVals-mean(x))^2/ssx)
# plot the upper and lower bounds of both intervals
lines(xVals,yVals+2*se1); lines(xVals,yVals-2*se1)
lines(xVals,yVals+2*se2); lines(xVals,yVals-2*se2)
```



Multivariate Regression

- linear models = most important applied statistical/machine learning technique
- **generalized linear model** extends simple linear regression (SLR) model

$$Y_i = \beta_1 X_{1i} + \beta_2 X_{2i} + \ldots + \beta_p X_{pi} + \epsilon_i = \sum_{k=1}^{p} X_{ik}\beta_j + \epsilon_i$$

  where $X_{1i} = 1$ typically, so that an intercept is included
- least squares/MLE for the model (under IID Gaussian errors) minimizes

$$\sum_{i=1}^{n} \left(Y_i - \sum_{k=1}^{p} X_{ki}\beta_j \right)^2$$

- **linearity of coefficients** is what defines a linear model as transformations of the variables (i.e. squaring) still yields a linear model

$$Y_i = \beta_1 X_{1i}^2 + \beta_2 X_{2i}^2 + \ldots + \beta_p X_{pi}^2 + \epsilon_i$$

- performing multivariate regression = pick any regressor and replace the outcome and all other regressors by their residuals against the chosen one

Derivation of Coefficients

- we know for simple univariate regression through the origin

$$E[Y_i] = X_{1i}\beta_1\hat{\beta}_1 = \frac{\sum_{i=1}^{n} X_i Y_i}{\sum_{i=1}^{n} X_i^2}$$

- we want to minimize

$$\sum_{i=1}^{n} (Y_i - X_{1i}\beta_1 - X_{2i}\beta_2 - \ldots - X_{pi}\beta_p)^2$$

- we begin by looking at the two variable model where

$$E[Y_i] = \mu_i = X_{1i}\beta_1 + X_{2i}\beta_2\hat{\mu}_i = X_{1i}\hat{\beta}_1 + X_{2i}\hat{\beta}_2$$

- from our previous derivations, to minimize the sum of squares, the following has to true

$$\sum_{i=1}^{n} (Y_i - \hat{\mu}_i)(\hat{\mu}_i - \mu_i) = 0$$

- plugging in $\hat{\mu}_i$ and μ_i

$$\begin{aligned}
\sum_{i=1}^{n} (Y_i - \hat{\mu}_i)(\hat{\mu}_i - \mu_i) &= \sum_{i=1}^{n} (Y_i - X_{1i}\hat{\beta}_1 - X_{2i}\hat{\beta}_2)(X_{1i}\hat{\beta}_1 + X_{2i}\hat{\beta}_2 - X_{1i}\beta_1 - X_{2i}\beta_2) \\
(simplifying) &= \sum_{i=1}^{n} (Y_i - X_{1i}\hat{\beta}_1 - X_{2i}\hat{\beta}_2) \left[X_{1i}(\hat{\beta}_1 - \beta_1) + X_{2i}(\hat{\beta}_2 - \beta_2) \right] \\
(expanding) &= \sum_{i=1}^{n} (Y_i - X_{1i}\hat{\beta}_1 - X_{2i}\hat{\beta}_2) X_{1i}(\hat{\beta}_1 - \beta_1) + \sum_{i=1}^{n} (Y_i - X_{1i}\hat{\beta}_1 - X_{2i}\hat{\beta}_2) X_{2i}(\hat{\beta}_2 - \beta_2)
\end{aligned}$$

- for the entire expression to equal to zero,

$$\sum_{i=1}^{n}(Y_i - X_{1i}\hat{\beta}_1 - X_{2i}\hat{\beta}_2)X_{1i}(\hat{\beta}_1 - \beta_1) = 0$$

$$(since\ \hat{\beta}_1,\ \beta_1\ don't\ depend\ on\ i) \Rightarrow \sum_{i=1}^{n}(Y_i - X_{1i}\hat{\beta}_1 - X_{2i}\hat{\beta}_2)X_{1i} = 0 \qquad (1)$$

$$\sum_{i=1}^{n}(Y_i - X_{1i}\hat{\beta}_1 - X_{2i}\hat{\beta}_2)X_{2i}(\hat{\beta}_2 - \beta_2) = 0$$

$$(since\ \hat{\beta}_2,\ \beta_2\ don't\ depend\ on\ i) \Rightarrow \sum_{i=1}^{n}(Y_i - X_{1i}\hat{\beta}_1 - X_{2i}\hat{\beta}_2)X_{2i} = 0 \qquad (2)$$

- we can hold $\hat{\beta}_1$ fixed and solve **(2)** for $\hat{\beta}_2$

$$\sum_{i=1}^{n}(Y_i - X_{1i}\hat{\beta}_1)X_{2i} - \sum_{i=1}^{n}X_{2i}^2\hat{\beta}_2 = 0$$

$$\sum_{i=1}^{n}(Y_i - X_{1i}\hat{\beta}_1)X_{2i} = \sum_{i=1}^{n}X_{2i}^2\hat{\beta}_2$$

$$\hat{\beta}_2 = \frac{\sum_{i=1}^{n}(Y_i - X_{1i}\hat{\beta}_1)X_{2i}}{\sum_{i=1}^{n}X_{2i}^2}$$

- plugging $\hat{\beta}_2$ back into **(1)**, we get

$$\sum_{i=1}^{n}\left[Y_i - X_{1i}\hat{\beta}_1 - \frac{\sum_{j=1}^{n}(Y_j - X_{1j}\hat{\beta}_1)X_{2j}}{\sum_{j=1}^{n}X_{2j}^2}X_{2i}\right]X_{1i} = 0$$

$$\sum_{i=1}^{n}\left[Y_i - X_{1i}\hat{\beta}_1 - \frac{\sum_{j=1}^{n}Y_jX_{2j}}{\sum_{j=1}^{n}X_{2j}^2}X_{2i} + \frac{\sum_{j=1}^{n}X_{1j}X_{2j}}{\sum_{j=1}^{n}X_{2j}^2}X_{2i}\hat{\beta}_1\right]X_{1i} = 0$$

$$\sum_{i=1}^{n}\left[\left(Y_i - \frac{\sum_{j=1}^{n}Y_jX_{2j}}{\sum_{j=1}^{n}X_{2j}^2}X_{2i}\right) - \hat{\beta}_1\left(X_{1i} - \frac{\sum_{j=1}^{n}X_{1j}X_{2j}}{\sum_{j=1}^{n}X_{2j}^2}X_{2i}\right)\right]X_{1i} = 0$$

$$\Rightarrow \left(Y_i - \frac{\sum_{j=1}^{n}Y_jX_{2j}}{\sum_{j=1}^{n}X_{2j}^2}X_{2i}\right) = \text{residual for } Y_i = X_{i2}\hat{\beta}_2 + \epsilon_i$$

$$\Rightarrow \left(X_{1i} - \frac{\sum_{j=1}^{n}X_{1j}X_{2j}}{\sum_{j=1}^{n}X_{2j}^2}X_{2i}\right) = \text{residual for } X_{i1} = X_{i2}\hat{\gamma} + \epsilon_i$$

- we can rewrite

$$\sum_{i=1}^{n}\left[\left(Y_i - \frac{\sum_{j=1}^{n}Y_jX_{2j}}{\sum_{j=1}^{n}X_{2j}^2}X_{2i}\right) - \hat{\beta}_1\left(X_{1i} - \frac{\sum_{j=1}^{n}X_{1j}X_{2j}}{\sum_{j=1}^{n}X_{2j}^2}X_{2i}\right)\right]X_{1i} = 0 \qquad (3)$$

as

$$\sum_{i=1}^{n}\left[e_{i,Y|X_1} - \hat{\beta}_1 e_{i,X_1|X_2}\right]X_{1i} = 0$$

where

$$e_{i,a|b} = a_i - \frac{\sum_{j=1}^{n}a_jb_j}{\sum_{j=1}^{n}b_j^2}b_i$$

which is interpreted as the residual when regressing b from a without an intercept

- solving **(3)**, we get

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n e_{i,Y|X_2} e_{i,X_1|X_2}}{\sum_{i=1}^n e_{i,X_1|X_2} X_{1i}} \qquad (4)$$

- to simplify the denominator, we will look at

$$\begin{aligned}
\sum_{i=1}^n e_{i,X_1|X_2}^2 &= \sum_{i=1}^n e_{i,X_1|X_2} \left(X_{1i} - \frac{\sum_{j=1}^n X_{1j} X_{2j}}{\sum_{j=1}^n X_{2j}^2} X_{2i} \right) \\
&= \sum_{i=1}^n e_{i,X_1|X_2} X_{1i} - \frac{\sum_{j=1}^n X_{1j} X_{2j}}{\sum_{j=1}^n X_{2j}^2} \sum_{i=1}^n e_{i,X_1|X_2} X_{2i} \\
&\quad (\textit{recall that } \sum_{i=1}^n e_i X_i = 0, \textit{ so the } 2^{nd} \textit{ term is } 0) \\
\Rightarrow \sum_{i=1}^n e_{i,X_1|X_2}^2 &= \sum_{i=1}^n e_{i,X_1|X_2} X_{1i}
\end{aligned}$$

- plugging the above equation back in to **(4)**, we get

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n e_{i,Y|X_2} e_{i,X_1|X_2}}{\sum_{i=1}^n e_{i,X_1|X_2}^2}$$

- **general case**
  - pick one regressor and to replace all other variables by the residuals of their regressions against that one

$$\sum_{i=1}^n (Y_i - X_{1i}\hat{\beta}_1 - \ldots - X_{pi}\hat{\beta}_p) X_k = 0$$

    for $k = 1, \ldots, p$ yields p equations with p unknowns
  - holding $\hat{\beta}_1, \ldots, \hat{\beta}_{p-1}$ constant, we get

$$\hat{\beta}_p = \frac{\sum_{i=1}^n (Y_i - X_{1i}\hat{\beta}_1 - \ldots - X_{p-1,i}\hat{\beta}_{p-1}) X_{pi}}{\sum_{i=1}^n X_{pi}^2}$$

  - plugging $\hat{\beta}_p$ back into the equation

$$\sum_{i=1}^n (e_{i,Y|X_p} - e_{i,X_1|X_p}\hat{\beta}_1 - \ldots - e_{i,X_{p-1}|X_p}\hat{\beta}_{p-1}) X_k = 0$$

  - since we know that

$$X_k = e_{i,X_i|X_p} + \frac{\sum_{i=1}^n X_{ki} X_{pi}}{\sum_{i=1}^n X_{pi}^2} X_p$$

    and that

$$\sum_{i=1}^n e_{i,X_i|X_p} X_{pi} = 0$$

    the equation becomes

$$\sum_{i=1}^n (e_{i,Y|X_p} - e_{i,X_1|X_p}\hat{\beta}_1 - \ldots - e_{i,X_{p-1}|X_p}\hat{\beta}_{p-1}) e_{i,X_k|X_p} = 0$$

  - this procedure reduces p LS equations and p unknowns to $p-1$ LS equations and $p-1$ unknowns
    - every variable is replaced by its residual with X_p

    * process iterates until **only** Y and **one variable remains**
    * or more intuitively, we take residuals over the confounding variables and do regression through the origin

- ***example***

  - for simple linear regression, $Y_i = \beta_1 X_{1i} + \beta_2 X_{2i}$ where $X_{2i} = 1$ is an intercept term
  - the residuals
$$e_{i,Y|X_2} = Y_i - \frac{\sum_{j=1}^n Y_j X_{2j}}{\sum_{j=1}^n X_{2j}^2} X_{2i} = Y_i - \bar{Y}$$
    * ***Note**: this is according to previous derivation of the slope of a regression line through the origin*
  - the residuals $e_{i,X_1|X_2} = X_{1i} - \frac{\sum_{j=1}^n X_{1j} X_{2j}}{\sum_{j=1}^n X_{2j}^2} X_{2i} = X_{1i} - \bar{X}_1$
    * ***Note**: this is according to previous derivation of the slope of a regression line through the origin*
  - Thus
$$\hat{\beta}_1 = \frac{\sum_{i=1}^n e_{i,Y|X_2} e_{i,X_1|X_2}}{\sum_{i=1}^n e_{i,X_1|X_2}^2} = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2} = Cor(X,Y)\frac{Sd(Y)}{Sd(X)}$$

Interpretation of Coefficients

- from the derivation in the previous section, we have
$$\hat{\beta}_1 = \frac{\sum_{i=1}^n e_{i,Y|X_2} e_{i,X_1|X_2}}{\sum_{i=1}^n e_{i,X_1|X_2}^2}$$

- this is interpreted as the effect of variable X_1 when the effects of all other variables have been removed from X_1 and the predicted result Y (holding everything else constant/adjusting for all other variables)

- the expected response is as follows
$$E[Y|X_1 = x_1, \ldots, X_p = x_p] = \sum_{k=1}^p x_k \beta_k$$
  so the expected change in the response through change in one variable is
$$\begin{aligned}
&E[Y|X_1 = x_1 + 1, \ldots, X_p = x_p] - E[Y|X_1 = x_1, \ldots, X_p = x_p] \\
=&(x_1 + 1)\beta_1 + \sum_{k=2}^p x_k \beta_k - \sum_{k=1}^p x_k \beta_k \\
=&(x_1 + 1)\beta_1 + \sum_{k=2}^p x_k \beta_k - (x_1 \beta_1 + \sum_{k=2}^p x_k \beta_k) \\
=&\beta_1
\end{aligned}$$

- therefore, interpretation of a multivariate regression coefficient $\rightarrow$ expected change in the response per unit change in the regressor, holding all of the other regressors fixed

- all of the SLR properties/calculations extend to generalized linear model
  - ***model*** $= Y_i = \sum_{k=1}^p X_{ik}\beta_k + \epsilon_i$ where $\epsilon_i \sim N(0, \sigma^2)$
  - ***fitted response*** $= \hat{Y}_i = \sum_{k=1}^p X_{ik}\hat{\beta}_k$

- ***residual*** $= e_i = Y_i - \hat{Y}_i$
- ***variance estimate*** $= \hat{\sigma}^2 = \frac{1}{n-p}\sum_{i=1}^{n} e_i^2$
- ***predicted responses at new values,*** $x_1, \ldots, x_p$ = plug x values into $\sum_{k=1}^{p} x_k \hat{\beta}_k$
- ***standard errors of coefficients*** $= \hat{\sigma}_{\hat{\beta}_k}$
- ***test/CI statistic*** $= \frac{\hat{\beta}_k - \beta_k}{\hat{\sigma}_{\hat{\beta}_k}}$ follows a T distribution with $n - p$ degrees of freedom
- ***predicted/expected response intervals*** = calculated using standard errors of predicted responses of $\hat{Y}_i$

Example: Linear Model with 2 Variables and Intercept

```
# simulate the data
n = 100; x = rnorm(n); x2 = rnorm(n); x3 = rnorm(n)
# equation = intercept + var1 + var2 + var3 + error
y = 1 + x + x2 + x3 + rnorm(n, sd = .1)
# residual of y regressed on var2 and var3
ey = resid(lm(y ~ x2 + x3))
# residual of y regressed on var2 and var3
ex = resid(lm(x ~ x2 + x3))
# estimate beta1 for var1
sum(ey * ex) / sum(ex ^ 2)
```

```
## [1] 0.9883021
```

```
# regression through the origin with xva1 with var2 var3 effect removed
coef(lm(ey ~ ex - 1))
```

```
##        ex
## 0.9883021
```

```
# regression for all three variables
coef(lm(y ~ x + x2 + x3))
```

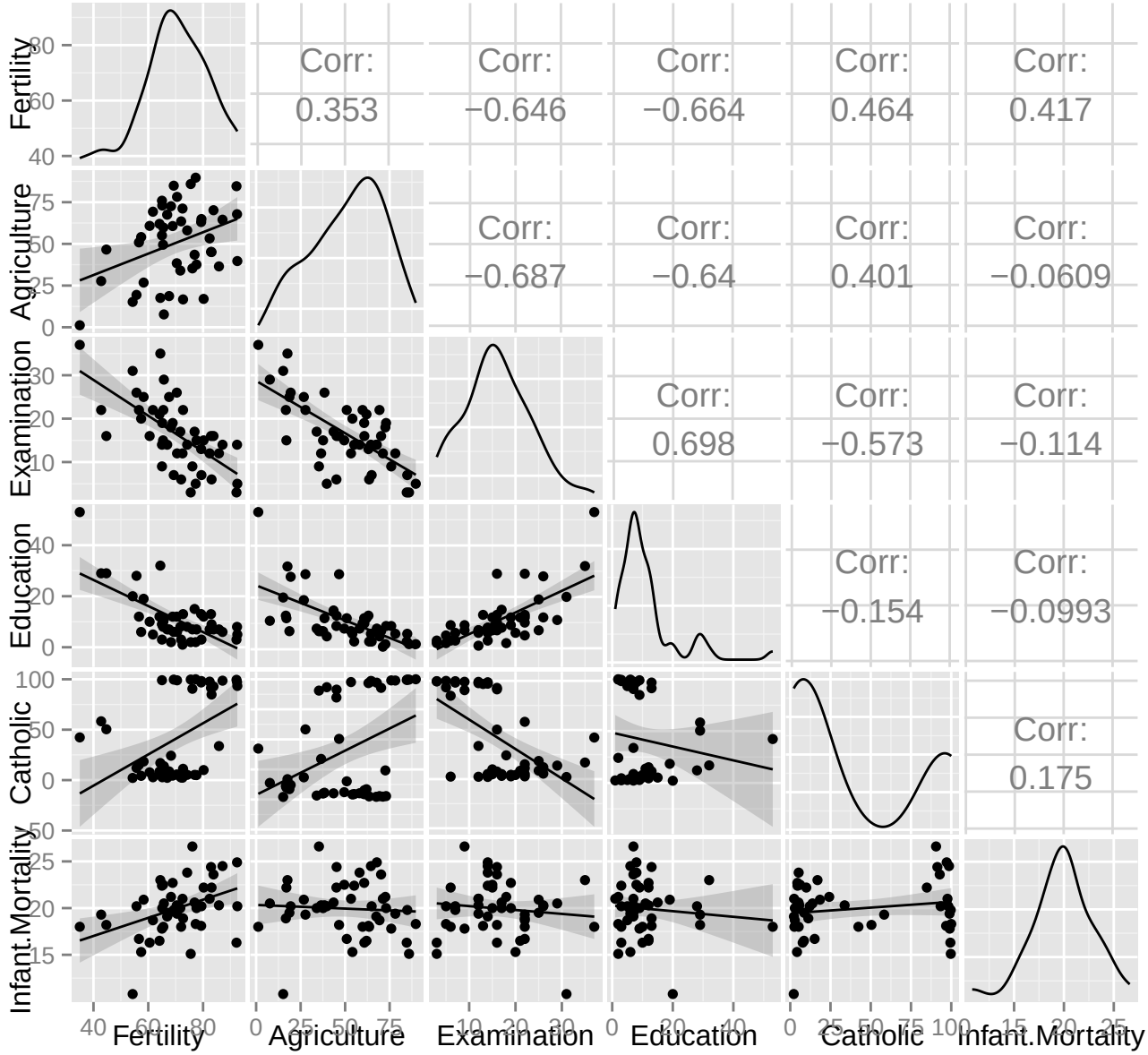
```
## (Intercept)           x          x2          x3
##   1.0093454   0.9883021   0.9973557   1.0127424
```

Example: Coefficients that Reverse Signs

- ***Note****: more information can be found at* `?swiss`
- data set is composed of standardized fertility measure and socio-economic indicators for each of 47 French-speaking provinces of Switzerland at about 1888
- data frame has 47 observations on 6 variables, each of which is in percent [0, 100]
  - ***Fertility*** = common standardized fertility measure → outcome
  - ***Agriculture*** = % of males involved in agriculture as occupation
  - ***Examination*** = % draftees receiving highest mark on army examination
  - ***Education*** = % education beyond primary school for draftees
  - ***Catholic*** = % catholic vs protestant
  - ***Infant.Mortality*** = live births who live less than 1 year

- [GGally package] `ggpairs(data)` = produces pair wise plot for the predictors similar to `pairs` in base package

```
# load dataset
require(datasets); data(swiss); require(GGally)
# produce pairwise plot using ggplot2
ggpairs(swiss, lower = list(continuous = "smooth"),params = c(method = "loess"))
```




```
# print coefficients of regression of fertility on all predictors
summary(lm(Fertility ~ . , data = swiss))$coefficients
```

```
##                    Estimate  Std. Error    t value     Pr(>|t|)
## (Intercept)      66.9151817 10.70603759  6.250229 1.906051e-07
## Agriculture      -0.1721140  0.07030392 -2.448142 1.872715e-02
```

```
## Examination      -0.2580082  0.25387820 -1.016268 3.154617e-01
## Education        -0.8709401  0.18302860 -4.758492 2.430605e-05
## Catholic          0.1041153  0.03525785  2.952969 5.190079e-03
## Infant.Mortality  1.0770481  0.38171965  2.821568 7.335715e-03
```

- interpretation for Agriculture coefficient
  - we expect an -0.17 **decrease** in standardized ***fertility*** for every 1% increase in percentage of ***males involved in agriculture*** in holding the remaining variables constant
  - since the p-value is 0.0187272, the t-test for $H_0 : \beta_{Agri} = 0$ versus $H_a : \beta_{Agri} \neq 0$ is ***significant***
- however, if we look at the unadjusted estimate (marginal regression) for the coefficient for Agriculture

```
# run marginal regression on Agriculture
summary(lm(Fertility ~ Agriculture, data = swiss))$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 60.3043752 4.25125562 14.185074 3.216304e-18
## Agriculture  0.1942017 0.07671176  2.531577 1.491720e-02
```

- interpretation for Agriculture coefficient
  - we expect an 0.19 **increase** in standardized ***fertility*** for every 1% increase in percentage of ***males involved in agriculture*** in holding the remaining variables constant
    - ***Note****: the coefficient* ***flipped signs***
  - since the p-value is 0.0149172, the t-test for $H_0 : \beta_{Agri} = 0$ versus $H_a : \beta_{Agri} \neq 0$ is ***significant***
- to see intuitively how a **sign change** is possible, we can look at the following simulated example

