Regression Models week 2

- add gausians erros

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

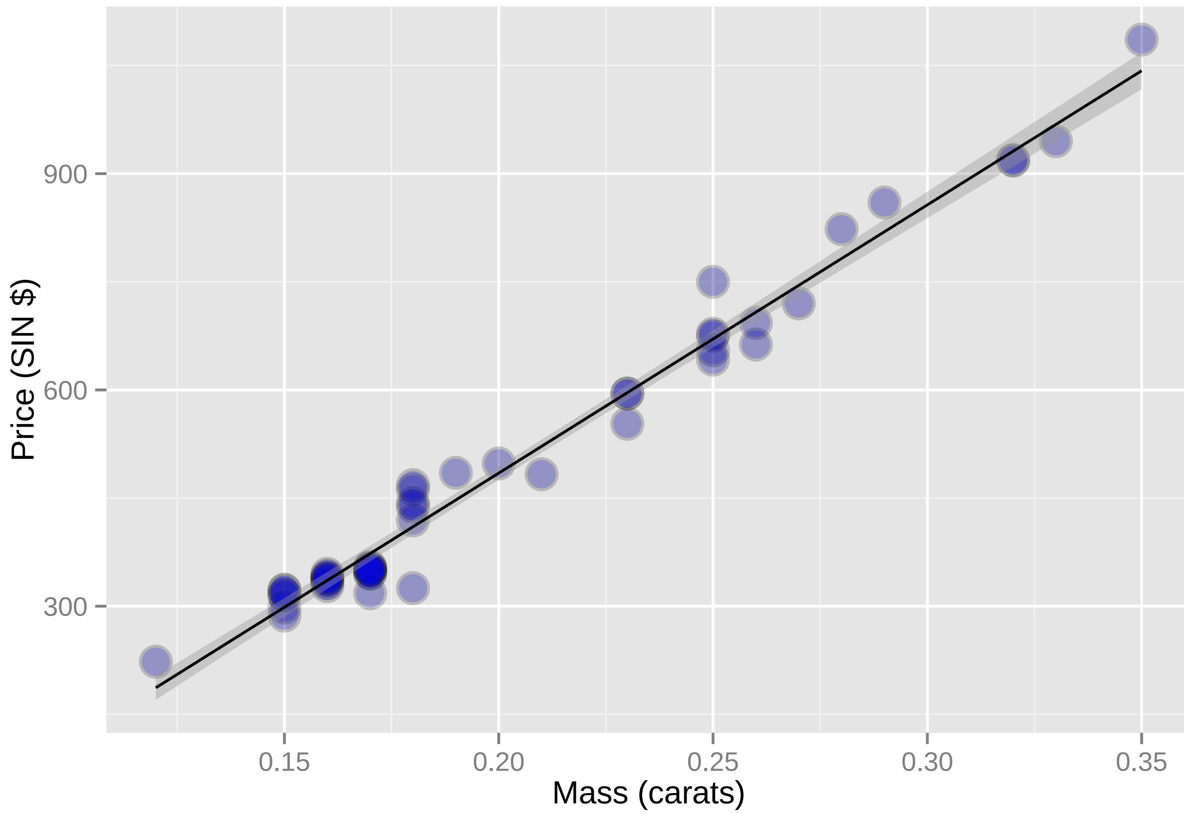
Example

using diamond prices

```
library(UsingR)
```

```
## Loading required package: MASS
## Loading required package: HistData
## Loading required package: Hmisc
## Loading required package: grid
## Loading required package: lattice
## Loading required package: survival
## Loading required package: Formula
## Loading required package: ggplot2
##
## Attaching package: 'Hmisc'
##
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
##
##
## Attaching package: 'UsingR'
##
## The following object is masked from 'package:ggplot2':
##
##     movies
##
## The following object is masked from 'package:survival':
##
##     cancer
```

```
data(diamond)
library(ggplot2)
g = ggplot(diamond, aes(x = carat, y = price))
g = g + xlab("Mass (carats)")
g = g + ylab("Price (SIN $)")
g = g + geom_point(size = 6, colour = "black", alpha = 0.2)
g = g + geom_point(size = 5, colour = "blue", alpha = 0.2)
g = g + geom_smooth(method = "lm", colour = "black")
g
```



Fitting the linear regression model

```
fit <- lm(price ~ carat, data = diamond)
summary(fit)
```

```
##
## Call:
## lm(formula = price ~ carat, data = diamond)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -85.159 -21.448  -0.869  18.972  79.370
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -259.63      17.32  -14.99   <2e-16 ***
## carat        3721.02      81.79   45.50   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 31.84 on 46 degrees of freedom
## Multiple R-squared:  0.9783, Adjusted R-squared:  0.9778
## F-statistic:  2070 on 1 and 46 DF,  p-value: < 2.2e-16
```

```r
coef(fit)
```

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

getting a better intercept

```r
fit2 <- lm(price ~ I(carat - mean(carat)), data = diamond)
coef(fit2)
```

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

- This gets the average price of the diamond for the mean size (0.2 carats)

```r
fit3 <- lm(price ~ I(carat*10), data = diamond)
coef(fit3)
```

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

Predicting the price of a diamond

```r
newx <- c(0.16, 0.27, 0.34)
coef(fit)[1] + coef(fit)[2] * newx
```

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

```r
predict(fit, newdata = data.frame(carat = newx))
```

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

- predict(fit) returns from the original diamond$carat data

Residuals

- residual variation: variation around the regression line
- residuals: the errors from the regression line

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

Least squares minimized $\sum_{i=1}^{n} e_i^2$

Properties of residuals

- $E[e_i] = 0$
- if an intercept is included, $\sum_{i=1}^{n} e_i = 0$
- if a regressor variable X is included, $\sum_{i=1}^{n} e_i X_i = 0$

```
data(diamond)
y <- diamond$price;
x <- diamond$carat
n <- length(y)
fit <- lm(y~x)
e <- resid(fit)
yhat <- predict(fit)
max(abs(e - (y - yhat)))
```

```
## [1] 9.485746e-13
```

```
max(abs(e - (y - coef(fit)[1] - coef(fit)[2] * x)))
```

```
## [1] 9.485746e-13
```

```
sum(e)
```

