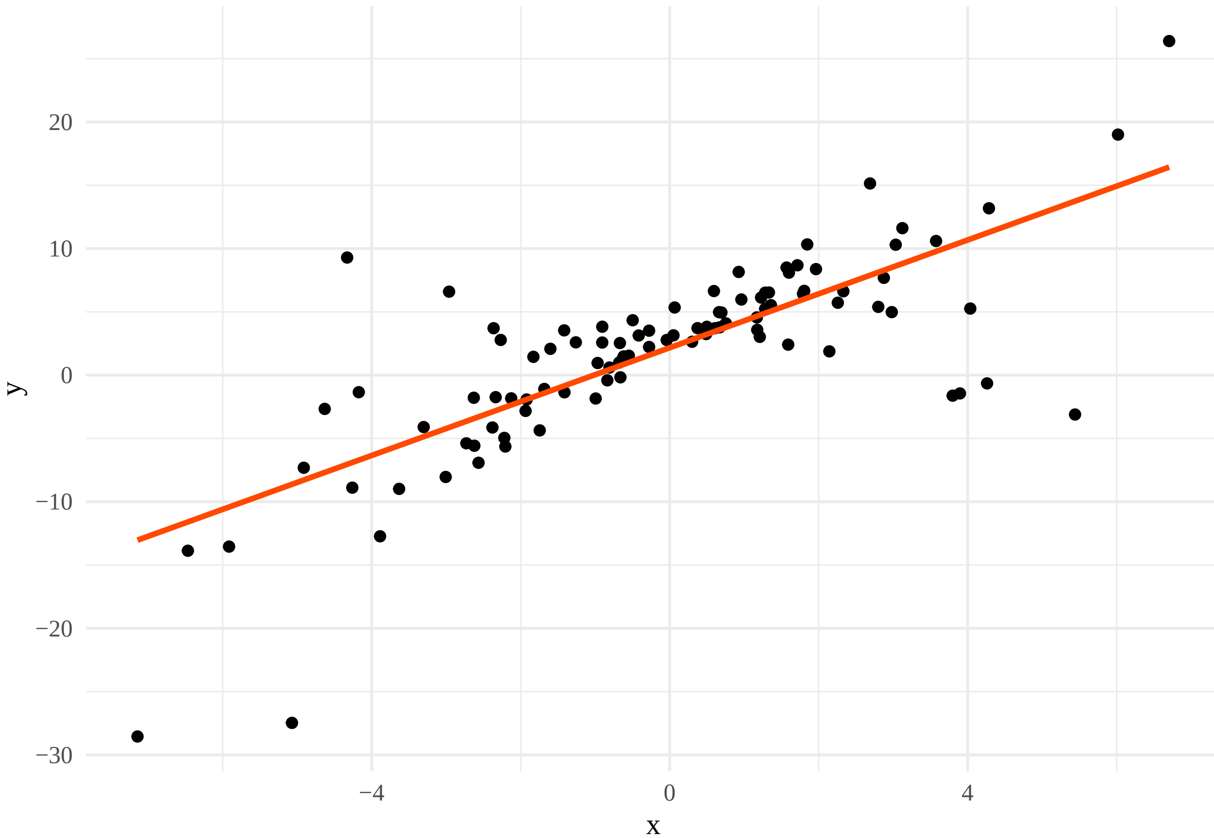
Chapter 10 of AEPV

DJM, Revised NAK

26 February 2019

Assumptions on Residuals




```
##             Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 2.164973  0.4980974  4.346484 3.383817e-05
## x           2.127688  0.1853945 11.476538 7.852372e-20
```

Questions: Whats wrong here? Where can we most reliably estimate the linear function? Where is the most unreliable area?

Standard Errors in Heteroskedacity

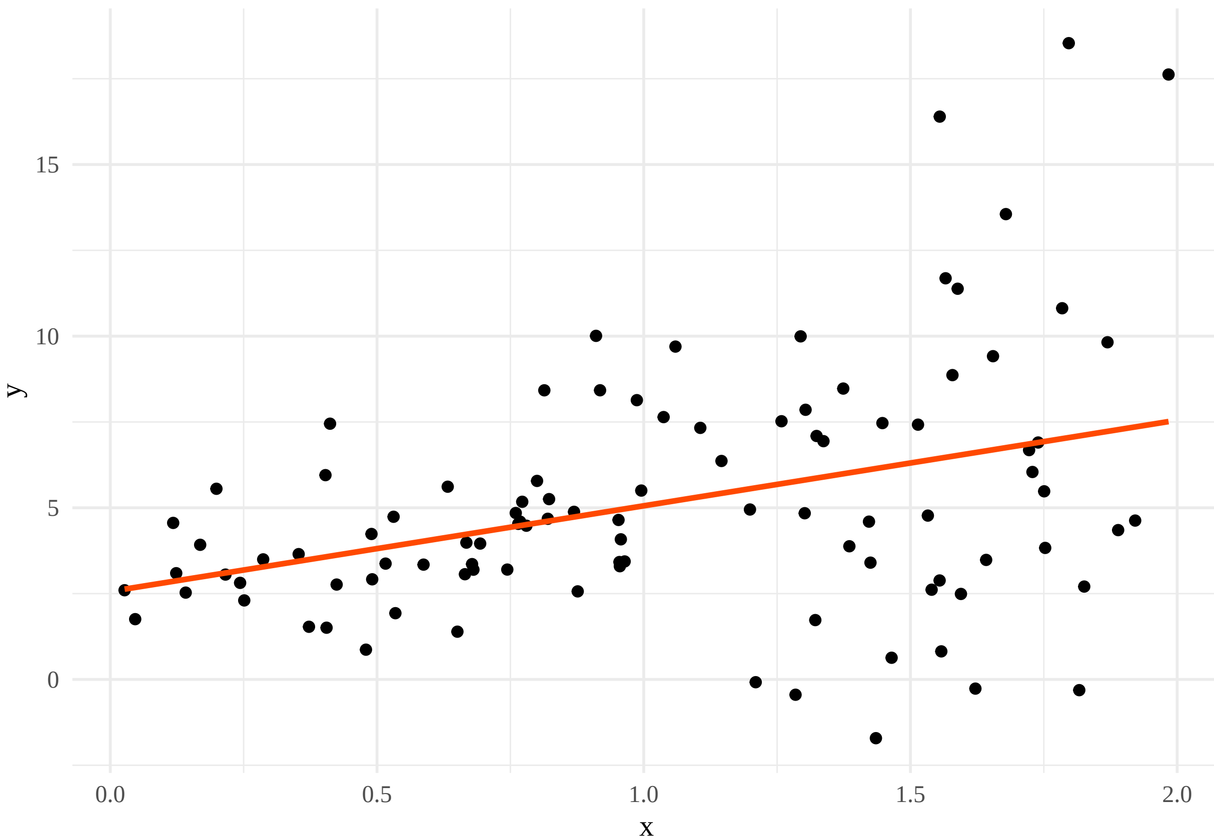
```
ols.heterosked.example = function(n) {
y = 3 - 2 * x + rnorm(n, 0, sapply(x, function(x) {
1 + 0.5 * x^2
}))
fit.ols = lm(y ~ x)
return(fit.ols$coefficients - c(3, -2))
}
```

```r
ols.heterosked.error.stats = function(n, m = 10000) {
ols.errors.raw = t(replicate(m, ols.heterosked.example(n)))
intercept.se = sd(ols.errors.raw[, "(Intercept)"])
slope.se = sd(ols.errors.raw[, "x"])
return(c(intercept.se = intercept.se, slope.se = slope.se))
}

ols.heterosked.error.stats(100)
```

```
## intercept.se     slope.se
##    0.6781456    0.5248545
```

Another One




```
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 2.563258  0.7385148 3.470828 0.0007731719
## x           2.494290  0.6341567 3.933239 0.0001565225
```

What about in this graph? Where is more reliable for estimating the line, and where is less reliable?