```
# simulate data
n <- 100; x2 <- 1 : n; x1 <- .01 * x2 + runif(n, -.1, .1); y = -x1 + x2 + rnorm(n, sd = .01)
# print coefficients
c("with x1" = summary(lm(y ~ x1))$coef[2,1],
    "with x1 and x2" = summary(lm(y ~ x1 + x2))$coef[2,1])
```

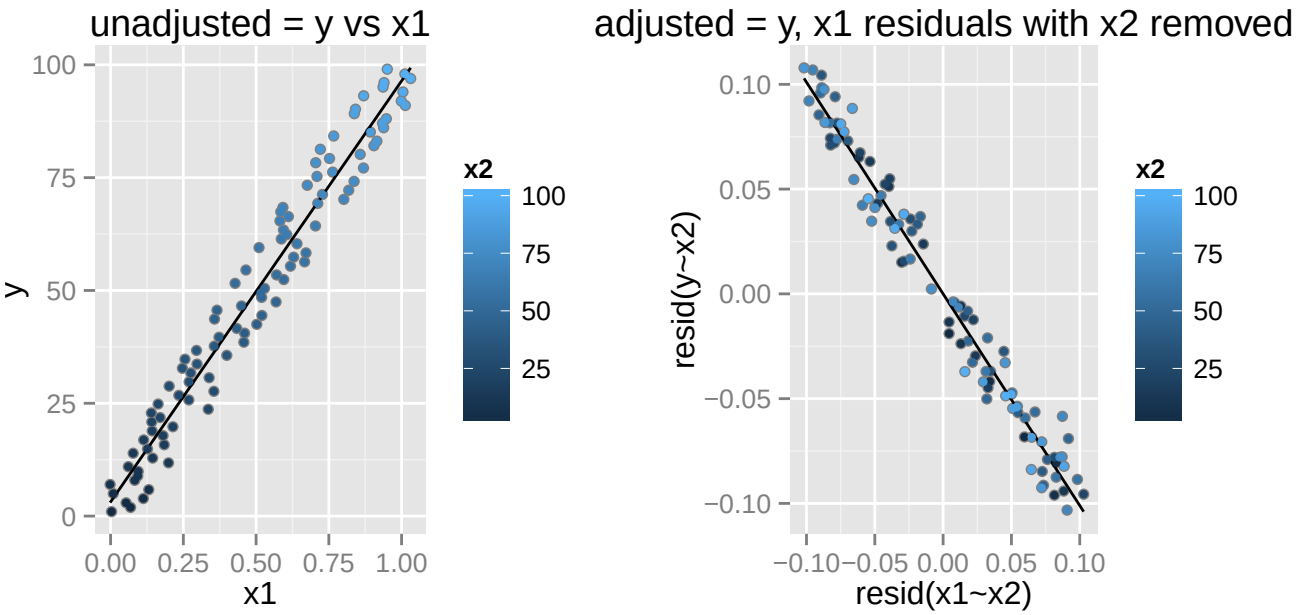
```
##        with x1 with x1 and x2
##      93.292440      -1.010287
```

```
# print p-values
c("with x1" = summary(lm(y ~ x1))$coef[2,4],
    "with x1 and x2" = summary(lm(y ~ x1 + x2))$coef[2,4])
```

```
##        with x1 with x1 and x2
##   2.910034e-69   2.645449e-76
```

```
# store all data in one data frame (ey and ex1 are residuals with respect to x2)
dat <- data.frame(y = y, x1 = x1, x2 = x2, ey = resid(lm(y ~ x2)), ex1 = resid(lm(x1 ~ x2)))
# plot y vs x1
g <- ggplot(dat, aes(y = y, x = x1, colour = x2)) +
    geom_point(colour="grey50", size = 2) +
    geom_smooth(method = lm, se = FALSE, colour = "black") + geom_point(size = 1.5) +
    ggtitle("unadjusted = y vs x1")
```

```r
# plot residual of y adjusted for x2 vs residual of x1 adjusted for x2
g2 <- ggplot(dat, aes(y = ey, x = ex1, colour = x2)) +
    geom_point(colour="grey50", size = 2) +
    geom_smooth(method = lm, se = FALSE, colour = "black") + geom_point(size = 1.5) +
    ggtitle("adjusted = y, x1 residuals with x2 removed") + labs(x = "resid(x1~x2)",
        y = "resid(y~x2)")
# combine plots
multiplot(g, g2, cols = 2)
```



- as we can see from above, the correlation between `y` and `x1` flips signs when adjusting for `x2`
- this effectively means that within each consecutive group/subset of points (each color gradient) on the left hand plot (unadjusted), there exists a negative relationship between the points while the overall trend is going up
- ***going back to the swiss data set***, the sign of the coefficient for Agriculture *reverses* itself with the inclusion of Examination and Education (both are ***negatively correlated*** with Agriculture)
  - correlation between Agriculture and Education: -0.64
  - correlation between Agriculture and Examination: -0.69
  - correlation between Education and Examination: 0.7
    - this means that the two variables are likely to be measuring the same things
- ***Note****: it is* ***difficult to interpret*** *and determine which one is the correct model* $\rightarrow$ *one* ***should not*** *claim positive correlation between Agriculture and Fertility simply based on marginal regression* `lm(Fertility ~ Agriculture, data=swiss)`

Example: Unnecessary Variables

- **unnecessary predictors** = variables that don't provide any new linear information, meaning that the variable are simply ***linear combinations*** (multiples, sums) of other predictors/variables

- when running a linear regression with unnecessary variables, R automatically drops the linear combinations and returns `NA` as their coefficients

```
# add a linear combination of agriculture and education variables
z <- swiss$Agriculture + swiss$Education
# run linear regression with unnecessary variables
lm(Fertility ~ . + z, data = swiss)$coef
```

```
##      (Intercept)      Agriculture      Examination        Education
##       66.9151817       -0.1721140       -0.2580082       -0.8709401
##         Catholic Infant.Mortality                z
##        0.1041153        1.0770481               NA
```

- as we can see above, the R dropped the unnecessary variable `z` by excluding it from the linear regression → `z`'s coefficient is `NA`

Dummy Variables

- **dummy variables** = binary variables that take on value of **1** when the measurement is in a particular group, and **0** when the measurement is not (i.e. in clinical trials, treated = 1, untreated = 1)
- in linear model form,

$$Y_i = \beta_0 + X_{i1}\beta_1 + \epsilon_i$$

where X_{i1} is a binary/dummy variable so that it is a 1 if measurement i is in a group and 0 otherwise

  - for people in the group, the mean or $\mu_{X_{i1}=1} = E[Y_i] = \beta_0 + \beta_1$
  - for people ***not*** in the group, the mean or $\mu_{X_{i1}=0} = E[Y_i] = \beta_0$
  - predicted mean for group = $\hat{\beta}_0 + \hat{\beta}_1$
  - predicted mean for not in group = $\hat{\beta}_0$
  - coefficient β_1 for X_{i1} is interpreted as the ***increase or decrease*** in the **mean** when comparing two groups (in vs not)
  - ***Note****: including a dummy variable that is 1 for not in the group would be* ***redundant*** *as it would simply be a linear combination* $1 - X_{i1}$

More Than 2 Levels

- for 3 factor levels, we would need 2 dummy variables and the model would be

$$Y_i = \beta_0 + X_{i1}\beta_1 + X_{i2}\beta_2 + \epsilon_i$$

- for this example, we will use the above model to analyze US political party affiliations (Democrats vs Republicans vs independents) and denote the variables as follows:

  - $X_{i1} = 1$ for Republicans and 0 otherwise
  - $X_{i2} = 1$ for Democrats and 0 otherwise
  - If i is Republican, $X_{i1} = 1$, $X_{i2} = 0$, $E[Y_i] = \beta_0 + \beta_1$
  - If i is Democrat, $X_{i1} = 0$, $X_{i2} = 1$, $E[Y_i] = \beta_0 + \beta_2$
  - If i is independent, $X_{i1} = 0$, $X_{i2} = 0$, $E[Y_i] = \beta_0$
  - β_1 compares Republicans to independents
  - β_2 compares Democrats to independents
  - $\beta_1 - \beta_2$ compares Republicans to Democrats

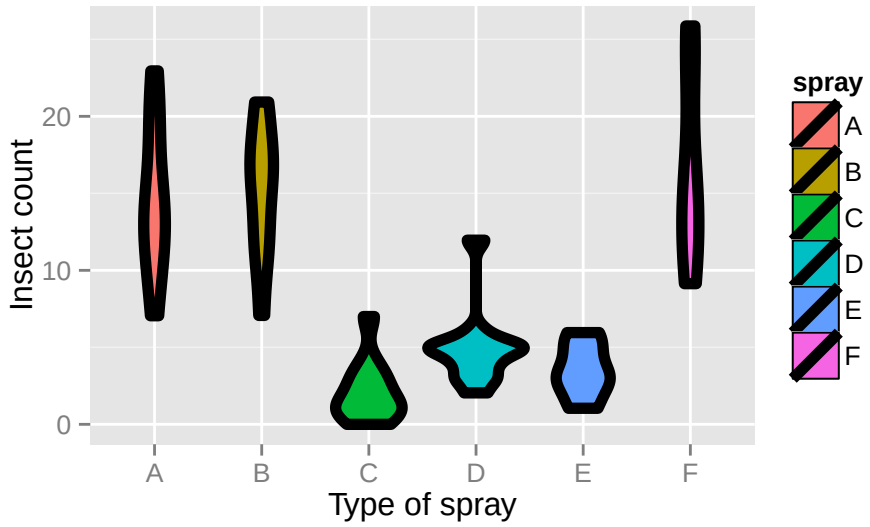
- ***Note****: choice of reference category (independent in this case) changes the interpretation*
  - **reference category** = the group whose binary variable has been eliminated
- the same principles explained above can be expanded to p level model

$$Y_i = \beta_0 + X_{i1}\beta_1 + X_{i2}\beta_2 + \ldots + X_{ip}\beta_p + \epsilon_i$$

Example: 6 Factor Level Insect Spray Data

- below is a violin plot of the 6 different types (A, B, C, D, E, and F) of insect sprays and their potency (kill count) from `InsectSprays` data set
  - ***Note****: the varying width of each bar indicates the density of measurement at each value*

```
# load insect spray data
data(InsectSprays)
ggplot(data = InsectSprays, aes(y = count, x = spray, fill  = spray)) +
    geom_violin(colour = "black", size = 2) + xlab("Type of spray") +
    ylab("Insect count")
```



Linear model fit with group A as reference category

```
# linear fit with 5 dummy variables
summary(lm(count ~ spray, data = InsectSprays))$coefficients
```

```
##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  14.5000000   1.132156 12.8074279 1.470512e-19
## sprayB        0.8333333   1.601110  0.5204724 6.044761e-01
## sprayC      -12.4166667   1.601110 -7.7550382 7.266893e-11
## sprayD       -9.5833333   1.601110 -5.9854322 9.816910e-08
## sprayE      -11.0000000   1.601110 -6.8702352 2.753922e-09
## sprayF        2.1666667   1.601110  1.3532281 1.805998e-01
```

- ***Note**: R automatically converts factor variables into $n-1$ dummy variables and uses the first category as reference*
  - mean of group A is therefore the default intercept
- the above coefficients can be interpreted as the difference in means between each group (B, C, D, E, and F) and group A (the intercept)
  - example: the mean of group B is 0.83 higher than the mean of group A, which is 14.5
  - means for group B/C/D/E/F = the intercept + their respective coefficient
- all t-tests are for comparisons of Sprays versus Spray A

Hard-coding the dummy variables

- this produces the exact same result as the command `lm(count ~ spray, data = InsectSprays)`

```
# hard coding dummy variables
lm(count ~ I(1 * (spray == 'B')) + I(1 * (spray == 'C')) +
           I(1 * (spray == 'D')) + I(1 * (spray == 'E')) +
           I(1 * (spray == 'F')), data = InsectSprays)$coefficients
```

```
##           (Intercept) I(1 * (spray == "B")) I(1 * (spray == "C"))
##            14.5000000             0.8333333           -12.4166667
## I(1 * (spray == "D")) I(1 * (spray == "E")) I(1 * (spray == "F"))
##            -9.5833333           -11.0000000             2.1666667
```

Linear model fit with all 6 categories

```
# linear fit with 6 dummy variables
lm(count ~ I(1 * (spray == 'B')) + I(1 * (spray == 'C')) +
           I(1 * (spray == 'D')) + I(1 * (spray == 'E')) +
           I(1 * (spray == 'F')) + I(1 * (spray == 'A')),
           data = InsectSprays)$coefficients
```

```
##           (Intercept) I(1 * (spray == "B")) I(1 * (spray == "C"))
##            14.5000000             0.8333333           -12.4166667
## I(1 * (spray == "D")) I(1 * (spray == "E")) I(1 * (spray == "F"))
##            -9.5833333           -11.0000000             2.1666667
## I(1 * (spray == "A"))
##                    NA
```

- as we can see from above, the coefficient for group `A` is `NA`
- this is because $X_{iA} = 1 - X_{iB} - X_{iC} - X_{iD} - X_{iE} - X_{iF}$, or the dummy variable for A is a linear combination of the rest of the dummy variables

Linear model fit with omitted intercept

- eliminating the intercept would mean that each group is compared to the value 0, which would yield 6 variables since A is no longer the reference category
- this means that the coefficients for the 6 variables are simply the ***mean*** of each group
  - when $X_{iA} = 1$, all the other dummy variables become 0, which means the linear model becomes

$$Y_i = \beta_A + \epsilon_i$$

  - then $E[Y_i] = \beta_A = \mu_A$
  - this makes sense because the ***best prediction*** for the kill count of spray of type A is the ***mean*** of recorded kill counts of A spray

```
# linear model with omitted intercept
summary(lm(count ~ spray - 1, data = InsectSprays))$coefficients
```

```
##         Estimate Std. Error   t value     Pr(>|t|)
## sprayA 14.500000   1.132156 12.807428 1.470512e-19
## sprayB 15.333333   1.132156 13.543487 1.001994e-20
## sprayC  2.083333   1.132156  1.840148 7.024334e-02
## sprayD  4.916667   1.132156  4.342749 4.953047e-05
## sprayE  3.500000   1.132156  3.091448 2.916794e-03
## sprayF 16.666667   1.132156 14.721181 1.573471e-22
```

```
# actual means of count by each variable
round(tapply(InsectSprays$count, InsectSprays$spray, mean), 2)
```

```
##     A     B     C     D     E     F
## 14.50 15.33  2.08  4.92  3.50 16.67
```

- all t-tests are for whether the groups are different than zero (i.e. are the expected counts 0 for that spray?)
- to compare between different categories, say B vs C, we can simply subtract the coefficients
- to reorient the model with other groups as reference categories, we can simply **reorder the levels** for the factor variable
  - `relevel(var, "l")` = reorders the factor levels within the factor variable `var` such that the specified level "l" is the reference/base/lowest level
    - ***Note**: R automatically uses the first/base level as the reference category during a linear regression*

```r
# reorder the levels of spray variable such that C is the lowest level
spray2 <- relevel(InsectSprays$spray, "C")
# rerun linear regression with releveled factor
summary(lm(count ~ spray2, data = InsectSprays))$coef
```

```
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept)  2.083333   1.132156 1.840148 7.024334e-02
## spray2A     12.416667   1.601110 7.755038 7.266893e-11
## spray2B     13.250000   1.601110 8.275511 8.509776e-12
## spray2D      2.833333   1.601110 1.769606 8.141205e-02
## spray2E      1.416667   1.601110 0.884803 3.794750e-01
## spray2F     14.583333   1.601110 9.108266 2.794343e-13
```

- it is important to note in this example that
  - counts are bounded from below by 0 → ***violates*** the assumption of normality of the errors
  - there are counts near zero → ***violates*** intent of the assumption → not acceptable in assuming normal distribution
  - variance does not appear to be constant across different type of groups → violates assumption
    - taking log(counts) + 1 may help (+1 since there are the zero values)
  - ***Poisson GLMs*** are better (don't have to worry about the assumptions) for fitting count data

Interactions

- **interactions** between variables can be added to a regression model to test how the outcomes change under different conditions
- we will use the data set from the Millennium Development Goal from the UN which can be found here
  - `Numeric` = values for children aged <5 years underweight (%)
  - `Sex` = records whether
  - `Year` = year when data was recorded
  - `Income` = income for the child's parents

```
# load in hunger data
hunger <- read.csv("hunger.csv")
# exclude the data with "Both Sexes" as values (only want Male vs Female)
hunger <- hunger[hunger$Sex!="Both sexes", ]
# structure of data
str(hunger)
```

```
## 'data.frame':    948 obs. of  12 variables:
##  $ Indicator     : Factor w/ 1 level "Children aged <5 years underweight (%)": 1 1 1 1 1 1 1 1 1 1 .
##  $ Data.Source   : Factor w/ 670 levels "NLIS_310005",..: 7 52 519 380 548 551 396 503 643 632 ...
##  $ PUBLISH.STATES: Factor w/ 1 level "Published": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Year          : int  1986 1990 2005 2002 2008 2008 2003 2006 2012 1999 ...
##  $ WHO.region    : Factor w/ 6 levels "Africa","Americas",..: 1 2 2 3 1 1 1 2 1 2 ...
##  $ Country       : Factor w/ 151 levels "Afghanistan",..: 115 104 97 69 57 54 71 13 115 144 ...
##  $ Sex           : Factor w/ 3 levels "Both sexes","Female",..: 3 3 3 2 2 3 3 3 3 2 ...
##  $ Display.Value : num  19.3 2.2 5.3 3.2 17 15.7 19.3 4 15.5 4.2 ...
##  $ Numeric       : num  19.3 2.2 5.3 3.2 17 15.7 19.3 4 15.5 4.2 ...
##  $ Low           : logi  NA NA NA NA NA NA ...
##  $ High          : logi  NA NA NA NA NA NA ...
##  $ Comments      : logi  NA NA NA NA NA NA ...
```

Model: % Hungry ~ Year by Sex

- this will include 2 models with 2 separate lines
- model for % hungry (H_F) vs year (Y_F) for females is

$$H_{Fi} = \beta_{F0} + \beta_{F1} Y_{Fi} + \epsilon_{Fi}$$

  - β_{F0} = % of females hungry at year 0
  - β_{F1} = decrease in % females hungry per year
  - ϵ_{Fi} = standard error (or everything we didn't measure)

- model for % hungry (H_M) vs year (Y_M) for males is

$$H_{Mi} = \beta_{M0} + \beta_{M1} Y_{Mi} + \epsilon_{Mi}$$

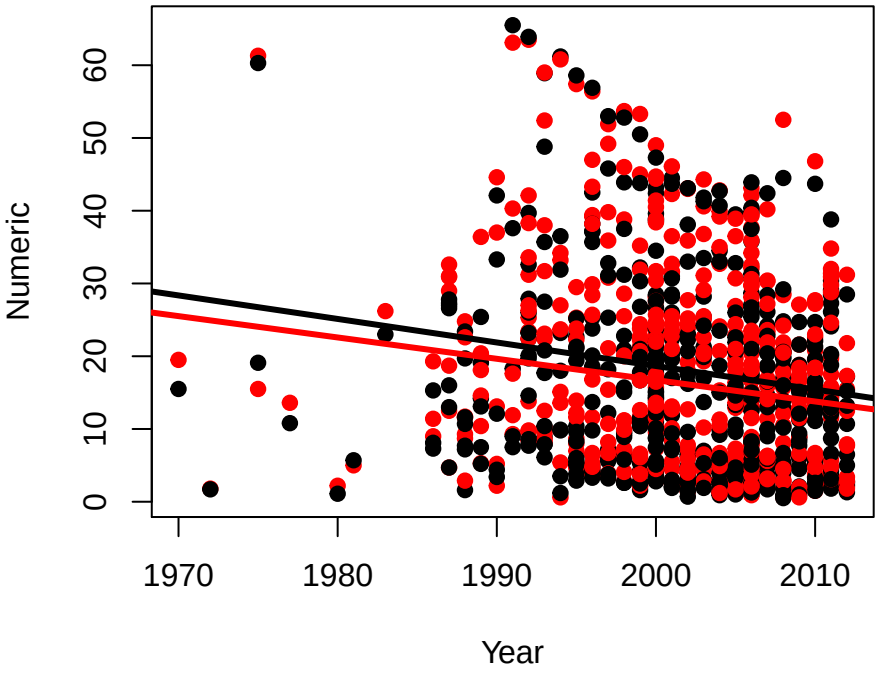
  - β_{M0} = % of males hungry at year 0
  - β_{M1} = decrease in % males hungry per year
  - ϵ_{Mi} = standard error (or everything we didn't measure)

- each line has **different** residuals, standard errors, and variances
- ***Note**: β_{F0} and β_{M0} are the interpolated intercept at year 0, which does hold any interpretable value for the model*

- it's possible to subtract the model by a meaningful value (% hungry at 1970, or average), which moves the intercept of the lines to something interpretable

- ***Note**: we are also assuming the error terms ϵ_{Fi} and ϵ_{Mi} are Gaussian distributions $\rightarrow$ mean = 0*

