Biost 518 / Biost 515
Applied Biostatistics II / Biostatistics II

Zimeng (Parker) Xie
University of Washington

Discussion Week 2:
Log transformations and robust standard errors in linear regression

January 17-19, 2018

Coming up

- Log transformations
- Robust standard errors

Log transformed variables

In class this week, we'll examine log transformations of variables in linear regression models.

Log transforming the outcome and/or predictor variables:

- May be scientifically relevant: examples include modeling rates of drug absorption into the body or concentrations of antibodies (which often differ in magnitude)
- Allows us to model relative changes in the outcome variable with the predictor (as percent or fold-changes)
- May stabilize the variance (more on this later!)

Regression with log-transformed outcome

Consider fitting a regression model with a log transformed outcome:

$$\log(Y|X) = b_0 + b_1 x + error$$

Exponentiating,

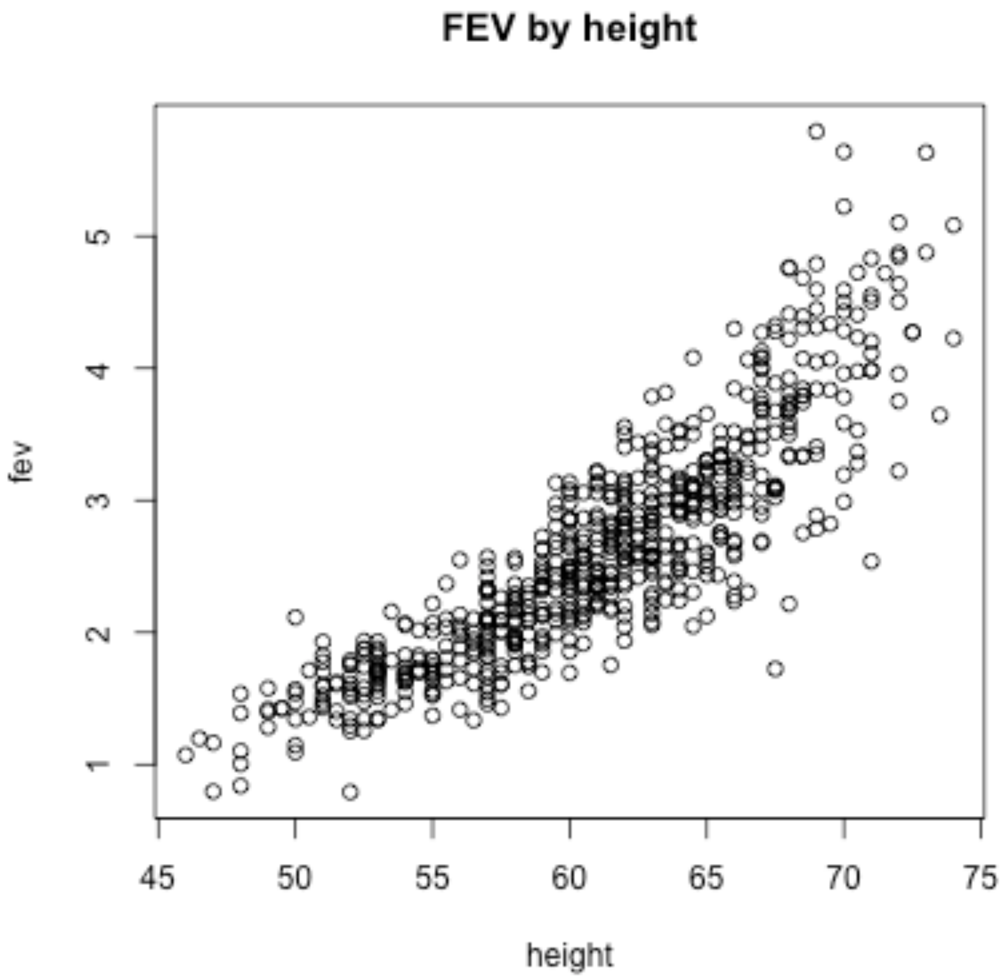
$$(Y \mid X) \approx \exp(b_0 + b_1 x)$$
$$(Y \mid X) = \exp(b_0)\exp(b_1 x)$$

- b_1 is the difference in mean $\log(Y)$ for a one-unit change in X
- $\exp(b_1)$ is the ratio of mean outcomes for groups differing by one unit of X
- $\frac{\log(k)}{b_1}$ is the change in X associated with a k-fold increase in geometric mean Y