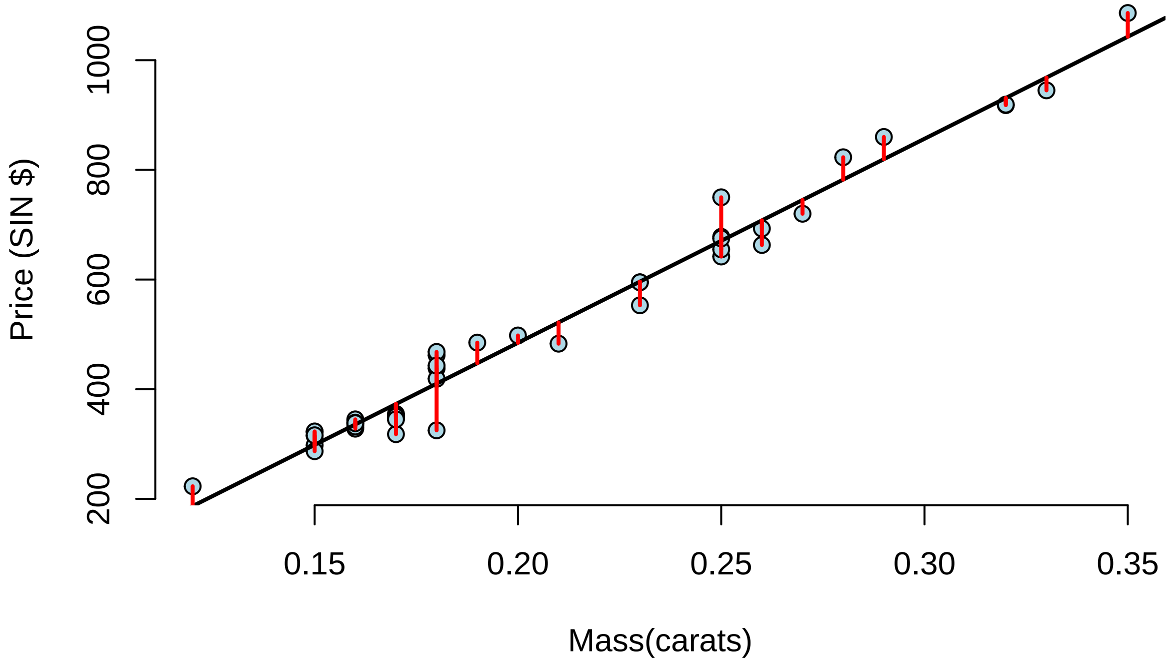
```
## [1] -1.865175e-14
```

```
sum(e*x)
```

```
## [1] 6.959711e-15
```

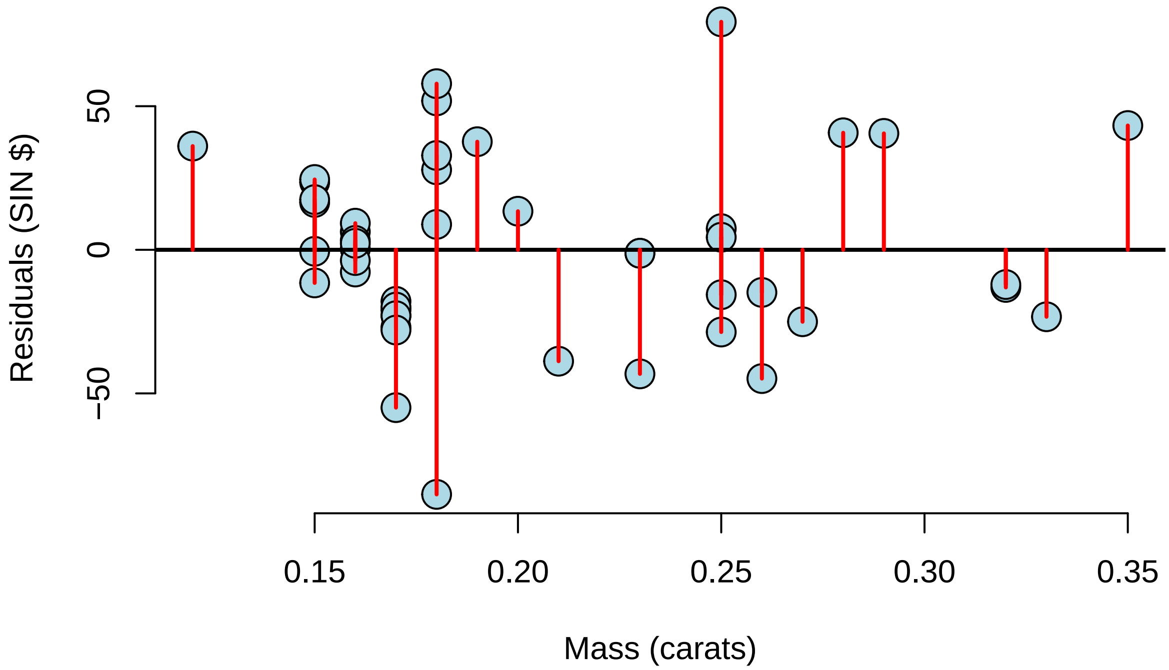
Plot

```
plot(diamond$carat, diamond$price,
     xlab = "Mass(carats)",
     ylab = "Price (SIN $)",
     bg = "lightblue",
     col = "black",
     cex = 1.1,
     pch = 21,
     frame = F)
abline(fit, lwd = 2)
for(i in 1:n)
  lines(c(x[i], x[i]), c(y[i],yhat[i]), col = "red", lwd = 2)
```