```
# run linear model with Numeric vs Year for male and females
male.fit <- lm(Numeric ~ Year, data = hunger[hunger$Sex == "Male", ])
female.fit <- lm(Numeric ~ Year, data = hunger[hunger$Sex == "Female", ])
# plot % hungry vs the year
plot(Numeric ~ Year, data = hunger, pch = 19, col=(Sex=="Male")*1+1)
# plot regression lines for both
abline(male.fit, lwd = 3, col = "black")
abline(female.fit, lwd = 3, col = "red")
```



Model: % Hungry ~ Year + Sex (Binary Variable)

- this will include 1 model with 2 separate lines with the ***same*** slope
- model for % hungry (H) vs year (Y) and dummy variable for sex (X) is

$$H_i = \beta_0 + \beta_1 X_i + \beta_2 Y_i + \epsilon_i^*$$

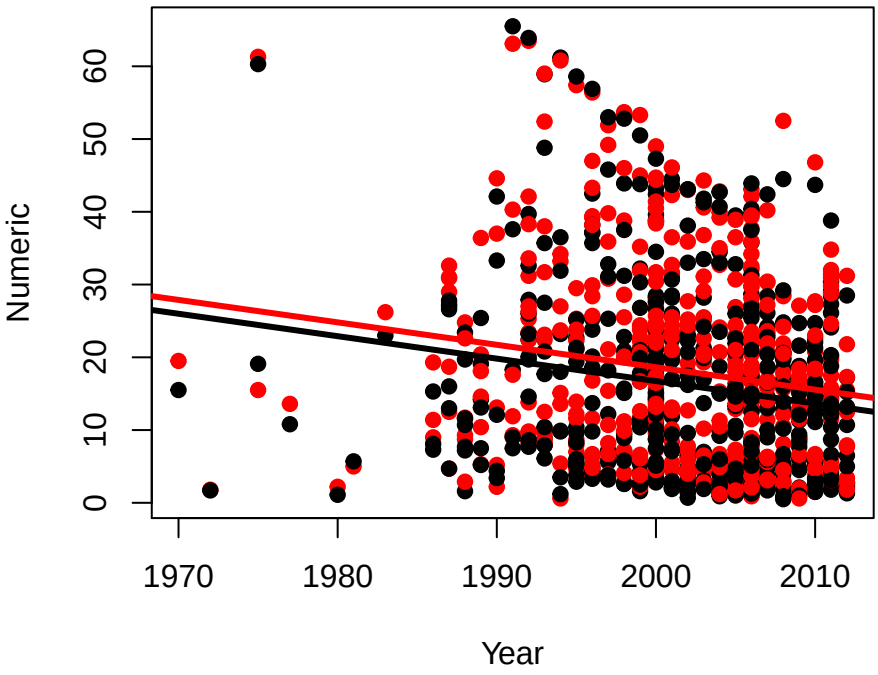
  - β_0 = % of females hungry at year 0
  - $\beta_0 + \beta_1$ = % of males hungry at year 0
    - ***Note**: the term $\beta_1 X_i$ is effectively an **adjustment** for the intercept for males and DOES NOT alter the slope in anyway*
    - β_1 = difference in means of males vs females
  - β_2 = decrease in % hungry (males and females) per year
    - this means that the slope is ***constant*** for both females and males
  - ϵ_i^* = standard error (or everything we didn't measure)
    - we are still assuming Gaussian error term

- `abline(intercept, slope)` = adds a line to the existing plot based on the intercept and slope provided
  - `abline(lm)` = plots the linear regression line on the plot

```
# run linear model with Numeric vs Year and Sex
both.fit <- lm(Numeric ~ Year+Sex, data = hunger)
# print fit
both.fit$coef
```

```
## (Intercept)        Year     SexMale
##  633.528289   -0.308397    1.902743
```

```
# plot % hungry vs the year
plot(Numeric ~ Year, data = hunger, pch = 19, col=(Sex=="Male")*1+1)
# plot regression lines for both (same slope)
abline(both.fit$coef[1], both.fit$coef[2], lwd = 3, col = "black")
abline(both.fit$coef[1]+both.fit$coef[3], both.fit$coef[2], lwd = 3, col = "red")
```



Model: % Hungry ~ Year + Sex + Year * Sex (Binary Interaction)

- this will include 1 model with an interaction term with binary variable, which produces 2 lines with ***different*** slopes
- we can introduce an interaction term to the previous model to capture the different slopes between males and females
- model for % hungry (H) vs year (Y), sex (X), and interaction between year and sex ($Y \times X$) is

$$H_i = \beta_0 + \beta_1 X_i + \beta_2 Y_i + \beta_3 X_i Y_i + \epsilon_i^+$$

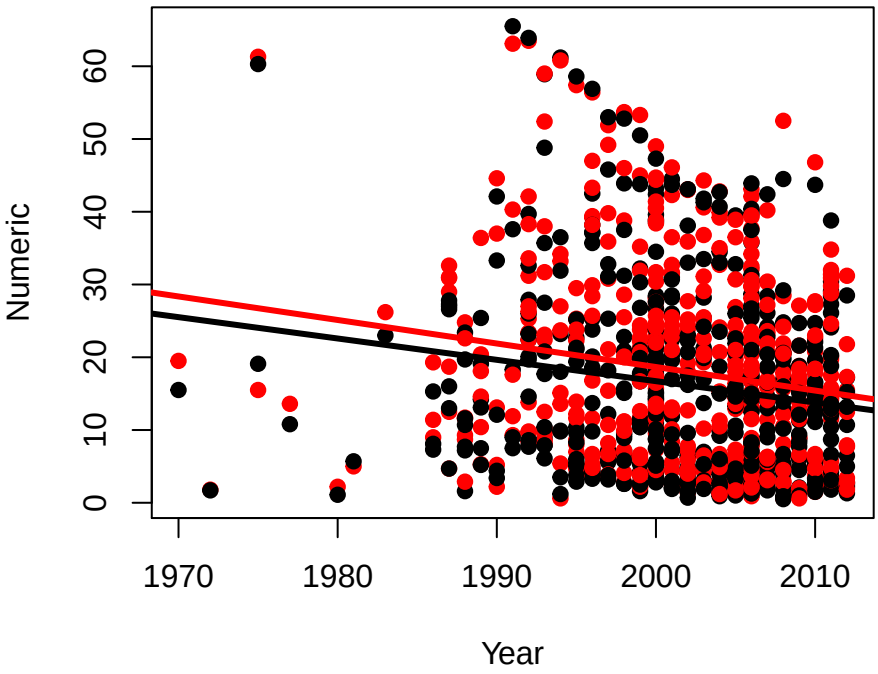
  - β_0 = % of females hungry at year 0
  - $\beta_0 + \beta_1$ = % of males hungry at year 0
    - β_1 = change in ***intercept*** for males

  - β_2 = decrease in % hungry (females) per year
  - $\beta_2 + \beta_3$ = decrease in % hungry (males) per year
    - β_3 = change in ***slope*** for males
  - ϵ_i^+ = standard error (or everything we didn't measure)
- expected value for males is $E[H_i]_M = (\beta_0 + \beta_1) + (\beta_2 + \beta_3)Y_i$
- expected value for females is $E[H_i]_F = \beta_0 + \beta_2 Y_i$
  - β_1 and β_3 are effectively adjusting the intercept and slope for males
- `lm(outcome ~ var1*var2)` = whenever an interaction is specified in `lm` function using the `*` operator, the individual terms are added automatically
  - `lm(outcome ~ var1+var2+var1*var2)` = builds the exact same model
  - `lm(outcome ~ var1:var2)` = builds linear model with ***only*** the interaction term (specified by `:` operator)

```
# run linear model with Numeric vs Year and Sex and interaction term
interaction.fit <- lm(Numeric ~ Year*Sex, data = hunger)
# print fit
interaction.fit$coef
```

```
##  (Intercept)         Year      SexMale Year:SexMale
## 603.50579986  -0.29339638  61.94771998  -0.03000132
```

```
# plot % hungry vs the year
plot(Numeric ~ Year, data = hunger, pch = 19, col=(Sex=="Male")*1+1)
# plot regression lines for both (different slope)
abline(interaction.fit$coef[1], interaction.fit$coef[2], lwd = 3, col = "black")
abline(interaction.fit$coef[1]+interaction.fit$coef[3],
    interaction.fit$coef[2]+interaction.fit$coef[4], lwd = 3, col = "red")
```



Example: % Hungry ~ Year + Income + Year * Income (Continuous Interaction)

- this will include 1 model with an interaction term with continuous variable, which produces a curve through the plot
- for **continuous interactions** (two continuous variables) with model

$$Yi = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{1i} X_{2i} + \epsilon_i$$

  the expected value for a given set of values x_1 and x_2 is defined as

$$E[Y_i | X_{1i} = x_1, X_{2i} = x_2] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$$

- holding X_2 constant and varying X_1 by 1, we have

$$\begin{aligned} \frac{\partial Y_i}{\partial X_{1i}} &= E[Y_i | X_{1i} = x_1 + 1, X_{2i} = x_2] - E[Y_i | X_{1i} = x_1, X_{2i} = x_2] \\ &= \beta_0 + \beta_1 (x_1 + 1) + \beta_2 x_2 + \beta_3 (x_1 + 1) x_2 - [\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2] \\ &= \beta_1 + \beta_3 x_2 \end{aligned}$$

  - ***Note**: this means that slope for X_{1i} **not** a constant and is **dependent** on X_{2i}*
  - β_1 is the slope for X_{1i} when $X_{2i} = 0$
- by the same logic, if we vary X_1 by 1 and find the change, and vary X_2 by 1 and find the change, we get

$$\begin{aligned} \frac{\partial}{\partial X_{2i}} \left(\frac{\partial Y_i}{\partial X_{1i}} \right) &= E[Y_i | X_{1i} = x_1 + 1, X_{2i} = x_2 + 1] - E[Y_i | X_{1i} = x_1, X_{2i} = x_2 + 1] \\ &\quad - \Big(E[Y_i | X_{1i} = x_1 + 1, X_{2i} = x_2] - E[Y_i | X_{1i} = x_1, X_{2i} = x_2] \Big) \\ &= \beta_0 + \beta_1 (x_1 + 1) + \beta_2 (x_2 + 1) + \beta_3 (x_1 + 1)(x_2 + 1) - [\beta_0 + \beta_1 x_1 + \beta_2 (x_2 + 1) + \beta_3 x_1 (x_2 + 1)] \\ &\quad - \Big(\beta_0 + \beta_1 (x_1 + 1) + \beta_2 x_2 + \beta_3 (x_1 + 1) x_2 - [\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2] \Big) \\ &= \beta_3 \end{aligned}$$

  - this can be interpreted as β_3 = the **expected change in** Y per **unit change in** X_1 per **unit change** of X_2
  - in other words, β_3 = the change in slope of X_1 per unit change of X_2
- coming back to the hunger data, model for % hungry (H) vs year (Y), income (I), and interaction between year and income ($Y \times I$) is

$$H_i = \beta_0 + \beta_1 I_i + \beta_2 Y_i + \beta_3 I_i Y_i + \epsilon_i^+$$

  - β_0 = % hungry children (whose parents have no income) at year 0
  - β_1 = change in % hungry children for **each dollar in income** in year 0
  - β_2 = change in % hungry children (whose parents have no income) **per year**
  - β_3 = change in % hungry children **per year** and for **each dollar in income**
    - if income is \$10,000, then the change in % hungry children **per year** will be $\beta_1 - 10000 \times \beta_3$
  - ϵ_i^+ = standard error (or everything we didn't measure)
- ***Note**: much care needs to be taken when interpreting these coefficients*

```
# generate some income data
hunger$Income <- 1:nrow(hunger)*10 + 500*runif(nrow(hunger), 0, 10) +
    runif(nrow(hunger), 0, 500)^1.5
# run linear model with Numeric vs Year and Income and interaction term
lm(Numeric ~ Year*Income, data = hunger)$coef
```

```
##   (Intercept)          Year        Income   Year:Income
##  1.489576e+03 -7.347779e-01 -7.101993e-02  3.541102e-05
```

Multivariable Simulation

- we will generate a series of simulated data so that we know the true relationships, and then run linear regressions to interpret and compare the results to truth
- **treatment effect** = effect of adding the treatment variable t to the regression model (i.e. how adding t changes the regression lines)
  - effectively measures how much the regression lines for the two groups separate with regression `lm(y ~ x + t)`
- **adjustment effect** = adjusting the regression for effects of x such that we just look at how t is ***marginally related*** to Y
  - ignore all variation of x and simply look at the group means of $t = 1$ vs $t = 0$

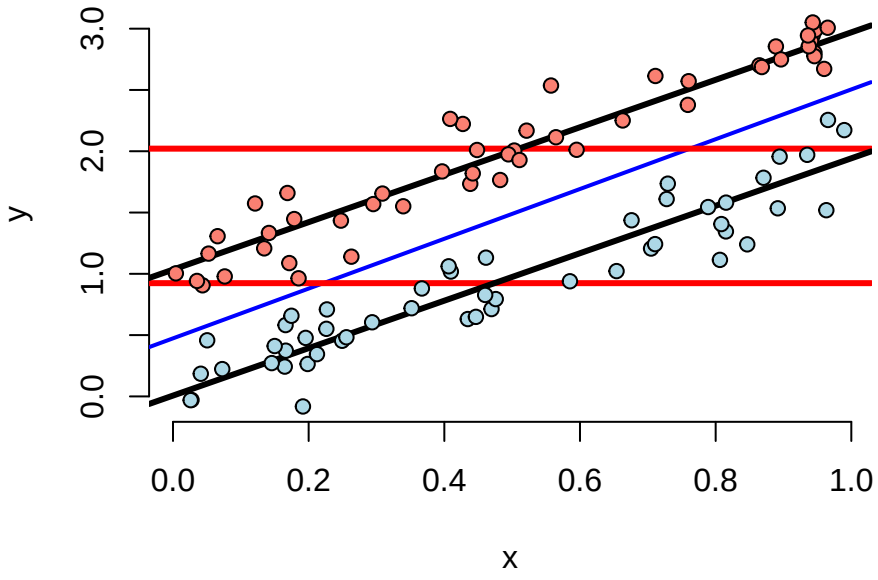
Simulation 1 - Treatment = Adjustment Effect

- the following code simulates the linear model,

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 t_i + \epsilon_i$$

  where $t = \{0, 1\} \rightarrow$ binary variable

```r
# simulate data
n <- 100; t <- rep(c(0, 1), c(n/2, n/2)); x <- c(runif(n/2), runif(n/2));
# define parameters/coefficients
beta0 <- 0; beta1 <- 2; beta2 <- 1; sigma <- .2
# generate outcome using linear model
y <- beta0 + x * beta1 + t * beta2 + rnorm(n, sd = sigma)
# set up axes
plot(x, y, type = "n", frame = FALSE)
# plot linear fit of y vs x
abline(lm(y ~ x), lwd = 2, col = "blue")
# plot means of the two groups (t = 0 vs t = 1)
abline(h = mean(y[1 : (n/2)]), lwd = 3, col = "red")
abline(h = mean(y[(n/2 + 1) : n]), lwd = 3, col = "red")
# plot linear fit of y vs x and t
fit <- lm(y ~ x + t)
# plot the two lines corresponding to (t = 0 vs t = 1)
abline(coef(fit)[1], coef(fit)[2], lwd = 3)
abline(coef(fit)[1] + coef(fit)[3], coef(fit)[2], lwd = 3)
# add in the actual data points
points(x[1 : (n/2)], y[1 : (n/2)], pch = 21, col = "black", bg = "lightblue", cex = 1)
points(x[(n/2 + 1) : n], y[(n/2 + 1) : n], pch = 21, col = "black", bg = "salmon", cex = 1)
```




```
# print treatment and adjustment effects
rbind("Treatment Effect" = lm(y~t+x)$coef[2], "Adjustment Effect" = lm(y~t)$coef[2])
```

```
##                            t
## Treatment Effect  1.027696
## Adjustment Effect 1.097452
```

- in the above graph, the elements are as follows:
  - *red* lines = means for two groups ($t = 0$ vs $t = 1$) $\rightarrow$ two lines representing `lm(y ~ t)`
  - *black* lines = regression lines for two groups ($t = 0$ vs $t = 1$) $\rightarrow$ two lines representing `lm(y ~ t + x)`
  - *blue* line = overall regression of y vs x $\rightarrow$ line representing `lm(y ~ x)`
- from the graph, we can see that
  - x variable is ***unrelated*** to group status t
    - distribution of each group (salmon vs light blue) of Y vs X is effectively the same
  - x variable is ***related*** to Y, but the intercept **depends** on group status t
  - group variable t is ***related*** to Y
    - relationship between t and Y disregarding $x \approx$ the same as holding x constant
    - difference in group means $\approx$ difference in regression lines
    - ***treatment effect*** (difference in regression lines) $\approx$ ***adjustment effect*** (difference in group means)

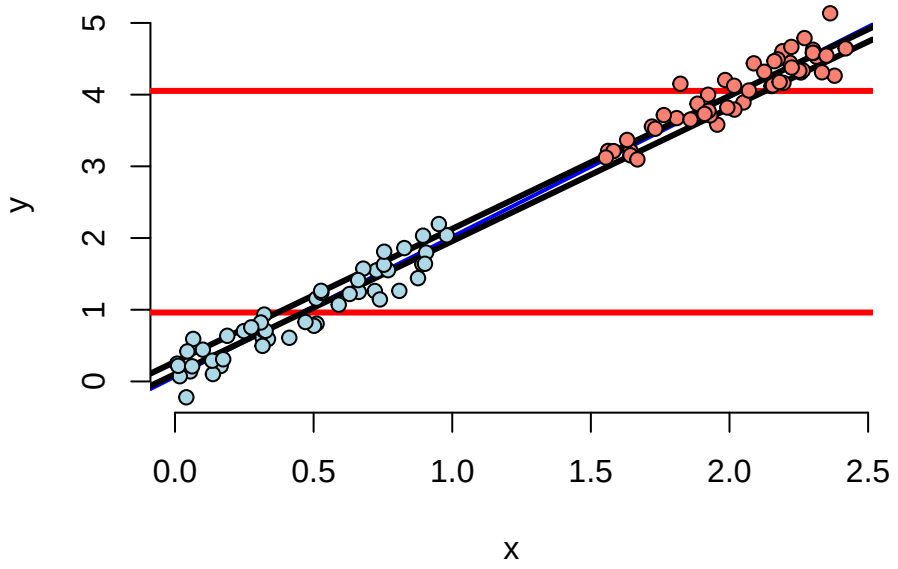
Simulation 2 - No Treatment Effect

- the following code simulates the linear model,

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 t_i + \epsilon_i$$

  where $t = \{0, 1\} \rightarrow$ binary variable
- in this case, β_2 is set to 0

```r
# simulate data
n <- 100; t <- rep(c(0, 1), c(n/2, n/2)); x <- c(runif(n/2), 1.5 + runif(n/2));
# define parameters/coefficients
beta0 <- 0; beta1 <- 2; beta2 <- 0; sigma <- .2
# generate outcome using linear model
y <- beta0 + x * beta1 + t * beta2 + rnorm(n, sd = sigma)
# set up axes
plot(x, y, type = "n", frame = FALSE)
# plot linear fit of y vs x
abline(lm(y ~ x), lwd = 2, col = "blue")
# plot means of the two groups (t = 0 vs t = 1)
abline(h = mean(y[1 : (n/2)]), lwd = 3, col = "red")
abline(h = mean(y[(n/2 + 1) : n]), lwd = 3, col = "red")
# plot linear fit of y vs x and t
fit <- lm(y ~ x + t)
# plot the two lines corresponding to (t = 0 vs t = 1)
abline(coef(fit)[1], coef(fit)[2], lwd = 3)
abline(coef(fit)[1] + coef(fit)[3], coef(fit)[2], lwd = 3)
# add in the actual data points
points(x[1 : (n/2)], y[1 : (n/2)], pch = 21, col = "black", bg = "lightblue", cex = 1)
points(x[(n/2 + 1) : n], y[(n/2 + 1) : n], pch = 21, col = "black", bg = "salmon", cex = 1)
```




```r
# print treatment and adjustment effects
rbind("Treatment Effect" = lm(y~t+x)$coef[2], "Adjustment Effect" = lm(y~t)$coef[2])
```

```
##                            t
## Treatment Effect  0.1717849
## Adjustment Effect 3.0907555
```

- in the above graph, the elements are as follows:
  - *red* lines = means for two groups ($t = 0$ vs $t = 1$) $\rightarrow$ two lines representing `lm(y ~ t)`
  - *black* lines = regression lines for two groups ($t = 0$ vs $t = 1$) $\rightarrow$ two lines representing `lm(y ~ t + x)`

    - in this case, both lines correspond to `lm(y ~ x)` since coefficient of t or $\beta_2 = 0$
  - *blue* line = overall regression of y vs x → line representing `lm(y ~ x)`
    - this is overwritten by the *black* lines
- from the graph, we can see that
  - x variable is ***highly related*** to group status t
    - clear shift in x with salmon vs light blue groups
  - x variable is ***related*** to Y, but the intercept **does not depend** on group status t
    - intercepts for both lines are the same
  - x variable shows ***similar relationships*** to Y for both groups ($t = 0$ vs $t = 1$, or salmon vs lightblue)
    - the x values of the two groups of points both seem to be linearly correlated with Y
  - group variable t is ***marginally related*** to Y when disregarding X
    - x values capture most of the variation
    - ***adjustment effect*** (difference in group means) is very large
  - group variable t is ***unrelated or has very little*** effect on Y
    - ***treatment effect*** is very small or non-existent
    - ***Note:*** *the groups ($t = 0$ vs $t = 1$) are* ***incomparable*** *since there is no data to inform the relationship between t and Y*
    - the groups (salmon vs lightblue) don't have any overlaps so we have no idea how they behave
    - this conclusion is based on the constructed alone

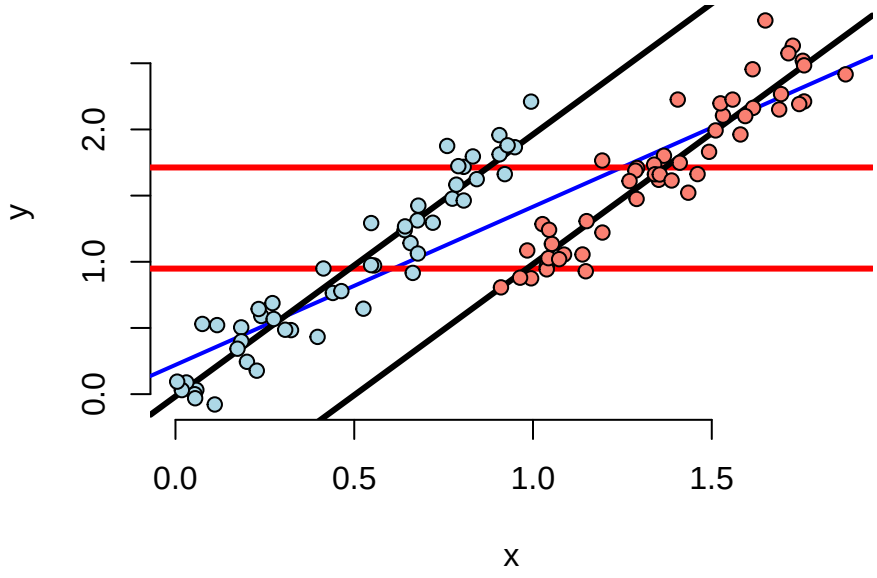
Simulation 3 - Treatment Reverses Adjustment Effect

- the following code simulates the linear model,

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 t_i + \epsilon_i$$

  where $t = \{0, 1\}$ → binary variable
- in this case, β_0 is set to 0 → no intercept

