STAT 215
Transformations and Influential Points

Colin Reimer Dawson

Oberlin College

10 October 2017

Outline

The Simple Linear Model

$$Y = \beta_0 + \beta_1 \cdot X + \varepsilon$$

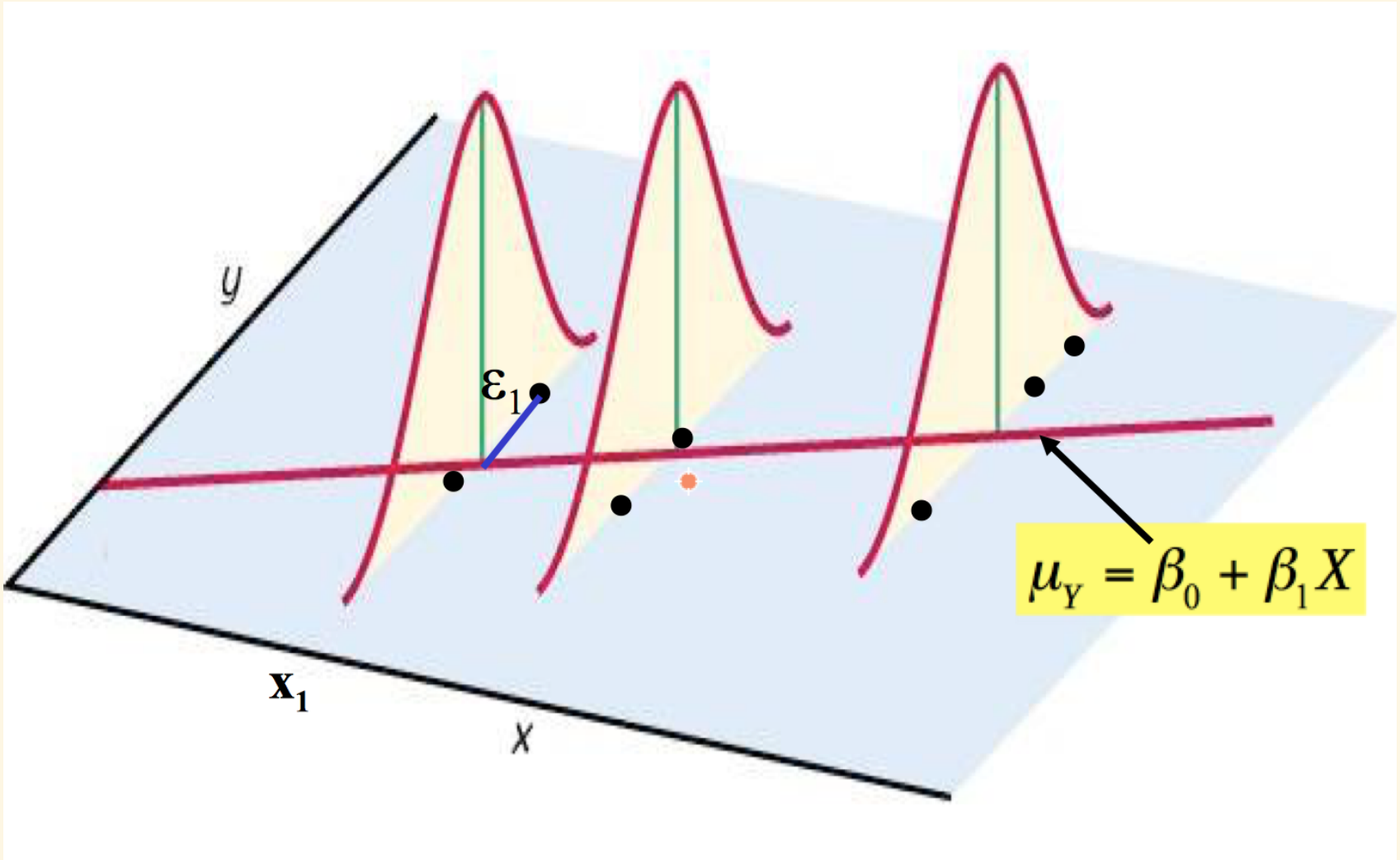
aka

$$\text{Response} = \text{Intercept} + \text{Slope} \cdot \text{Predictor} + \text{Random Error}$$

Standard form: Assume the $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon)$ and are independent

Parameters to estimate: β_0, β_1 and σ_ε

SLM Visualized



Conditions for SLM

Pattern

1. Mean Y at each X is a linear function of X:

$$\mu_Y(X) = f(X) = \beta_0 + \beta_1 X$$

Residuals

2. Zero mean: Residuals centered at 0
3. Constant variance: Same variability at all X (Homoskedasticity)
4. Independence: No relationship among errors
5. Normality (for standard form): At each X, Ys are Normally distributed

What to Do If Conditions are Violated?

