Formulas Regression Models

02/05/2020

Some basics

X= predictor/independent variable; Y= outcome/dependent variable

a) sum of least square of mean:$\sum_{i=1}^{n} = (Y_i - \bar{Y})^2$

- this value is smallest at the empirical mean (μ) because this is where the distance between observed and estimated data points is the smallest; the physical middle of the data

b) sum of least square of mean when weights known: $\sum_{i=1}^{n}(w(Y - \bar{Y})^2)$

- when want to calc the minimum μ then: sum(x*w)/sum(w)

c) centering a random variable: $Y - \bar{Y} = Y_c$ –> $mean(Y_c) = 0$

d) scaling a random variable: $\frac{X_i}{s}$ –> $sd = 1$

e) normalizing data: $Z_i = \frac{X_i - \bar{X}}{s}$ –> $\mu = 0$ and $s = 1$

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

g) Correlation: normalized data –> $Cor(X, Y) = \frac{Cov(X,Y)}{S_x S_y}$

- correclation between outcome/predictor variables indifferent: $Cor(Y, X) = Cor(X, Y)$
- Cor(Y, X)=Cor(Yn, Xn)
- must: $-1 < Cor(Y, X) < 1$
- $cor(Y, X) = \pm 1$ (perfect correlation), $cor(Y, X) = 0$ (no relationship)

h) regression to origin (calc slope of best fit line):

- center intercept by centering each variable($Y - \bar{Y} = Y_c$) –> new intercept will be at the intersect of mean(X_c) and mean(Y_c)
- calculate the lowest mean square error (mse) $\sum(Y_i - X_i\beta)^2$ for the best slope ($\downarrow \hat{\beta}_1 \Rightarrow \downarrow mse$)
- Rcode feature same results:
  - coef(lm(I(Y – mean(Y)) ~ I(X- mean(X)) -1, data=df))[2] –> slope
    - "-1" so regression go through origin
  - sum(Y_c * X_c)/sum(X_c^2)

Ordinary least squares

a) slope: $\hat{\beta}_1 = cor(Y,X)\frac{sd(Y)}{sd(X)}$, intercept: $\hat{\beta}_o = \bar{Y} - \hat{\beta}_1\bar{X}$

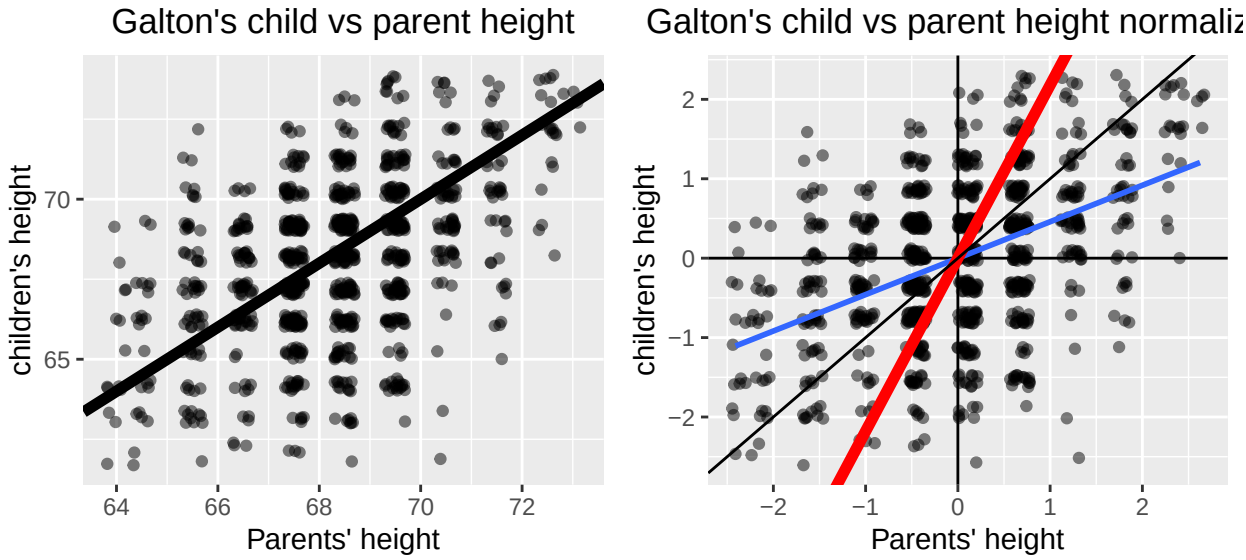
- Rcode: beta1=cor(y, x) x sd(y)/sd(x); beta0=mean(y)-beta1 x mean(x)
- Rcode: $\hat{\beta}_o$=coef(lm(y ~ x))[1]; $\hat{\beta}_1$=coef(lm(y ~ x))[2]
- Rcode: lmVar=lm(Y ~ X); then to get complete pic: summary($lmVar$)
- if X=outcome and Y=predictor then $\hat{\beta}_1 = cor(X,Y)\frac{sd(X)}{sd(Y)}$

b) best line fit data: $Y_i = \beta_o + X_1\beta_1$

- slope is still the same when center data points: $\hat{\beta}_1 = \frac{\sum Y_c X_c}{\sum X_c^2}$ when $\hat{\beta}_o = 0$
- normalize data $\Rightarrow \hat{\beta}_1 = cor(Y,X) \Rightarrow Y_i = cor(Y,X)X_i$

Regression to mean

concept is when comparing values, the extremeties of their respective distribution move towards the mean over time



- child(Y) and parent(X) variables normalized:
  - $\hat{\beta}_1 = cor(Y,X)$ (black line)
  - best fit line (blue line)
  - $Y_i = cor(Y,X)X_i$; $X_i = cor(Y,X)Y_i$
  - regression line passes through mean(Xn) and mean(Yn) intersection
  - this eventually regresses to mean (0,0) unless $cor(Y,X) = 1$ then no regression to mean
  - when X=outcome and Y=predictor, then $\hat{\beta}_1 = \frac{1}{cor(Y,X)}$ (red slope line)
- geom_jitter() and jitter(): used to create some noise in a variable and graph it with frequency emphasized in various degrees. ex: plot(jitter(*var1*, *num*) ~ *var2*, *dataFrame*)
- residuals: distance between X_i and their estimates on regression line.