```
# simulate data
n <- 100; t <- rep(c(0, 1), c(n/2, n/2)); x <- c(runif(n/2), .9 + runif(n/2));
# define parameters/coefficients
beta0 <- 0; beta1 <- 2; beta2 <- -1; sigma <- .2
# generate outcome using linear model
y <- beta0 + x * beta1 + t * beta2 + rnorm(n, sd = sigma)
# set up axes
plot(x, y, type = "n", frame = FALSE)
# plot linear fit of y vs x
abline(lm(y ~ x), lwd = 2, col = "blue")
# plot means of the two groups (t = 0 vs t = 1)
abline(h = mean(y[1 : (n/2)]), lwd = 3, col = "red")
abline(h = mean(y[(n/2 + 1) : n]), lwd = 3, col = "red")
# plot linear fit of y vs x and t
fit <- lm(y ~ x + t)
# plot the two lines corresponding to (t = 0 vs t = 1)
abline(coef(fit)[1], coef(fit)[2], lwd = 3)
abline(coef(fit)[1] + coef(fit)[3], coef(fit)[2], lwd = 3)
# add in the actual data points
points(x[1 : (n/2)], y[1 : (n/2)], pch = 21, col = "black", bg = "lightblue", cex = 1)
points(x[(n/2 + 1) : n], y[(n/2 + 1) : n], pch = 21, col = "black", bg = "salmon", cex = 1)
```




```
# print treatment and adjustment effects
rbind("Treatment Effect" = lm(y~t+x)$coef[2], "Adjustment Effect" = lm(y~t)$coef[2])
```

```
##                              t
## Treatment Effect  -0.9801131
## Adjustment Effect  0.7640190
```

- in the above graph, the elements are as follows:
  - *red* lines = means for two groups ($t = 0$ vs $t = 1$) $\rightarrow$ two lines representing `lm(y ~ t)`
  - *black* lines = regression lines for two groups ($t = 0$ vs $t = 1$) $\rightarrow$ two lines representing `lm(y ~ t + x)`
  - *blue* line = overall regression of y vs x $\rightarrow$ line representing `lm(y ~ x)`
- from the graph, we can see that
  - disregarding/adjusting for x, the mean for salmon group is ***higher*** than the mean of the blue group (***adjustment effect*** is positive)
  - when adding t into the linear model, the treatment actually reverses the orders of the group $\rightarrow$ the mean for salmon group is ***lower*** than the mean of the blue group (***treatment effect*** is negative)
  - group variable t is ***related*** to x
  - some points overlap so it is possible to compare the subsets two groups holding x fixed

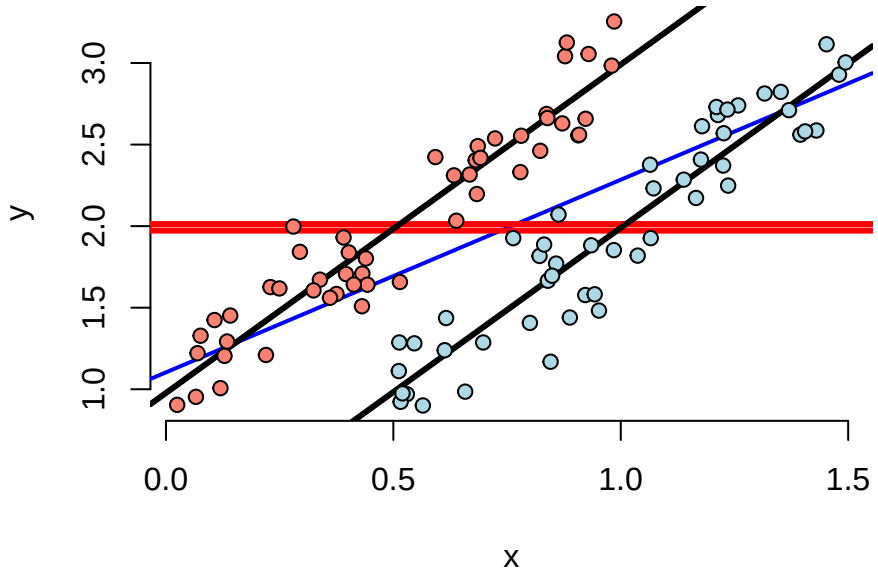
Simulation 4 - No Adjustment Effect

- the following code simulates the linear model,

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 t_i + \epsilon_i$$

  where $t = \{0, 1\}$ $\rightarrow$ binary variable
- in this case, β_0 is set to 0 $\rightarrow$ no intercept

```r
# simulate data
n <- 100; t <- rep(c(0, 1), c(n/2, n/2)); x <- c(.5 + runif(n/2), runif(n/2));
# define parameters/coefficients
beta0 <- 0; beta1 <- 2; beta2 <- 1; sigma <- .2
# generate outcome using linear model
y <- beta0 + x * beta1 + t * beta2 + rnorm(n, sd = sigma)
# set up axes
plot(x, y, type = "n", frame = FALSE)
# plot linear fit of y vs x
abline(lm(y ~ x), lwd = 2, col = "blue")
# plot means of the two groups (t = 0 vs t = 1)
abline(h = mean(y[1 : (n/2)]), lwd = 3, col = "red")
abline(h = mean(y[(n/2 + 1) : n]), lwd = 3, col = "red")
# plot linear fit of y vs x and t
fit <- lm(y ~ x + t)
# plot the two lines corresponding to (t = 0 vs t = 1)
abline(coef(fit)[1], coef(fit)[2], lwd = 3)
abline(coef(fit)[1] + coef(fit)[3], coef(fit)[2], lwd = 3)
# add in the actual data points
points(x[1 : (n/2)], y[1 : (n/2)], pch = 21, col = "black", bg = "lightblue", cex = 1)
points(x[(n/2 + 1) : n], y[(n/2 + 1) : n], pch = 21, col = "black", bg = "salmon", cex = 1)
```




```r
# print treatment and adjustment effects
rbind("Treatment Effect" = lm(y~t+x)$coef[2], "Adjustment Effect" = lm(y~t)$coef[2])
```

```
##                             t
## Treatment Effect  1.00082093
## Adjustment Effect 0.04008387
```

- in the above graph, the elements are as follows:
  - *red* lines = means for two groups ($t = 0$ vs $t = 1$) $\rightarrow$ two lines representing `lm(y ~ t)`
  - *black* lines = regression lines for two groups ($t = 0$ vs $t = 1$) $\rightarrow$ two lines representing `lm(y ~ t + x)`

  - *blue* line = overall regression of y vs x $\rightarrow$ line representing `lm(y ~ x)`
- from the graph, we can see that
  - no ***clear relationship*** between group variable t and Y
    - two groups have similar distributions with respect to Y
  - ***treatment effect*** is substantial
    - separation of regression lines is large
  - ***adjustment effect*** is effectively 0 as there are no large differences between the means of the groups
  - group variable t is ***not related*** to x
    - distribution of each group (salmon vs light blue) of Y vs X is effectively the same
  - lots of direct evidence for comparing two groups holding X fixed

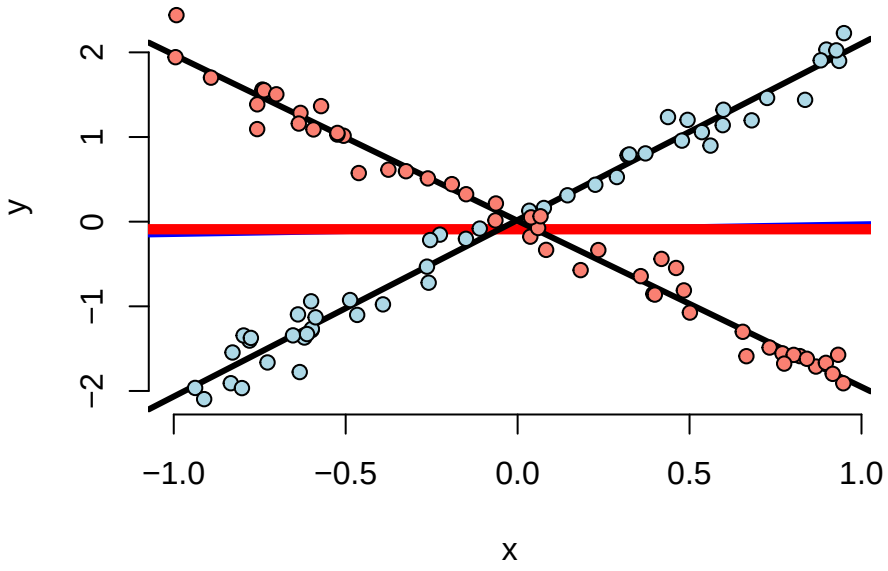
Simulation 5 - Binary Interaction

- the following code simulates the linear model,

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 t_i + \beta_3 x_i t_i + \epsilon_i$$

  where $t = \{0, 1\}$ $\rightarrow$ binary variable
- in this case, β_0 and β_2 are set to 0

```
# simulate data
n <- 100; t <- rep(c(0, 1), c(n/2, n/2)); x <- c(runif(n/2, -1, 1), runif(n/2, -1, 1));
# define parameters/coefficients
beta0 <- 0; beta1 <- 2; beta2 <- 0; beta3 <- -4; sigma <- .2
# generate outcome using linear model
y <- beta0 + x * beta1 + t * beta2 + t * x * beta3 + rnorm(n, sd = sigma)
# set up axes
plot(x, y, type = "n", frame = FALSE)
# plot linear fit of y vs x
abline(lm(y ~ x), lwd = 2, col = "blue")
# plot means of the two groups (t = 0 vs t = 1)
abline(h = mean(y[1 : (n/2)]), lwd = 3, col = "red")
abline(h = mean(y[(n/2 + 1) : n]), lwd = 3, col = "red")
# plot linear fit of y vs x and t and interaction term
fit <- lm(y ~ x + t + I(x * t))
# plot the two lines corresponding to (t = 0 vs t = 1)
abline(coef(fit)[1], coef(fit)[2], lwd = 3)
abline(coef(fit)[1] + coef(fit)[3], coef(fit)[2] + coef(fit)[4], lwd = 3)
# add in the actual data points
points(x[1 : (n/2)], y[1 : (n/2)], pch = 21, col = "black", bg = "lightblue", cex = 1)
points(x[(n/2 + 1) : n], y[(n/2 + 1) : n], pch = 21, col = "black", bg = "salmon", cex = 1)
```



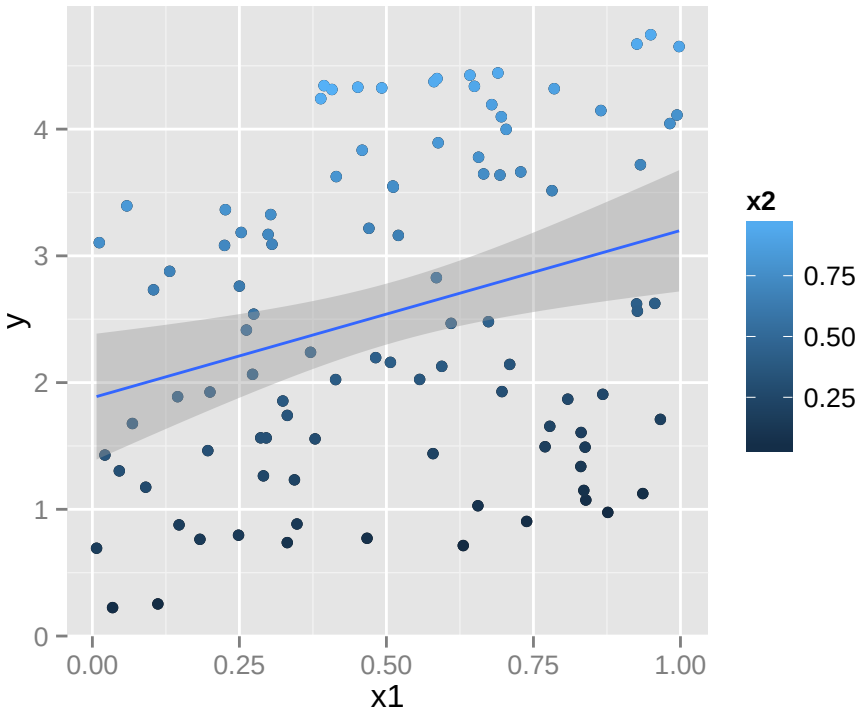
- in the above graph, the elements are as follows:
  - *red* lines = means for two groups ($t = 0$ vs $t = 1$) $\rightarrow$ two lines representing `lm(y ~ t)`
  - *black* lines = regression lines for two groups ($t = 0$ vs $t = 1$) $\rightarrow$ two lines representing `lm(y ~ t + x + t*x)`
  - *blue* line = overall regression of y vs x $\rightarrow$ line representing `lm(y ~ x)`
    - this is completely meaningless in this case
- from the graph, we can see that
  - ***treatment effect*** does not apply since it varies with x
    - impact of treatment/group variable **reverses itself** depending on x
  - ***adjustment effect*** is effectively zero as the means of the two groups are very similar
  - both intercept and slope of the two lines depend on the group variable t
  - group variable and x are ***unrelated***
  - lots of information for comparing group effects holding x fixed

Simulation 6 - Continuous Adjustment

- the following code simulates the linear model,

$$Y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \epsilon_i$$

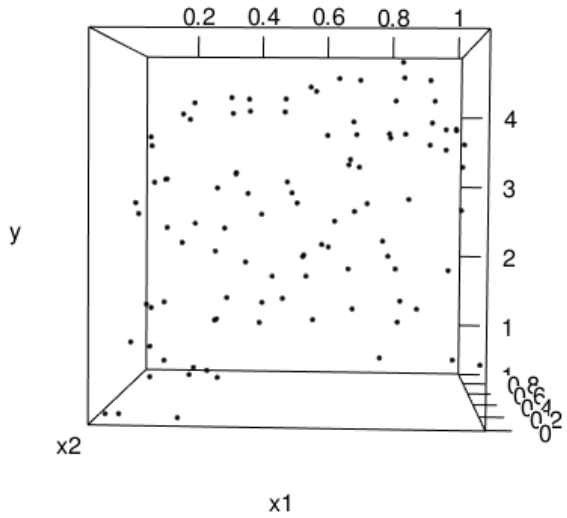
```
# simulate data
p <- 1; n <- 100; x2 <- runif(n); x1 <- p * runif(n) - (1 - p) * x2
# define parameters/coefficients
beta0 <- 0; beta1 <- 1; beta2 <- 4 ; sigma <- .01
# generate outcome using linear model
y <- beta0 + x1 * beta1 + beta2 * x2 + rnorm(n, sd = sigma)
# plot y vs x1 and x2
qplot(x1, y) + geom_point(aes(colour=x2)) + geom_smooth(method = lm)
```



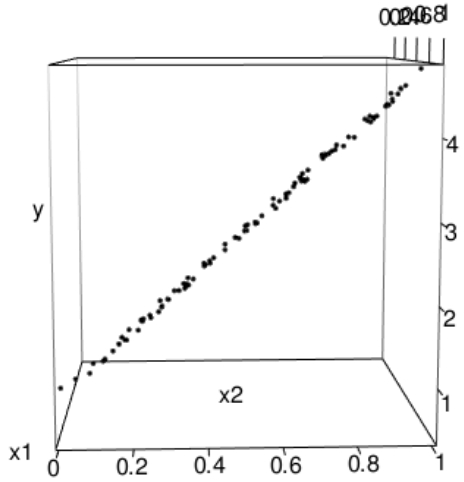
- in the above graph, we plotted y vs x_1 with x_2 denoted as the gradient of color from blue to white
- to investigate the bivariate relationship more clearly, we can use the following command from the `rgl` package to generate ***3D plots***

```
rgl::plot3d(x1, x2, y)
```

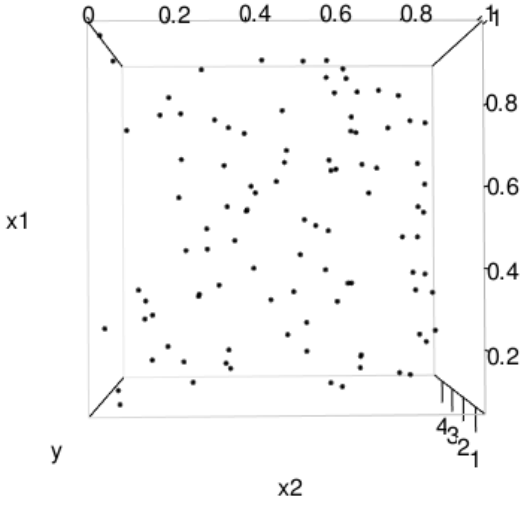
- y ***vs*** x_1



- y ***vs*** x_2

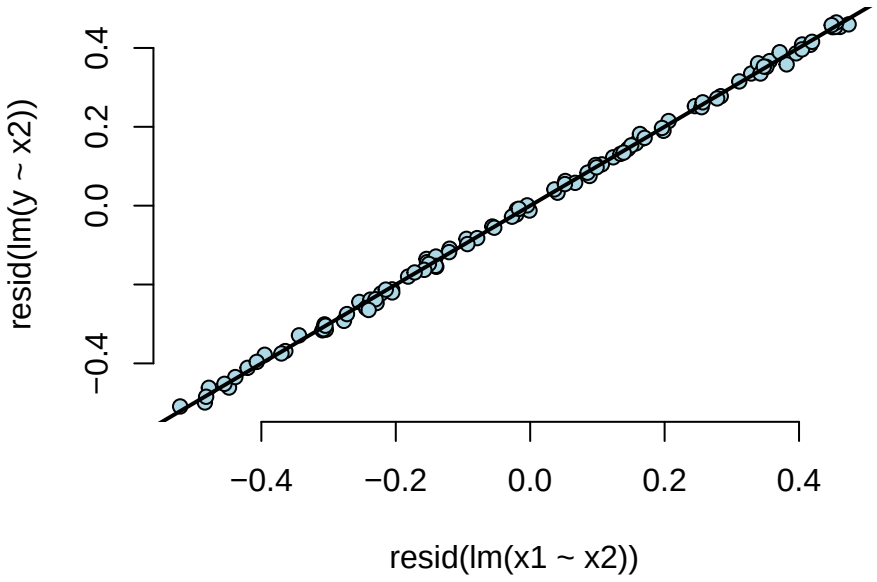


- x_1 ***vs*** x_2



- residual plot with effect of x_2 removed from both y and x_1

```
# plot the residuals for y and x1 with x2 removed
plot(resid(lm(x1 ~ x2)), resid(lm(y ~ x2)), frame = FALSE,
    col = "black", bg = "lightblue", pch = 21, cex = 1)
# add linear fit line
abline(lm(I(resid(lm(y ~ x2))) ~ I(resid(lm(x1 ~ x2)))), lwd = 2)
```



- from the generated plots above, we can see that
  - x_1 is ***unrelated*** to x_2
  - x_2 ***strongly correlated*** to y
  - relationship between x_1 and y (loosely correlated $\rightarrow R^2 = 0.09$) ***largely unchanged*** by when x_2 is considered
    - x_2 captures the vast majority of variation in data
    - there exists almost no residual variability after removing x_2

Summary and Considerations

- modeling multivariate relationships is ***difficult***
  - modeling for prediction is fairly straight forward
  - interpreting the regression lines is much harder, as adjusting for variables can have profound effect on variables of interest
- it is often recommended to explore with simulations to see how inclusion or exclusion of another variable affects the relationship of variable of interest and the outcome
- variable selection simply affects associations between outcome and predictors, using the model to formulate causal relationship are even more difficult (entire field dedicated to this $\rightarrow$ ***causal inference***)

Residuals and Diagnostics

- recall that the **generalized linear model** is defined as

$$Y_i = \sum_{k=1}^{p} X_{ik}\beta_j + \epsilon_i$$

  where $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$
- the **predicted outcome**, $\hat{Y}_i$, is defined as

$$\hat{Y}_i = \sum_{k=1}^{p} X_{ik}\hat{\beta}_j$$

- the **residuals**, e_i, is defined as

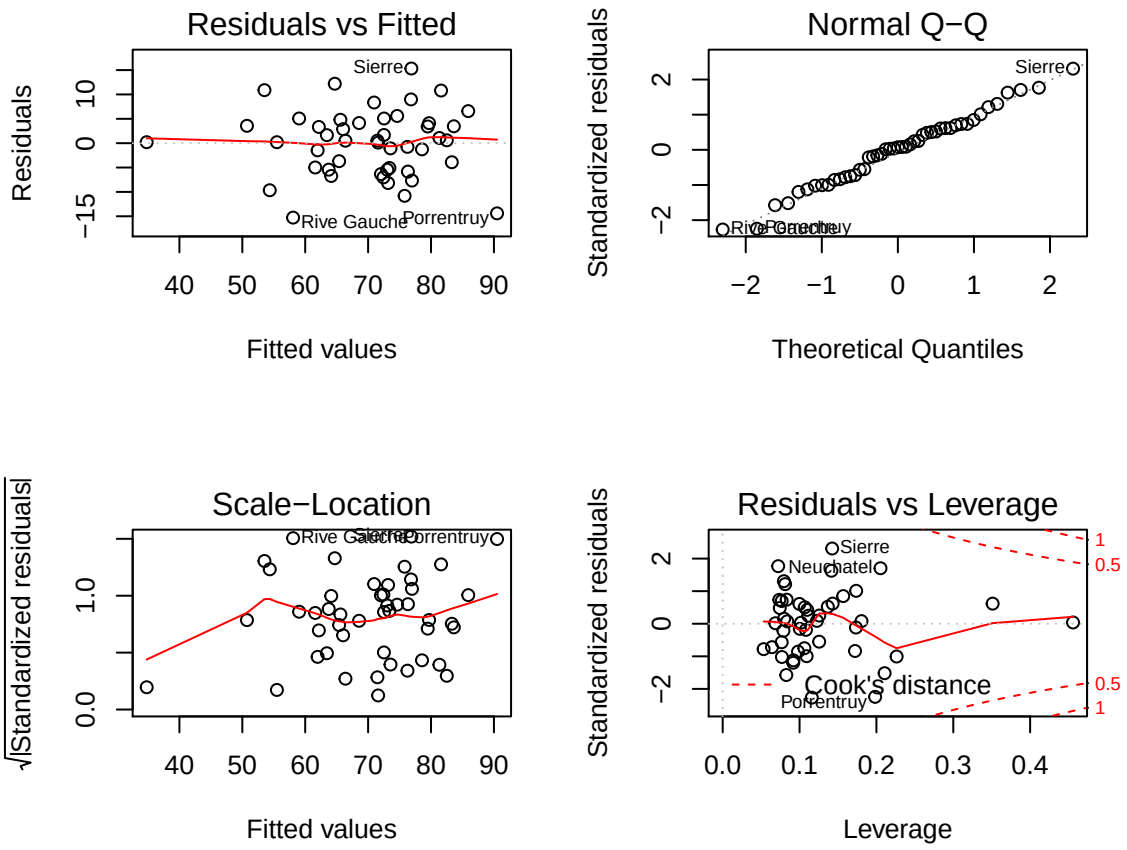
$$e_i = Y_i - \hat{Y}_i = Y_i - \sum_{k=1}^{p} X_{ik}\hat{\beta}_j$$

- the unbiased estimate for **residual variation** is defined as

$$\hat{\sigma}^2_{resid} = \frac{\sum_{i=1}^{n} e_i^2}{n-p}$$

  where the denominator is $n - p$ so that $E[\hat{\sigma}^2] = \sigma^2$
- to evaluate the fit and residuals of a linear model generated by R (i.e. `fit <- lm(y~x)`, we can use the `plot(fit)` to produce a series of ***4 diagnostic plots***
  - ***Residuals vs Fitted*** = plots ordinary residuals vs fitted values → used to detect patterns for missing variables, heteroskedasticity, etc
  - ***Scale-Location*** = plots standardized residuals vs fitted values → similar residual plot, used to detect patterns in residuals
  - ***Normal Q-Q*** = plots theoretical quantiles for standard normal vs actual quantiles of standardized residuals → used to evaluate normality of the errors
  - ***Residuals vs Leverage*** = plots cooks distances comparison of fit at that point vs potential for influence of that point → used to detect any points that have substantial influence on the regression model
- ***example***

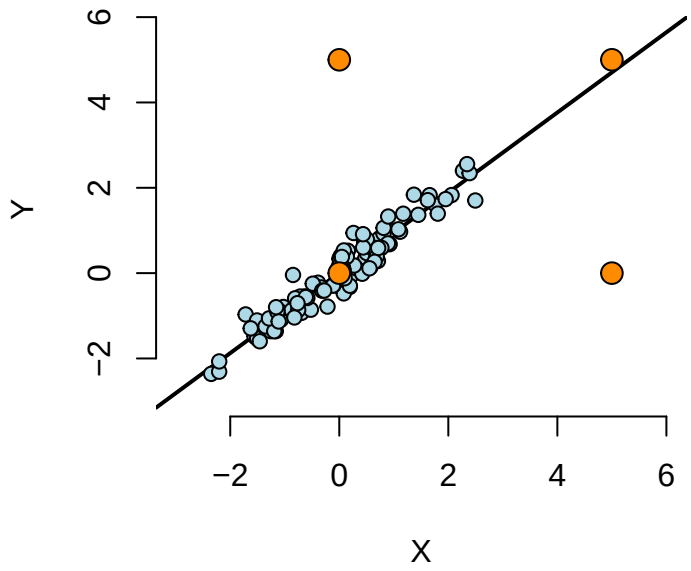
```
# load swiss data and
data(swiss)
# run linear regression on Fertility vs all other predictors
fit <- lm(Fertility ~ . , data = swiss)
# generate diagnostic plots in 2 x 2 panels
par(mfrow = c(2, 2)); plot(fit)
```



Outliers and Influential Points

- **outlier** = an observation that is distant from the other observations of the data set
  - can be results of spurious or real processes
  - can conform to the regression relationship (i.e being marginally outlying in X or Y, but not outlying given the regression relationship)