Interpretation hint: compare pairs of obs. (x_1, y_1) *and* (x_2, y_2)

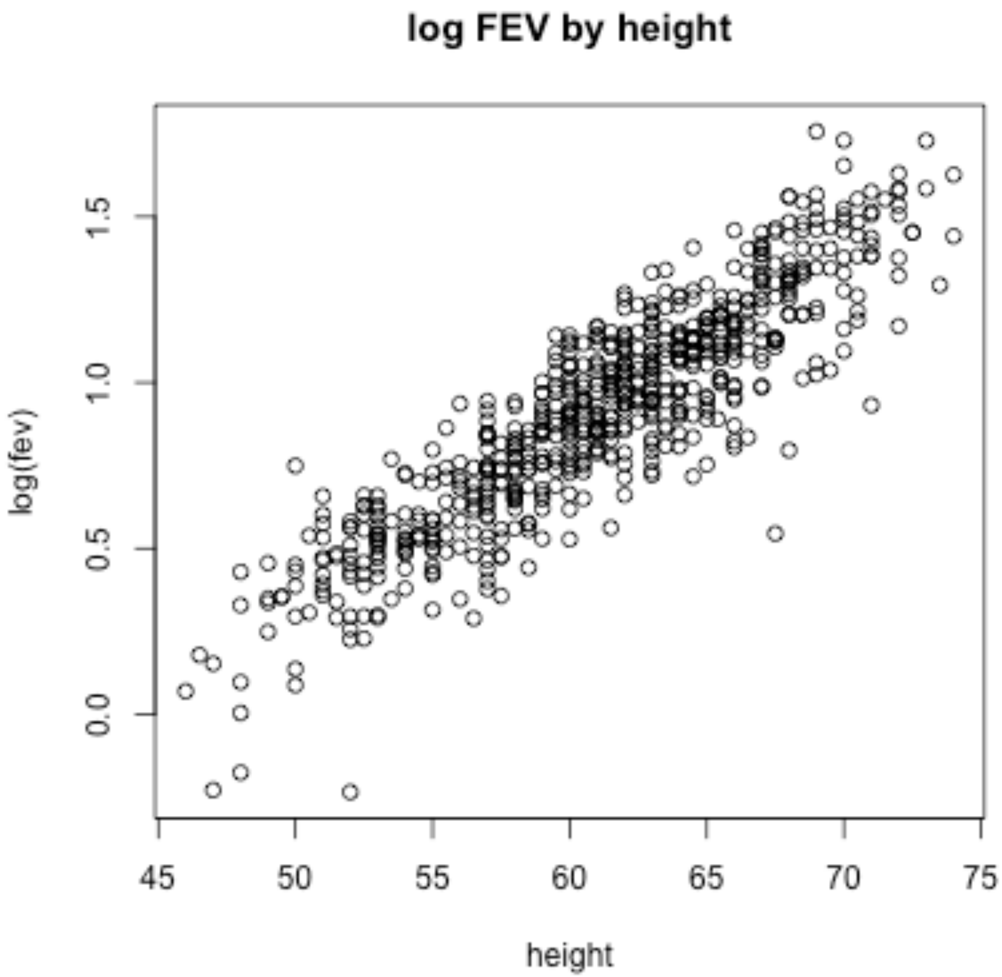
Example: FEV dataset

A first example; untransformed FEV and heights from 654 children.



Example: FEV dataset

Log-transformed FEV by height



Example: FEV dataset

Make a scatterplot of the data with `plot()`:

```
> plot( fev ~ height, data = fevdat, main = "FEV by height");
```

To fit the regression line, use `lm()`:

```
> lm.fev <- lm( fev ~ height, data = fevdat );
```

To overlay the regression line, use `lines()` and `predict()`:

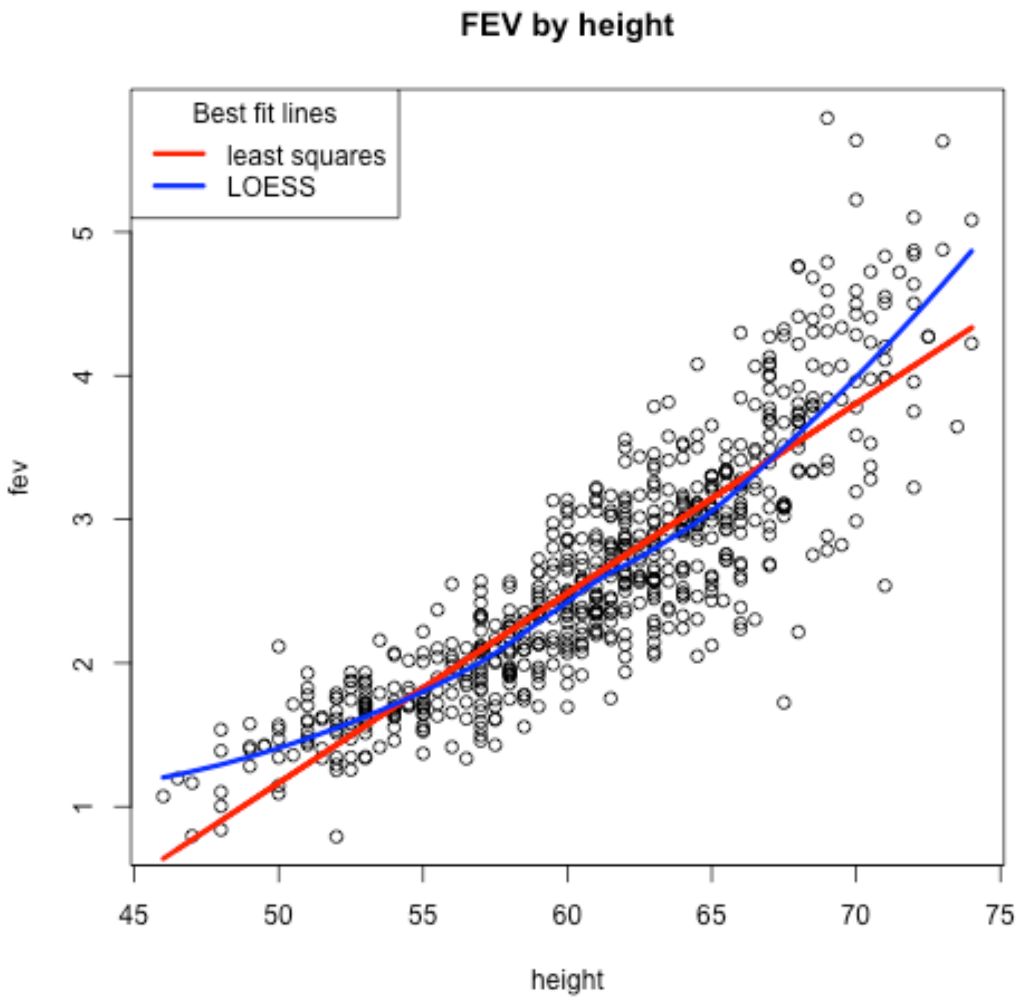
```
> lines( fevdat$height,
   predict(lm.fev, data.frame(height=fevdat$height)),
   col="red", lwd=3 );
```

Make a loess smoothed curve with `loess()`:

```
> loess.fev <- loess(fev~height,col="red",lwd=2,data=fevdat);
> ord <- order( fevdat$height );
> lines( fevdat$height[ ord ],loess.fev$fitted[ ord ], col =
   "blue", lwd = 3 );
```