Ordinary Least Squares: A review

In Ordinary Least Squares, we are trying to minimize the sum of squared errors:

$$\widehat{\beta} = \underset{\beta}{\text{argmin}} \sum_{i=1}^{n} (y_i - \underline{x}_i^\top \beta)^2 = (X^\top X)^{-1} X^\top Y$$

The hat matrix is

$$\widehat{Y} = X\widehat{\beta} = X(X^\top X)^{-1} X^\top Y = HY$$

The Gauss-Markov theorem says if:

1. $Y_i = \underline{x}_i^\top \beta + \epsilon_i$
2. $\mathbb{E}[\epsilon_i] = 0$
3. $\mathbb{V}[\epsilon_i] = \sigma^2 < \infty$
4. $\text{Cov}[\epsilon_i,\ \epsilon_j] = 0$

Then $\widehat{\beta} = (X^\top X)^{-1} X^\top Y$ has the smallest variance of all possible unbiased estimators for β.

In linear models theory, we call the line based off of $\widehat{\beta}$ the **Best Linear Unbiased Estimator** (**BLUE**) of the true line/coefficcients.

Weighting in the Least Squares formula

Weighted least-squares (WLS) is based on the following:

$$\widehat{\beta} = \underset{\beta}{\text{argmin}} \sum_{i=1}^{n} w_i (y_i - \underline{x}_i^\top \beta)^2 = (X^\top W X)^{-1} X^\top W Y$$

- If some of those assumptions for G-M are violated, in particular, if $\mathbb{V}[\epsilon_i]$ depends on x_i (notated like $\sigma^2(x_i)$), then we lose the optimality of OLS.
- Aside: Gauss-Markov is a commonly used justification for OLS in applied work. The logic goes like this: (1) unbiased is good, (2) G-M says OLS is the best linear model which is unbiased. The problem is that (1) is wrong. Unbiased may be good, but often a little bias is better.

The main question here is, how do we choose the weights? $w_i =??$

Choosing w_i

Lets consider the data from the first slide.

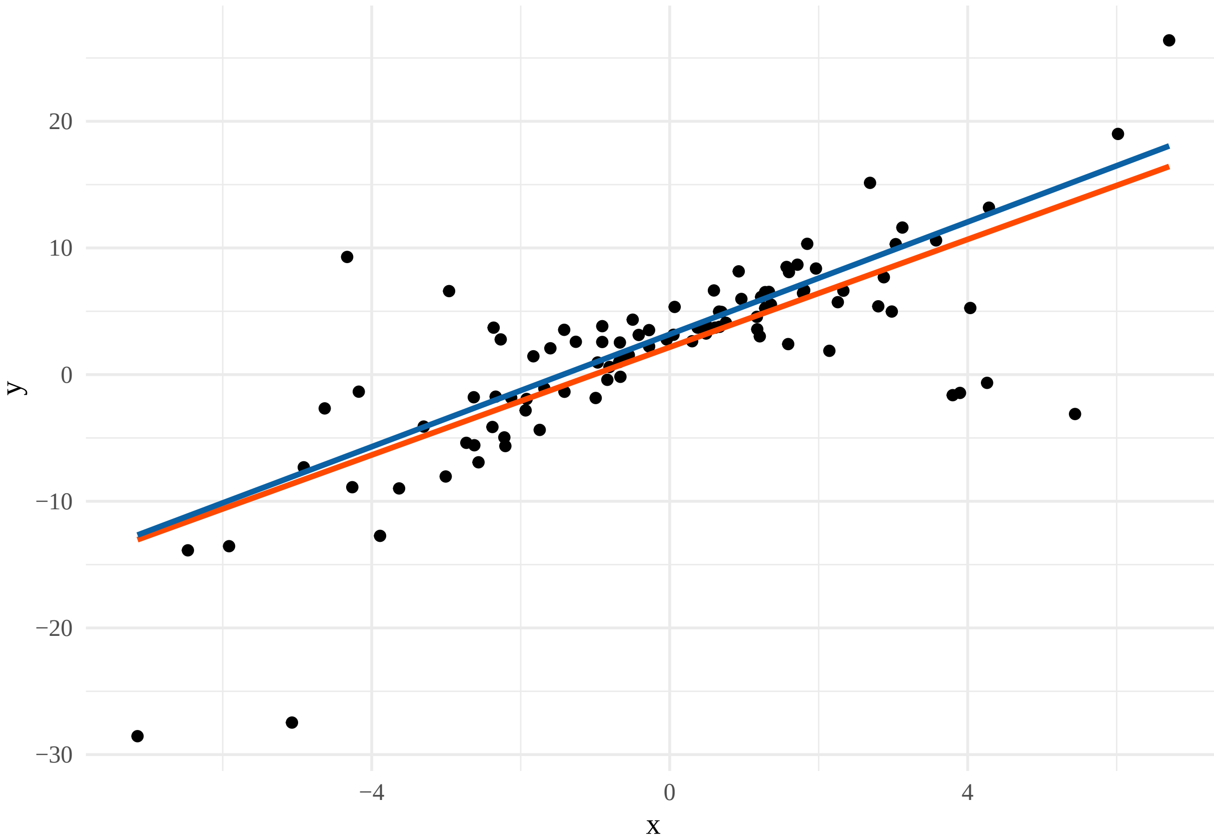
We could set $w_i = 0$ for a certain range of observations, and w_i for other observations.

For example, use $w_i = 1$ for $|x_i| \leq 2$ and $w_i = 0$ otherwise.

$$\widehat{\beta} = \begin{pmatrix} \widehat{\beta}_0 \\ \widehat{\beta}_1 \end{pmatrix} = \underset{\beta_0, \beta_1}{\text{argmin}} \sum_{i \in S} (y_i - (\beta_0 + \beta_1 x_i))^2, \text{ where } S = \{i : |x_i| \leq 2\}$$

```r
#Make weights
w = rep(NA,n)
w[abs(dfHetero$x) <= 2] <- 1
w[abs(dfHetero$x) > 2] <- 0
dfHetero$w <- w
wlm1 <- lm(y ~ x, dfHetero, weights=w) # For Next Part
```

```
ggplot(dfHetero, aes(x,y)) + geom_point() +
  geom_smooth(method='lm', se = F, color = red) +
  geom_smooth(method='lm', aes(x = x, y = y, weight=w),  se=FALSE,color=blue)
```



Is that any better?