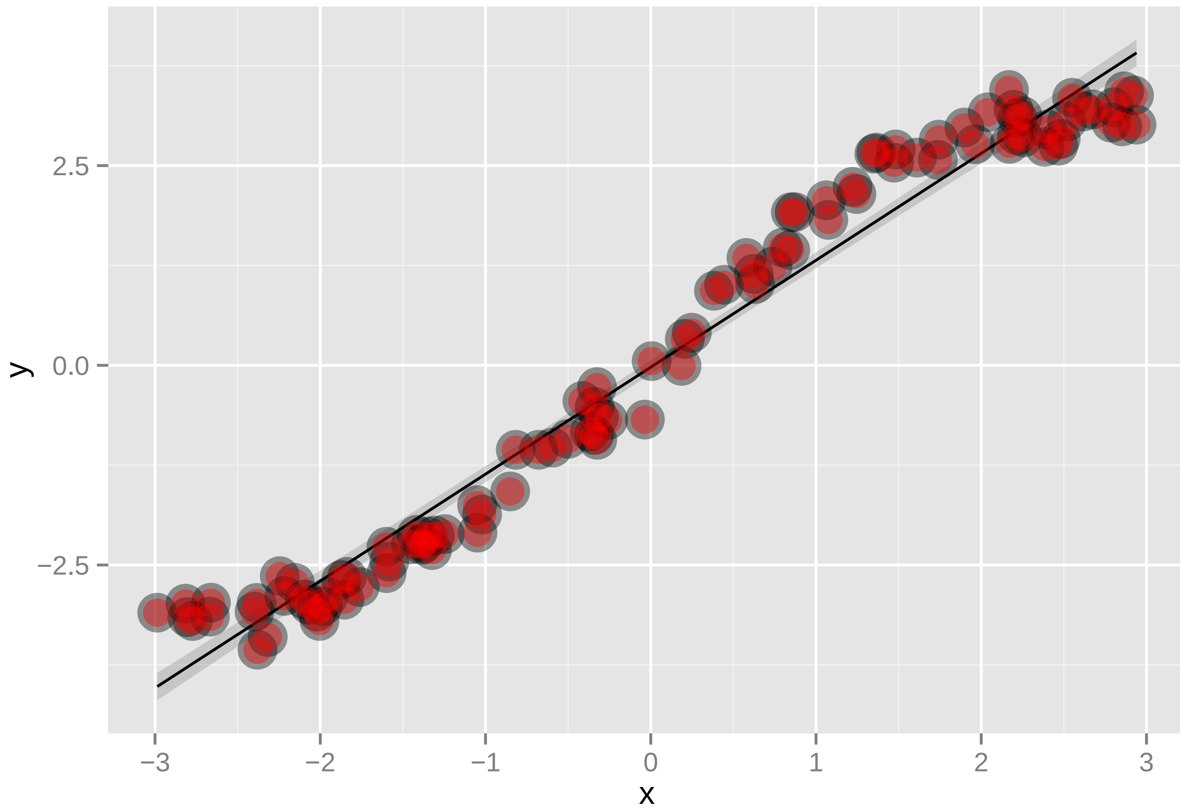
Residuals on the y axis

```
plot(x, e,
     xlab = "Mass (carats)",
     ylab = "Residuals (SIN $)",
     bg = "lightblue",
     col = "black", cex = 2, pch = 21, frame = F)
abline(h = 0, lwd = 2)
for(i in 1:n)
  lines(c(x[i],x[i]), c(e[i], 0), col = "red", lwd = 2)
```



non-linear data

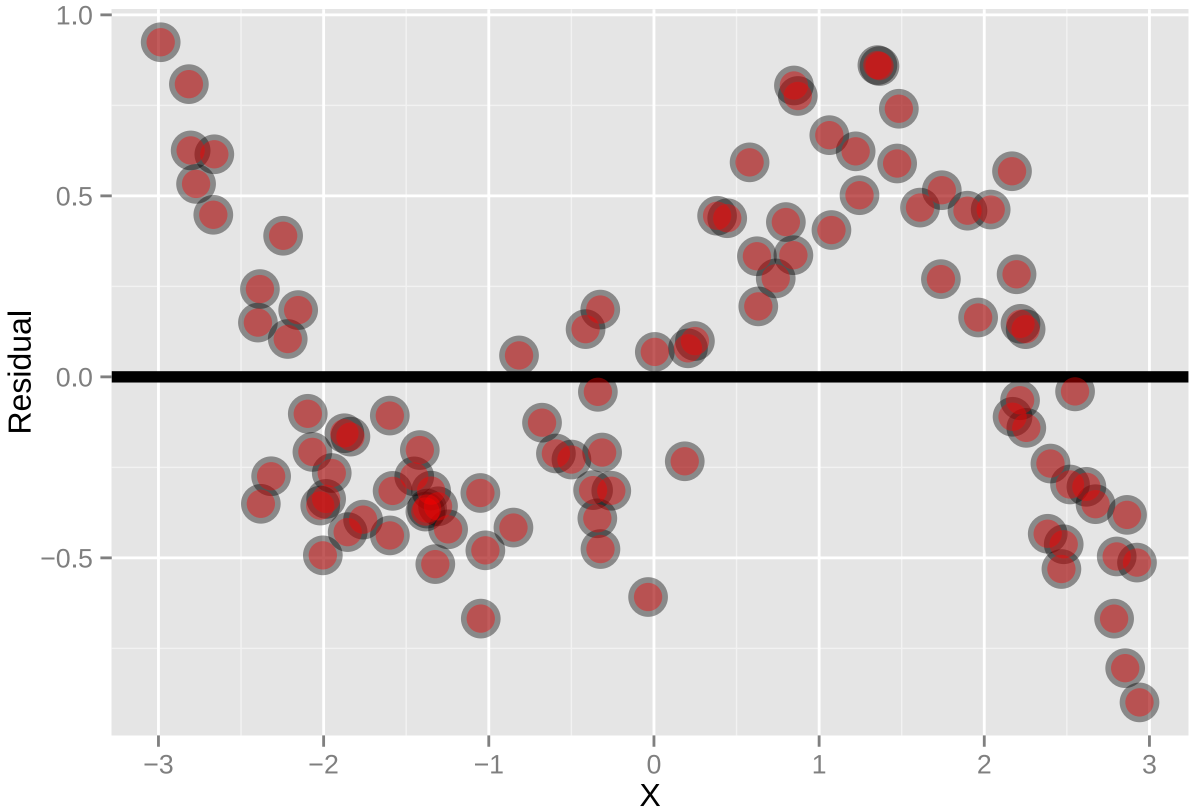
```
x = runif(100, -3, 3)
y = x + sin(x) + rnorm(100, sd = .2)
library(ggplot2)
g = ggplot(data.frame(x = x, y = y), aes(x = x, y = y))
g  = g + geom_smooth(method = "lm", colour = "black")
g = g + geom_point(size = 7, colour = "black", alpha = 0.4)
g = g + geom_point(size = 5, colour = "red", alpha = 0.4)
g
```