What if we have...

- Lack of normality of residuals
- Patterns (e.g. curvature) in residuals
- Non-constant variance ("heteroscedasticity")
- Outliers: influential points, large residuals

Transformations and Outliers

Data Transformations

Can be used to

- "Unskew" residual distribution
- Address non-linearity
- Stabilize (homogenize) variance
- Reduce influence of outliers

Example: Year Length on Different Planets

Cases: Planets in our solar system

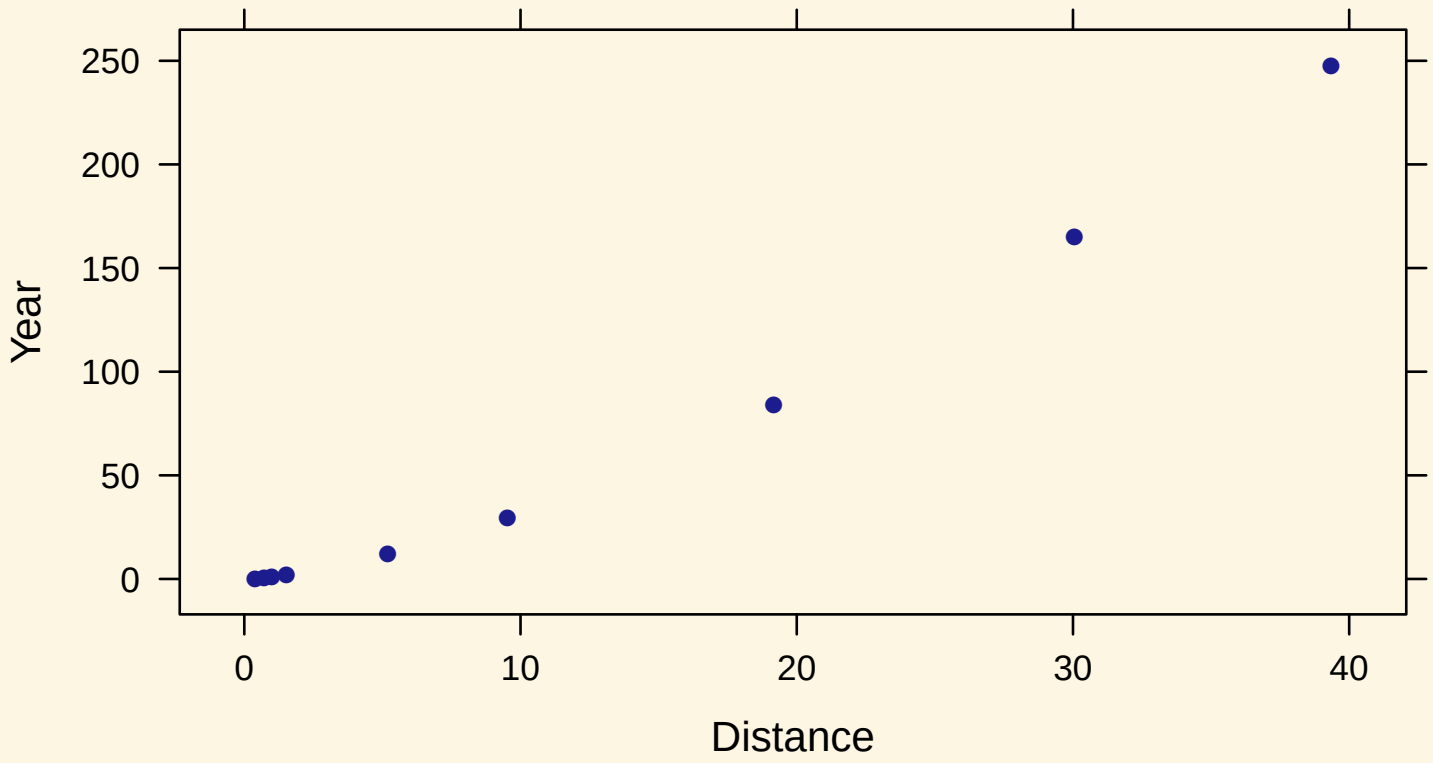
Y : Length (days) of a year on each planet

X : Distance (km) from the sun

Can we model Y as a function of X?