Checking The Models (1)

Model 1 Summary:

```
##
## Call:
## lm(formula = y ~ x, data = dfHetero)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.8477  -1.3648   0.5328   1.9607  16.3377
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.1650     0.4981   4.346 3.38e-05
## x             2.1277     0.1854  11.477  < 2e-16
##
## Residual standard error: 4.971 on 98 degrees of freedom
## Multiple R-squared:  0.5734, Adjusted R-squared:  0.569
## F-statistic: 131.7 on 1 and 98 DF,  p-value: < 2.2e-16
```

Weighted Model 1 Summary:

```
##
## Call:
## lm(formula = y ~ x, data = dfHetero, weights = w)
##
## Weighted Residuals:
##    Min     1Q Median     3Q    Max
## -4.299 -0.479  0.000  0.000  3.498
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.1801     0.2251   14.13  < 2e-16
## x             2.2183     0.1962   11.31  5.7e-16
##
## Residual standard error: 1.691 on 55 degrees of freedom
## Multiple R-squared:  0.6992, Adjusted R-squared:  0.6937
## F-statistic: 127.9 on 1 and 55 DF,  p-value: 5.704e-16
```

Choosing w_i (Part 2)

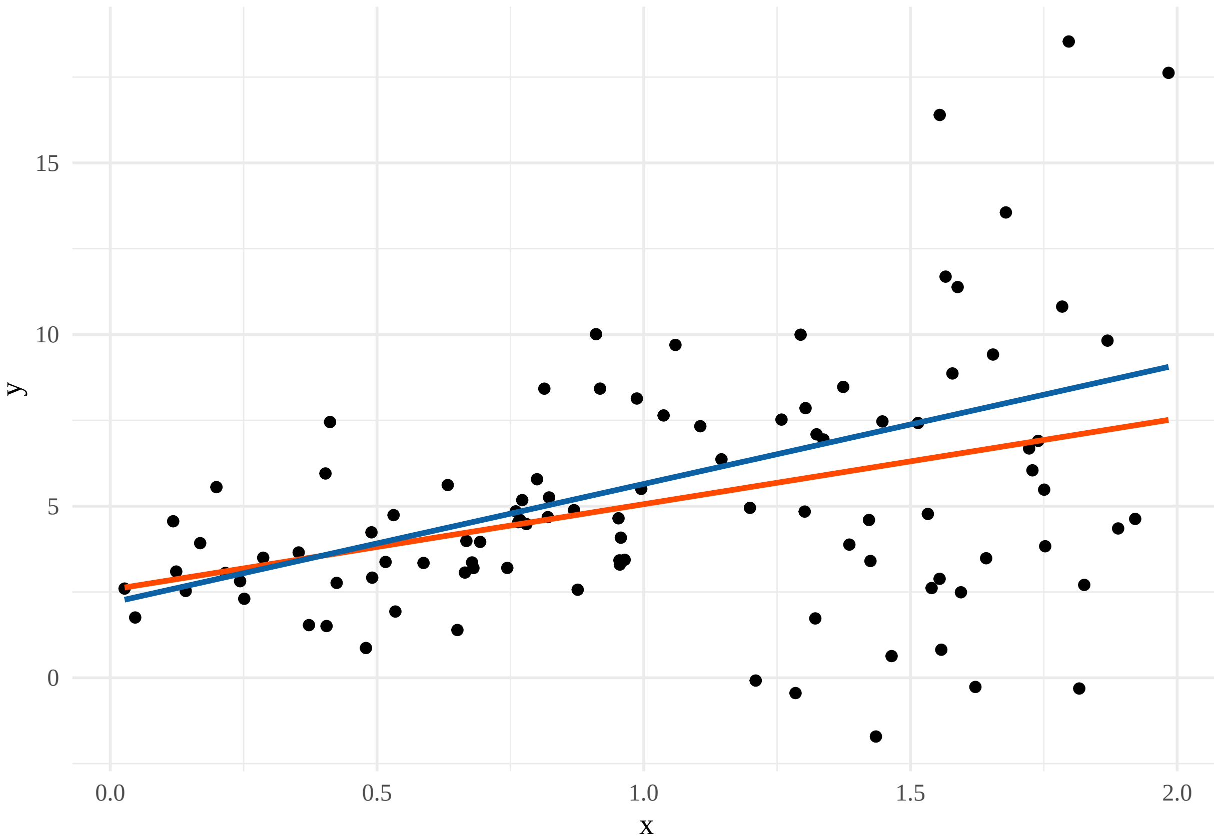
Now lets look at the other data from the beginning

Use $w_i = 1$ for $x_i < 1.2$ and $w_i = 0$ otherwise.

$$\widehat{\beta} = \begin{pmatrix} \widehat{\beta}_0 \\ \widehat{\beta}_1 \end{pmatrix} = \underset{\beta_0,\beta_1}{\operatorname{argmin}} \sum_{i \in S} (y_i - (\beta_0 + \beta_1 x_i))^2, \text{ where } S = \{i : x_i < 1.2\}$$

```r
#Make weights
w = rep(NA,n)
w[dfHetero2$x < 1.2] <- 1
w[dfHetero2$x >= 1.2] <- 0
dfHetero2$w <- w
wlm2 <- lm(y ~ x, dfHetero2, weights=w) # For Next Part

ggplot(dfHetero2, aes(x,y)) + geom_point() +
  geom_smooth(method='lm', se = F, color = red) +
  geom_smooth(method='lm', aes(x = x, y = y, weight=w),  se=FALSE,color=blue)
```



Checking The Models (Part 2)

Model 2 Summary:

```
##
## Call:
## lm(formula = y ~ x, data = dfHetero2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.8547 -1.6219 -0.1132  1.5305 11.4908
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.5633     0.7385   3.471 0.000773
## x             2.4943     0.6342   3.933 0.000157
##
## Residual standard error: 3.377 on 98 degrees of freedom
## Multiple R-squared:  0.1363, Adjusted R-squared:  0.1275
## F-statistic: 15.47 on 1 and 98 DF,  p-value: 0.0001565
```

Weighted Model 2 Summary:

```
##
## Call:
## lm(formula = y ~ x, data = dfHetero2, weights = w)
##
```

```
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0455 -0.3595  0.0000  0.0081  4.6723
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.1810     0.5418   4.026 0.000175
## x             3.4659     0.7677   4.515 3.39e-05
##
## Residual standard error: 1.779 on 55 degrees of freedom
## Multiple R-squared:  0.2704, Adjusted R-squared:  0.2571
## F-statistic: 20.38 on 1 and 55 DF,  p-value: 3.392e-05
```

The General Issue With Heteroskedacity

So suppose $\mathbb{V}[\epsilon_i] = \sigma^2(x_i)$. That is our "homoskedasticity" assumption is violated. Should we care?

What if we just use OLS (that is `lm`) anyway?

Some things don't change.