```
# generate data
n <- 100; x <- rnorm(n); y <- x + rnorm(n, sd = .3)
# set up axes
plot(c(-3, 6), c(-3, 6), type = "n", frame = FALSE, xlab = "X", ylab = "Y")
# plot regression line for y vs x
abline(lm(y ~ x), lwd = 2)
# plot actual (x, y) pairs
points(x, y, cex = 1, bg = "lightblue", col = "black", pch = 21)
# plot 4 points of interest
points(0, 0, cex = 1.5, bg = "darkorange", col = "black", pch = 21)
points(0, 5, cex = 1.5, bg = "darkorange", col = "black", pch = 21)
points(5, 5, cex = 1.5, bg = "darkorange", col = "black", pch = 21)
points(5, 0, cex = 1.5, bg = "darkorange", col = "black", pch = 21)
```



- different outliers can have ***varying*** degrees of ***influence***
  - influence = actual effect on model fit
  - leverage = potential for influence
- in the plot above, we examine 4 different points of interest (in orange)
  - **lower left**: low leverage, low influence, ***not*** an outlier in any sense
  - **upper left**: low leverage, low influence, classified as outlier because it does not conform to the regression relationship
    - ***Note****: this point, though far away from the rest,* ***does not*** *impact the regression line since it lies in the middle of the data range because the regression line must always pass through the mean/center of observations*
  - **upper right**: high leverage, low influence, classified as outlier because it lies far away from the rest of the data
    - ***Note****: this point has low influence on regression line because it conforms to the overall regression relationship*
  - **lower right**: high leverage, high influence, classified as outlier because it lies far away from the rest of the data AND it does not conform to the regression relationship

Influence Measures

- there exists many pre-written functions to measure influence of observations already in the `stats` package in R
  - ***Note****: typing in* `?influence.measures` *in R will display the detailed documentation on all available functions to measure influence*
  - ***Note****: the* `model` *argument referenced in the following functions is simply the linear fit model generated by the* `lm` *function (i.e.* `model <- lm(y~x)`*)*
  - `rstandard(model)` = standardized residuals → residuals divided by their standard deviations
  - `rstudent(model)` = standardized residuals → residuals divided by their standard deviations, where the i^{th} data point was deleted in the calculation of the standard deviation for the residual to follow a t distribution
  - `hatvalues(model)` = measures of leverage
  - `dffits(model)` = change in the predicted response when the i^{th} point is deleted in fitting the model

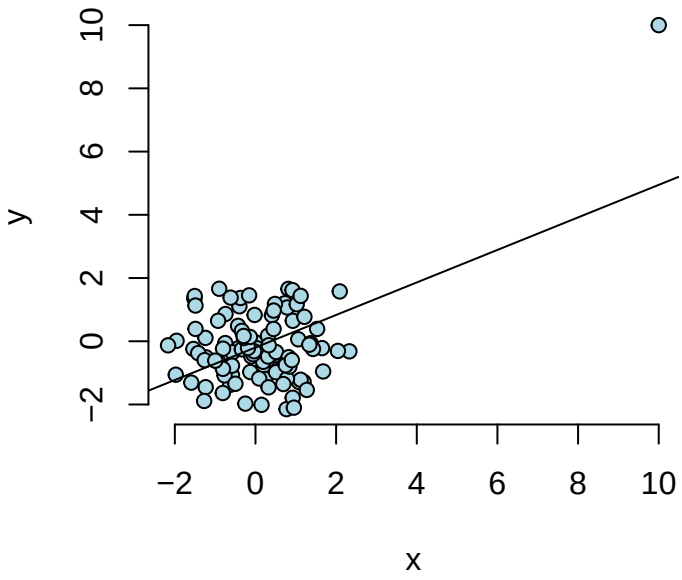
    - effectively measures influence of a point on prediction
  - `dfbetas(model)` = change in individual coefficients when the i^{th} point is deleted in fitting the model
    - effectively measures influence of the individual coefficients
  - `cooks(model).distance` = overall change in coefficients when the i^{th} point is deleted
  - `resid(model)` = returns ordinary residuals
  - `resid(model)/(1-hatvalues(model))` = returns *PRESS* residuals (i.e. the leave one out cross validation residuals)
    - PRESS residuals measure the differences in the response and the predicted response at data point i, where it was not included in the model fitting
    - effectively measures the prediction error based on model constructed with every other point but the one of interest

Using Influence Measures

- the purpose of these functions are to probe the given data in different ways to ***diagnose*** different problems
  - patterns in **residual plots** (most important tool) → generally indicate some poor aspect of model fit
    - heteroskedasticity → non-constant variance
    - missing model terms
    - temporal patterns → residuals versus collection order/index exhibit pattern
  - **residual Q-Q plots** plots theoretical quantile vs actual quantiles of residuals
    - investigates whether the errors follow the standard normal distribution
  - **leverage measures** (hat values) measures the potential to influence the regression model
    - only depend on x or predictor variables
    - can be useful for diagnosing data entry errors
  - **influence measures** (i.e. dfbetas) measures actual influence of points on the regression model
    - evaluates how deleting or including this point impact a particular aspect of the model
- it is important to to understand these functions/tools and use carefully in the ***appropriate context***
- not all measures have ***meaningful absolute scales***, so it may be useful to apply these measures to different values in the same data set but ***almost never*** to different datasets

Example - Outlier Causing Linear Relationship

```
# generate random data and point (10, 10)
x <- c(10, rnorm(n)); y <- c(10, c(rnorm(n)))
# plot y vs x
plot(x, y, frame = FALSE, cex = 1, pch = 21, bg = "lightblue", col = "black")
# perform linear regression
fit <- lm(y ~ x)
# add regression line to plot
abline(fit)
```



- data generated
  - 100 points are randomly generated from the standard normal distribution
  - point (10, 10) added to the data set
- there is no regression relationship between X and Y as the points are simply random noise
- the regression relationship was able to be constructed precisely because of the presence of the point (10, 10)
  - $R^2 = 0.26044$
  - a single point has created a strong regression relationship where there shouldn't be one
    - point (10, 10) has high leverage and high influence
  - we can use diagnostics to detect this kind of behavior
- `dfbetas(fit)` = difference in coefficients for including vs excluding each data point
  - the function will return a `n x m` dataframe, where n = number of values in the original dataset, and m = number of coefficients
  - for this example, the coefficients are β_0 (intercept), and β_1 (slope), and we are interested in the slope

```
# calculate the dfbetas for the slope the first 10 points
round(dfbetas(fit)[1 : 10, 2], 3)
```

```
##      1      2      3      4      5      6      7      8      9     10
##  7.375 -0.078 -0.090 -0.098  0.003 -0.118 -0.001  0.025 -0.133 -0.031
```

- as we can see from above, the slope coefficient would ***change dramatically*** if the first point (10, 10) is left out
- `hatvalues(fit)` = measures the potential for influence for each point

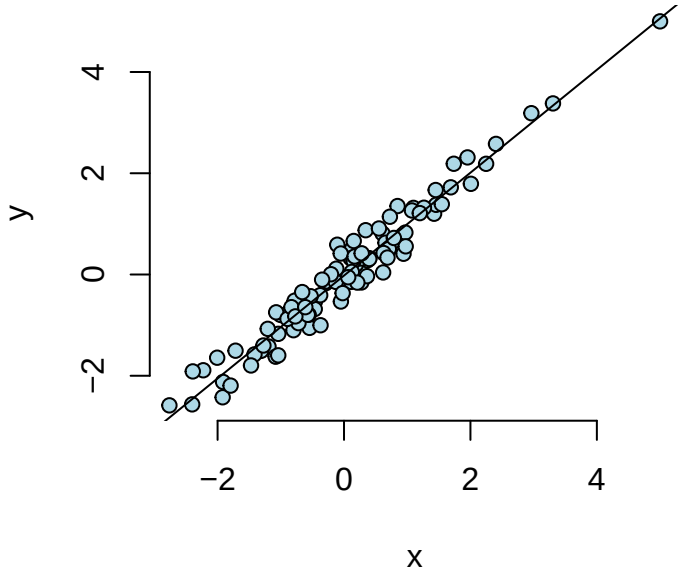
```
# calculate the hat values for the first 10 points
round(hatvalues(fit)[1 : 10], 3)
```

```
##     1     2     3     4     5     6     7     8     9    10
## 0.508 0.020 0.012 0.013 0.010 0.013 0.010 0.013 0.023 0.012
```

- again, as we can see from above, the ***potential for influence is very large*** for the first point (10, 10)

Example - Real Linear Relationship

```
# generate data
x <- rnorm(n); y <- x + rnorm(n, sd = .3)
# add an outlier that fits the relationship
x <- c(5, x); y <- c(5, y)
# plot the (x, y) pairs
plot(x, y, frame = FALSE, cex = 1, pch = 21, bg = "lightblue", col = "black")
# perform the linear regression
fit2 <- lm(y ~ x)
# add the regression line to the plot
abline(fit2)
```



- data generated
  - 100 directly correlated points are generated
  - point (5, 5) added to the data set
- there is a linear relationship between X and Y
  - $R^2 = 0.9517188$
  - point (5, 5) has high leverage and low influence

```
# calculate the dfbetas for the slope the first 10 points
round(dfbetas(fit2)[1 : 10, 2], 3)
```

```
##      1      2      3      4      5      6      7      8      9     10
## -0.093 -0.029 -0.012 -0.007  0.080  0.114  0.063 -0.035 -0.006  0.026
```

```r
# calculate the hat values for the first 10 points
round(hatvalues(fit2)[1 : 10], 3)
```

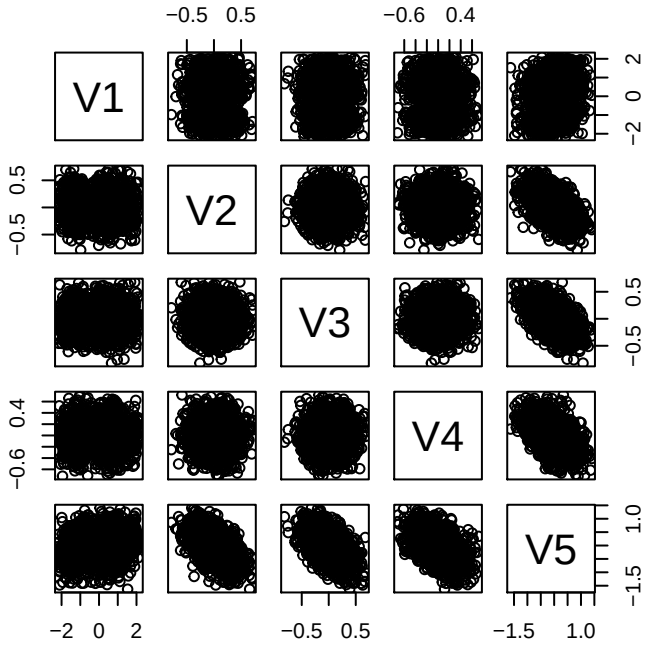
```
##     1     2     3     4     5     6     7     8     9    10
## 0.164 0.013 0.010 0.011 0.012 0.014 0.014 0.014 0.011 0.012
```

- as we can see from above, the point (5, 5) no longer has a large `dfbetas` value (indication of low influence) but still has a substantial `hatvalue` (indication of high leverage)
  - this is in line with out expectations

Example - Stefanski TAS 2007

- taken from Leonard A. Stefanski's paper Residual (Sur)Realism
- data set can be found here
- the data itself exhibit no sign of correlation between the variables

```r
# read data
data <- read.table('http://www4.stat.ncsu.edu/~stefanski/NSF_Supported/Hidden_Images/orly_owl_files/orly
    header = FALSE)
# construct pairwise plot
pairs(data)
```



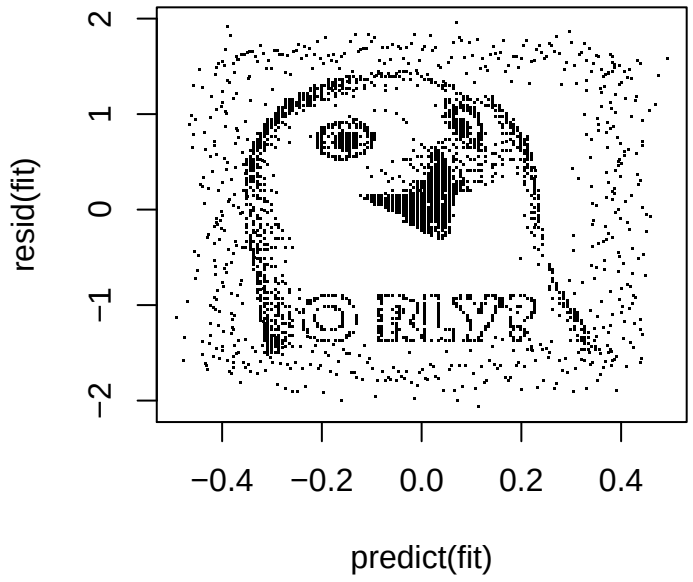

```r
# perform regression on V1 with all other predictors (omitting the intercept)
fit <- lm(V1 ~ . - 1, data = data)
# print the coefficient for linear model
summary(fit)$coef
```

```
##     Estimate Std. Error   t value     Pr(>|t|)
## V2 0.9856157 0.12798121  7.701253 1.989126e-14
```

```
## V3 0.9714707 0.12663829  7.671225 2.500259e-14
## V4 0.8606368 0.11958267  7.197003 8.301184e-13
## V5 0.9266981 0.08328434 11.126919 4.778110e-28
```

- as we can see from above, the p-values for the coefficients indicate that they are significant
- if we take a look at the residual plot, an interesting pattern appears

```
# plot the residuals vs fitted values
plot(predict(fit), resid(fit), pch = '.')
```



Model Selection

> "A model is a lense through which to look at your data" – **Scott Zeger**
>
> "There's no such thing as a correct model" – **Brian Caffo**

- **goal for modeling** = find ***parsimonious, interpretable representations*** of data that enhance our understanding
- whichever model connects data to a true, parsimonious statement would be ***best*** model
- like nearly all aspects of statistics, good modeling decisions are context dependent
- good model for prediction $\neq$ model to establish causal effects
  - prediction model may tolerate more variables and variability

Rumsfeldian Triplet

> "There are known knowns. These are things we know that we know. There are known unknowns. That is to say, there are things that we know we don't know. But there are also unknown unknowns. There are things we don't know we don't know." – **Donald Rumsfeld**

- **known knowns** = regressors that we know and have, which will be evaluated to be included in the model
- **known unknowns** = regressors that we but don't have but would like to include in the model
  - didn't or couldn't collect the data
- **unknown unknowns** = regressors that we don't know about that we should have included in the model

General Rules

- **omitting variables** $\rightarrow$ generally results in ***increased bias*** in coefficients of interest
  - exceptions are when the omitted variables are uncorrelated with the regressors (variables of interest/included in model)
    - ***Note****: this is why randomize treatments/trials/experiments are the norm; it's the best strategy to balance confounders, or maximizing the probability that the treatment variable is uncorrelated with variables not in the model*
    - often times, due to experiment conditions or data availability, we cannot randomize
    - however, if there are too many unobserved confounding variables, even randomization won't help
- **including irrelevant/unnecessary variables** $\rightarrow$ generally ***increases standard errors*** (estimated standard deviation) of the coefficients
  - ***Note****: including* ***any*** *new variables increases true standard errors of other regressors, so it is not wise to idly add variables into model*
- whenever highly correlated variables are included in the same model $\rightarrow$ the standard error and therefore the***variance*** of the model ***increases*** $\rightarrow$ this is known as **variance inflation**
  - actual increase in standard error of coefficients for adding a regressor = estimated by the ratio of the estimated standard errors minus 1, or in other words

    $$\Delta_{\sigma \mid adding\ x_2} = \frac{\hat{\sigma}_{y \sim x_1 + x_2}}{\hat{\sigma}_{y \sim x_1}} - 1$$

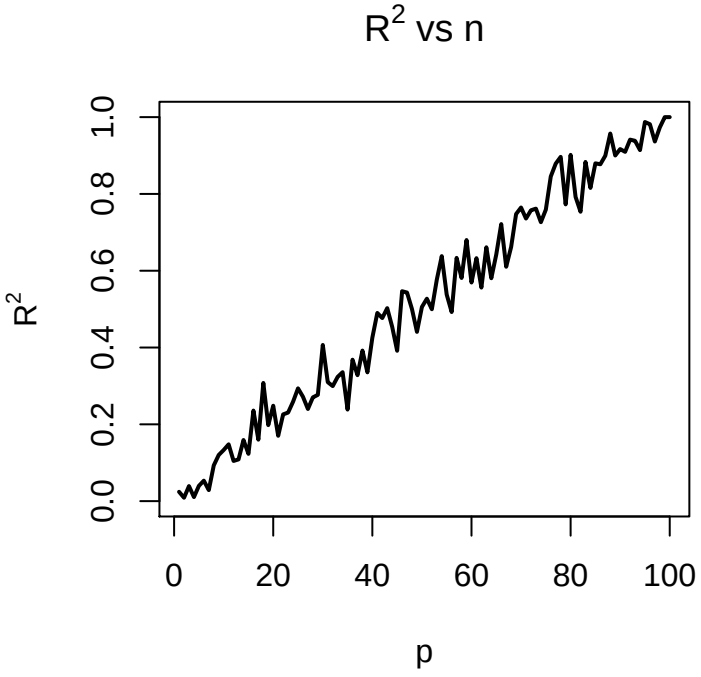
    for all coefficients for the regression model

    - ***example***: if standard error of the β_1 of `y~x1+x2` = 1.5 and standard error for the β_1 of `y~x1` = 0.5, then the actual increase in standard error of the β_1 = 1.5/0.5 - 1 = 200%
  - ***Note****: when the regressors added are orthogonal (statistically independent) to the regressor of interest, then there is no variance inflation*
    - **variance inflation factor** (VIF) = the increase in the variance for the i_{th} regressor compared to the ideal setting where it is orthogonal to the other regressors
    - $\sqrt{VIF}$ = increase in standard error
  - ***Note****: variance inflation is only part of the picture in that sometimes we will* ***include variables*** *even though they dramatically inflate the variation because it is an* ***empirical part of the relationship*** *we are attempting to model*
- as the number of *non-redundant* variables increases or approaches n, the model ***approaches a perfect fit*** for the data
  - R^2 monotonically increases as more regressors are included
  - Sum of Squared Errors (SSE) monotonically decreases as more regressors are included

Example - R^2 **v** n

- for the simulation below, no actual regression relationship exist as the data generated are simply standard normal noise
- it is clear, however, that as p, the number of regressors included in the model, approaches n, the R^2 value approaches 1.0, which signifies perfect fit