- incorrect models (in this case linear) are also important.
- however, we may get meaningful insight by looking at the residuals

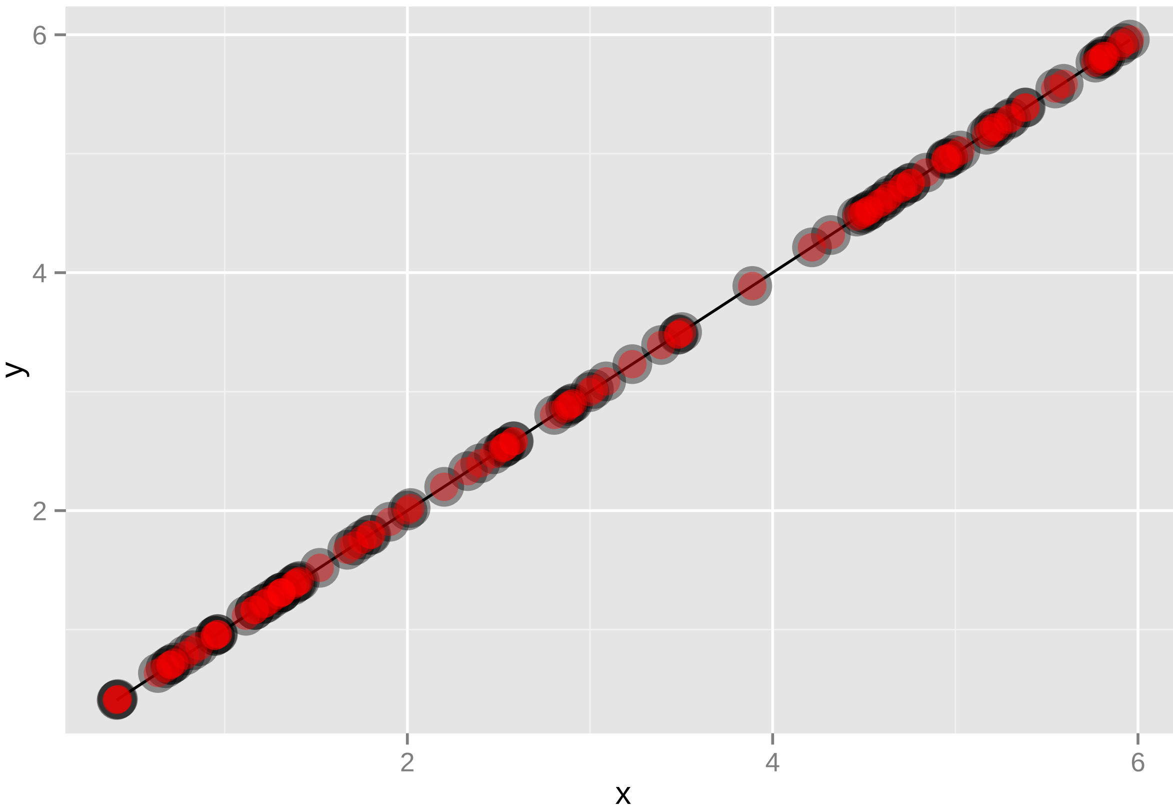
Residual

```
g = ggplot(data.frame(x  =x , y = resid(lm(y~x))), aes(x = x, y = y))
g = g + geom_hline(yintercept = 0, size = 2)
g = g + geom_point(size = 7, colour = "black", alpha = 0.4)
g = g + geom_point(size = 5, colour = "red", alpha = 0.4)
g = g + xlab("X") + ylab("Residual")
g
```



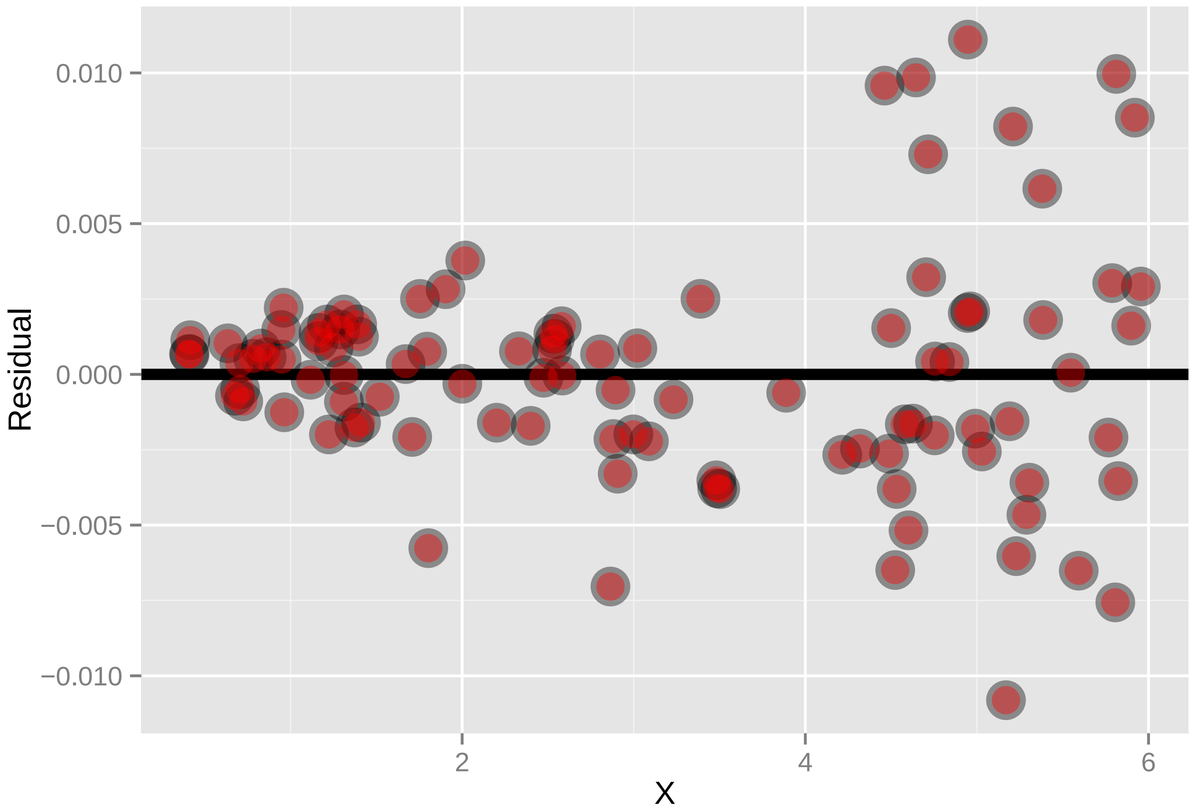
Heteroskedasticity

```
x <- runif(100,0,6)
y <- x + rnorm(100, mean = 0, sd = .001*x)
g = ggplot(data.frame(x = x, y = y), aes(x = x, y = y))
g = g + geom_smooth(method = "lm", colour = "black")
g = g + geom_point(size = 7, colour = "black", alpha = 0.4)
g  = g + geom_point(size = 5, colour = "red", alpha = 0.4)
g
```