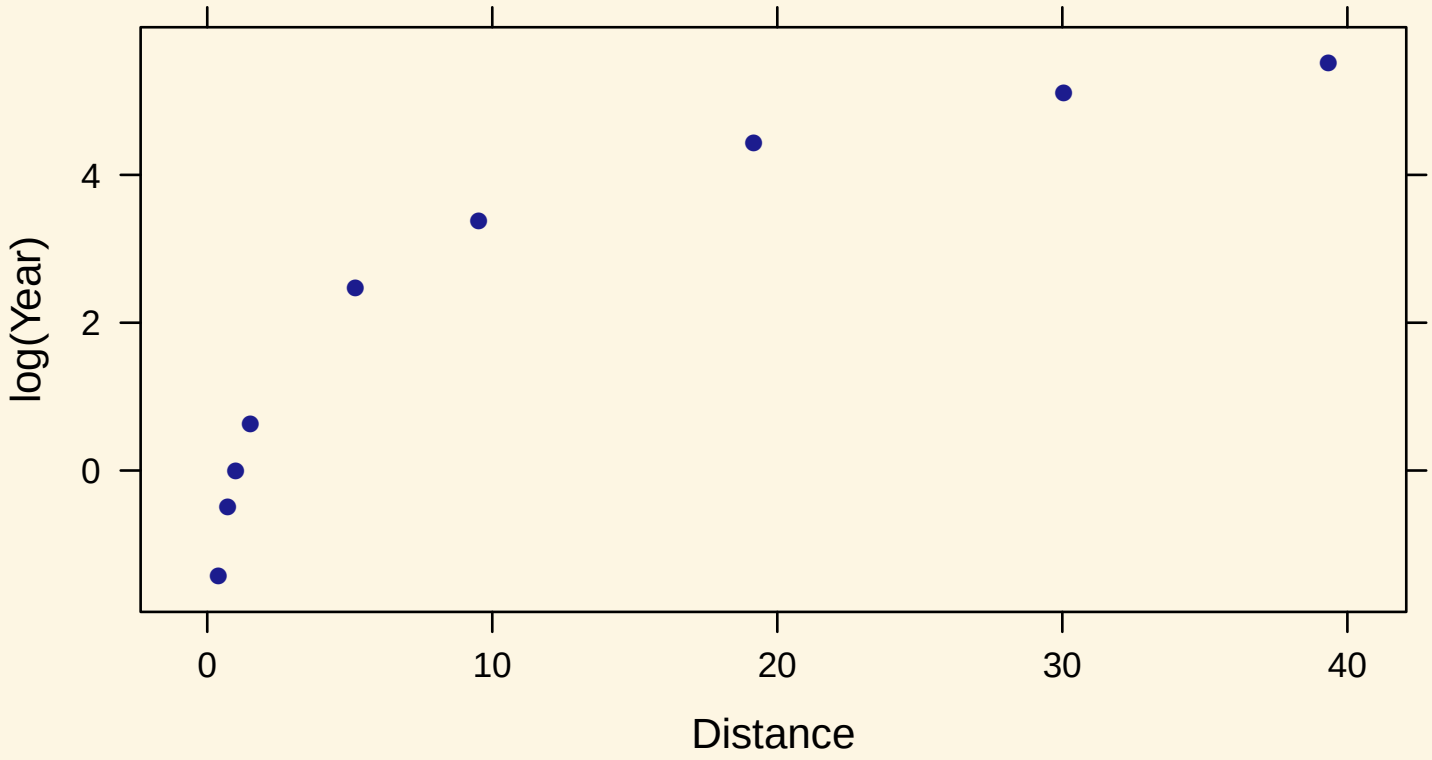
Example: Year Length on Different Planets

```
library("mosaic")
Planets <- read.file("http://colindawson.net/data/Planets.csv")
xyplot(Year ~ Distance, data = Planets) # not linear
```