  - residual μ=0 $\Rightarrow$ residuals are balanced amoung the data points
  - cov(*lmVar$residuals*, X)=0

- var(data)=var(estimate) + var(residuals) **aka** var(dependentVar)=var(independentVar) + var(independentVar)

Statistical linear regression models

a) Regression coefficients:

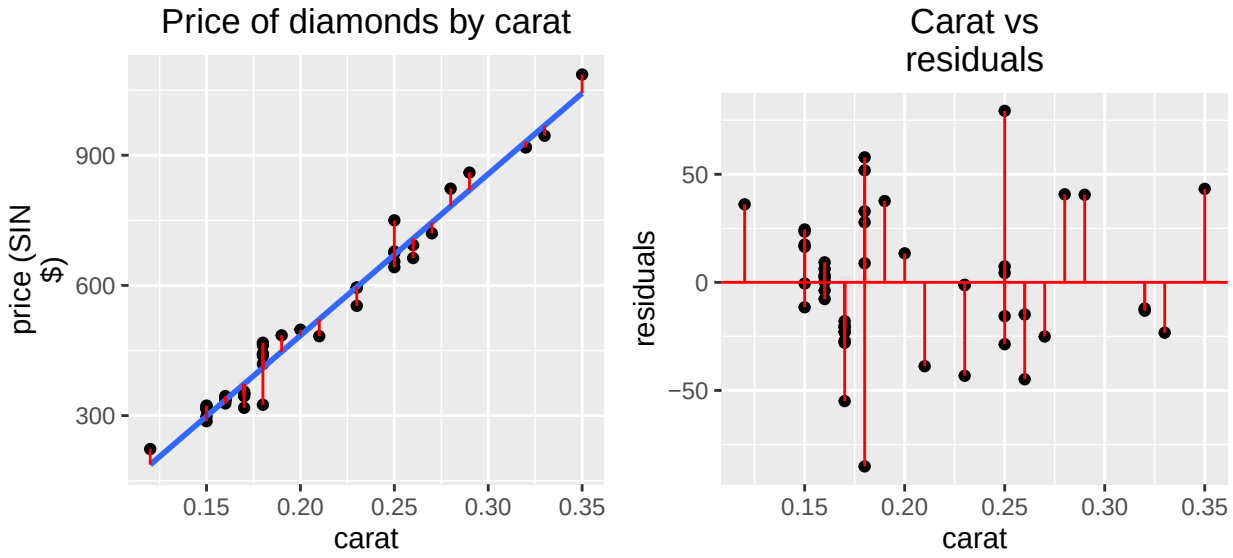
- β_o: when $X = 0$ then $Y = \beta_o + \beta_1 0$ (not realisticly possible/interesting)
  - X by a: $\tilde{\beta}_o + \beta_1(X_i - a)$ (change intercept, not slope)
  - $a = \bar{X} \Rightarrow$ the expected outcome and mean predictor
- β_1: expected Y change when X change by 1 unit $\Rightarrow \beta_o + \beta_1(x+1) - (\beta_o + \beta_1 x)$
  - changing X units: $\beta_o + \frac{\beta_1}{a}(X_i a) + \epsilon_i$ ($Xa \Rightarrow \frac{\beta_1}{a}$)
    - ϵ_i=gaussian error
- Regression for prediction in R:
  - calc regression line: lmVar <- lm(Y ~ X, data=dataFrame)
  - print out slope/intercept: coef(lmVar)
  - center X to interpret intercept: lmVar <- lm(Y~ I(X - mean(X)), data=dataFrame)
  - change scale X by $\frac{1}{num}$: lmVar <- lm(Y ~ I(X * num), data=dataFrame)
  - predict Y w/regression when $X = newX$ method1: lmVar <- coef(lmVar)[1] + coef(lmVar)[2] * newX
    - if newX outside X data range then cant rely on this calc
  - predict Y w/regression method2: predict(lmVar, newdata=data.frame(xVar=newX))
  - stats on coefficients (i.e. pVal): summary(lmVar)$coef

Residuals

a) residuals: $e_i = Y_i - \hat{Y}_i$ where observed=Y_i and predicted=$\hat{Y}_i$, aka vertical distance btw observed data point and regression line

- least squares minimizes the sum of the squared residual: $\sum_{i=1}^{n} e_i^2$
- when intercept included: $\sum_{i=1}^{n} e_i = 0$ and $mean(e_i) = 0$; when regressor/predictor var. included: $\sum_{i=1}^{n} e_i X_i = 0$
- Rcode: fit <- lm(yVar ~ xVar, data=dataFrame)
  - extract residuals: e <- resid(fit)
  - calc predicted outcome($\hat{Y}_i$): yhat <- predict(fit); yhat <- coef(fit)[1] - coef(fit)[2] * xVar
  - calc residuals around $\bar{Y}$: e1 <- resid(lm(Y ~ 1, data=dataFrame))
- Residual vs Xvar: expand residual length for insite into data
  - when residual size pretty small, then use e vs X plot to detect distribution about the regression line (i.e. heteroskedasticity)