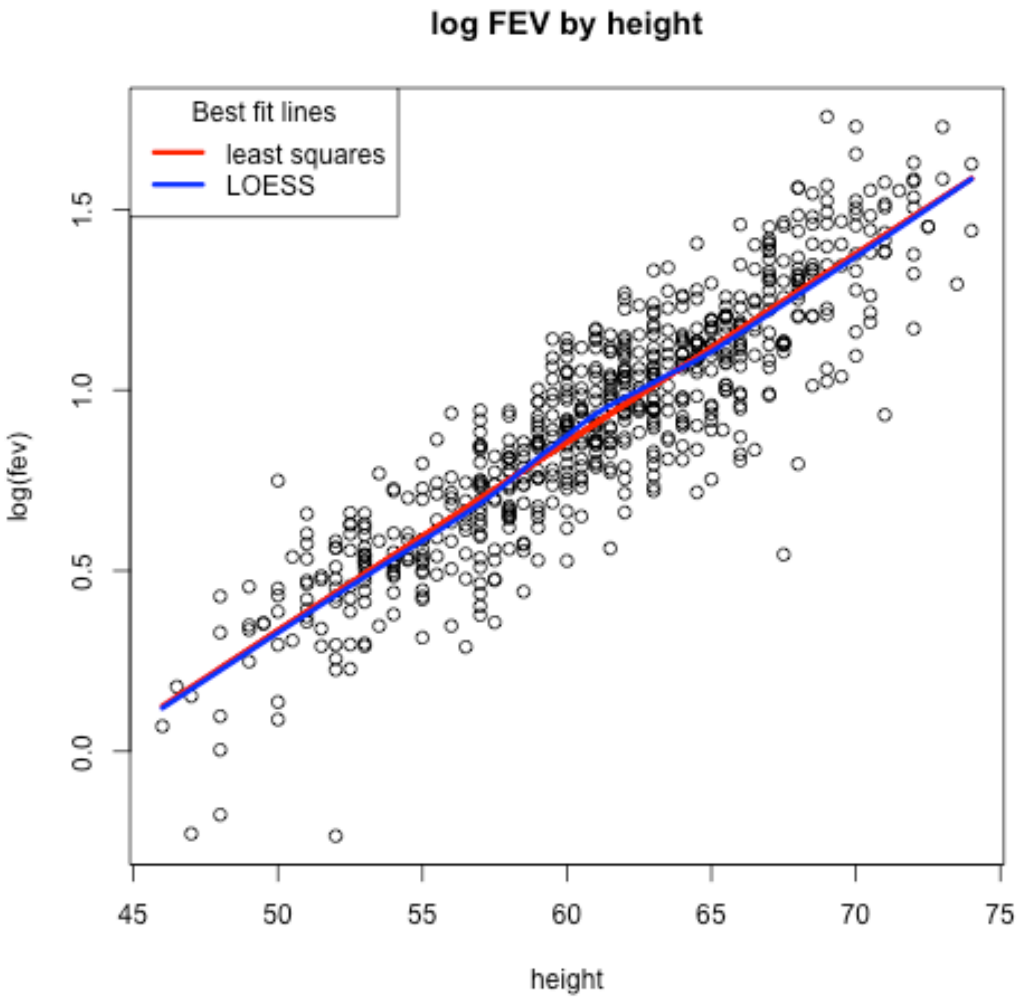
Example: FEV dataset

Note the nonlinear relationship, heteroscedasticity (cone-shaped)



Example: FEV dataset

Improved linear fit, homoscedastic (const. variation about regression line)



Regression commands in R

`uwIntroStats`' `regress()` has verbose output, including robust SE's:

```
> lmobj <- regress( "mean", fev ~ height, data = fevdat );
```

```
Call:
regress(fnctl = "mean", formula = fev ~ height, data = fevdat)
Residuals:
Min        1Q    Median        3Q       Max
-1.75167 -0.26619 -0.00401  0.24474  2.11936
Coefficients:
Estimate   Naive SE    Robust SE     95%L       95%H         F stat     df Pr(>F)
[1] Intercept      -5.433      0.1815      0.2008       -5.827     -5.038          731.83 1   < 0.00005
[2] height          0.1320    2.955e-03   3.415e-03      0.1253     0.1387        1493.41 1   < 0.00005
Residual standard error: 0.4307 on 652 degrees of freedomMultiple R-squared:  0.7537,
Adjusted R-squared:  0.7533 F-statistic:  1493 on 1 and 652 DF,  p-value: < 2.2e-16
```

To extract the coefficients, the `coef()` function is probably the easiest approach:

```
> coef(lmobj)
```

You can also refer to `lmobj`**`$augCoefficients`**`;`

Log transformed response

We can instruct `regress()` to model the outcome using the geometric mean:

```
> lmgm <- regress(”geometric mean", fev~height, data=fevdat);
> lmgm
```

```
Coefficients:
Raw Model:
Estimate  Naive SE   Robust SE       F stat    df Pr(>F)
[1] Intercept     -2.271     0.06353    0.06855          1097.78 1  < 0.00005
[2] height        0.05212   1.035e-03  1.123e-03         2155.08 1  < 0.00005
Transformed Model:
e(Est)     e(95%L)    e(95%H)          F stat    df Pr(>F)
[1] Intercept      0.1032   0.09018    0.1180          1097.78 1  < 0.00005
[2] height         1.054     1.051     1.056           2155.08 1  < 0.00005
```

`uwIntroStats`’ output returns you the transformed estimates for free!