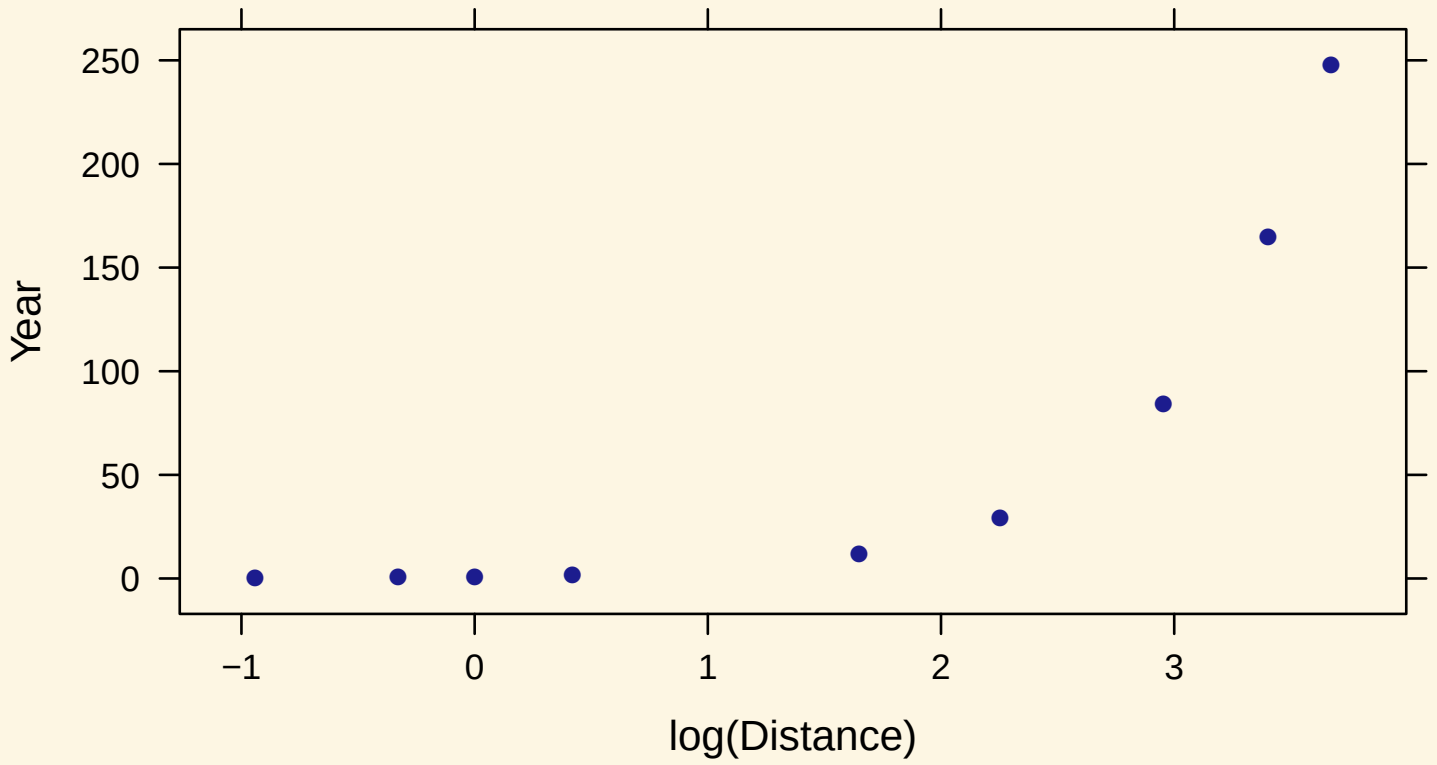
Transforming X and Y

```
xyplot(log(Year) ~ Distance, data = Planets) ## overcorrected
```