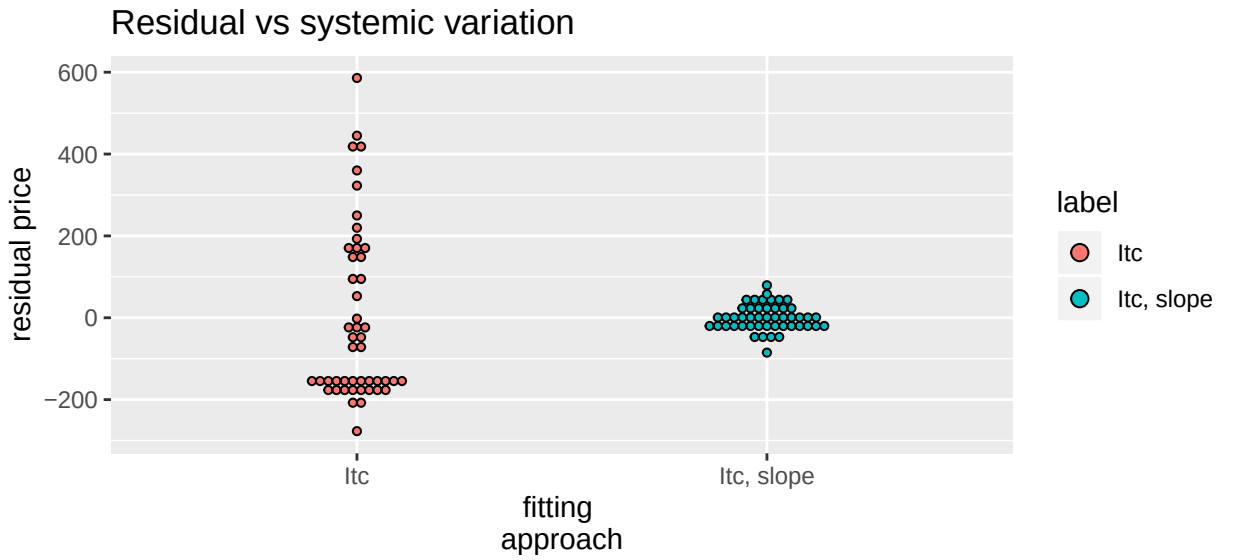
```
library(UsingR); data(diamond); library(ggplot2); fit <- lm(price ~ carat, data=diamond)
#graph X vs Y with delineated residual size basics first
g1 <- ggplot(diamond, aes(x=carat, y=price)) + geom_point()
#graph predicted outcome --> regression line --> vertical segements connex observed to predicted
g1 <- g1 + geom_point(aes(y=predict(fit)), alpha=0.005) +
geom_smooth(method="lm", se=F) + geom_segment(aes(xend=carat,yend=predict(fit)),
col="red")+ labs(title="Price of diamonds by carat", x="carat", y="price (SIN
$)") + theme(plot.title = element_text(hjust=0.5))
#graph X vs residuals
g2 <- ggplot(data=diamond, aes(x=carat, y=resid(fit))) + geom_point() +
geom_point(aes(y=0), alpha=0.005) + geom_hline(yintercept=0, color="red") +
geom_segment(aes(xend=carat, yend=0), col="red") + labs(title="Carat vs
residuals", x="carat", y="residuals") + theme(plot.title = element_text(hjust=0.5))
grid.arrange(g1, g2, nrow=1)
```



- calc residual variation for population variation: $\hat{\sigma}^2 = \frac{1}{n-2}\sum_{i=1}^{n} e_i^2$ where $n-2$ is to account for coefficients of regression
  - Rcode Var of error: fit <- lm(Y ~ X, data=dataFrame); n <- length(Y); summary(fit)$sigma^2
  - Rcode SD of error (aka sigma): method1: sqrt(sum(resid(fit)^2)/(n-2)); method2: sqrt(sum(fit$residuals^2)/(n-2)); method3: sqrt(deviance(fit)/(n-2))
- Variance summary: totalVar=residualVar(Yonly)+regressionVar(w/X): $\sum_{i=1}^{n}(Y_i - \bar{Y})^2 = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2$
- defines proportion of total variability explained by model: variation decrease w/xVar inclusion

```
library(UsingR); data(diamond); library(ggplot2)
#compare Y distribution alone (residual variation) vs Y distribution with
#regression coefficient influence (systematic variation)
#calc residuals around Ymean and regression line
e <- c(resid(lm(price ~ 1, data=diamond)), resid(fit))
#factor for (Yvariation)intercept only and intercept + Xvar residuals
label <- factor(c(rep("Itc", nrow(diamond)), rep("Itc, slope", nrow(diamond))))
#build box scatter plot
```

```
g1 <- ggplot(data.frame(e=e, label=label), aes(y=e, x=label, fill=label)) +
geom_dotplot(binaxis="y", stackdir="center", binwidth=20) + labs(x="fitting
approach", y="residual price", title="Residual vs systemic variation")
g1
```



- R squared:% of totalVar explained by linear regression variation: $R^2 = \frac{\sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$ aka $\frac{intercept+slope}{interceptonly}$; also sample correlation squared
  - must: $0 \leq R^2 \leq 1$
  - R^2= cor(X, Y)^2
  - value not reveal whole story: dramatic different data models can have same R^2 values
  - calc in R quick: fit <- lm(Y ~ X); summary(lm(Y ~ X, data=dataFrame))$r.squared; OR cor(Y, X)^2
  - calc long in R: fit1 <- lm(Y ~ X); fit2 <- lm(Y ~ 1); sseNum <- sum((predict(fit1) - X)^2); sseDen <- sum((predict(fit2) - X)^2); sseNum/sseDen
  - calc long in R: sTot <- sum((Y-mean(Y))^2); sRes <- deviance(fit); 1-(sRes/sTot)

Regression Inference

a) variance of coeffiecients: $Var(\hat{\beta}_1) = \frac{\sigma^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}$ and $Var(\hat{\beta}_o) = (\frac{1}{n} + \frac{\bar{X}^2}{\sum_{i=1}^{n}(X_i - \bar{x})^2})\sigma^2$

- more variance in X $\Rightarrow$ smaller variance value $\Rightarrow$ better estimate of regression slope
- Rcode:
  - establish basics: n <- legnth(dataFrame); b1 <- cor(Y, X) * sd(Y)/sd(X); b0 <- mean(Y) - b1 * mean(x); e <- Y - b0 - b1 * X
  - variance: sigma <- sqrt(sum(e^2)/(n-2)); (squared sum X) ssx <- sum((x - mean(x))^2)
  - SE of coefficients: seB0 <- (1/n + mean(x)^2 / ssx)^0.5 * sigma; seB1 <- sigma/sqrt(ssx)
  - t-stats based on $H_o : \beta_{o/1} = 0$: tB0 <- b0/seB0; tB1 <- b1/seB1

- p-Values under $H_o(2 - sidedt - test)$: pB0 <- 2 * pt(abs(tB0), df=n-2, lower.tail=F); pB1 <- 2 * pt(abs(tb1), df=n-2, lower.tail=F)
  - when pB1=0, no linear relationship between X and Y
- shortcut to all calculations: summary(fit)$coefficients