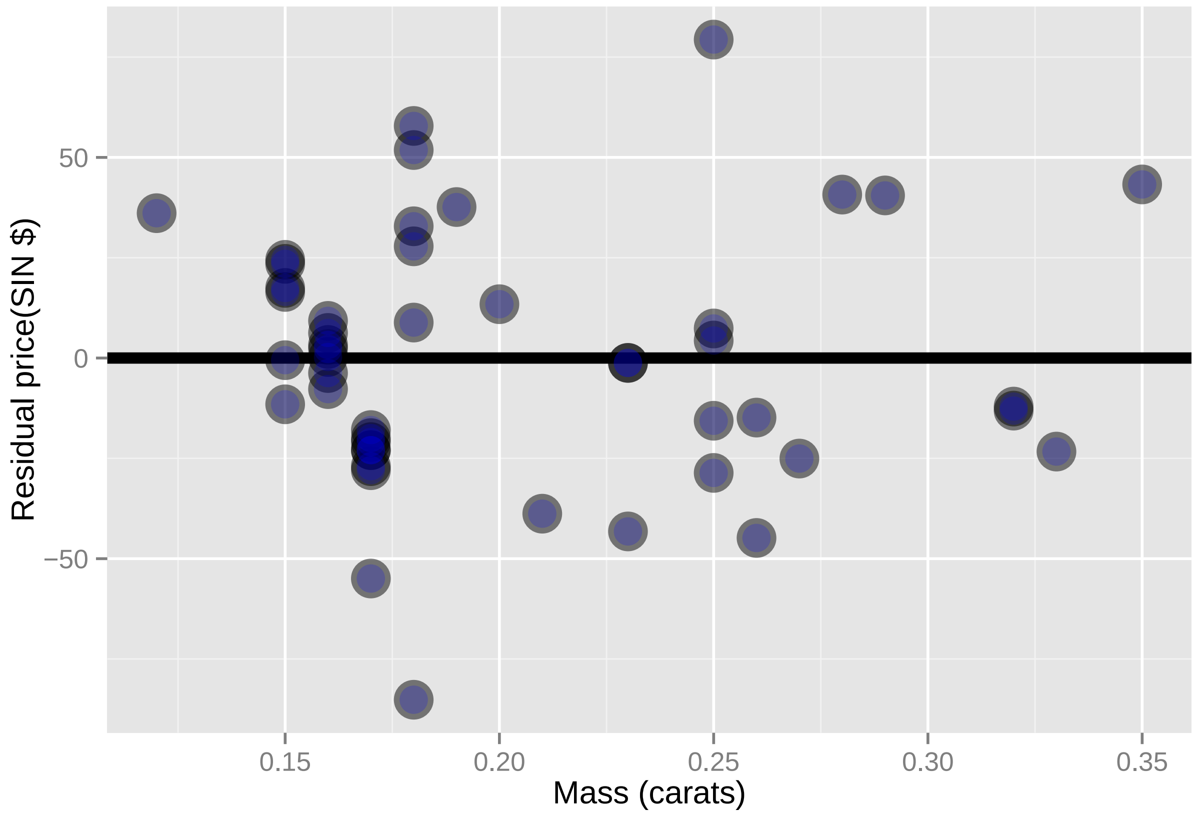
Getting rid of the blank space

```
g = ggplot(data.frame(x = x, y = resid(lm(y~x))), aes(x = x, y = y))
g = g + geom_hline(yintercept  = 0, size = 2)
g = g + geom_point(size = 7, colour = "black", alpha = 0.4)
g = g + geom_point(size = 5, colour = "red", alpha = 0.4)
g = g + xlab("X") + ylab("Residual")
g
```