1. We still have that $\mathbb{E}\left[\widehat{\beta}\right] = \beta$. That is OLS **is** still unbiased.
2. We still have that OLS minimizes the sum of squared residuals: among all lines, OLS makes $\sum_{i=1}^{n}(x_i^\top \widehat{\beta} - y_i)^2$ as small as possible.

Some things **do** change.

1. OLS no longer has the best variance of all unbiased estimators (WLS does).
2. The standard errors that `R` produces are wrong. They make it seem "more certain" than is correct (could use the bootstrap to fix it though).
3. So are the F-tests and p-values (again, the bootstrap).

Optimal WLS for Heteroskedacity: $\sigma^2(x)$

So WLS is fairly general. But for now, let's focus on how to use it for heteroskedasticity.

Suppose you **know** the following:

1. $Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$.
2. $\mathbb{E}[\epsilon_i] = 0$
3. $\mathbb{V}[\epsilon_i] = \sigma^2(x_i)$ ($\sigma^2(\cdot)$ is a function).

It can be shown that the optimal weights are $w_i = \frac{1}{\sigma^2(\underline{x}_i)}$, making no assumption about the probability distribution of the errors, besides what is above.

This means, that the optimal $\widehat{\beta}$ vector is found by minimizing

$$\sum_{i=1}^{n} \frac{(y_i - \underline{x}_i^\top \widehat{\beta})^2}{\sigma_i^2(\underline{x}_i)}$$

See section 10.2.2.1 of Shalizi's book if you are curious… (Actually, I don't recommend that.)

Weighting in Kernel Regression, an aside

Try to recall linear smoothers, and Kernel Regression in particular.

Kernel Regression can be written as a sort of Weighted Least Squares solution

$$\widehat{c} = \underset{c}{\operatorname{argmin}} \sum_{j=1}^{n} \sum_{i=1}^{n} w_{ij}(y_i - c_j)^2 \quad w_{ij} = \frac{K((x_i - x_j)/h)}{\sum_{i=1}^{n} K((x_i - x_j)/h)}$$

This is locally constant regression.

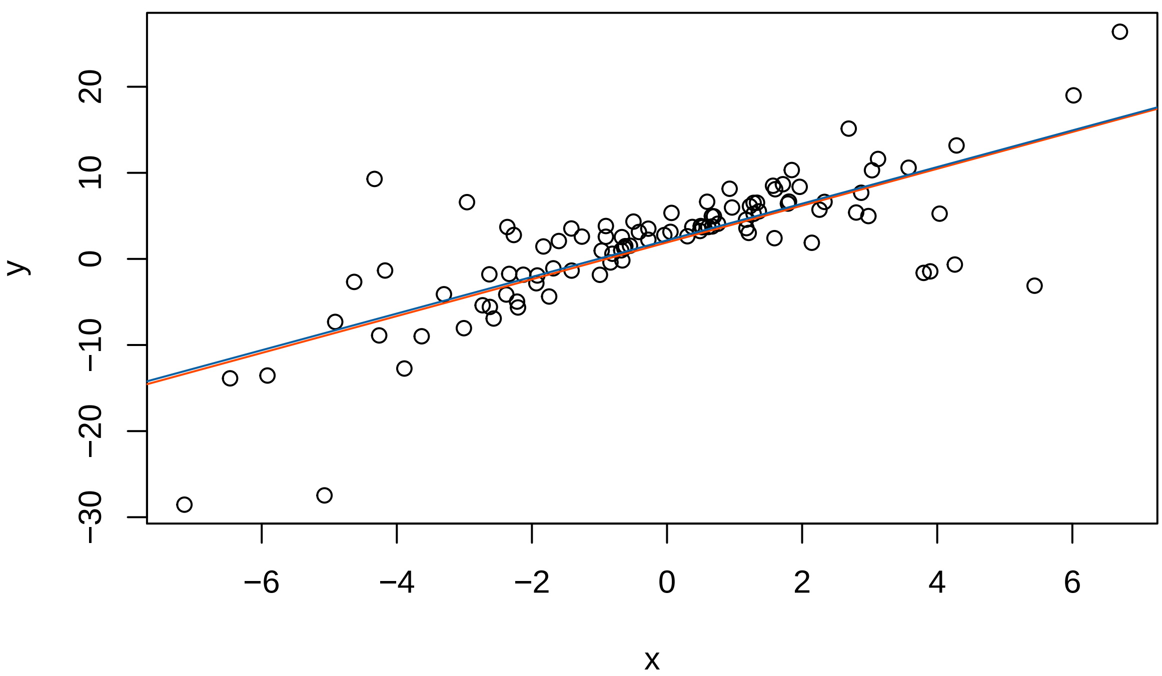
You don't need to understand this formula, but it can be useful, and it provides some justification for WLS based on previous ideas.

Using Optimal Weights in LM

```
#Make weights

dfHetero$w <- 1/(1+x^2/2) # First example had Var(residuals) = (1+x^2/2)
opt.wlm1 <- lm(y ~ x, dfHetero, weights=w) # For Next Part

with(dfHetero, plot(x,y))
abline(opt.wlm1, col = red)
abline(lm1, col = blue)
```



Comparing the different models

Model 1 Summary:

```
##
## Call:
```

```
## lm(formula = y ~ x, data = dfHetero)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.8477  -1.3648   0.5328   1.9607  16.3377
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.1650     0.4981   4.346 3.38e-05
## x             2.1277     0.1854  11.477  < 2e-16
##
## Residual standard error: 4.971 on 98 degrees of freedom
## Multiple R-squared:  0.5734, Adjusted R-squared:  0.569
## F-statistic: 131.7 on 1 and 98 DF,  p-value: < 2.2e-16
```

___ Pseudo Weights Model 1 Summary___:

```
##
## Call:
## lm(formula = y ~ x, data = dfHetero, weights = w)
##
## Weighted Residuals:
##    Min     1Q Median     3Q    Max
## -4.299 -0.479  0.000  0.000  3.498
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.1801     0.2251   14.13  < 2e-16
## x             2.2183     0.1962   11.31  5.7e-16
##
## Residual standard error: 1.691 on 55 degrees of freedom
## Multiple R-squared:  0.6992, Adjusted R-squared:  0.6937
## F-statistic: 127.9 on 1 and 55 DF,  p-value: 5.704e-16
```

Optimal Weights Model 1 Summary:

```
##
## Call:
## lm(formula = y ~ x, data = dfHetero, weights = w)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -16.8430  -0.9417   0.6378   1.8065  13.9880
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.9154     0.5136   3.729 0.000322
## x             2.1391     0.1926  11.105  < 2e-16
##
## Residual standard error: 4.16 on 98 degrees of freedom
## Multiple R-squared:  0.5572, Adjusted R-squared:  0.5527
## F-statistic: 123.3 on 1 and 98 DF,  p-value: < 2.2e-16
```

Simulating the differencce between OLS and WLS

We will compare the simulated OLS and WLS Standard Errors from the model at the very beginning

```
## intercept.se     slope.se
##    0.3066330    0.3553108
```

```
## intercept.se     slope.se
##    0.2734971    0.3190810
```

If we knew $\sigma^2(x)$, this would be easy as... pie. Unfortunately, it is never that easy.