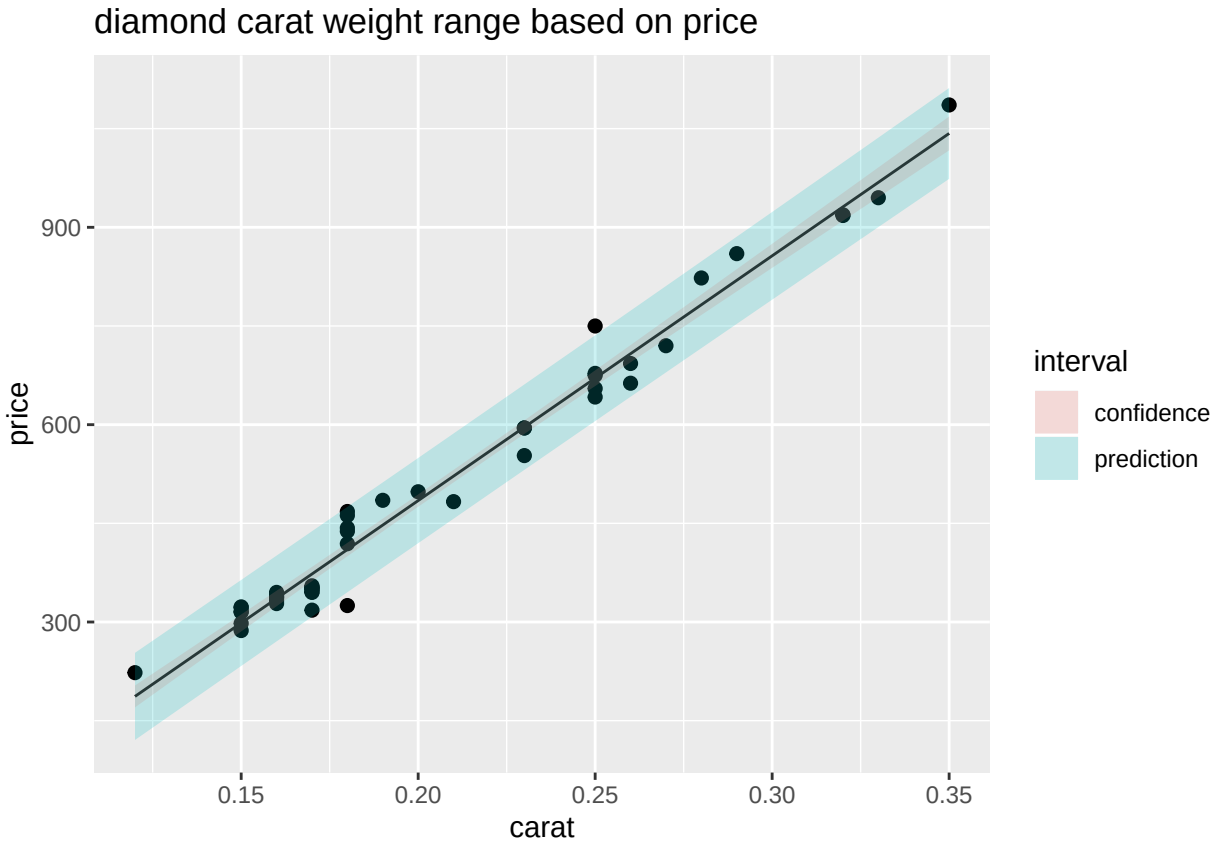
b) CI Rcode: fit <- lm(Y ~ X, data=dataFrame); sumCoef <- summary(fit)$coefficients

- intercept: sumCoef[1,1] + c(-1,1) x qt(0.975, df=fit$df) x sumCoef[1,2]
- slope: sumCoef[2,1] + c(-1,1) x qt(0.975, df=fit$df) x sumCoef[2,2]
- previous 2 only 1 tail test, for 2 tail test: confint(fit)

c) Prediction: SE needed to create prediction interval

- SE at x_o of regression: $\hat{\sigma}\sqrt{\frac{1}{n} + \frac{(x_o - \bar{X})^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}}$; of prediction: $\hat{\sigma}\sqrt{1 + \frac{1}{n} + \frac{(x_o - \bar{X})^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}}$
  - when $x_o = \bar{X}$, prediction error smallest; ↑ variability amoung X values ⇒ ↓ prediction error
  - both SE width most narrow at $\bar{X}$; ↑ n ⇒ CI for regression model gets smaller but predcition CI stay same cause must include inherent variability
  - when intercept/slope on regression line, then CI has 0 width
- Rcode for Y value interval based on mean $\bar{X}$: fit(Y ~ X, data=dataFrame)
  - condifenceCI: predict(fit, newdata=data.frame(X=mean(X)), interval="confidence"); predictionCI: predict(fit, newdata=data.frame(X=mean(X)), interval="prediction")

```
library(UsingR); data(diamond); library(ggplot2)
x <- diamond$carat; y <- diamond$price; fit <- lm(y ~ x)
#create new set of X values that want to predict the Y values for (in this case 10)
x2 <- data.frame(x = seq(min(x), max(x), length = 10))
#calc for X, the range in Y (ie for a given carat, on avg, whats the CI price)
p1 <- data.frame(predict(fit, newdata= x2,interval = "confidence"))
#calc interval around Y?
p2 <- data.frame(predict(fit, newdata = x2,interval = "prediction"))
#label each CI calc as confidence or prediction
p1$interval = "confidence"; p2$interval = "prediction"
#include predictor values from x2 in each of the CI variables (p1 & p2)
p1$x = x2$x; p2$x = x2$x
#combine the CI variables together and label the outcome variable column (y)
dat = rbind(p1, p2); names(dat)[1] = "y"
#build graph with regression line and x/y values used to calc CI
g <- ggplot(dat, aes(x = x, y = y)) + geom_line() + geom_point(data =
data.frame(x = x, y=y), aes(x = x, y = y), size = 2) + labs(x="carat",
y="price", title="diamond carat weight range based on price")
#add the 2 CI intervals
g <- g+ geom_ribbon(aes(ymin = lwr, ymax = upr, fill = interval), alpha = 0.2); g
```



Multivariable Regression

- relationship btw X and Y while accounting for other variables by removing the linear relationship of other variables from regressor and response by taking residuals
- in lm(Y ~ X, data=df) intercept coefficient of special regressor which has the same value (1) at every sample, this is included by default
- centering variable ⇒ elimnate intercept regressor (gaussian elimination) ⇒ replaces the variables by its residuals of its regression against 1
- $\sum_{i=1}^{n}(Y_i - X_{1i}\beta_1 - X_{2i}\beta_2)^2$: the regression estimate for each xVar slope is regression through origin with elimination of the other xVars from response and predictor
- 2 xVar linear regression: $Y_i = \beta_1 X_{1i} + \beta_2 X_{2i}$; $X_{2i} = 1$ intercept term; $\hat{\beta} = cor(X, Y)\frac{sd(Y)}{sd(X)}$
- Rcode: linear regression model ex: y= 1 + x1 + x2 + x3
  - remove x2 and x3 residuals from x1 and y: ey=resid(lm(y ~ x2 + x3)); ex=resid(lm(x1 ~ x2 + x3))
  - regression through origin fit with residuals: sum(ey * ex)/sum(ex^2)
  - shortway in R: coef(lm(ey ~ ex - 1))
  - full linear model with x2 and x3 residuals removed: coef(lm(y ~ x1 + x2 + x3))
- multivariant regresion coefficient is the expected change in the response per unit change in regressor, holding all other regressors fixed