```
# set number of measurements
n <- 100
# set up the axes of the plot
plot(c(1, n), 0 : 1, type = "n", xlab = "p", ylab = expression(R^2),
    main = expression(paste(R^2, " vs n")))
# for each value of p from 1 to n
r <- sapply(1 : n, function(p){
    # create outcome and p regressors
    y <- rnorm(n); x <- matrix(rnorm(n * p), n, p)
    # calculate the R^2
    summary(lm(y ~ x))$r.squared
})
# plot the R^2 values and connect them with a line
lines(1 : n, r, lwd = 2)
```



Adjusted R^2

- recall that R^2 is defined as the percent of total variability that is explained by the regression model, or

$$R^2 = \frac{\text{regression variation}}{\text{total variation}} = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} = 1 - \frac{Var(e_i)}{Var(Y_i)}$$

- Estimating R^2 with the above definition is ***acceptable*** when there is a *single* variable, but it becomes less and ***less helpful*** as the *number of variables increases*
  - as we have shown previously, R^2 always increases as more variables are introduced and is thus ***biased***
- **adjusted** R^2 is a better estimate of variability explained by the model and is defined as

$$R^2_{adj} = 1 - \frac{Var(e_i)}{Var(Y_i)} \times \frac{n-1}{n-k-1}$$

  where n = number of observations, and k = number of predictors in the model
  - since k is always greater than zero, the adjusted R^2 is ***always smaller*** than the unadjusted R^2
  - adjusted R^2 also penalizes adding large numbers of regressors, which would have inflated the unadjusted R^2

Example - Unrelated Regressors

- in the simulation below, outcome y is only related to x_1
  - x_2 and x_3 are random noise
- we will run 1000 simulations of 3 linear regression models, and calculate the ***standard error of the slope***
  - y vs x_1

  - y vs $x_1 + x_2$
  - y vs $x_1 + x_2 + x_3$

```
# simulate data
n <- 100; nosim <- 1000
# generate 3 random noise, unrelated variables
x1 <- rnorm(n); x2 <- rnorm(n); x3 <- rnorm(n);
# calculate beta1s of three different regression
betas <- sapply(1 : nosim, function(i){
    # generate outcome as only related to x1
    y <- x1 + rnorm(n, sd = .3)
    # store beta1 of linear regression on y vs x1
    c(coef(lm(y ~ x1))[2],
        # store beta1 of linear regression on y vs x1 and x2
        coef(lm(y ~ x1 + x2))[2],
        # store beta1 of linear regression on y vs x1 x2 and x3
        coef(lm(y ~ x1 + x2 + x3))[2])
})
# calculate the standard error of the beta1s for the three regressions
beta1.se <- round(apply(betas, 1, sd), 5)
# print results
rbind("y ~ x1" = c("beta1SE" = beta1.se[1]),
      "y ~ x1 + x2" = beta1.se[2],
      "y ~ x1 + x2 + x3" = beta1.se[3])
```

```
##                   beta1SE.x1
## y ~ x1               0.03145
## y ~ x1 + x2          0.03145
## y ~ x1 + x2 + x3     0.03180
```

- as we can see from the above result, if we include unrelated regressors x_2 and x_3, the ***standard error increases***

Example - Highly Correlated Regressors / Variance Inflation

- in the simulation below, outcome y is related to x_1
  - x_2 and x_3 are highly correlated with x_1
  - x_3 is more correlated with x_1 than x_2
- we will run 1000 simulations of 3 linear regression models, and calculate the ***standard error of*** β_1, the coefficient of x_1
  - y vs x_1
  - y vs $x_1 + x_2$
  - y vs $x_1 + x_2 + x_3$

```
# generate number of measurements and trials
n <- 100; nosim <- 1000
# generate random variables that are correlated with each other
x1 <- rnorm(n); x2 <- x1/sqrt(2) + rnorm(n) /sqrt(2)
x3 <- x1 * 0.95 + rnorm(n) * sqrt(1 - 0.95^2);
# calculate the beta1s for 1000 trials
```

```
betas <- sapply(1 : nosim, function(i){
    # generate outcome as only related to x1
    y <- x1 + rnorm(n, sd = .3)
    # store beta1 of linear regression on y vs x1
    c(coef(lm(y ~ x1))[2],
        # store beta1 of linear regression on y vs x1 and x2
        coef(lm(y ~ x1 + x2))[2],
        # store beta1 of linear regression on y vs x1 x2 and x3
        coef(lm(y ~ x1 + x2 + x3))[2])
})
# calculate the standard error of the beta1 for the three regressions
beta1.se <- round(apply(betas, 1, sd), 5)
# print results
rbind("y ~ x1" = c("beta1SE" = beta1.se[1]),
      "y ~ x1 + x2" = beta1.se[2],
      "y ~ x1 + x2 + x3" = beta1.se[3])
```

```
##                  beta1SE.x1
## y ~ x1              0.03210
## y ~ x1 + x2         0.04163
## y ~ x1 + x2 + x3    0.10384
```

- as we can see from above, adding highly correlated regressors ***drastically increases*** the standard errors of the coefficients
- to estimate the actual change in variance, we can use the ratio of estimated variances for the β_1 coefficient for the different models
  - `summary(fit)$cov.unscaled` = returns p x p covariance matrix for p coefficients, with the diagonal values as the true variances of coefficients
    - `summary(fit)$cov.unscaled[2,2]` = true variance for the β_1

```
# generate outcome that is correlated with x1
y <- x1 + rnorm(n, sd = .3)
# store the variance of beta1 for the 1st model
a <- summary(lm(y ~ x1))$cov.unscaled[2,2]
# calculate the ratio of variances of beta1 for 2nd and 3rd models with respect to 1st model
c(summary(lm(y ~ x1 + x2))$cov.unscaled[2,2],
    summary(lm(y~ x1 + x2 + x3))$cov.unscaled[2,2]) / a - 1
```

```
## [1] 0.6820404 9.0116426
```

```
# alternatively, the change in variance can be estimated by calculating ratio of trials variance
temp <- apply(betas, 1, var); temp[2 : 3] / temp[1] - 1
```

```
##        x1        x1
## 0.6817101 9.4636880
```

- as we can see from the above results
  - adding x_2 increases the variance by approximately 1 fold
  - adding x_2 and x_3 increases the variance by approximately 9 folds
- the estimated values from the 1000 trials are ***different but close*** to the true increases, and they will approach the true values as the number of trials increases

Example: Variance Inflation Factors

- we will use the `swiss` data set for this example, and compare the following models
  - Fertility vs Agriculture
  - Fertility vs Agriculture + Examination
  - Fertility vs Agriculture + Examination + Education

```
# load swiss data
data(swiss)
# run linear regression for Fertility vs Agriculture
fit <- lm(Fertility ~ Agriculture, data = swiss)
# variance for coefficient of Agriculture
a <- summary(fit)$cov.unscaled[2,2]
# run linear regression for Fertility vs Agriculture + Examination
fit2 <- update(fit, Fertility ~ Agriculture + Examination)
# run linear regression for Fertility vs Agriculture + Examination + Education
fit3 <- update(fit, Fertility ~ Agriculture + Examination + Education)
# calculate ratios of variances for Agriculture coef for fit2 and fit3 w.r.t fit1
c(summary(fit2)$cov.unscaled[2,2], summary(fit3)$cov.unscaled[2,2]) / a - 1
```

```
## [1] 0.8915757 1.0891588
```

- as we can see from above
  - adding Examination variable to the model increases the variance by 89%
  - adding Examination and Education variables to the model increases the variance by 109%
- we can also calculate the **variance inflation factors** for all the predictors and see how variance will change by adding each predictor (assuming all predictor are orthogonal/independent of each other)
  - [`car` library] `vit(fit)` = returns the variance inflation factors for the predictors of the given linear model

```
# load car library
library(car)
# run linear regression on Fertility vs all other predictors
fit <- lm(Fertility ~ . , data = swiss)
# calculate the variance inflation factors
vif(fit)
```

```
##      Agriculture      Examination        Education         Catholic
##         2.284129         3.675420         2.774943         1.937160
## Infant.Mortality
##         1.107542
```

```
# calculate the standard error inflation factors
sqrt(vif(fit))
```

```
##      Agriculture      Examination        Education         Catholic
##         1.511334         1.917138         1.665816         1.391819
## Infant.Mortality
##         1.052398
```

- as we can see from the above results, Education and Examination both have relatively higher inflation factors, which makes sense as the two variables are likely to be correlated with each other

Residual Variance Estimates

- assuming that the model is linear with additive iid errors (with finite variance), we can mathematically describe the impact of omitting necessary variables or including unnecessary ones
  - **underfitting** the model → variance estimate is ***biased*** → $E[\hat{\sigma}^2] \neq \sigma^2$
  - **correctly fitting** or **overfitting** the model → variance estimate is ***unbiased*** → $E[\hat{\sigma}^2] = \sigma^2$
    - however, if unnecessary variables are included, the variance estimate is ***larger*** than that of the correctly fitted variables → $Var(\hat{\sigma}_{overfitted}) \geq Var(\hat{\sigma}_{correct})$
    - in other words, adding unnecessary variables increases the variability of estimate for the true model

Covariate Model Selection

- automated covariate/predictor selection is difficult
  - the space of models explodes quickly with interactions and polynomial terms
  - ***Note**: in the **Practical Machine Learning** class, many modern methods for traversing large model spaces for the purposes of prediction will be covered*
- principal components analysis (PCA) or factor analytic models on covariates are often useful for reducing complex covariate spaces
  - find linear combinations of variables that captures the most variation
- good experiment design can often eliminate the need for complex model searches during analyses
  - randomization, stratification can help simply the end models
  - unfortunately, control over the design is ***often limited***
- it is also viable to manually explore the covariate space based on understanding of the data
  - use covariate adjustment and multiple models to probe that effect of adding a particular predictor on the model
  - ***Note**: this isn't a terribly systematic or efficient approach, but it tends to teach you a lot about the the data through the process*
- if the models of interest are nested (i.e. one model is a special case of another with one or more coefficients set to zero) and without lots of parameters differentiating them, it's fairly possible to use nested likelihood ratio tests (ANOVA) to help find the best model
  - **Analysis of Variance** (ANOVA) works well when adding one or two regressors at a time
    - `anova(fit1, fit2, fit3)` = performs ANOVA or analysis of variance (or deviance) tables for a series of nested linear regressions models
  - ***Note**: it is extremely important to get the order of the models correct to ensure the results are sensible*
  - an example can be found here
- another alternative to search through different models is the **step-wise search** algorithm that repeatedly adds/removes regressors one at a time to find the best model with the least Akaike Information Criterion (AIC)
  - `step(lm, k=df)` = performs step wise regression on a given linear model to find and return best linear model
    - `k=log(n)` = specifying the value of `k` as the log of the number of observation will force the step-wise regression model to use Bayesian Information Criterion (BIC) instead of the AIC
    - ***Note**: both BIC and AIC penalizes adding parameters to the regression model with an additional penalty term; the penalty is **larger** for BIC than AIC*
  - `MASS::stepAIC(lm, k = df)` = more versatile, rigorous implementation of the step wise regression
  - an example can be found here

Example: ANOVA

- we will use the `swiss` data set for this example, and compare the following nested models
  - Fertility vs Agriculture
  - Fertility vs Agriculture + Examination + Education
  - Fertility vs Agriculture + Examination + Education + Catholic + Infant.Mortality

```
# three different regressions that are nested
fit1 <- lm(Fertility ~ Agriculture, data = swiss)
fit3 <- update(fit, Fertility ~ Agriculture + Examination + Education)
fit5 <- update(fit, Fertility ~ Agriculture + Examination + Education + Catholic + Infant.Mortality)
# perform ANOVA
anova(fit1, fit3, fit5)
```

```
## Analysis of Variance Table
##
## Model 1: Fertility ~ Agriculture
## Model 2: Fertility ~ Agriculture + Examination + Education
## Model 3: Fertility ~ Agriculture + Examination + Education + Catholic +
##     Infant.Mortality
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     45 6283.1
## 2     43 3180.9  2    3102.2 30.211 8.638e-09 ***
## 3     41 2105.0  2    1075.9 10.477 0.0002111 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- the ANOVA function returns a formatted table with the follow information
  - `Res.Df` = residual degrees of freedom for the models
  - `RSS` = residual sum of squares for the models, measure of fit
  - `Df` = change in degrees of freedom from one model to the next
  - `Sum of Sq` = difference/change in residual sum of squares from one model to the next
  - `F` = F statistic, measures the ratio of two scaled sums of squares reflecting different sources of variability

    $$F = \frac{\frac{RSS_1 - RSS_2}{p_2 - p_1}}{\frac{RSS_2}{n - p_2}}$$

    where p_1 and p_2 = number of parameters in the two models for comparison, and n = number of observations
  - `Pr(>F)` = p-value for the F statistic to indicate whether the change in model is significant or not
- from the above result, we can see that both going from first to second, and second to third models result in significant reductions in RSS and ***better model fits***

Example: Step-wise Model Search

- we will use the `mtcars` data set for this example, and perform step-wise regression/model selection algorithm on the following initial model
  - Miles Per Gallon vs Number of Cylinder + Displacement + Gross Horse Power + Rear Axle Ratio + Weight

```r
# load the mtcars data starting regression model
data(mtcars); fit <- lm(mpg ~ cyl + disp + hp + drat + wt, data = mtcars)
# step-wise search using BIC
step(fit, k = log(nrow(mtcars)))
```

```
## Start:  AIC=73.75
## mpg ~ cyl + disp + hp + drat + wt
##
##        Df Sum of Sq    RSS    AIC
## - drat  1     3.018 170.44 70.854
## - disp  1     6.949 174.38 71.584
## - cyl   1    15.411 182.84 73.100
## <none>              167.43 73.748
## - hp    1    21.066 188.49 74.075
## - wt    1    77.476 244.90 82.453
##
## Step:  AIC=70.85
## mpg ~ cyl + disp + hp + wt
##
##        Df Sum of Sq    RSS    AIC
## - disp  1     6.176 176.62 68.528
## - hp    1    18.048 188.49 70.609
## <none>              170.44 70.854
## - cyl   1    24.546 194.99 71.694
## - wt    1    90.925 261.37 81.069
##
## Step:  AIC=68.53
## mpg ~ cyl + hp + wt
##
##        Df Sum of Sq    RSS    AIC
## - hp    1    14.551 191.17 67.595
## - cyl   1    18.427 195.05 68.237
## <none>              176.62 68.528
## - wt    1   115.354 291.98 81.147
##
## Step:  AIC=67.6
## mpg ~ cyl + wt
##
##        Df Sum of Sq    RSS    AIC
## <none>              191.17 67.595
## - cyl   1     87.15 278.32 76.149
## - wt    1    117.16 308.33 79.426

##
## Call:
## lm(formula = mpg ~ cyl + wt, data = mtcars)
##
## Coefficients:
## (Intercept)          cyl           wt
##      39.686       -1.508       -3.191
```

- as we can see from above, the best model that captures most of the variability in the data is simply `mpg ~ cyl + wt`

General Linear Models Overview

- limitations of linear models:
  - response can be discrete (i.e. 0, 1, etc.) or strictly positive → linear response models don't make much sense
  - if outcome must be positive, Gaussian errors (± errors) don't make sense as negative outcomes are possible
  - transformations on predictors (log + 1) are often hard to interpret
    - modeling the data on the scale that it was collected is most ideal
    - even for interpretable transformations, *natural logarithms* specifically, aren't applicable for negative/zero values
- **general linear models** = introduced in 1972 RSSB paper by Nelder and Wedderburn and has ***3*** parts
  1. exponential family model for response/outcome (i.e. Gaussian, Bernoulli distribution)
  2. systematic component for linear predictor → incorporates the information about the independent variables into the model
     - denoted by $\eta = X\beta$ where X is a matrix of independent variables/predictors and β is the coefficients
  3. link function that connects means of the outcome/distribution to linear predictor
     - the relationship is defined as $\eta = g(\mu)$, or the linear predictor η is a function of the mean of the distribution μ

Simple Linear Model

- *exponential family distribution*: Gaussian distribution, assumed $Y_i \sim N(\mu_i, \sigma^2)$
- *linear predictor*: $\eta_i = \sum_{k=1}^{p} X_{ik}\beta_k$
- *link function* : $g(\mu) = \eta = \mu$
  - for linear models, $g(\mu) = \mu$, so $\eta_i = \mu_i$
- **result**: the same likelihood model (see derivation)as the additive Gaussian error linear model

$$Y_i = \sum_{k=1}^{p} X_{ik}\beta_k + \epsilon_i$$

where $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$

Logistic Regression

- *exponential family distribution*: binomial/Bernoulli distribution, assumed $Y_i \sim Bernoulli(\mu_i)$ where the probability of success is μ_i
  - due to the properties of the binomial/Bernoulli distribution, $E[Y_i] = \mu_i$ where $0 \leq \mu_i \leq 1$
- *linear predictor*: $\eta_i = \sum_{k=1}^{p} X_{ik}\beta_k$
- *link function* : $g(\mu) = \eta = \log\left(\frac{\mu}{1-\mu}\right)$
  - **odds** for success for a binomial/Bernoulli distribution is defined as

$$\text{odds} = \frac{p}{1-p}$$

  - **logit** is defined as

$$\log(\text{odds}) = \log\frac{\mu}{1-\mu}$$

    - ***Note**: the* log *here is the **natural** log*
  - **inverse logit** is defined as
$$\mu_i = \frac{\exp(\eta_i)}{1+\exp(\eta_i)}$$
  - complement of inverse logit is
$$1-\mu_i = \frac{1}{1+\exp(\eta_i)}$$

- **result**: the likelihood model
$$\begin{aligned}
L(\beta) &= \prod_{i=1}^{n} \mu_i^{y_i}(1-\mu_i)^{1-y_i} \\
(plug\ in\ \mu_i\ and\ 1-\mu_i\ from\ above) &= \prod_{i=1}^{n} \left(\frac{\exp(\eta_i)}{1+\exp(\eta_i)}\right)^{y_i} \left(\frac{1}{1+\exp(\eta_i)}\right)^{1-y_i} \\
(multiply\ 2^{nd}\ term\ by\ \frac{exp(\eta_i)}{exp(\eta_i)}) &= \prod_{i=1}^{n} \left(\frac{\exp(\eta_i)}{1+\exp(\eta_i)}\right)^{y_i} \left(\frac{\exp(\eta_i)}{1+\exp(\eta_i)}\right)^{1-y_i} \left(\frac{1}{\exp(\eta_i)}\right)^{1-y_i} \\
(simplify) &= \prod_{i=1}^{n} \left(\frac{\exp(\eta_i)}{1+\exp(\eta_i)}\right) \left(\frac{1}{\exp(\eta_i)}\right)^{1-y_i} \\
(simplify) &= \prod_{i=1}^{n} \left(\frac{\exp(\eta_i)}{1+\exp(\eta_i)}\right) \exp(\eta_i)^{y_i-1} \\
(simplify) &= \prod_{i=1}^{n} \frac{\exp(\eta_i)^{y_i}}{1+\exp(\eta_i)} \\
(change\ form\ of\ numerator) &= \exp\left(\sum_{i=1}^{n} y_i\eta_i\right) \prod_{i=1}^{n} \frac{1}{1+\exp(\eta_i)} \\
(substitute\ \eta_i) \Rightarrow L(\beta) &= \exp\left(\sum_{i=1}^{n} y_i\Big(\sum_{k=1}^{p} X_{ik}\beta_k\Big)\right) \prod_{i=1}^{n} \frac{1}{1+\exp\left(\sum_{k=1}^{p} X_{ik}\beta_k\right)}
\end{aligned}$$
  - maximizing the likelihood $L(\beta)$ (solving for $\frac{\partial L}{\partial \beta} = 0$) would return a set of optimized coefficients β that will fit the data

Poisson Regression

- *exponential family distribution*: Poisson distribution, assumed $Y_i \sim Poisson(\mu_i)$ where $E[Y_i] = \mu_i$
- *linear predictor*: $\eta_i = \sum_{k=1}^{p} X_{ik}\beta_k$
- *link function* : $g(\mu) = \eta = \log(\mu)$
  - ***Note**: the* log *here is the **natural** log*
  - since e^x is the inverse of $\log(x)$, then $\eta_i = \log(\mu_i)$ can be transformed into $\mu_i = e^{\eta_i}$

- **result**: the likelihood model

$$L(\beta) = \prod_{i=1}^{n} (y_i!)^{-1} \mu_i^{y_i} e^{-\mu_i}$$
$$(substitute\ \mu_i = e^{\eta_i}) = \prod_{i=1}^{n} \frac{(e^{\eta_i})^{y_i}}{y_i! e^{e^{\eta_i}}}$$
$$(transform) = \prod_{i=1}^{n} \frac{\exp(\eta_i y_i)}{y_i! \exp(e^{\eta_i})}$$
$$(taking\ \log\ of\ both\ sides)\ \mathcal{L}(\beta) = \sum_{i=1}^{n} \eta_i y_i - \sum_{i=1}^{n} e^{\eta_i} - \sum_{i=1}^{n} log(y_i!)$$
$$(since\ y_i\ is\ given,\ we\ can\ ignore\ \log y_i!)\ \mathcal{L}(\beta) \propto \sum_{i=1}^{n} \eta_i y_i - \sum_{i=1}^{n} e^{\eta_i}$$
$$(substitute\ \eta_i = \sum_{k=1}^{p} X_{ik}\beta_k) \Rightarrow \mathcal{L}(\beta) \propto \sum_{i=1}^{n} y_i \left(\sum_{k=1}^{p} X_{ik}\beta_k \right) - \sum_{i=1}^{n} \exp\left(\sum_{k=1}^{p} X_{ik}\beta_k \right)$$

  - maximizing the log likelihood $\mathcal{L}(beta)$ (solving for $\frac{\partial \mathcal{L}}{\partial \beta} = 0$) would return a set of optimized coefficients β that will fit the data

Variances and Quasi-Likelihoods

- in each of the linear/Bernoulli/Poisson cases, the ***only*** term in the likelihood functions that depend on the ***data*** is

$$\sum_{i=1}^{n} y_i \eta_i = \sum_{i=1}^{n} y_i \sum_{k=1}^{p} X_{ik}\beta_k = \sum_{k=1}^{p} \beta_k \sum_{i=1}^{n} X_{ik} y_i$$