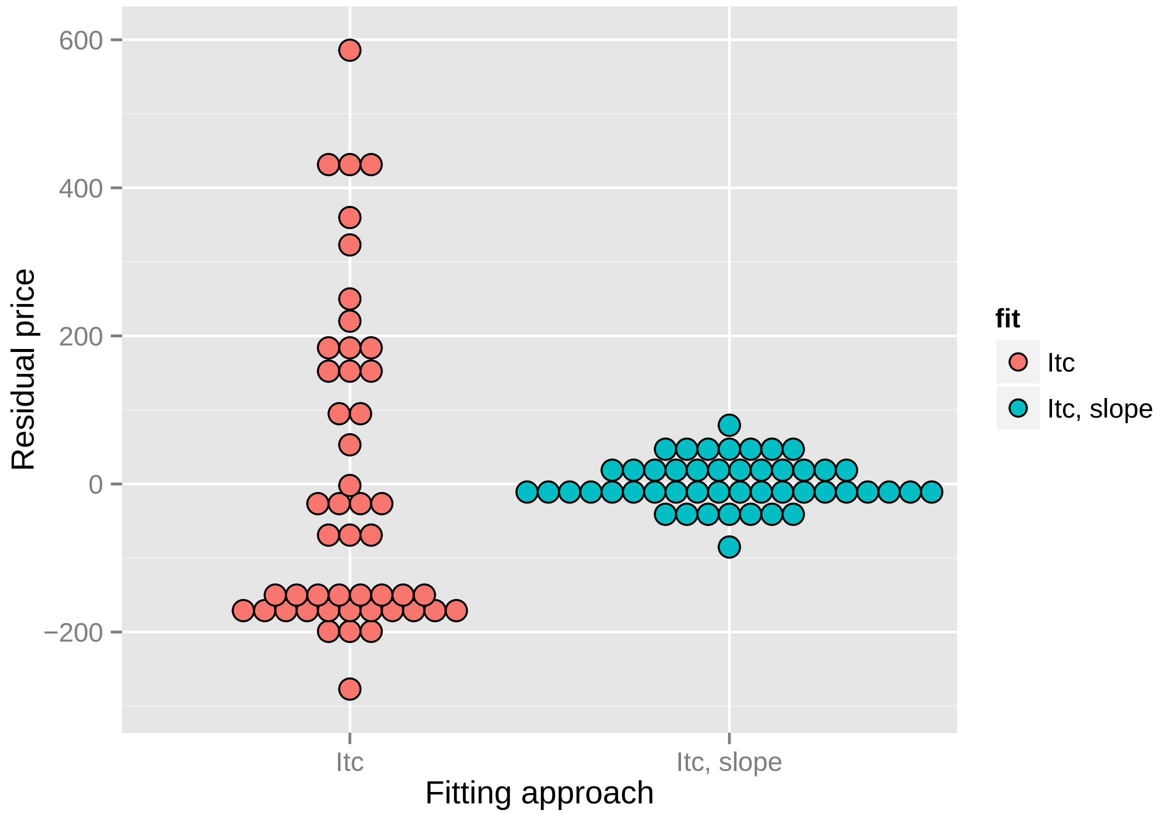
- this trend that the error gets bigger is called heteroskedasticity

```
diamond$e <- resid(lm(price ~ carat, data = diamond))
g = ggplot(diamond, aes(x = carat, y = e))
g = g + xlab("Mass (carats)")
g = g + ylab("Residual price(SIN $)")
g = g + geom_hline(yintercept = 0, size = 2)
g = g + geom_point(size = 7, colour = "black", alpha = 0.5)
g = g + geom_point(size = 5, colour = "blue", alpha = 0.2)
g
```



Diamond data residual plot

```
e = c(resid(lm(price ~ 1, data = diamond)),    ## variation around the average price
      resid(lm(price~carat, data = diamond)))  ## variation around carat
fit = factor(c(rep("Itc", nrow(diamond)),
               rep("Itc, slope", nrow(diamond))))
g = ggplot(data.frame(e = e, fit = fit), aes(y = e, x = fit, fill = fit))
g  = g + geom_dotplot(binaxis = "y", size = 2, stackdir = "center")
g = g + xlab("Fitting approach")
g = g + ylab("Residual price")
g
```



- most of the variation can be explained by the regression to carat

Residual Variance

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

- n-2 as this loses 2 degress of freedom, sum(e) = 0, sum(e*x) = 0 $$

```
y <- diamond$price
x <- diamond$carat
n <- length(y)
fit <- lm(y ~ x)
summary(fit)$sigma
```

```
## [1] 31.84052
```