- Simpson's Paradox: unadjusted and adjusted effects can be the reverse of each other, X/Y relationship changes when Z is taken into account

```
#necessary packages
library(ggplot2); library(gridExtra); library(knitr); library(kableExtra)
#simulate vars where x1 has a negative adjusted effect on Y and depends on x2
n=100; x2=1:n; x1=0.01*x2+runif(n, -0.1, 0.1); y=-x1 + x2 + rnorm(n, sd=0.01)
#coefficient tables unadjusted/adjusted
t1 <- summary(lm(y ~ x1))$coef; t2 <- summary(lm(y ~ x1 + x2))$coef
kable(t1, "latex", escape=F, booktabs=T, caption="Unadjusted model with x2 residual effect") %>%
kable_styling(latex_option=c("striped", "hold_position"), position="c")
```

Table 1: Unadjusted model with x2 residual effect

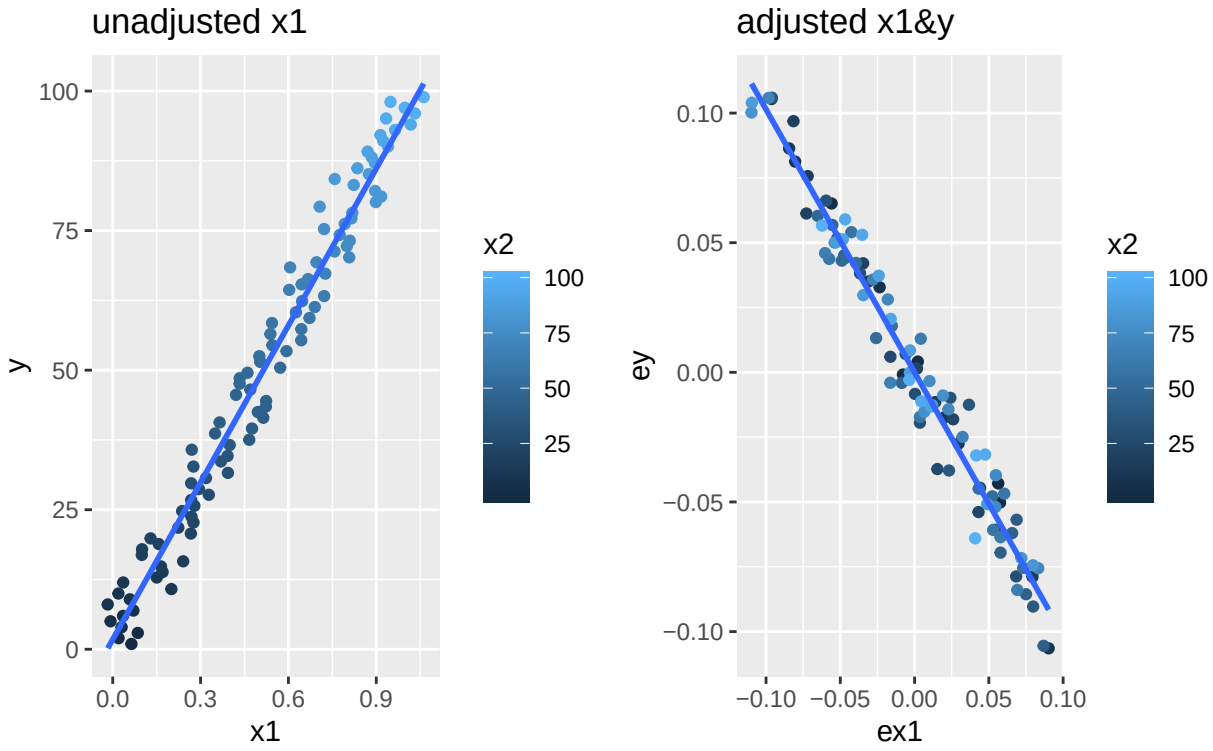
	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.764639	1.001111	1.762681	0.0810718
x1	93.733851	1.680818	55.766801	0.0000000

```
kable(t2, "latex", escape=F, booktabs=T, caption="Adjusted model") %>%
kable_styling(latex_option=c("striped", "hold_position"), position="c")
```

Table 2: Adjusted model

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.000781	0.0020123	0.3881186	0.6987787
x1	-1.014267	0.0192303	-52.7430947	0.0000000
x2	1.000173	0.0001999	5002.4132739	0.0000000

```
#df for graphs along with residuals for y and x1
dat=data.frame(y=y, x1=x1, x2=x2, ey=resid(lm(y~ x2)),ex1=resid(lm(x1 ~ x2)))
#plot unadjusted regression
p1 <- ggplot(dat, aes(x=x1, y=y, color=x2)) + geom_point() +
geom_smooth(method=lm, se=F) + labs(title="unadjusted x1")
#plot adjusted regression of x1 and y residuals after regressing x2
p2 <- ggplot(dat, aes(x=ex1, y=ey, color=x2)) + geom_point() +
geom_smooth(method=lm, se=F) + labs(title="adjusted x1&y")
grid.arrange(p1, p2, nrow=1)
```



simulated x1 and x2 have oppsoite behaviors. when regression model run without correction (incorporate x2), the slope was calculated incorrectly and it appeared that x1 and x2 had the same behavior. When x1 residuals were isolated and compared to y residuals, the true relationship between the 3 (y, x1, x2) became visible

- Rcode: lm(Y ~ . , data=df): "." will include all other variables in df (all Xs) in regression model; note: when there are a repeat of variables and there is a repeat of explanation of variations, R will print out NA as the coefficient for the repeated variable
- **in lm(Y ~ X1 + X2 + X3) etc., coefficients of the independent variables (X1/2/3) is the estimated change in the dependent variable (Y) when independent variable changes by +1% when all other independent variables are held constant**
- factor variables as predictors: to avoid repeat variation, R automatically assigns lowest factor as the dummy/reference variable for coefficient interpretation
  - relevel in R: newXvar <- relevel(xVvar, "defaultLevel")

lm(Y ~ X)

```
##                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
```