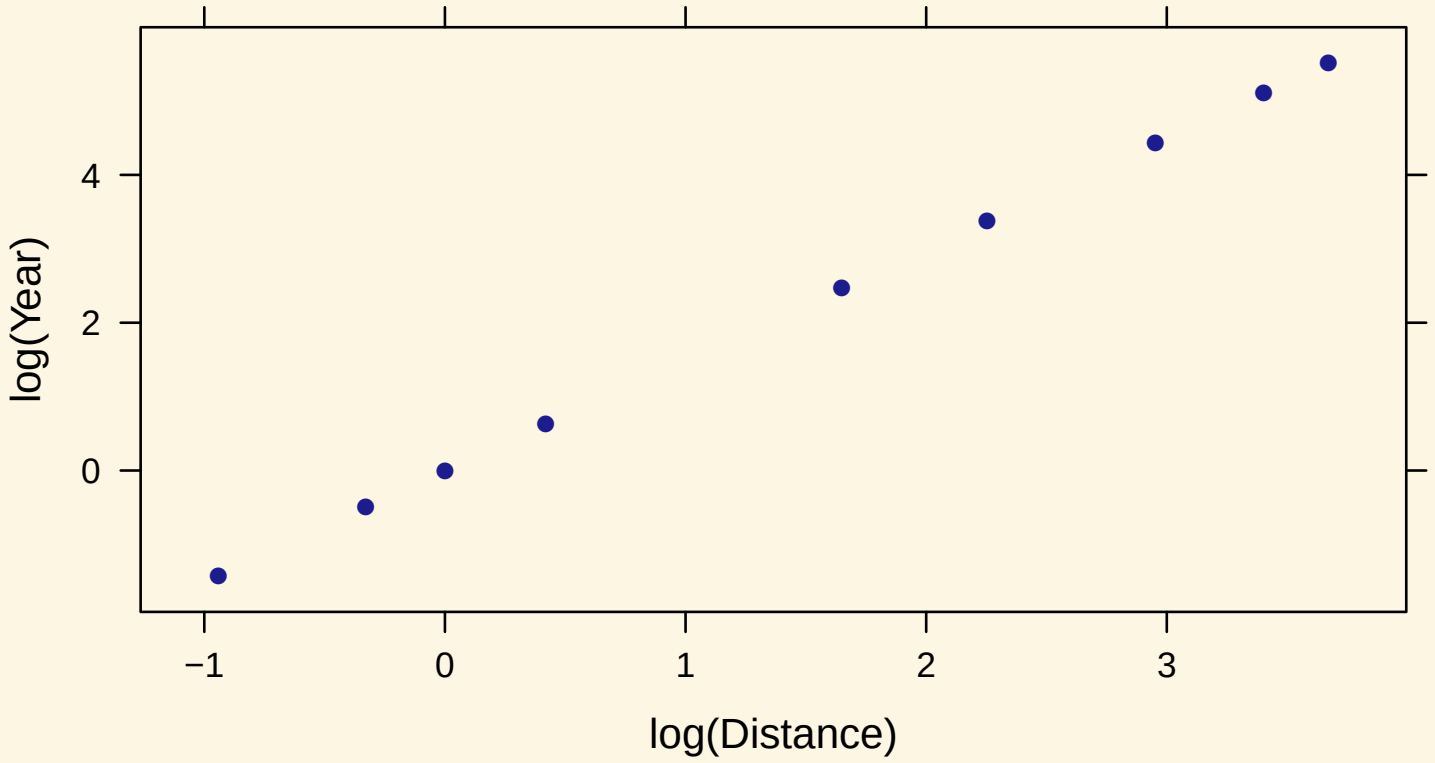
Transforming X and Y

```
xyplot(Year ~ log(Distance), data = Planets) ## wrong direction
```



Transforming X and Y

```
xyplot(log(Year) ~ log(Distance), data = Planets) ## linear!
```



Interpreting the Transformed Relationship

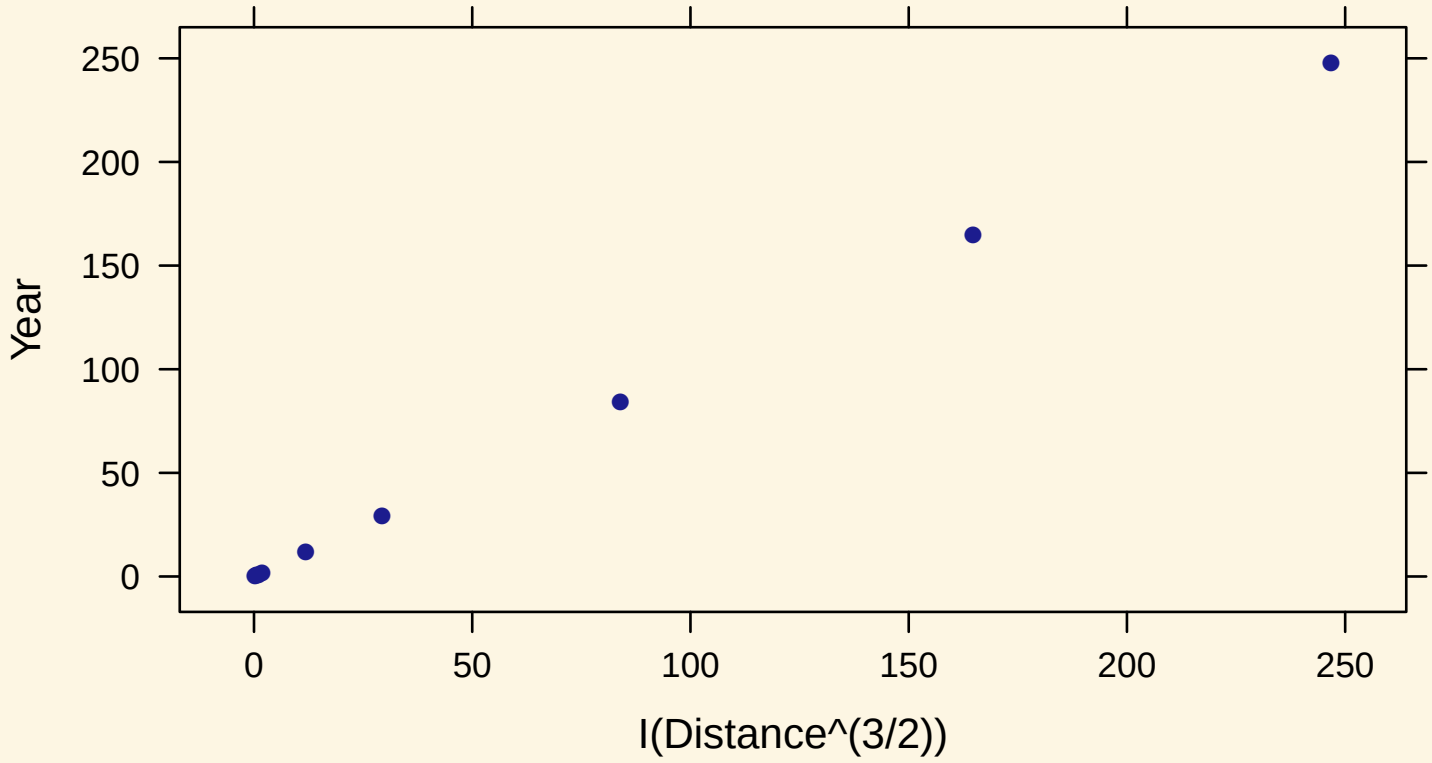
```
LogLogModel <- lm(log(Year) ~ log(Distance), data = Planets)
coef(LogLogModel)

      (Intercept) log(Distance)
     -0.003433939   1.502061101
```