This means that our new issue is estimating $\sigma^2(x)$.

Variances and Conditional Variances

In general, for a random variable X, the variance is defined as:

$$\mathbb{V}\left[X\right] = \mathbb{E}\left[\left(X - \mathbb{E}\left[X\right]^2\right)\right]$$

Let's consider the variance of the residuals: $\epsilon_i = y_i - \underline{x}_i^\top \beta$

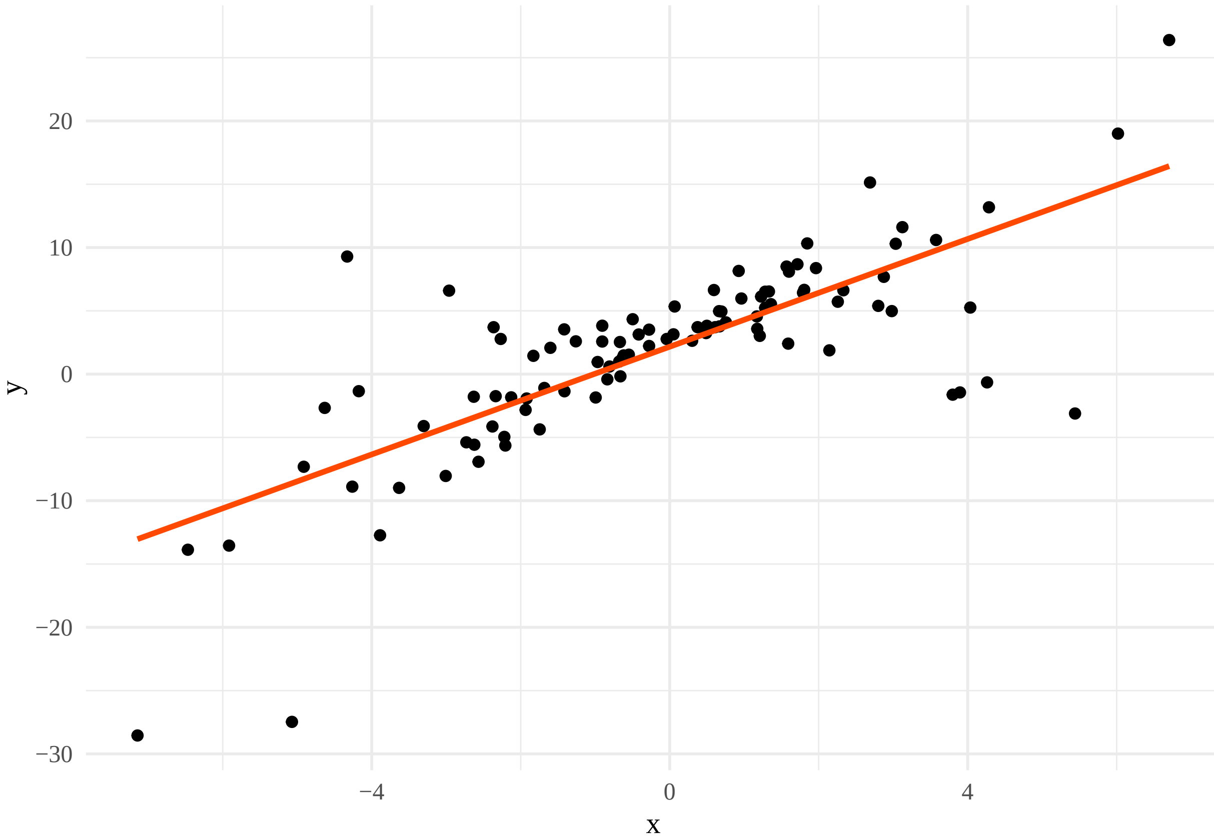
$$\begin{aligned}\sigma^2(x_i) = \mathbb{V}\left[\epsilon_i|x_i\right] &= \mathbb{E}\left[\epsilon_i - \mathbb{E}\left[\epsilon_i|x_i\right]^2|x_i\right] \\ &= \mathbb{E}\left[\epsilon_i^2|x_i\right]\end{aligned}$$

What is our estimate of this expectation?

Estimating $\sigma^2(x)$, An Iterative Process

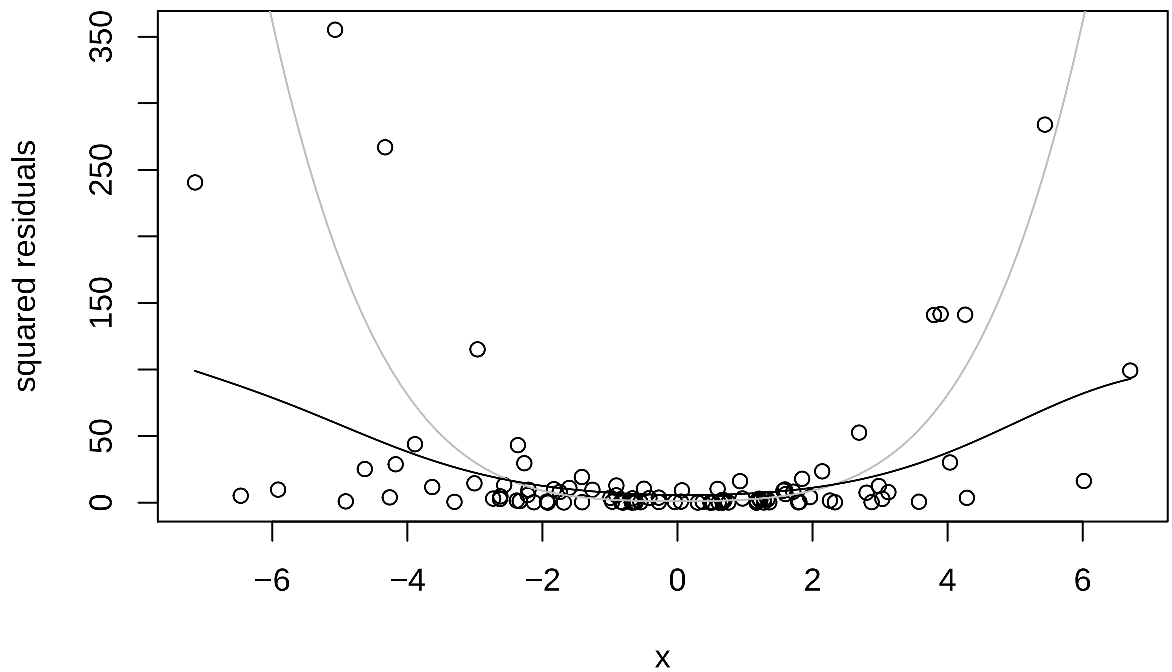
1. Use `lm` to estimate β_0 and β_1 to get the estimated regression line $\widehat{\mu}(x)$.
2. Use your estimated regression line to calculate the squared residuals, $e_i^2 = (y_i - \widehat{\mu}(x_i))^2$.
3. Use nonparametric regression to get $\widehat{\sigma}^2(x)$, which is an estimate of $\mathbb{E}\left[\epsilon_i^2|x_i\right]$
4. Use this estimate "know" $\sigma^2(x)$ and use WLS (with `lm(y~x, weights=1/sig2)`)
5. You could stop here. But since you now have "better" estimates of β_1 and β_0, it's better to iterate 2 and 3 until some convergence.
6. Ok. Something converged, so you return the last estimates of β_0 and β_1. But the SEs are not right quite right since we are only estimating $\sigma^2(x)$
7. To get "correct" SEs, use the bootstrap:
    a. Non-parametric: repeat 1-5 B times on resampled data. This can be rather slow...
    b. Model-based: this is actually pretty hard here, better not to do it.