- intercept=sprayA mean; mean sprayB-F=intercept+respective coefficient
- inferential stats for sprayA is 1-sample t-test

- inferential stats for spraysB-F is 2 smaple t-test sprayA vs others respectively

lm(Y ~ X -1)

```
##         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
```

- when default intercept (-1) removed, then each coefficient represents mean of respective independent variables
- simi to performing 1-sample t-test for each variable

lm(Y ~ X1 + factor(X2))

- $Y_i = (\beta_o + \beta_2) + X_{i1}\beta_1 + \epsilon_i$ when $X_{i2} = 1 \Rightarrow \beta_2$ is the change in the intercept of y and x1 between factor 0 and 1 in x2; lm(gestation ~ age + factor(smokeBin, data=babies))

```
##                       Estimate Std. Error     t value   Pr(>|t|)
## (Intercept)       284.0276493 2.28045764 124.5485310 0.00000000
## age                -0.1559675 0.07869519  -1.9819185 0.04771011
## factor(smokeBin)1  -0.6404363 0.91541925  -0.6996098 0.48430296
```

- β_2 is factor(smokeBin)1; intercept=smokeBin0; smokeBin1=intercept+factor(smokeBin)1

lm(Y ~ X1 * factor(X2))

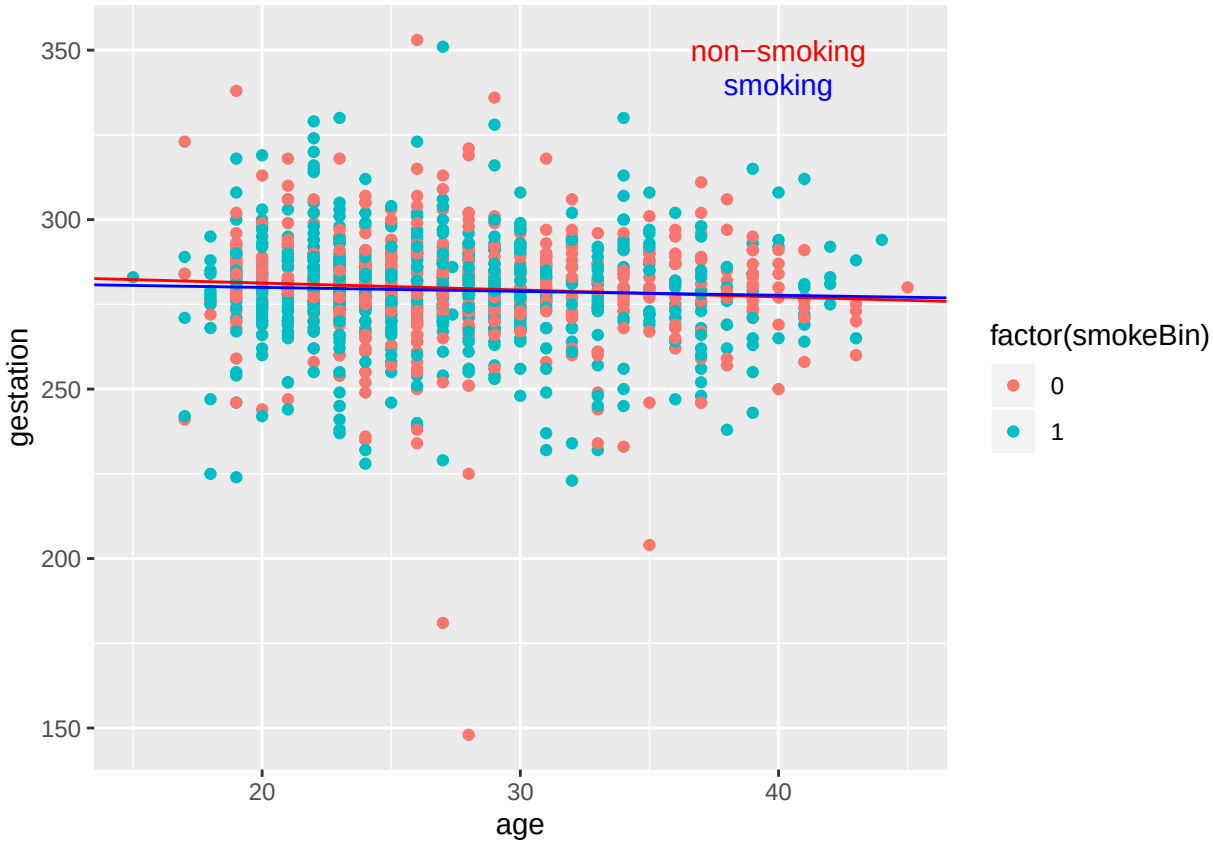
- $Y_i = (\beta_o + \beta_2) + X_{i1}(\beta_1 + \beta_3) + \epsilon_i$ when $X_{i2} = 1 \Rightarrow \beta_2$ is the change in intercept and β_3(interaction term) is the change in slope; lm(gestation ~ age * factor(smokeBin), data=babies)

```
##                           Estimate Std. Error    t value   Pr(>|t|)
## (Intercept)           285.35063066  3.3015741 86.4286611 0.00000000
## age                    -0.20382316  0.1168361 -1.7445219 0.08131749
## factor(smokeBin)1      -3.03546247  4.4169046 -0.6872375 0.49206242
## age:factor(smokeBin)1   0.08763613  0.1581073  0.5542827 0.57948598
```

- β_2 is factor(smokeBin)1; β_3 is age:factor(smokeBin)1
- gestation at smokeBin0=intercept; gestation at smokeBin1=intercept+factor(smokeBin)1; motherAge with smokeBin0 at gestation=age; motherAge with gestation at smokeBin1=age+age:factor(smokeBin)1

```r
#necessary packages
library(UsingR); data(babies); library(dplyr)
#cleanup df (smoke=1 non-smoke=0)
babies=mutate(babies, smokeBin= 1 * (smoke > 0))
#get rid of data outliers
babies$age[babies$age > 50] <- mean(babies$age)
```

```r
babies$gestation[babies$gestation > 500] <- mean(babies$gestation)
#create regression model
fit <- lm(gestation ~ age * factor(smokeBin), data=babies)
#create graph
g <- ggplot(babies, aes(x=age, y=gestation, color=factor(smokeBin))) + geom_point()
#non-smoke regression line
g <- g + geom_abline(intercept=coef(fit)[1], slope=coef(fit)[2], color="red") +
annotate("text", x=40, y=350, label="non-smoking", color="red")
#smoke regression line
g <- g + geom_abline(intercept=coef(fit)[1]+coef(fit)[3],
slope=coef(fit)[2]+coef(fit)[4], color="blue") + annotate("text", x=40, y=340,
label="smoking", color="blue"); g
```