- this means that we don't need need all of the data collected to maximize the likelihoods/find the coefficients β, but ***only*** need $\sum_{i=1}^{n} X_{ik} y_i$
  - ***Note****: this simplification is a consequence of choosing "**canonical**" link functions, $g(\mu)$, to be in specific forms*
- [Derivation needed] all models achieve their ***maximum*** at the root of the **normal equations**

$$\sum_{i=1}^{n} \frac{(Y_i - \mu_i)}{Var(Y_i)} W_i = 0$$

  where $W_i = \frac{\partial g^{-1}(\mu_i)}{\mu_i}$ or the derivative of the inverse of the link function
  - ***Note****: this is similar to deriving the least square equation where the middle term must be set to 0 to find the solution (see Derivation for β)*
  - ***Note****: $\mu_i = g^{-1}(\eta_i) = g^{-1}\left(\sum_{k=1}^{p} X_{ik}\beta_k\right)$, the normal functions are really functions of β*
- the variance, $Var(Y_i)$, is defined as
  - ***linear model***: $Var(Y_i) = \sigma^2$, where σ is constant
  - ***binomial model***: $Var(Y_i) = \mu_i(1 - \mu_i)$
  - ***Poisson model***: $Var(Y_i) = \mu_i$
- for binomial and Poisson models, there are ***strict relationships*** between the mean and variance that can be easily tested from the data:
  - binomial: mean = μ_i, variance = $\mu_i(1 - \mu_i)$
  - Poisson: mean = μ_i, variance = μ_i

- it is often relevant to have a ***more flexible*** variance model (i.e. data doesn't follow binomial/Poisson distributions exactly but are approximated), even if it doesn't correspond to an actual likelihood, so we can add an extra parameter, ϕ, to the normal equations to form **quasi-likelihood normal equations**

$$binomial: \sum_{i=1}^{n} \frac{(Y_i - \mu_i)}{\phi\mu_i(1-\mu_i)} W_i = 0 Poisson: \sum_{i=1}^{n} \frac{(Y_i - \mu_i)}{\phi\mu_i} W_i = 0$$

  where $W_i = \frac{\partial g^{-1}(\mu_i)}{\mu_i}$ or the derivative of the inverse of the link function
  - for R function `glm()`, its possible to specify for the model to solve using quasi-likelihood normal equations instead of normal equations through the parameter `family = quasi-binomial` and `family = quasi-poisson` respectively
  - ***Note****: the quasi-likelihoods models generally same properties as normal GLM*

Solving for Normal and Quasi-Likelihood Normal Equations

- normal equations have to be solved ***iteratively***
  - the results are $\hat{\beta}_k$, estimated coefficients for the predictors
  - for quasi-likelihood normal equations, $\hat{\phi}$ will be part of the results as well
  - in R, Newton/Raphson's algorithm is used to solve the equations
  - ***asymptotics*** are used for inference of results to broader population (see ***Statistical Inference*** course)
  - ***Note****: many of the ideas, interpretation, and conclusions derived from simple linear models are applicable to GLMs*
- **predicted linear predictor responses** are defined as

$$\hat{\eta} = \sum_{k=1}^{p} X_k \hat{\beta}_k$$

- **predicted mean responses** can be solved from

$$\hat{\mu} = g^{*1}(\hat{\eta})$$

- **coefficients** are interpreted as the ***expected change in the link function*** of the expected response ***per unit change*** in X_k holding other regressors constant, or

$$\beta_k = g(E[Y|X_k = x_k + 1, X_{\sim k} = x_{\sim k}]) - g(E[Y|X_k = x_k, X_{\sim k} = x_{\sim k}])$$

General Linear Models - Binary Models

- **Bernoulli/binary** models are frequently used to model outcomes that have two values
  - alive vs dead
  - win vs loss
  - success vs failure
  - disease vs healthy
- **binomial outcomes** = collection of exchangeable binary outcomes (i.e. flipping coins repeatedly) for the same covariate data
  - in other words, we are interested in the count of predicted 1s vs 0s rather individual outcomes of 1 or 0

Odds

- **odds** are useful in constructing logistic regression models and fairly easy to interpret
  - imagine flipping a coin with success probability p
    - if heads, you win X
    - if tails, you lose Y
  - how should X and Y be set so that the game is ***fair***?

$$E[earnings] = Xp - Y(1-p) = 0 \Rightarrow \frac{Y}{X} = \frac{p}{1-p}$$

  - odds can be interpreted as "How much should you be willing to pay for a p probability of winning a dollar?"
    - if $p > 0.5$, you have to pay more if you lose than you get if you win
    - if $p < 0.5$, you have to pay less if you lose than you get if you win
- odds are **NOT** probabilities
- odds ratio of 1 = no difference in odds or 50% - 50%
  - $p = 0.5 \Rightarrow odds = \frac{0.5}{1-0.5} = 1$
  - log odds ratio of 0 = no difference in odds
    - $p = 0.5 \Rightarrow odds = \log\left(\frac{0.5}{1-0.5}\right) = \log(1) = 0$
- odds ratio < 0.5 or > 2 commonly a "moderate effect"
- **relative risk** = ratios of probabilities instead of odds, and are often easier to interpret but harder to estimate

$$\frac{Pr(W_i|S_i = 10)}{Pr(W_i|S_i = 0)}$$

  - ***Note**: relative risks often have **boundary problems** as the range of* $\log(p)$ *is* $(-\infty,\ 0]$ *where as the range of logit* $\frac{p}{1-p}$ *is* $(-\infty, \infty)$
  - for small probabilities Relative Risk $\approx$ Odds Ratio but **they are not the same**!

Example - Baltimore Ravens Win vs Loss

- the data for this example can be found here
  - the data contains the records 20 games for Baltimore Ravens, a professional American Football team
  - there are 4 columns

    * `ravenWinNum` = 1 for Raven win, 0 for Raven loss
    * `ravenWin` = W for Raven win, L for Raven loss
    * `ravenScore` = score of the Raven team during the match
    * `opponentScore` = score of the Raven team during the match

```
# load the data
load("ravensData.rda")
head(ravensData)
```

```
##   ravenWinNum ravenWin ravenScore opponentScore
## 1           1        W         24             9
## 2           1        W         38            35
## 3           1        W         28            13
## 4           1        W         34            31
## 5           1        W         44            13
## 6           0        L         23            24
```

Example - Simple Linear Regression

- **simple linear regression** can be used model win vs loss for the Ravens

$$W_i = \beta_0 + \beta_1 S_i + \epsilon_i$$

  - W_i = binary outcome, 1 if a Ravens win, 0 if not
  - S_i = number of points Ravens scored
  - β_0 = probability of a Ravens win if they score 0 points
  - β_1 = increase in probability of a Ravens win for each additional point
  - ϵ_i = residual variation, error

- the expected value for the model is defined as

$$E[W_i|S_i, \beta_0, \beta_1] = \beta_0 + \beta_1 S_i$$

- however, the model wouldn't work well as the predicted results ***won't*** be 0 vs 1
  - the error term, ϵ_i, is assumed to be continuous and normally distributed, meaning that the prediction will likely be a decimal
  - therefore, this is ***not*** a good assumption for the model

```
# perform linear regression
summary(lm(ravenWinNum ~ ravenScore, data = ravensData))
```

```
##
## Call:
## lm(formula = ravenWinNum ~ ravenScore, data = ravensData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7302 -0.5076  0.1824  0.3215  0.5719
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.285032   0.256643   1.111   0.2814
```

```
## ravenScore  0.015899   0.009059   1.755   0.0963 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4464 on 18 degrees of freedom
## Multiple R-squared:  0.1461, Adjusted R-squared:  0.09868
## F-statistic:  3.08 on 1 and 18 DF,  p-value: 0.09625
```

- as expected, the model produces a poor fit for the data ($R^2_{adj} = 0.0987$)
- adding a threshold to the predicted outcome (i.e. if $\hat{W}_i < 0.5, \hat{W}_i = 0$) and using the model to predict the results would be ***viable***
  - however, the coefficients for the model are ***not very interpretable***

Example - Logistic Regression

- **probability** of Ravens win is defined as

$$Pr(W_i|S_i, \beta_0, \beta_1)$$

- **odds** is defined as

$$\frac{Pr(W_i|S_i, \beta_0, \beta_1)}{1 - Pr(W_i|S_i, \beta_0, \beta_1)}$$

  which ranges from 0 to ∞
- log odds or **logit** is defined as

$$\log\left(\frac{Pr(W_i|S_i, \beta_0, \beta_1)}{1 - Pr(W_i|S_i, \beta_0, \beta_1)}\right)$$

  which ranges from $-\infty$ to ∞
- we can use the link function and linear predictors to construct the **logistic regression** model

$$g(\mu_i) = \log\left(\frac{\mu_i}{1-\mu_i}\right) = \eta_i$$

$$(substitute\ \mu_i = Pr(W_i|S_i, \beta_0, \beta_1))\ g(\mu_i) = \log\left(\frac{Pr(W_i|S_i, \beta_0, \beta_1)}{1 - Pr(W_i|S_i, \beta_0, \beta_1)}\right) = \eta_i$$

$$(substitute\ \eta_i = \beta_0 + \beta_1 S_i) \Rightarrow\ g(\mu_i) = \log\left(\frac{Pr(W_i|S_i, \beta_0, \beta_1)}{1 - Pr(W_i|S_i, \beta_0, \beta_1)}\right) = \beta_0 + \beta_1 S_i$$

  which can also be written as

$$Pr(W_i|S_i, \beta_0, \beta_1) = \frac{\exp(\beta_0 + \beta_1 S_i)}{1 + \exp(\beta_0 + \beta_1 S_i)}$$

- for the model

$$\log\left(\frac{Pr(W_i|S_i, \beta_0, \beta_1)}{1 - Pr(W_i|S_i, \beta_0, \beta_1)}\right) = \beta_0 + \beta_1 S_i$$

  - β_0 = log odds of a Ravens win if they score zero points
  - β_1 = log odds ratio of win probability for each point scored (compared to zero points)

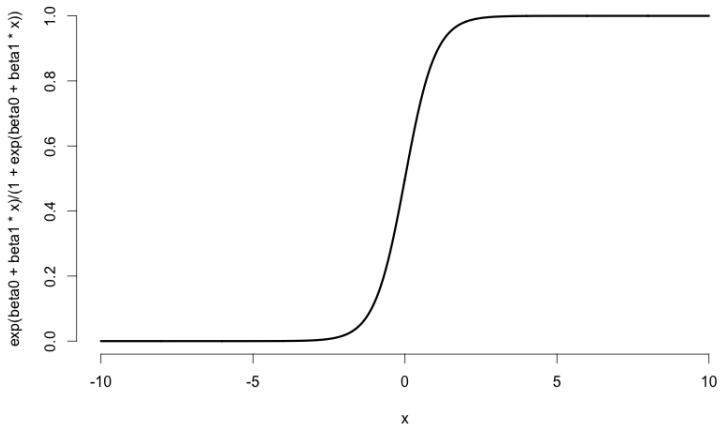
$$\beta_1 = \log\left(odds(S_i = S_i + 1)\right) - \log\left(odds(S_i = S_i)\right) = \log\left(\frac{odds(S_i = S_i + 1)}{odds(S_i = S_i)}\right)$$

  - $\exp(\beta_1)$ = odds ratio of win probability for each point scored (compared to zero points)

$$\exp(\beta_1) = \frac{odds(S_i = S_i + 1)}{odds(S_i = S_i)}$$

- we can leverage the `manupulate` function vary β_0 and β_1 to fit logistic regression curves for simulated data

```
# set x values for the points to be plotted
x <- seq(-10, 10, length = 1000)
# "library(manipulate)" is needed to use the manipulate function
manipulate(
    # plot the logistic regression curve
    plot(x, exp(beta0 + beta1 * x) / (1 + exp(beta0 + beta1 * x)),
         type = "l", lwd = 3, frame = FALSE),
    # slider for beta1
    beta1 = slider(-2, 2, step = .1, initial = 2),
    # slider for beta0
    beta0 = slider(-2, 2, step = .1, initial = 0)
    )
```



- we can use the `glm(outcome ~ predictor, family = "binomial")` to fit a logistic regression to the data

```
# run logistic regression on data
logRegRavens <- glm(ravenWinNum ~ ravenScore, data = ravensData,family="binomial")
# print summary
summary(logRegRavens)
```

```
##
## Call:
## glm(formula = ravenWinNum ~ ravenScore, family = "binomial",
##     data = ravensData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7575  -1.0999   0.5305   0.8060   1.4947
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -1.68001    1.55412  -1.081     0.28
## ravenScore   0.10658    0.06674   1.597     0.11
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 24.435  on 19  degrees of freedom
## Residual deviance: 20.895  on 18  degrees of freedom
## AIC: 24.895
##
## Number of Fisher Scoring iterations: 5
```

- as we can see above, the coefficients β_0 and β_1 are -1.68, 0.107, which are interpreted to be the log odds ratios
- we can convert the log ratios as well as the log confidence intervals to ratios and confidence intervals (in the same units as the data)

```
# take e^coefs to find the log ratios
exp(logRegRavens$coeff)
```

```
## (Intercept)  ravenScore
##   0.1863724   1.1124694
```

```
# take e^log confidence interval to find the confidence intervals
exp(confint(logRegRavens))
```

```
## Waiting for profiling to be done...
```

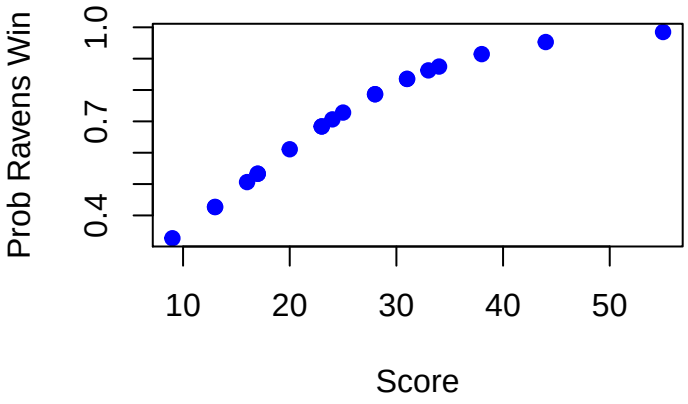
```
##                    2.5 %   97.5 %
## (Intercept) 0.005674966 3.106384
## ravenScore  0.996229662 1.303304
```

- ***Note**: $\exp(x) \approx 1 + x$ for small values (close to 0) of x, this can be a quick way to estimate the coefficients*
- we can interpret the slope, β_1 as 11.247 % increase in probability of winning for every point scored
- we can interpret the intercept, β_0 as 0.186 is the odds for Ravens winning if they scored 0 points
  - ***Note**: similar to the intercept of a simple linear regression model, the intercept should be interpreted carefully as it is an extrapolated value from the model and may not hold practical meaning*
- to calculate specific probability of winning for a given number of points

$$Pr(W_i|S_i, \hat{\beta}_0, \hat{\beta}_1) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 S_i)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 S_i)}$$

- the resulting logistic regression curve can be seen below

```
# plot the logistic regression
plot(ravensData$ravenScore,logRegRavens$fitted,pch=19,col="blue",xlab="Score",ylab="Prob Ravens Win")
```



Example - ANOVA for Logistic Regression

- ANOVA can be performed on a single logistic regression, in which it will analyze the change in variances with addition of parameters in the model, or multiple nested logistic regression (similar to linear models)

```
# perform analysis of variance
anova(logRegRavens,test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: ravenWinNum
##
## Terms added sequentially (first to last)
##
##
##            Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                           19     24.435
## ravenScore  1   3.5398        18     20.895  0.05991 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- ANOVA returns information about the model, link function, response, as well as analysis of variance for adding terms
  - `Df` = change in degrees of freedom
    - the value 1 refers to adding the `ravenScore` parameter (slope)
  - `Deviance` = measure of goodness of model fit compare to the previous model
  - `Resid. Dev` = residual deviance for current model
  - `Pr(>Chi)` = used to evaluate the significance of the added parameter
    - in this case, the `Deviance` value of 3.54 is used to find the corresponding p-value from the Chi Squared distribution, which is 0.06
      - ***Note**: Chi Squared distribution with 1 degree of freedom is simply the squared of normal distribution, so z statistic of 2 corresponds to 95% for normal distribution indicates that deviance of 4 corresponds to approximately 5% in the Chi Squared distribution (which is what our result shows)*

Further resources

- Wikipedia on Logistic Regression
- Logistic regression and GLMs in R
- Brian Caffo's lecture notes on: Simpson's Paradox, Retrospective Case-control Studies
- Open Intro Chapter on Logistic Regression

General Linear Models - Poisson Models

- **Poisson distribution** is a useful model for counts and rates
  - **rate** = count per unit of time
  - linear regression with transformation is an alternative
- count data examples
  - calls to a call center
  - number of flu cases in an area
  - number of cars that cross a bridge
- rate data examples
  - percent of children passing a test
  - percent of hits to a website from a country
  - radioactive decay
- Poisson model examples
  - modeling web traffic hits incidence rates
  - approximating binomial probabilities with small p and large n
  - analyzing contingency table data (tabulated counts for categorical variables)

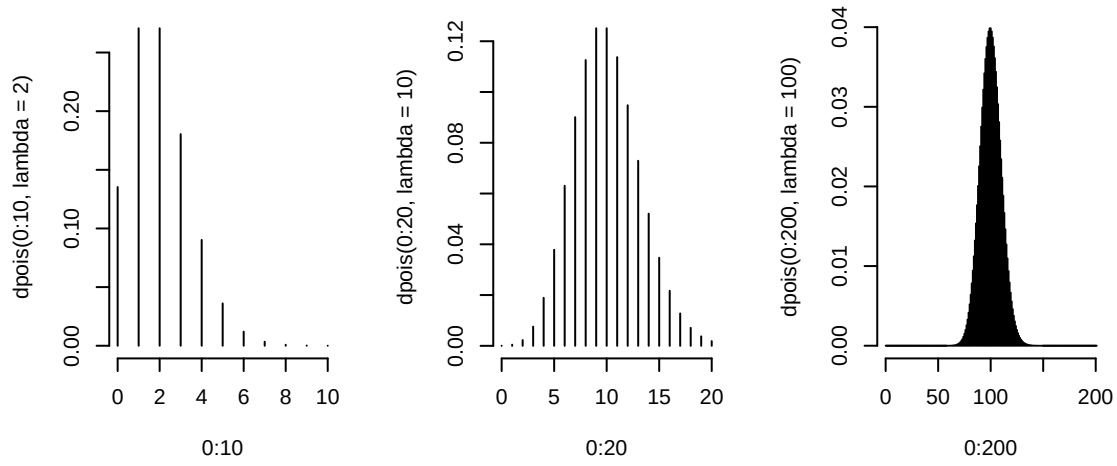
Properties of Poisson Distribution

- a set of data X is said to follow the Poisson distribution, or $X \sim Poisson(t\lambda)$, if

$$P(X = x) = \frac{(t\lambda)^x e^{-t\lambda}}{x!}$$

  where $x = 0, 1, \ldots$
  - λ = rate or expected count per unit time
  - t = monitoring time
- **mean** of Poisson distribution is $E[X] = t\lambda$, thus $E[X/t] = \lambda$
- **variance** of the Poisson is $Var(X) = t\lambda = \mu(mean)$
  - ***Note**: Poisson approaches a Gaussian/normal distribution as $t\lambda$ gets large*
- below are the Poisson distributions for various values of λ

```
# set up 1x3 panel plot
par(mfrow = c(1, 3))
# Poisson distribution for t = 1, and lambda = 2
plot(0 : 10, dpois(0 : 10, lambda = 2), type = "h", frame = FALSE)
# Poisson distribution for t = 1, and lambda = 10
plot(0 : 20, dpois(0 : 20, lambda = 10), type = "h", frame = FALSE)
# Poisson distribution for t = 1, and lambda = 100
plot(0 : 200, dpois(0 : 200, lambda = 100), type = "h", frame = FALSE)
```

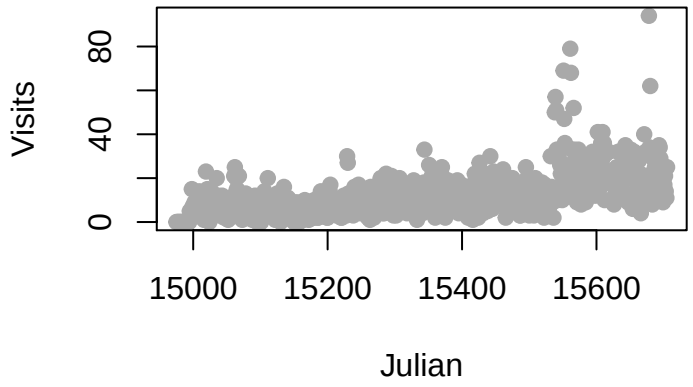


- as we can see from above, for large values of λ, the distribution looks like the Gaussian

Example - Leek Group Website Traffic

- for this example, we will be modeling the daily traffic to Jeff Leek's web site: http://biostat.jhsph.edu/~jleek/
  - the data comes from Google Analytics and was extracted by the `rga` package that can be found at http://skardhamar.github.com/rga/
- for the purpose of the example, the time is ***always*** one day, so $t = 1$, Poisson mean is interpreted as web hits per day
  - if $t = 24$, we would be modeling web hits per hour
- the data can be found here

```
# laod data
load("gaData.rda")
# convert the dates to proper formats
gaData$julian <- julian(gaData$date)
# plot visits vs dates
plot(gaData$julian,gaData$visits,pch=19,col="darkgrey",xlab="Julian",ylab="Visits")
```



Example - Linear Regression

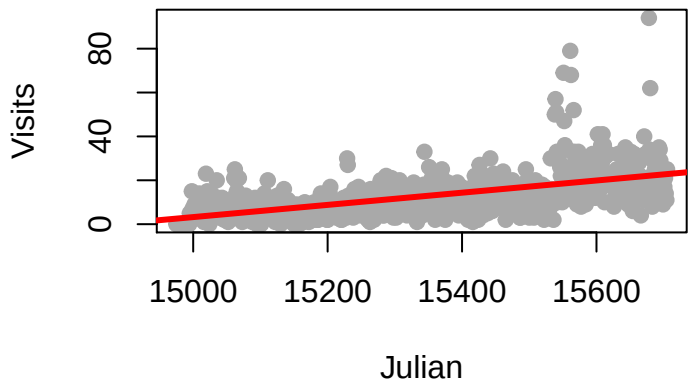
- the traffic can be modeled using linear model as follows

$$NH_i = \beta_0 + \beta_1 JD_i + \epsilon_i$$

  - NH_i = number of hits to the website
  - JD_i = day of the year (Julian day)
  - β_0 = number of hits on Julian day 0 (1970-01-01)
  - β_1 = increase in number of hits per unit day
  - ϵ_i = variation due to everything we didn't measure
- the expected outcome is defined as

$$E[NH_i|JD_i, \beta_0, \beta_1] = \beta_0 + \beta_1 JD_i$$