- "For each one unit increase in $log(Distance)$ the log year length increases by 1.5 units"
- Or, approximately: "Year length is proportional to Distance to the 1.5 power"

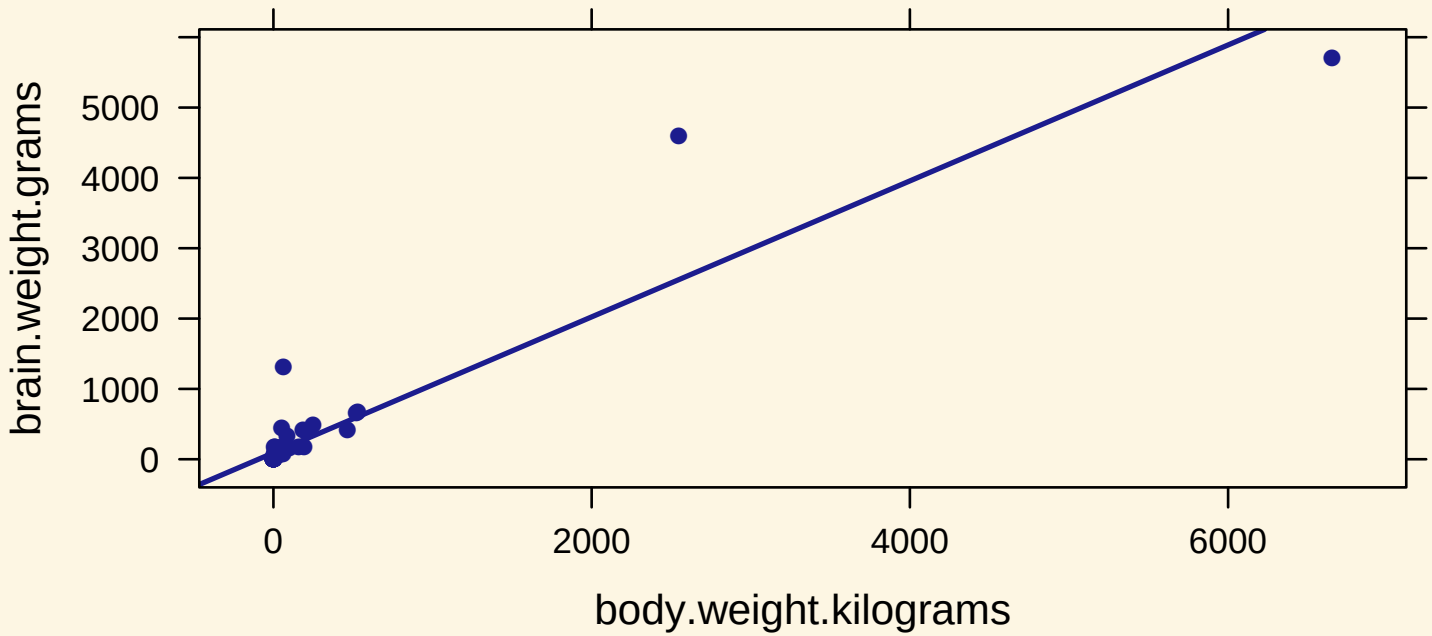
Year Length and Distance

```
xyplot(Year ~ I(Distance^(3/2)), data = Planets)
```