Ajustment effects

- Marginal effect: difference between $\bar{Y}$ in regression models for residual variation (without xVar effect)
- regresion between 2 parallel regression lines: $Y = \beta_o + \beta_1 T + \beta_2 X + \epsilon$; β_1 is change in intercept btw 2 regression groups; β_2 common slope btw 2 regression groups
- high marginal effect & low β_1: value of xVar determines which regression group it corresponds to; usually occur in treatment studies
- regression btw 2 parallel regression lines w/some overlap: Simposon's paradox; β_1 and marginal effect have opposite results where high $\bar{Y}$ in β_1 is opposite in marginal effect and vice versa
- also possible: small marginal effect but high β_1 when xVar taken into account

- regression between 2 regression lines where they dont have a common β_2: $Y = \beta_o + \beta_1 T + \beta_3 T_x X + \epsilon$ where $\beta_3 T_x X$ is an interaction term; means no treatment effect

Revisiting Residuals

- plotting residuals for detecting systematic patterns and large outlying observations: plot(fit); 4 graphs in total
  - to print specific plots: plot(fit, which=1), change the num for which to specify other plots
  - residuals vs fitted: y=0 is delineated in red bc intercept is included and the residuals must sum upt to 0, values therefore lie above and below line
  - normal Q-Q: residual quantile vs normal/theoreitcal quantile; determine normality of error, points should fall along a diagnol line
- categorizing outliers:
  - leverage is how far a data point is from $\bar{X}$
  - influence is the degree of impact the outlier has on the regression model, determined by comparing the regression model with vs without outlier; this is also reflected when examining ratio of residuals with vs without outlier, ratio=0 not influential, ratio=1 influential
  - outlier outside Y range but within X range has little effect on fit model since thre are many X points to couteract its effect $\Rightarrow$ low leverage and influence
  - outlier outside X and Y range but lies on the regression line $\Rightarrow$ high leverage but low influence
  - outlier outside X range but within Y range doesn't conform to the linear model $\Rightarrow$ high leverage and influence
- influence measures Rcommands: (?influence.measure); best way to examine results is via plot(function(fit))
  - standardize residuals across experiments: threshold residuals using T-cutoff
    - rstandard(): $\frac{resids}{sd(resids)}$, not t-distributed, internally standardized; manual calculation: fit=lm(Y ~ X), sigma=sqrt(deviance(fit)/df.residual(fit)), rstd=resid(fit)/(sigma*sqrt(1-hatvalues(fit)))
    - rstudent(): like rstandard but ith data point deleted in calc to follow t-distribution, externally standardized; manual calculation: fit=lm(Y ~ X), fit1=lm(Y ~ X, data=df[-1,]), sigma=sqrt(deviance(fit1)/df.residual(fit1)), resid(fit)[1]/(sigma*sqrt(1-hatvalues(fit)[1]))
  - leverage measurement: measured by hat diagnols (hatvalues()), range 0-1 with closer to 1 value indicating greater potential leverage
  - influence measurement:
    - dffits(): change predicted response when ith point deleted from model; check influence in fitted values
    - dfbetas(): change ind coefficients when ith point deleted from model; dfbeta(fit)[,1] is intercept, dfbeta(fit)[,2] is slope
    - cooks.distance(): overall change coefficients when ith point deleted; manual method: fit=lm(Y ~ X), fit1=lm(Y ~ X, data=df[-1,]), dy=predict(fit1, df)-predict(fit, df), sigma=sqrt(deviance(fit)/df.residual(fit)), sum(dy^2)/(2*sigma^2)
  - residual/influence measure: method1: resid(fit)/(1-hatvalues(fit)); method2: fit=lm(Y ~ X), fit2=lm(Y ~ X, data=df[-1,]), resno=df[1, "y"] - prediction(fit2, df[1,]), 1-resid(fit)[1]/resno
  - PRESS residuals: compare diff btw response and predicted values at ith point when included vs not included in model

Model Selection

- parsimony is key: keep models as simple as possible for interpretation
- Rimsfeldian triplet: **1) known knowns (required regressors for inclusion in model)**, 2) known unknowns: (desired regressoes to include but not have ; proxy variable to make an educated guess on the uncollected variables), 3) unknown unknowns (unknown regressors that should be included in model)
- radomization for overcoming hurdles: unknown unknowns, treatment vs control, A/B testing
- Effects of variable exclusion/inclusion:
  - exclusion: when ommited variable is **not** orthogonal (stat independent) to target variable, will result in bias of target coefficient; when ommited variable is orthogonal to target variable then no impact on target variable coefficient
  - inclusion: when include variable that shouldn't of been included, actual standard error of regression variable abnormally increased; increasing the number of regressors increases R^2 $\Rightarrow$ adjusted R^2 better since it accounts for number of variables included in model, get via summary(fit)$adj.r.squared
- variance inflation: including new variables into model increases actual standard error of other regressors; orthogonal variable inclusion vs unorthogonal variable inclusion (respectively)

```
##         x1         x1         x1
## 0.02515625 0.02561336 0.02562730
```

```
##         x1         x1         x1
## 0.03058153 0.04010550 0.04971533
```