Robust standard errors

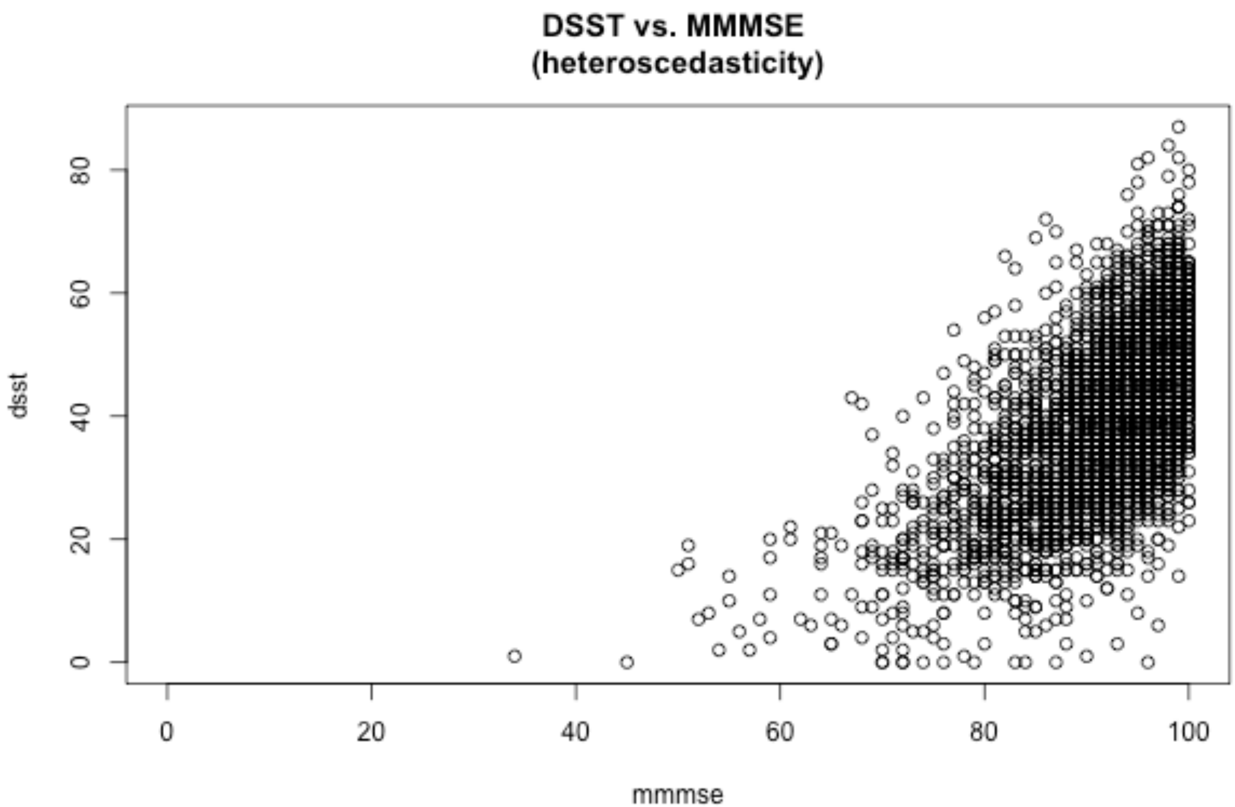
When these are available, robust standard errors are a quick way to perform inference with the same robustness properties as the bootstrap -- think t test allowing for unequal variances, but for multiple groups!

- Used when groups defined by the predictor variable have different variability in the response
- Examples:
  - Weight measurements get more variable as we age
  - Public universities have greater variability in enrollment than private universities/colleges

We illustrate them with the DSST dataset from the previous weeks' discussion, in more detail:

Example: DSST dataset

We return to examining cognitive function as measured by the digit symbol substitution test (DSST – a test of attention) and mental status, as measured by the modified mini mental status exam (MMMSE). First, a plot of the data;



Example: DSST dataset

"Ordinary" linear regression:

```
> lm.normal <- lm( dsst ~ mmmse, data = "mri" );
> summary( lm.normal )$coef; # just the coefficients

              Estimate Std. Error   t value      Pr(>|t|)
(Intercept) -62.432239 2.30288892 -27.11040 3.382432e-147
mmmse         1.112746 0.02508311  44.36235  0.000000e+00
```