An Example Using the First Model




```
##             Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 2.164973  0.4980974  4.346484 3.383817e-05
## x           2.127688  0.1853945 11.476538 7.852372e-20
```

Using Kernel Regression to Estimate $\sigma^2(x)$



Iterations for first model

```
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 2.164973  0.4980974  4.346484 3.383817e-05
## x           2.127688  0.1853945 11.476538 7.852372e-20
```

```
var1 <- npreg(residuals(lm1)^2 ~ x, data = dfHetero )
wlm1 <- lm(y ~ x, dfHetero, weights = 1/fitted(var1))
summary(wlm1)$coefficients

var2 <- npreg(residuals(wlm1)^2 ~ x, data = dfHetero )
wlm2 <- lm(y ~ x, dfHetero, weights = 1/fitted(var2))
summary(wlm2)$coefficients

var3 <- npreg(residuals(wlm2)^2 ~ x, data = dfHetero )
wlm3 <- lm(y ~ x, dfHetero, weights = 1/fitted(var3))
summary(wlm3)$coefficients
```

```
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 2.878506  0.2861681 10.05879 9.021827e-17
## x           2.013979  0.1738771 11.58277 4.651842e-20

##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 2.907036  0.2757588 10.54195 8.093778e-18
## x           2.009604  0.1733555 11.59239 4.436706e-20

##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 2.907659  0.2755368 10.55271 7.671609e-18
## x           2.009335  0.1733433 11.59165 4.452920e-20
```

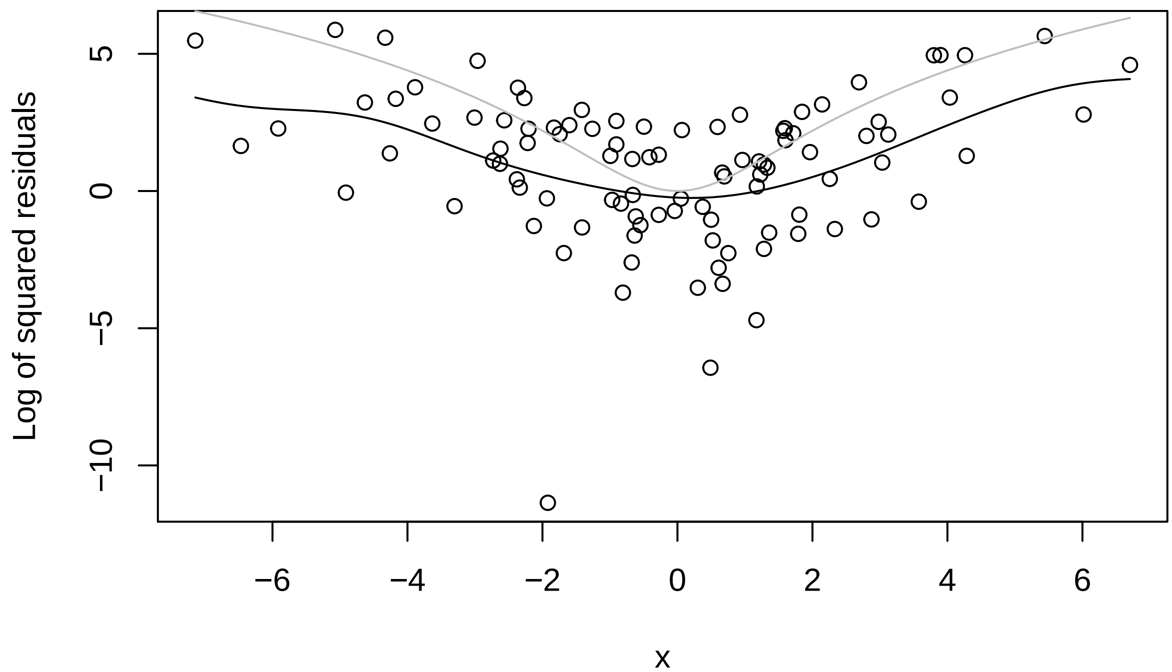
Simplified Iterative Function

```
iterative.wls <- function(x, y, tol = 0.01, max.iter = 100) {
iteration <- 1
old.coefs <- NA
regression <- lm(y ~ x)
coefs <- coefficients(regression)
while (is.na(old.coefs) || ((max(coefs - old.coefs) > tol) && (iteration <
max.iter))) {
variance <- npreg(residuals(regression)^2 ~ x)
old.coefs <- coefs
iteration <- iteration + 1
regression <- lm(y ~ x, weights = 1/fitted(variance))
coefs <- coefficients(regression)
}
return(list(regression = regression, variance = variance, iterations = iteration))
}
```

Alternative, $\log$ of residuals

1. Use `lm` to estimate β_0 and β_1.
2. Use your estimated regression line to calculate the squared residuals, $e_i^2 = (y_i - \widehat{\mu}(x_i))^2$.
3. Calculate $\log(\widehat{e_i}^2)$ and use `npreg` to estimate $\log \sigma^2(x)$.
4. Now pretend that you "know" $\sigma^2(x)$ (take exp of your estimate from 2.) and use WLS (with `lm(y~x, weights=1/sig2)`)
5. You could stop here. But since you now have "better" estimates of β_1 and β_0, it's better to iterate 2 and 3 until some convergence.
6. Ok. Something converged, so you return the last estimates of β_0 and β_1. But the SEs are not right (because you "know" $\sigma^2(x)$ but you don't **know** it).
7. To get SEs, use the bootstrap:
   a. Non-parametric: repeat 1-5 B times on resampled data. Still slow.
   b. Model-based: this is actually pretty hard here, better not to do it.

Looking at Log of e_i^2



Same Example, Now using the log

```
logvar1 <- npreg(log(residuals(lm1)^2) ~ x, data = dfHetero )
wlm1 <- lm(y ~ x, dfHetero, weights = 1/exp(fitted(logvar1)))
summary(wlm1)$coefficients

logvar2 <- npreg(log(residuals(wlm1)^2) ~ x, data = dfHetero )
wlm2 <- lm(y ~ x, dfHetero, weights = 1/exp(fitted(logvar2)))
summary(wlm2)$coefficients

logvar3 <- npreg(log(residuals(wlm2)^2) ~ x, data = dfHetero )
wlm3 <- lm(y ~ x, dfHetero, weights = 1/exp(fitted(logvar3)))
summary(wlm3)$coefficients
```

```
##             Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 2.972818  0.2454297 12.11271 3.458792e-21
## x           2.059395  0.1665938 12.36177 1.027915e-21

##             Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 3.028201  0.2277862 13.29405 1.153945e-23
## x           2.033225  0.1690139 12.02993 5.183114e-21

##             Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 3.027455  0.2278962 13.28436 1.208497e-23
## x           2.029709  0.1684034 12.05266 4.637998e-21
```

A Bigger Example

This is a (slightly modified) portion of a real job interview.