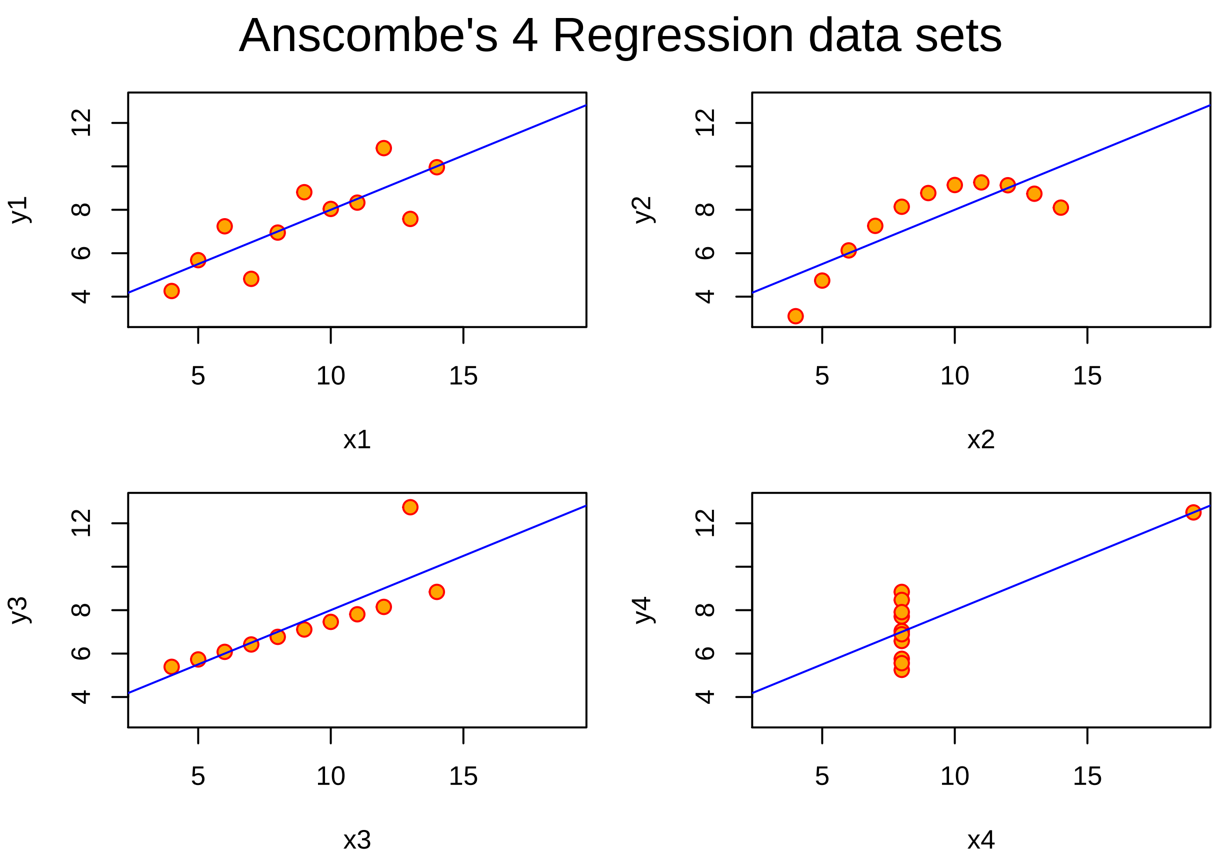
```
sqrt(sum(resid(fit)^2) / (n-2))
```

```
## [1] 31.84052
```

```
data(anscombe)
example(anscombe)
```

```
##
## anscmb> require(stats); require(graphics)
##
## anscmb> summary(anscombe)
##        x1             x2             x3             x4    
##  Min.   : 4.0   Min.   : 4.0   Min.   : 4.0   Min.   : 8  
##  1st Qu.: 6.5   1st Qu.: 6.5   1st Qu.: 6.5   1st Qu.: 8  
##  Median : 9.0   Median : 9.0   Median : 9.0   Median : 8  
##  Mean   : 9.0   Mean   : 9.0   Mean   : 9.0   Mean   : 9  
##  3rd Qu.:11.5   3rd Qu.:11.5   3rd Qu.:11.5   3rd Qu.: 8  
##  Max.   :14.0   Max.   :14.0   Max.   :14.0   Max.   :19  
##        y1               y2               y3              y4        
##  Min.   : 4.260   Min.   :3.100   Min.   : 5.39   Min.   : 5.250  
##  1st Qu.: 6.315   1st Qu.:6.695   1st Qu.: 6.25   1st Qu.: 6.170  
##  Median : 7.580   Median :8.140   Median : 7.11   Median : 7.040  
##  Mean   : 7.501   Mean   :7.501   Mean   : 7.50   Mean   : 7.501  
##  3rd Qu.: 8.570   3rd Qu.:8.950   3rd Qu.: 7.98   3rd Qu.: 8.190  
##  Max.   :10.840   Max.   :9.260   Max.   :12.74   Max.   :12.500  
##
## anscmb> ##-- now some "magic" to do the 4 regressions in a loop:
## anscmb> ff <- y ~ x
##
## anscmb> mods <- setNames(as.list(1:4), paste0("lm", 1:4))
##
## anscmb> for(i in 1:4) {
## anscmb+   ff[2:3] <- lapply(paste0(c("y","x"), i), as.name)
## anscmb+   ## or   ff[[2]] <- as.name(paste0("y", i))
## anscmb+   ##      ff[[3]] <- as.name(paste0("x", i))
## anscmb+   mods[[i]] <- lmi <- lm(ff, data = anscombe)
## anscmb+   print(anova(lmi))
## anscmb+ }
## Analysis of Variance Table
##
## Response: y1
##           Df Sum Sq Mean Sq F value  Pr(>F)   
## x1         1 27.510 27.5100   17.99 0.00217 **
## Residuals  9 13.763  1.5292                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: y2
##           Df Sum Sq Mean Sq F value   Pr(>F)   
## x2         1 27.500 27.5000  17.966 0.002179 **
## Residuals  9 13.776  1.5307                    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: y3
##           Df Sum Sq Mean Sq F value   Pr(>F)   
## x3         1 27.470 27.4700  17.972 0.002176 **
## Residuals  9 13.756  1.5285                    
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: y4
##           Df Sum Sq Mean Sq F value   Pr(>F)
## x4         1 27.490 27.4900  18.003 0.002165 **
## Residuals  9 13.742  1.5269
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## anscmb> ## See how close they are (numerically!)
## anscmb> sapply(mods, coef)
##                   lm1      lm2       lm3       lm4
## (Intercept) 3.0000909 3.000909 3.0024545 3.0017273
## x1          0.5000909 0.500000 0.4997273 0.4999091
##
## anscmb> lapply(mods, function(fm) coef(summary(fm)))
## $lm1
##              Estimate Std. Error  t value    Pr(>|t|)
## (Intercept) 3.0000909  1.1247468 2.667348 0.025734051
## x1          0.5000909  0.1179055 4.241455 0.002169629
##
## $lm2
##             Estimate Std. Error  t value    Pr(>|t|)
## (Intercept) 3.000909  1.1253024 2.666758 0.025758941
## x2          0.500000  0.1179637 4.238590 0.002178816
##
## $lm3
##              Estimate Std. Error  t value    Pr(>|t|)
## (Intercept) 3.0024545  1.1244812 2.670080 0.025619109
## x3          0.4997273  0.1178777 4.239372 0.002176305
##
## $lm4
##              Estimate Std. Error  t value    Pr(>|t|)
## (Intercept) 3.0017273  1.1239211 2.670763 0.025590425
## x4          0.4999091  0.1178189 4.243028 0.002164602
##
##
## anscmb> ## Now, do what you should have done in the first place: PLOTS
## anscmb> op <- par(mfrow = c(2, 2), mar = 0.1+c(4,4,1,1), oma =  c(0, 0, 2, 0))
##
## anscmb> for(i in 1:4) {
## anscmb+   ff[2:3] <- lapply(paste0(c("y","x"), i), as.name)
## anscmb+   plot(ff, data = anscombe, col = "red", pch = 21, bg = "orange", cex = 1.2,
## anscmb+        xlim = c(3, 19), ylim = c(3, 13))
## anscmb+   abline(mods[[i]], col = "blue")
## anscmb+ }
```




```
##
## anscmb> mtext("Anscombe's 4 Regression data sets", outer = TRUE, cex = 1.5)
##
## anscmb> par(op)
```

Inference in regression

- beta's can be predicted by the data
- $\frac{\hat{\beta}-\beta}{\sigma}$ follow a t distruibution

$$\sigma^2_{\hat{\beta}_1} = Var(\hat{\beta}_1) = \sigma^2 / \sum_{i=1}^{n} (X_i - \overline{X})^2$$