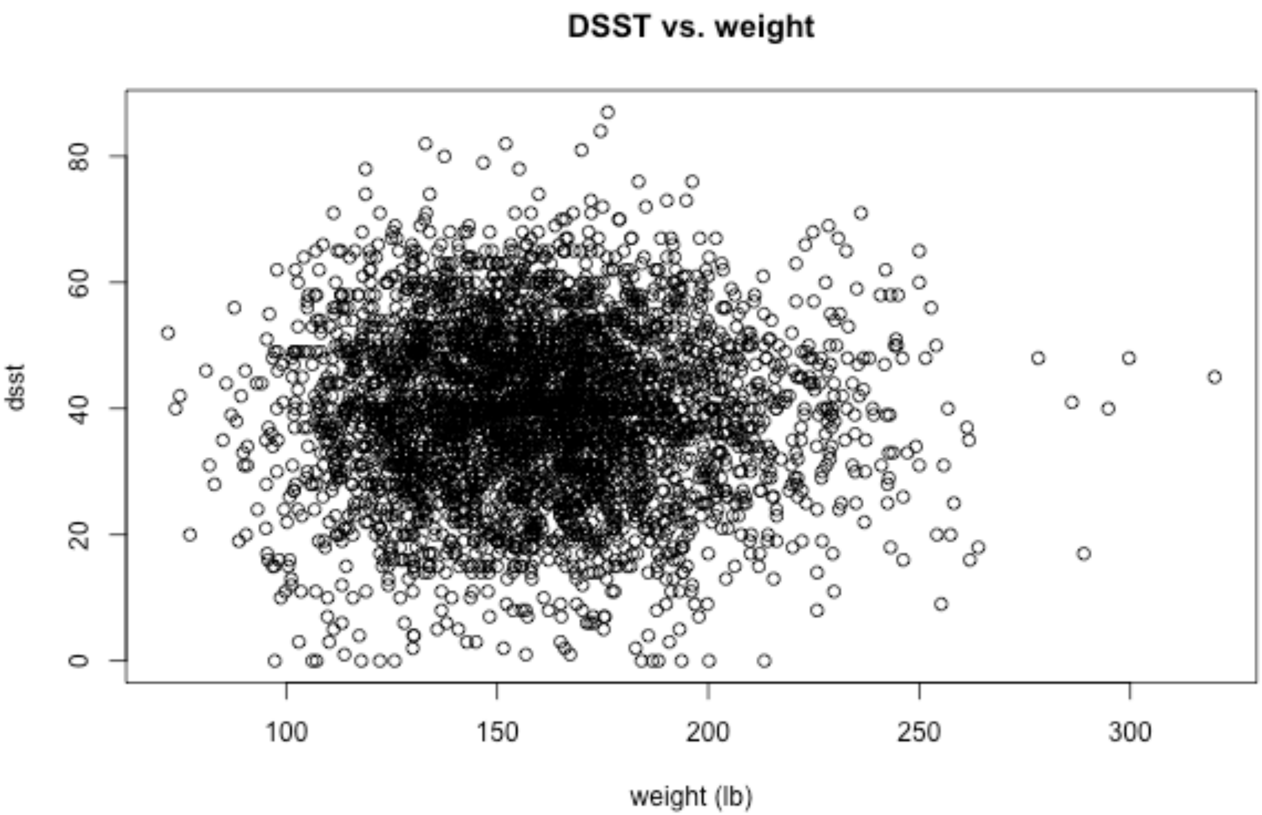
Linear regression with robust standard errors:

```
> lm.robust <- regress( "mean", dsst ~ mmmse, data = mri );
> coef( lm.robust );

            Estimate Naive SE Robust SE 95%L    95%H t value    Pr(>|t|)
(Intercept) -62.4322 2.30288   2.454 -67.244 -57.6203 -25.438 3.227e-131
mmmse         1.1127 0.02508   0.026   1.060   1.1650  41.680 3.914e-309
```

Example: DSST dataset

What happens with robust standard errors when variances are constant? To examine this, we examine the relationship between DSST and weight;



Example: DSST dataset

"Ordinary" linear regression:

```
> lm.wt.normal <- lm( dsst ~ weight, data = mri );
> summary( lm.wt.normal )$coef;
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	3.921142e+01	1.187054276	33.032543	8.218782e-209
weight	9.797549e-04	0.007303103	0.134156	8.932869e-01

Linear regression with robust SEs:

```
> lm.wt.robust <- regress("mean", dsst~weight, data=mri );
> coef( lm.wt.robust );
```

	Estimate	Naive SE	Robust SE	95%L	95%H	t value	Pr(>|t|)
(Intercept)	3.921e+01	1.18705	1.19848	36.8616	41.5612	32.7174	2.3139e-205
weight	9.797e-04	0.00730	0.00732	-0.0133	0.0153	0.1336	8.9366e-01

Questions?