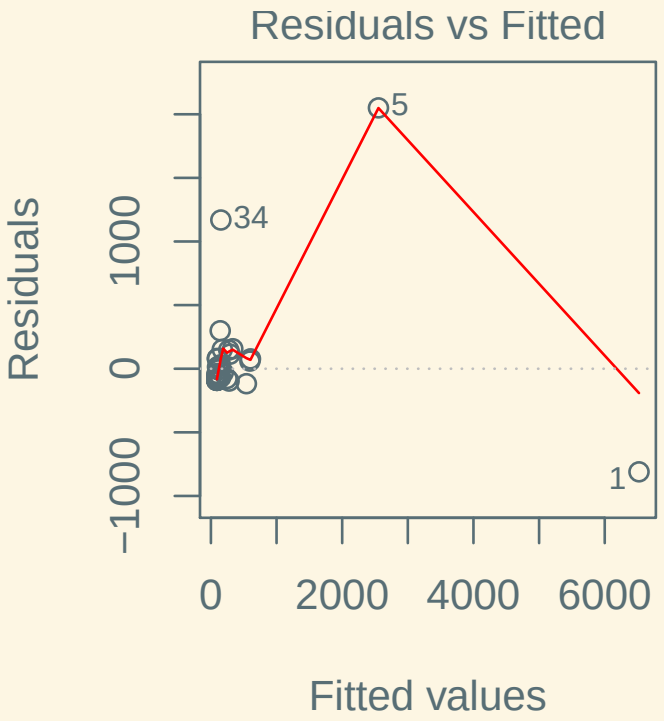
Brain and Body Weight of Terrestrial Mammals

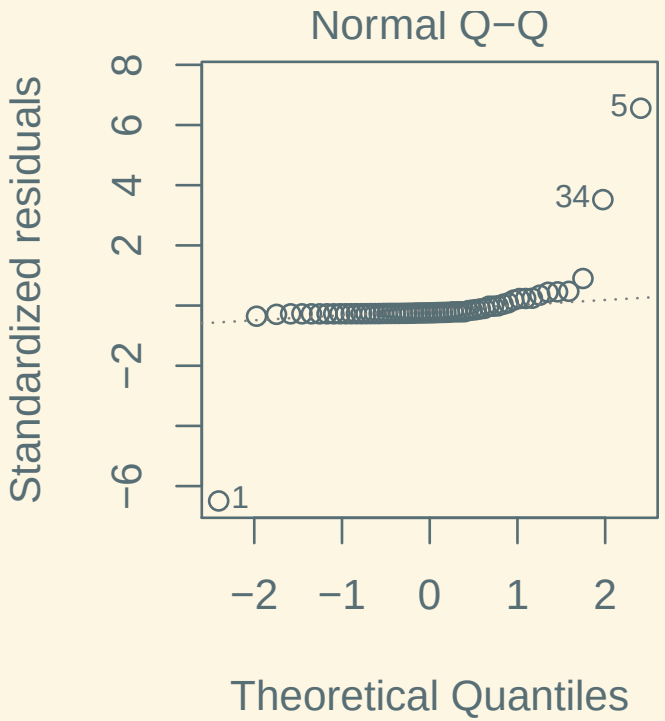
```
library("mosaic")
BrainBodyWeight <- read.file("http://colindawson.net/data/BrainBodyWeight.csv")
xyplot(
    brain.weight.grams ~ body.weight.kilograms,
    data = BrainBodyWeight, type = c("p", "r"))
```