- you want more variation in the predictor

$$\sigma^2_{\hat{\beta}_0} = Var(\hat{\beta}_0) = \left(\frac{1}{n} + \frac{\overline{X}^2}{\sum_{i=1}^{n} (X_i - \overline{X})^2} \sigma^2 \right)$$

$$\frac{\hat{\beta}_j - \beta_j}{\hat{\sigma}_{\hat{\beta}_j}}$$

- this follows a t distruibtuion, df = n-2

```r
library(UsingR)
data(diamond)
y <- diamond$price
x <- diamond$carat
n <- length(y)
beta1 <- cor(x,y) * sd(y) / sd(x)
beta0 <- mean(y) - beta1 * mean(x)
e <- y - beta0 - beta1 * x
sigma <- sqrt(sum(e^2)/(n-2))
ssx <- sum((x - mean(x))^2)
seBeta0 <- (1 / n + mean(x)^2 / ssx) ^.5 * sigma
seBeta1 <- sigma/ sqrt(ssx)
tBeta0 <- beta0 / seBeta0
tBeta1 <- beta1 / seBeta1
pBeta0 <- 2 * pt(abs(tBeta0), df = n-2, lower.tail = F)
pBeta1 <- 2 * pt(abs(tBeta1), df = n-2, lower.tail = F)
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")
```

Easy way

```r
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
```

```r
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
```

coefs

```r
sumCoef <- summary(fit)$coefficients
sumCoef[1,1] + c(-1,1) * qt(.975, df = fit$df) * sumCoef[1,2]
```

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

```r
(sumCoef[2,1] + c(-1,1) * qt(.975, df = fit$df) * sumCoef[2,2]) / 10
```

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

- 95% confident that 0.1 carats increase will result in 355 - 388 SID increase

Prediction

- $\hat{\beta}_0 + \hat{\beta}_1 x_0$ should make sense
- line at x0,

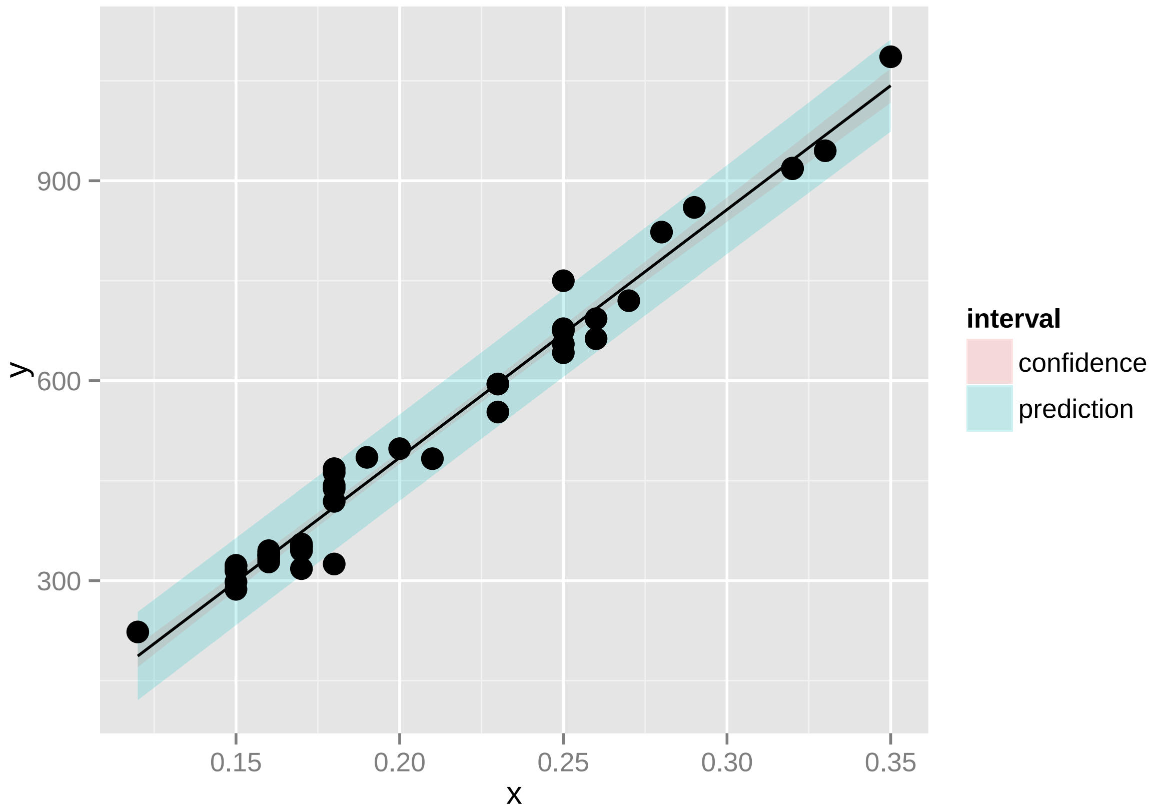
$$\hat{\sigma}\sqrt{\frac{1}{n} + \frac{(x_0 - \overline{X})^2}{\sum_{i=1}^{n}(X_i - \overline{X})^2}}$$

- prediction interval se at x_0,

$$\hat{\sigma}\sqrt{1 + \frac{1}{n} + \frac{(x_0 - \overline{X})^2}{\sum_{i=1}^{n}(X_i - \overline{X})^2}}$$

```
library(ggplot2)
newx = data.frame(x = seq(min(x), max(x), length = 100))
p1 = data.frame(predict(fit, newdata = newx, interval = ("confidence")))
p2 = data.frame(predict(fit, newdata = newx, interval = ("prediction")))
p1$interval = "confidence"
p2$interval = "prediction"
p1$x = newx$x
p2$x = newx$x
dat = rbind(p1, p2)
names(dat)[1] = "y"

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 = g + geom_point(data = data.frame(x = x, y = y), aes(x = x, y = y), size = 4)
g
```



900
600
300
y
0.15
0.20
0.25
0.30
0.35
x
interval
confidence
prediction