```
# plot the visits vs dates
plot(gaData$julian,gaData$visits,pch=19,col="darkgrey",xlab="Julian",ylab="Visits")
# perform linear regression
lm1 <- lm(gaData$visits ~ gaData$julian)
# plot regression line
abline(lm1,col="red",lwd=3)
```



Example - log Outcome

- if we are interested in relative increases in web traffic, we can the natural log of the outcome, so the linear model becomes

$$\log(NH_i) = \beta_0 + \beta_1 JD_i + \epsilon_i$$

  - $\log(NH_i)$ = number of hits to the website
  - JD_i = day of the year (Julian day)
  - β_0 = log number of hits on Julian day 0 (1970-01-01)
  - β_1 = increase in log number of hits per unit day
  - ϵ_i = variation due to everything we didn't measure
- when we take the natural log of outcomes and fit a regression model, the exponentiated coefficients estimate quantities based on the geometric means rather than the measured values
  - $e^{E[\log(Y)]}$ = geometric mean of Y
    - geometric means are defined as

$$e^{\frac{1}{n}\sum_{i=1}^{n}\log(y_i)} = (\prod_{i=1}^{n} y_i)^{1/n}$$

      which is the estimate for the **population geometric mean**

    - as we collect infinite amount of data, $\prod_{i=1}^{n} y_i)^{1/n} \rightarrow E[\log(Y)]$
  - e^{β_0} = estimated geometric mean hits on day 0
  - e^{β_1} = estimated relative increase or decrease in geometric mean hits per day
  - ***Note**: not we can not take the natural log of zero counts, so often we need to adding a constant (i.e. 1) to construct the log model*
    - adding the constant changes the interpretation of coefficient slightly
    - e^{β_1} is now the relative increase or decrease in geometric mean hits **+ 1** per day

- the expected outcome is

$$E[\log(NH_i|JD_i, \beta_0, \beta_1)] = \beta_0 + \beta_1 JD_i$$

```
round(exp(coef(lm(I(log(gaData$visits + 1)) ~ gaData$julian))), 5)
```

```
##   (Intercept) gaData$julian
##       0.00000       1.00231
```

- as we can see from above, the daily increase in hits is 0.2%

Example - Poisson Regression

- the Poisson model can be constructed as log of the mean

$$\log\left(E[NH_i|JD_i, \beta_0, \beta_1]\right) = \beta_0 + \beta_1 JD_i$$

  or in other form

$$E[NH_i|JD_i, \beta_0, \beta_1] = \exp\left(\beta_0 + \beta_1 JD_i\right)$$

  - NH_i = number of hits to the website
  - JD_i = day of the year (Julian day)
  - β_0 = expected number of hits on Julian day 0 (1970-01-01)
  - β_1 = expected increase in number of hits per unit day
  - ***Note**: Poisson model differs from the log outcome model in that the coefficients are interpreted naturally as expected value of outcome where as the log model is interpreted on the log scale of outcome*

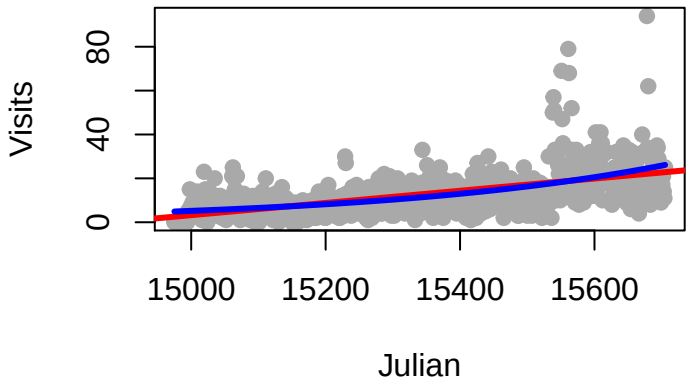
- we can transform the Poisson model to

$$E[NH_i|JD_i, \beta_0, \beta_1] = \exp\left(\beta_0 + \beta_1 JD_i\right) = \exp\left(\beta_0\right)\exp\left(\beta_1 JD_i\right)$$

  - $\beta_1 = E[NH_i|JD_i + 1, \beta_0, \beta_1] - E[NH_i|JD_i, \beta_0, \beta_1]$
  - β_1 can therefore be interpreted as the **relative** increase/decrease in web traffic hits per one day increase

- `glm(outcome~predictor, family = "poisson")` = performs Poisson regression

```
# plot visits vs dates
plot(gaData$julian,gaData$visits,pch=19,col="darkgrey",xlab="Julian",ylab="Visits")
# construct Poisson regression model
glm1 <- glm(gaData$visits ~ gaData$julian,family="poisson")
# plot linear regression line in red
abline(lm1,col="red",lwd=3)
# plot Poisson regression line in
lines(gaData$julian,glm1$fitted,col="blue",lwd=3)
```

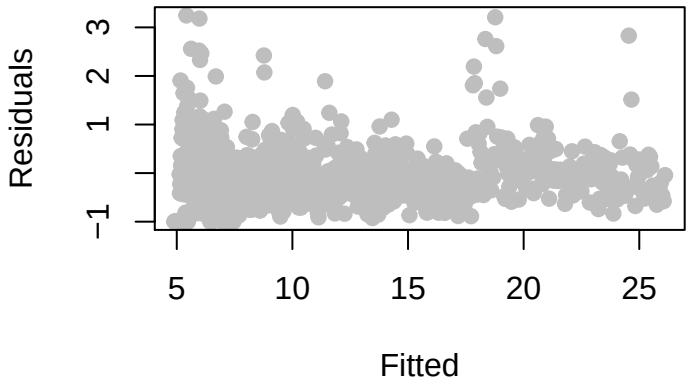


- ***Note**: the Poisson fit is non-linear since it is linear only on the log of the mean scale*

Example - Robust Standard Errors with Poisson Regression

- variance of the Poisson distribution is defined to be the mean of the distribution, so we would expect the variance to increase with higher values of X
- below is the residuals vs fitted value plot for the Poisson regression model

```
# plot residuals vs fitted values
plot(glm1$fitted,glm1$residuals,pch=19,col="grey",ylab="Residuals",xlab="Fitted")
```



- as we can see from above, the residuals don't appear to be increasing with higher fitted values
- even if the mean model is correct in principle, there could always be a certain degree of ***model mis-specification***
- to account for mis-specifications for the model, we can use
    1. `glm(outcome~predictor, family = "quasi-poisson")` = introduces an additional multiplicative factor ϕ to denominator of model so that the variance is $\phi\mu$ rather than just μ (see Variances and Quasi-Likelihoods)
    2. more generally, *robust standard errors* (effectively constructing wider confidence intervals) can be used
- **model agnostic standard errors**, implemented through the `sandwich` package, is one way to calculate the robust standard errors
    - algorithm assumes the mean relationship is specified correctly and attempts to get a general estimates the variance that isn't highly dependent on the model

- it uses assumption of large sample sizes and asymptotics to estimate the confidence intervals that is robust to model mis-specification
- ***Note:*** *more information can be found at* [*http://stackoverflow.com/questions/3817182/vcovhc-and-confidence-interval*](http://stackoverflow.com/questions/3817182/vcovhc-and-confidence-interval)

```
# load sandwich package
library(sandwich)
# compute
confint.agnostic <- function (object, parm, level = 0.95, ...)
{
    cf <- coef(object); pnames <- names(cf)
    if (missing(parm))
        parm <- pnames
    else if (is.numeric(parm))
        parm <- pnames[parm]
    a <- (1 - level)/2; a <- c(a, 1 - a)
    pct <- stats:::format.perc(a, 3)
    fac <- qnorm(a)
    ci <- array(NA, dim = c(length(parm), 2L), dimnames = list(parm,
                                                               pct))
    ses <- sqrt(diag(sandwich::vcovHC(object)))[parm]
    ci[] <- cf[parm] + ses %o% fac
    ci
}
# regular confidence interval from Poisson Model
confint(glm1)
```

```
##                       2.5 %        97.5 %
## (Intercept)   -34.346577587 -31.159715656
## gaData$julian   0.002190043   0.002396461
```

```
# model agnostic standard errors
confint.agnostic(glm1)
```

```
##                       2.5 %        97.5 %
## (Intercept)   -36.362674594 -29.136997254
## gaData$julian   0.002058147   0.002527955
```

- as we can see from above, the robust standard error produced slightly wider confidence intervals

Example - Rates

- if we were to model the percentage of total web hits that are coming from the *Simply Statistics* blog, we could construct the following model

$$E[NHSS_i|JD_i, \beta_0, \beta_1]/NH_i = \exp(\beta_0 + \beta_1 JD_i)$$
$$(take\ \log\ of\ both\ sides)\ \log(E[NHSS_i|JD_i, \beta_0, \beta_1]) - \log(NH_i) = \beta_0 + \beta_1 JD_i$$
$$(move\ \log(NH_i)\ to\ right\ side)\ \log(E[NHSS_i|JD_i, \beta_0, \beta_1]) = \log(NH_i) + \beta_0 + \beta_1 JD_i$$

  - when **offset** term, $\log(NH_i)$, is present in the Poisson model, the interpretation of the coefficients will be relative to the offset quantity

  - it's important to recognize that the fitted response doesn't change
  - ***example***: to convert the outcome from daily data to hourly, we can add a factor 24 so that the model becomes

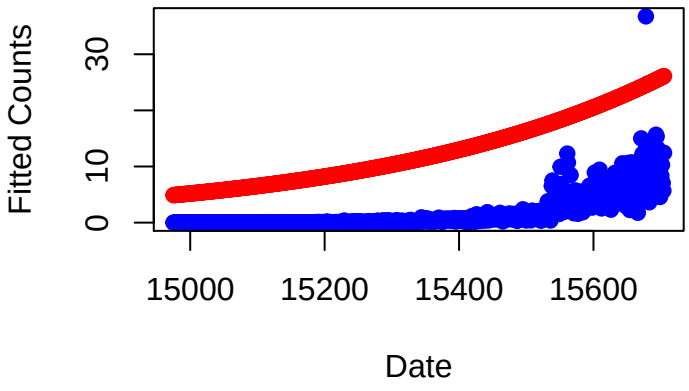
$$E[NHSS_i|JD_i, \beta_0, \beta_1]/24 = \exp(\beta_0 + \beta_1 JD_i)$$

$$(take\ \log\ of\ both\ sides)\ \log(E[NHSS_i|JD_i, \beta_0, \beta_1]) - \log(24) = \beta_0 + \beta_1 JD_i$$

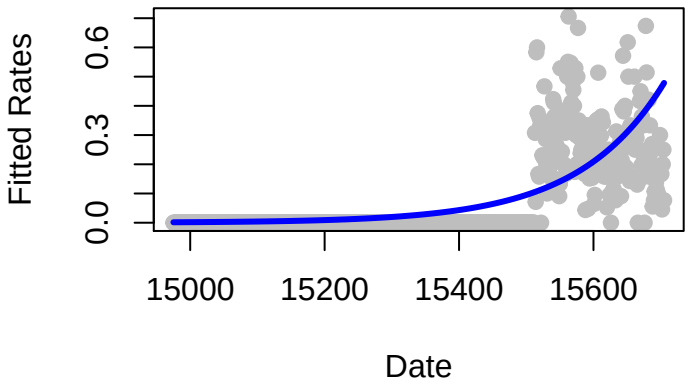
$$(move\ \log(24)\ to\ right\ side)\ \log(E[NHSS_i|JD_i, \beta_0, \beta_1]) = \log(24) + \log(NH_i) + \beta_0 + \beta_1 JD_i$$

- back to the rates model, we fit the Poisson model now with an offset so that the model is interpreted with respect to the number of visits
  - `glm(outcome ~ predictor, offset = log(offset), family = "poisson")` = perform Poisson regression with offset
  - `glm(outcome ~ predictor + log(offset))` = produces the same result

```
# perform Poisson regression with offset for number of visits
glm2 <- glm(gaData$simplystats ~ julian(gaData$date),offset=log(visits+1),
            family="poisson",data=gaData)
# plot the fitted means (from simply statistics)
plot(julian(gaData$date),glm2$fitted,col="blue",pch=19,xlab="Date",ylab="Fitted Counts")
# plot the fitted means (total visit)
points(julian(gaData$date),glm1$fitted,col="red",pch=19)
```




```
# plot the rates for simply stats
plot(julian(gaData$date),gaData$simplystats/(gaData$visits+1),col="grey",xlab="Date",
     ylab="Fitted Rates",pch=19)
# plot the fitted rates for simply stats (visit/day)
lines(julian(gaData$date),glm2$fitted/(gaData$visits+1),col="blue",lwd=3)
```



- ***Note***: *we added 1 to the* `log(visits)` *to address 0 values*

Further Resources

- Log-linear mMdels and Multi-way Tables
- Wikipedia on Poisson Regression
- Wikipedia on Overdispersion
- Regression Models for count data in R
- `pscl` package -
  - often time in modeling counts, they maybe more zero counts in the data than anticipated, which the regular Poisson model doesn't account for
  - the function `zeroinfl` fits zero inflated Poisson (ziP) models to such data

Fitting Functions

- **scatterplot smoothing** = fitting functions (multiple linear models, piece-wise zig-zag lines) to data in the form $Y_i = f(X_i) + \epsilon_i$
- consider the model

$$Y_i = \beta_0 + \beta_1 X_i + \sum_{k=1}^{d} (x_i - \xi_k)_+ \gamma_k + \epsilon_i$$

  where $(a)_+ = a$ if $a > 0$ and 0 otherwise and $\xi_1 \leq ... \leq \xi_d$ are known **knot points**
- the mean function

$$E[Y_i] = \beta_0 + \beta_1 X_i + \sum_{k=1}^{d} (x_i - \xi_k)_+ \gamma_k$$

  is continuous at the knot points
  - for $\xi_k = 5$, the expected value for Y_i as x_i approaches 5 from the left is

$$\begin{aligned} E[Y_i]_{\xi=5|left} &= \beta_0 + \beta_1 x_i + (x_i - 5)_+ \gamma_k \\ (since\ x_i < 5)\ E[Y_i]_{\xi=5|left} &= \beta_0 + \beta_1 x_i \end{aligned}$$

  - the expected value for Y_i as x_i approaches 5 from the right is

$$\begin{aligned} E[Y_i]_{\xi=5|right} &= \beta_0 + \beta_1 x_i + (x_i - 5)_+ \gamma_k \\ (since\ x_i > 5)\ E[Y_i]_{\xi=5|right} &= \beta_0 + \beta_1 x_i + (x_i - 5)\gamma_k \\ (simplify)\ E[Y_i]_{\xi=5|right} &= \beta_0 - 5\gamma_k + (\beta_1 + \gamma_k) x_i \end{aligned}$$

  - as we can see from above, the right side is just another line with different intercept $(\beta_0 - 5\gamma_k)$ and slope $(\beta_1 + \gamma_k)$
  - so as x approaches 5, both sides converge

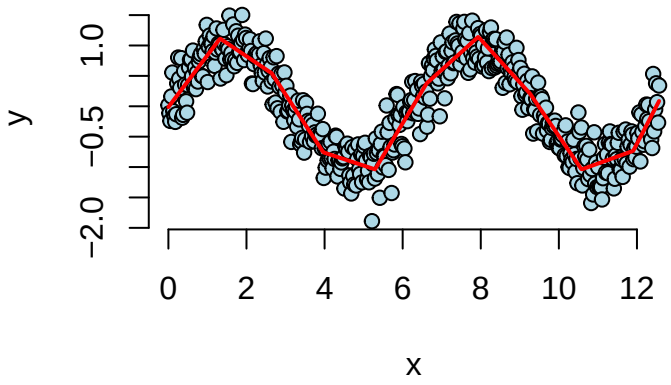
Considerations

- **basis** = the collection of regressors
- single knot point terms can fit *hockey-stick-like* processes
- these bases can be used in GLMs (as an additional term/predictor) as well
- issue with these approaches is the **large** number of parameters introduced
  - requires some method of ***regularization***, or penalize for large number of parameters (see **Practical Machine Learning** course)
  - introducing large number of knots have significant consequences

Example - Fitting Piecewise Linear Function

```
# simulate data
n <- 500; x <- seq(0, 4 * pi, length = n); y <- sin(x) + rnorm(n, sd = .3)
# define 20 knot points
knots <- seq(0, 8 * pi, length = 20);
# define the ()+ function to only take the values that are positive after the knot pt
splineTerms <- sapply(knots, function(knot) (x > knot) * (x - knot))
# define the predictors as X and spline term
xMat <- cbind(x, splineTerms)
# fit linear models for y vs predictors
```

```r
yhat <- predict(lm(y ~ xMat))
# plot data points (x, y)
plot(x, y, frame = FALSE, pch = 21, bg = "lightblue")
# plot fitted values
lines(x, yhat, col = "red", lwd = 2)
```



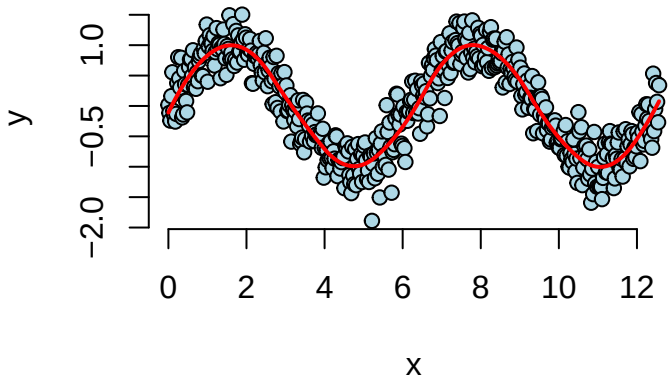
Example - Fitting Piecewise Quadratic Function

- adding squared terms makes it ***continuous*** AND ***differentiable*** at the knot points, and the model becomes

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \sum_{k=1}^{d} (x_i - \xi_k)_+^2 \gamma_k + \epsilon_i$$

  where $(a)_+^2 = a^2$ if $a > 0$ and 0 otherwise
- adding cubic terms makes it twice continuously differentiable at the knot points, etcetera

```r
# define the knot terms in the model
splineTerms <- sapply(knots, function(knot) (x > knot) * (x - knot)^2)
# define the predictors as x, x^2 and knot terms
xMat <- cbind(x, x^2, splineTerms)
# fit linear models for y vs predictors
yhat <- predict(lm(y ~ xMat))
# plot data points (x, y)
plot(x, y, frame = FALSE, pch = 21, bg = "lightblue")
# plot fitted values
lines(x, yhat, col = "red", lwd = 2)
```



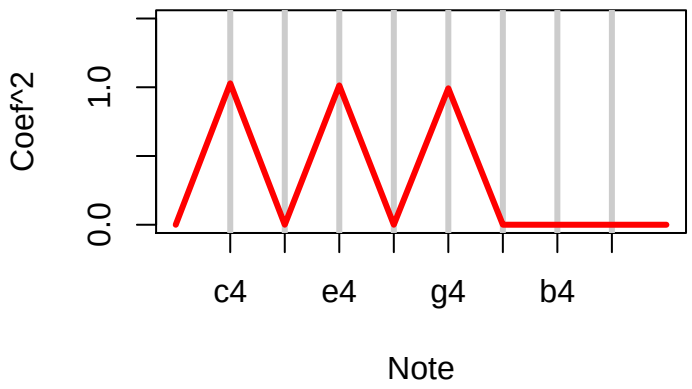
Example - Harmonics using Linear Models

- **discrete Fourier transforms** = instance of linear regression model, use sin and cosine functions as basis to fit data
- to demonstrate this, we will generate 2 seconds of sound data using sin waves, simulate a chord, and apply linear regression to find out which notes are playing

```
# frequencies for white keys from c4 to c5
notes4 <- c(261.63, 293.66, 329.63, 349.23, 392.00, 440.00, 493.88, 523.25)
# generate sequence for 2 seconds
t <- seq(0, 2, by = .001); n <- length(t)
# define data for c4 e4 g4 using sine waves with their frequencies
c4 <- sin(2 * pi * notes4[1] * t); e4 <- sin(2 * pi * notes4[3] * t);
g4 <- sin(2 * pi * notes4[5] * t)
# define data for a chord and add a bit of noise
chord <- c4 + e4 + g4 + rnorm(n, 0, 0.3)
# generate profile data for all notes
x <- sapply(notes4, function(freq) sin(2 * pi * freq * t))
# fit the chord using the profiles for all notes
fit <- lm(chord ~ x - 1)
```

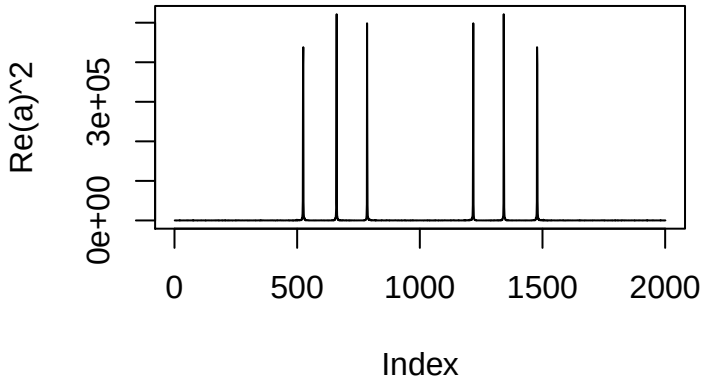
- after generating the data and running the linear regression, we can plot the results to see if the notes are correctly identified

```
# set up plot
plot(c(0, 9), c(0, 1.5), xlab = "Note", ylab = "Coef^2", axes = FALSE, frame = TRUE, type = "n")
# set up axes
axis(2)
axis(1, at = 1 : 8, labels = c("c4", "d4", "e4", "f4", "g4", "a4", "b4", "c5"))
# add vertical lines for each note
for (i in 1 : 8) abline(v = i, lwd = 3, col = grey(.8))
# plot the linear regression fits
lines(c(0, 1 : 8, 9), c(0, coef(fit)^2, 0), type = "l", lwd = 3, col = "red")
```



- as we can see from above, the correct notes were identified
- we can also use the **Fast Fourier Transforms** to identify the notes
  - `fft(data)` = performs fast Fourier transforms on provided data
  - `Re(data)` = subset to only the real components of the complex data

```r
# perform fast fourier transforms on the chord matrix
a <- fft(chord)
# plot only the real components of the fft
plot(Re(a)^2, type = "l")
```



- ***Note**: the algorithm checks for all possible notes at all frequencies it can detect, which is why the peaks are very high in magnitude*
- ***Note**: the symmetric display of the notes are due to periodic symmetries of the sine functions*