It is a very simple application of heteroskedasticity.

Heteroskedasticity appears frequently with financial data, so those companies like to see if you can handle it.

The set up

The dataset `jobInt` contains data from a simple linear model with heteroskedastic noise.

```
set.seed(02-26-2019)
n=250
x = rnorm(n, sd=1.5)
sigma.x <- function(x) (5*(sin(x)^2)+2)*(x>=0) + (x^2+1)*(x<0)
y = -1+2*x + sigma.x(x)*rnorm(n)
jobInt = data.frame(x=x, y=y)
```

In other words, for $i = 1, \ldots, 250$,

$$y_i = \beta_0 + \beta_1 x_i + \sigma(x_i)\epsilon_i \qquad \epsilon_i \sim \mathrm{N}(0, 1).$$

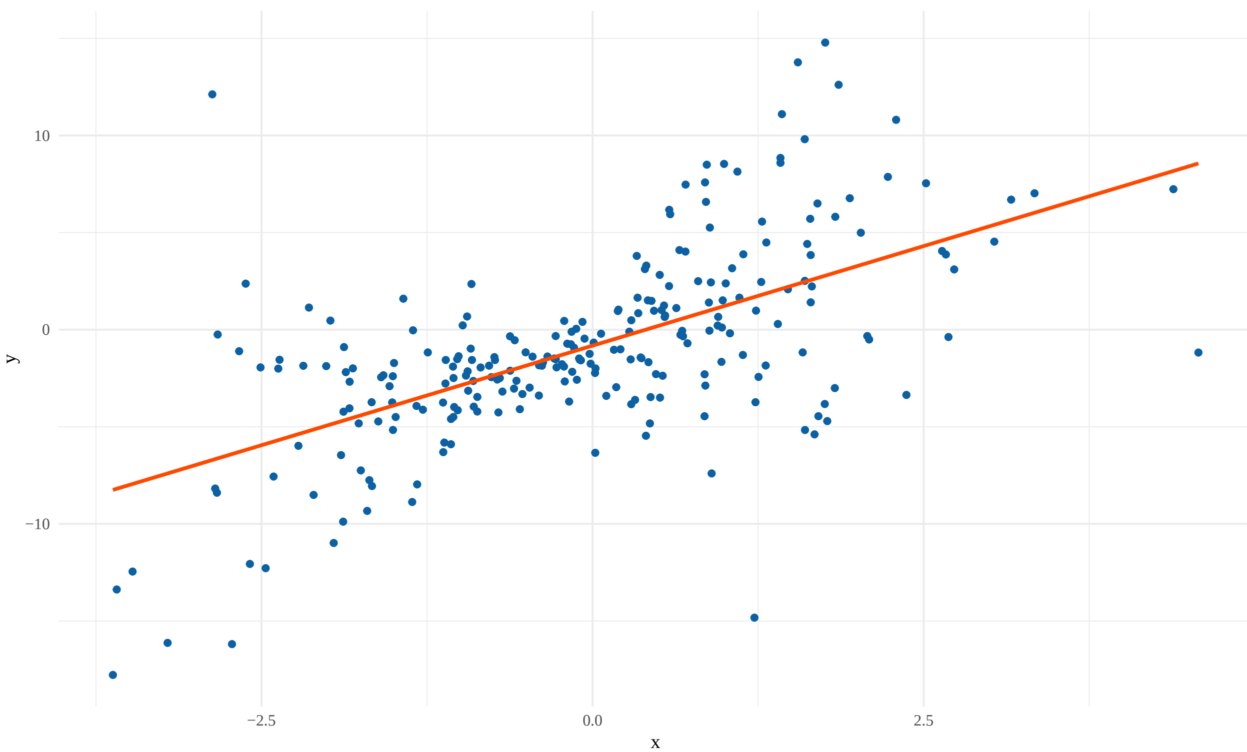
You know nothing about (the function) $\sigma(\cdot)$.

Your goal is to estimate $(\beta_0,\ \beta_1)$ as well as possible, and provide a CI.

How do I do this?

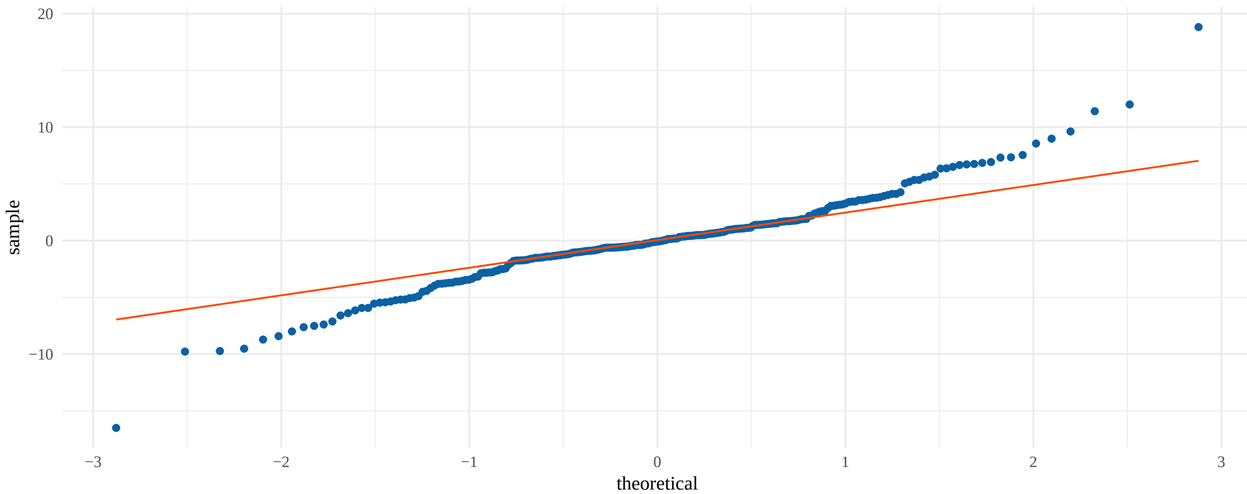
First things first, EDA.

```
ggplot(jobInt, aes(x,y)) + geom_point(color=blue) +
  geom_smooth(method='lm',se=FALSE,color=red)
```