- when additional regressors are orthogonal to the target regressor, there is no variance inflation
- calc variance of X1 effect when include other variables: fit=lm(Y ~ X1); fit2=lm(Y ~ X1 + X2); fit3=lm(Y ~ X1 + X2 + X3) $\Rightarrow$ relative variance increase w/X2: summary(fit2)$cov.unscaled[2,2]/summary(fit1)$cov.un relative variance increase w/X2&X3: summary(fit3)$cov.unscales[2,2]/summary(fit)$cov.unscaled[2,2]
- variance inflation factor (VIF): ratio of theoretical estimates; variance with ith regressor inclusion divided by variance of including unorthogonal regressors; increase in variance for the ith regressor vs when its ideally orthogonal to other regressors; $sqrtVIF$ is increase in standard error inflation rahter than variance
  - calc in R: fit=lm(Y ~ ., data=df); library(car); vif(fit) OR sqrt(vif(fit)) for sd inflation; this calculates the amount of times the coefficient's variance is greater than if the variable wasn't correlated to other regressors
- Fitting model impacts: **underfit model**(omit necessary covariates, then variance estimate is biased); **correctly/overfit model**(include all necessary and some unecessary covariates, then variance estimate is unbiased BUT variance of the variance larger w/unecessary covariate inclusion)
- Nested model testing: model of interested nested and w/o many differentiating parameters; to determine which added variables contribute to the best model
  - Adding irrelevant regresors to model: as increasing number of regressors in model fit to number of df data points, deviance (residual sum of squares) reaches 0/reduces residual degrees of freedom $\Rightarrow$ anova quantifies significance of added regressors

```r
#load data
library(datasets); data(swiss)
#nest models of interest
fit1=lm(Fertility ~ Agriculture, data=swiss); fit3=update(fit1, Fertility ~ Agriculture
+ Examination + Education); fit5=update(fit1, Fertility ~ Agriculture + Examination +
Education + Catholic + Infant.Mortality)
#test model selection
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
```

- Res.DF= num of vals-num of parameters; RSS=residual sums of squares also deviance(); DF=excess DF to btw models AKA parameters added/subtracted; F= s-stat; p-values= test whether the new variables added to each model are above zero or not, indicates whether the added variables are significant to the model or not
- F calc manual: n=length(df); regfit1=2; regfit3=4; fit1=lm(Y ~ X1); fit3=lm(Y ~ X1 + X2 + X3); d=deviance(fit)/n-regfit3; n=(deviance(fit1)-deviance(fit3))/regfit3-regfit1; n/d
- p-value calc manual: pf(n/d, 2, 43, lower.tail=FALSE); this test can be affected by residual abnormality, conduct Shapairo-Wilk test to test normality: shapiro.test(fit$residuals), if th p-value of this test fails to reject normality (greater than 0.05) then anova confidence of variance is valid

Generalized Linea Models

- family of models including linear models that overcomes some of the obsticles in linear models; components are **exponential family**(response), **systematic component**(linear predictor), **link function**(connex mean response to linear predictor)
  - linear predictor in linear/binomial/poisson: $\eta_i = \sum_{k=1}^{p} X_{ki}\beta_k$
  - linear function: $g(\mu) = \eta \Rightarrow g(\mu) = \mu$ (linear); $g(\mu) = \eta = log(\frac{\mu}{1-\mu})$ (binomial); $g(\mu) = \eta = log(\mu)$ (poisson)
  - estimates and SE obtained numerically and based on large sample size: $0 = \sum_{i=1}^{n} \frac{(Y_i-\mu_i)}{var(Y_i)} W_i$
  - variances: $var(Y_i) = \sigma^2$ (linear); $var(Y_i) = \mu_i(1-\mu_i)$ (binomial); $var(Y_i) = \mu_i$ (poisson); quasi-likelihood used for more felxible variance model, in R: quasipoisson() and quasibinomial()
- Binary GLMs: model outcomes that can only take 2 values; can't use linear model to formualte basic assumptions of the data; best way is to transform the data into probability and calcualte the log of odds (logit)
  - odds: function of proportion; $p = probability$ and $o = odds \Rightarrow p = \frac{o}{(1+o)}$ and $o = \frac{p}{(1-p)}$

- probs are 0 to 1; odds are 1 to ∞
- logit: log of odds; logit: $g = logit(p) = log(\frac{p}{(1-p)})$; inverse logit to calc prob: $expit(g) = \frac{e^g}{(1+e^{-g}} = \frac{1}{1+e^{-g}} = p$
- logistic regression model: $logit(p_i) = \beta_o + \beta_1 x_i$; also $P(Y_i = 1|X_i = x_i, \beta_o, \beta_1) = p_i = \frac{exp(\beta_o+\beta_1 x_i)}{1+exp(\beta_o+\beta_1 x_i}$
- run binary regression model in R: logGML=glm(df$Y$ ~ df$X$, family="binomial", data=df); predict values based on glm model: lodds=predict(logGML, data.frame=values) ⇒ given as log values so convert to prob ⇒ exp(lodds)/(1+exp(lodds))
- to incorporate total count: logGML=glm(cbind(Y, yTotal - Y) ~ X, family="binomial", data=df)
- to plot probability of outcome: plot(df$Outcome, logGLM$fitted)
- exponentiate results to interpret: exp(logGLM$coeff); exp(confit(logGLM))
- calculate CI probabbility: exp(confit(logGML))
- change in residual variation due to a variable determines if variable is an important predictor or not, in GLMs, deviance does this. In anova(logGML), deviance residual value is the difference btw deviance of model (intercept+slope) and model with only intercept. This value s centrally chi-squared distributed with 1 degree of freedom. $H_o : \beta_1 = 0$, for rejection the deviance of residual value should be greater than qchisq(0.95, 1)

- Poisson GLMs: model outcomes that model rate data when the upper bound is known; considers poisson distribution; proportions treated as rates when n is large and success prob is small
  - poisson distribution info: mean is $E[X] = t\lambda$; variance is $Var(X) = t\lambda$; as $t\lambda$ gets larger poisson reaches normal distribution
  - $log(\lambda)$ assumed to be a linear function of the predictors/data; $log(\lambda) = \beta_o + \beta_1 * rate$; $\lambda = exp(\beta_o)exp(\beta_1)^r ate$; exp($\beta_1$) estimates percentage change in rate
  - issues/solutions w/using linear model: i) response is non-negative ⇒ natural log/sqrt/cubeRt transformation of outcome, ii) gaussian error isn't a proper approximation for small counts
  - model for non-zero data: $log(\mu_i) = \beta_o + \beta_1 x_i$; e^{β_o} is expected mean of outcome when $x_i = 0$; e^{β_1} is the relative change in outcome for unit change in regressor
  - R: lm(I(log(Y + 1)) ~ X) note: + 1 when data contains zeros; logGLM=glm(Y ~ X, family="poisson", data=df)
  - when mean not equal to variance: assume quasi-Poisson model where variance is a constant multiple of mean; logGLM=glm(Y ~ X, family="quasipoisson", data=df)
  - fit rates/proportions in Poisson model: $log(\frac{\mu_i}{Y_i}) = \beta_o + \beta_1 x_i$ where Y_i is the num of hits and μ_i is expected count, time variable would be included as an offset; logGLM=glm(Y ~ X, offset=log(timeVar + 1), family="poisson", data=df)