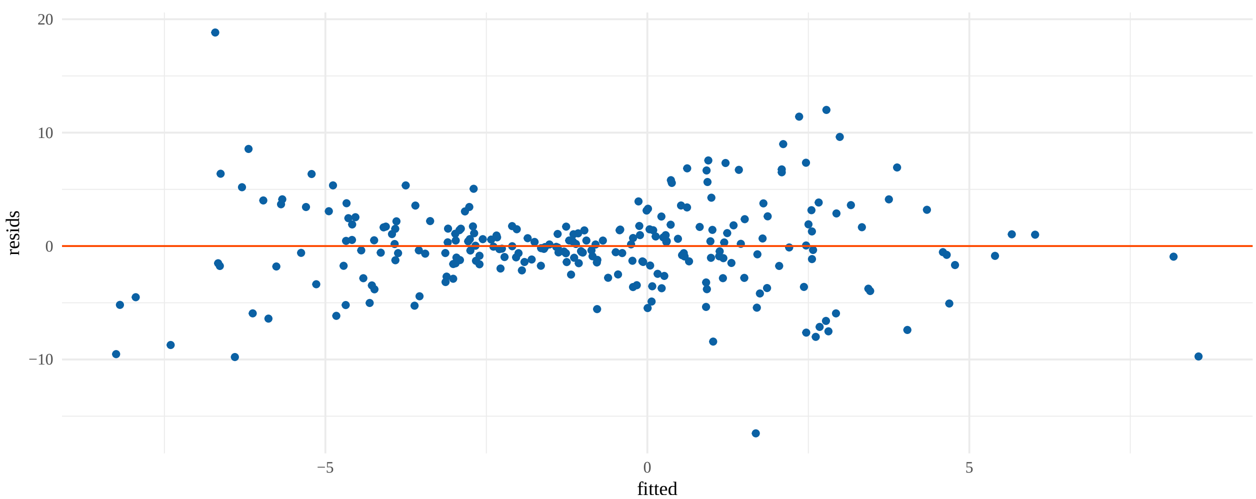


```r
basicMod = lm(y~x)
```

QQ-Plot of Residuals



Examine Residuals vs. Fitted values



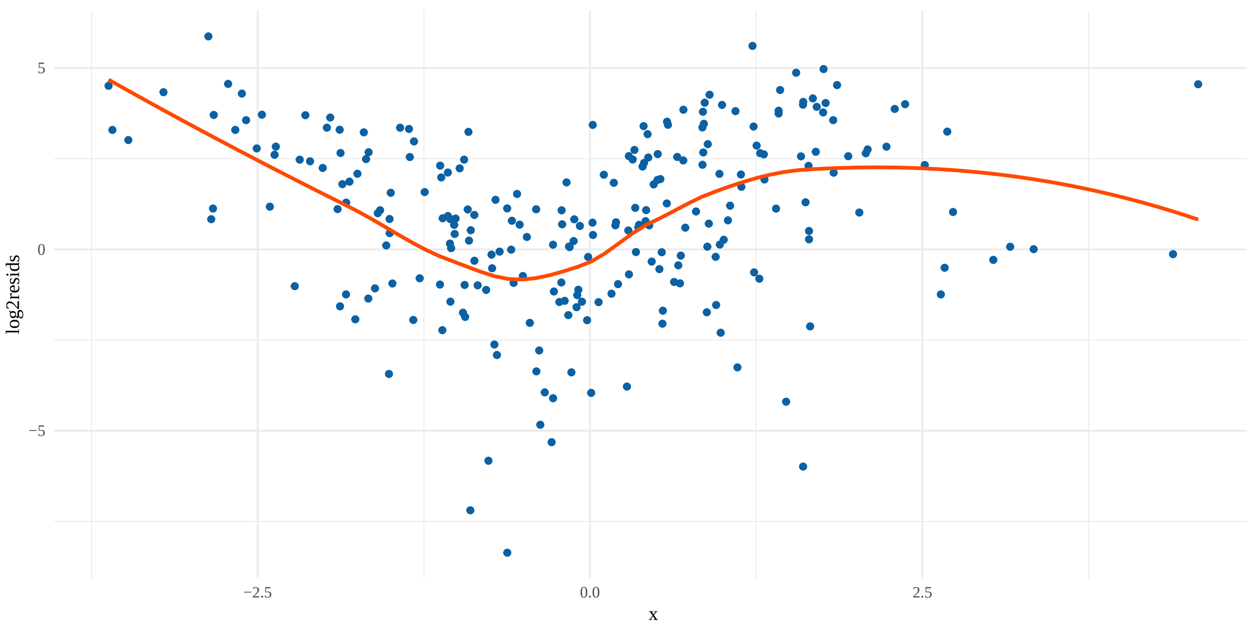
Code for Iterative WLS using Log of Squared Residuals

This code takes in data and does steps 1-5. It is **not** optimized for speed, but for readability, so run with care.

```r
heteroWLS <- function(dataFrame, tol = 1e-4, maxit = 100, track=FALSE){
  # inputs: a data object, optional: tolerance, max.iterations, and progress tracker (prints)
  # outputs: estimated betas and weights
  require(np)
  ols = lm(y~x, data=dataFrame)
  b = coefficients(ols)
  conv = FALSE
  for(iter in 1:maxit){ # don't let this run forever
```

```
    if(conv) break # if the b's stop moving, get out of the loop
    logSqResids = log(residuals(ols)^2)
    winv = exp(predict(npreg(logSqResids~x, data=dataFrame, tol=1e-2, ftol=1e-2)))
    winv[winv < tol] = tol # zero inverse weights are bad, make them small
    ols = lm(y~x, weights = 1/winv, data=dataFrame) #weights are 1 / estim.variance
    newb = coefficients(ols)
    conv.crit = sum((b-newb)^2) # calculate how much b moved
    if(track) cat('\n', iter, '/', maxit, ' conv.crit = ', conv.crit) # print progress
    conv = (conv.crit < tol) # check if the b's changed much
    b = newb # update the coefficient estimates
  }
  return(list(betas=b, weights = winv, log2resids = log(residuals(ols)^2)))
}
```

Log of Squared Residuals in OLS Model



Running Code (Slow...)

```
start.time <-proc.time()[[3]]

resampWLS <- function(dataFrame,...){ # ... means options passed on
  rowSamp = sample(1:nrow(dataFrame), size=nrow(dataFrame), replace=TRUE)
  return(heteroWLS(dataFrame[rowSamp,],...)$betas) # passed things on if desired
}

B = 100 #
alp = .05
origBetas = heteroWLS(jobInt)
time.1 <- proc.time()[[3]] - start.time

bootBetas <- replicate(B, resampWLS(jobInt, maxit=20))
```

```r
qq = apply(bootBetas, 1, quantile, probs=c(1-alp/2, alp/2))
CI = cbind(origBetas$betas, 2*origBetas$betas - t(qq))
colnames(CI) = c('coef', rev(colnames(CI)[2:3]))

time.2 <- proc.time()[[3]] - time.1
```

```r
# Time to get WLS function to converge on weights
time.1
```

```
## [1] 15.541
```

```r
# Time to get bootstrapped CIs
time.2
```

```
## [1] 81.033
```

```r
CI
```

```
##                  coef     2.5%      97.5%
## (Intercept) -1.002996 -1.521497 -0.7480195
## x            1.959554  1.374641  2.4224998
```

Log of Squared Residuals in WLS Model