Brain and Body Weight of Terrestrial Mammals

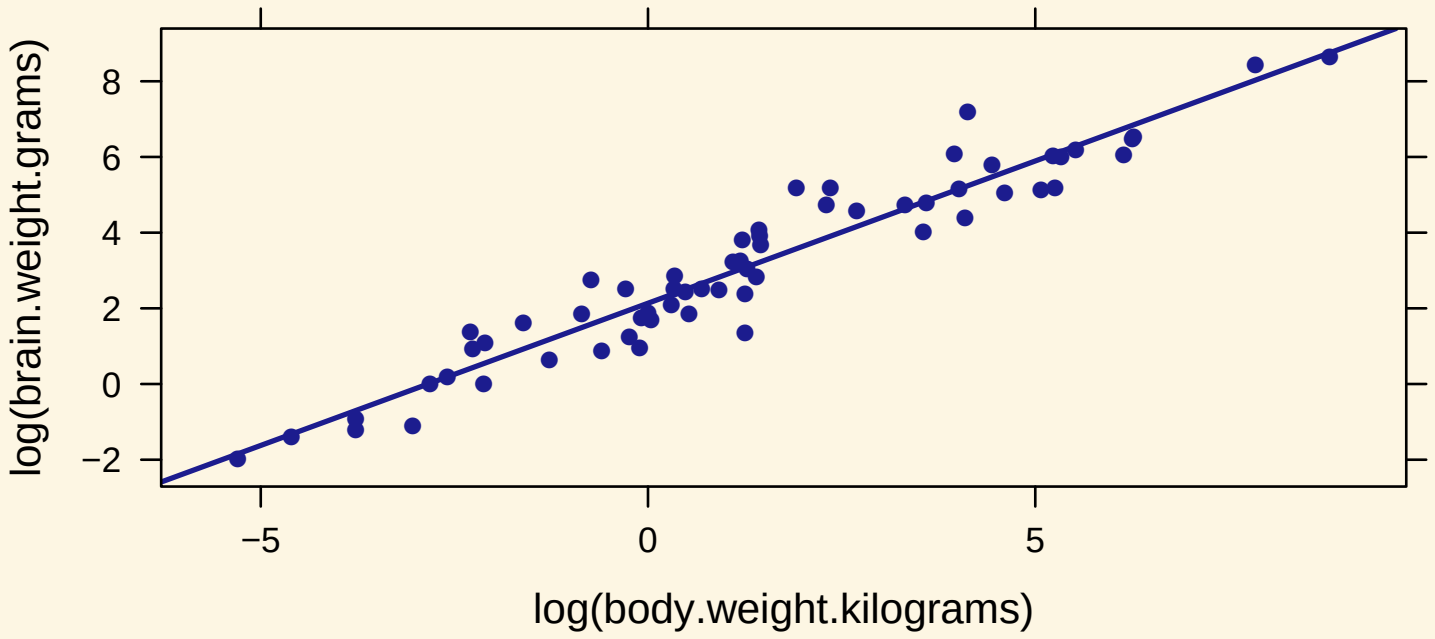
```
brain.model <-
    lm(brain.weight.grams ~ body.weight.kilograms, data = BrainBodyWeight)
par(mfrow = c(1,2)) # to create a 1-by-2 plotting grid
plot(brain.model, which = 1) #residuals by predicted
plot(brain.model, which = 2) #quantile-quantile
```





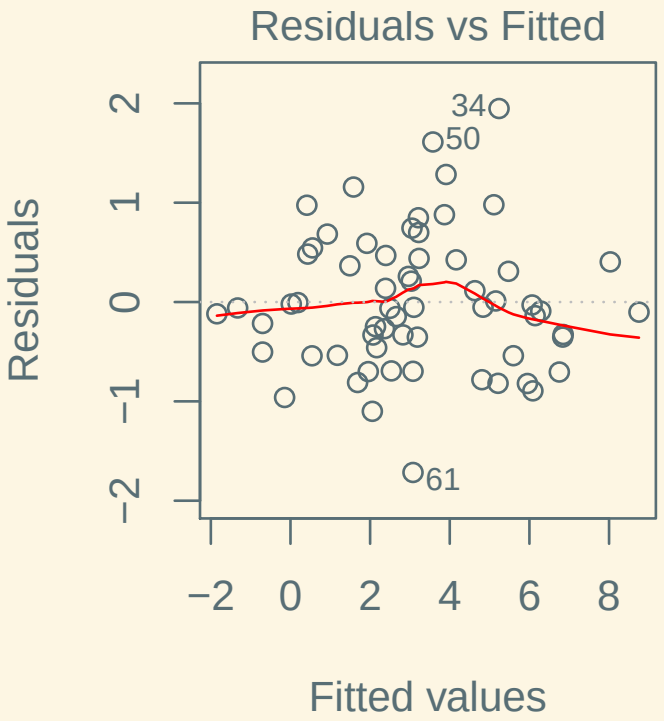
Log Brain and Log Body Weight

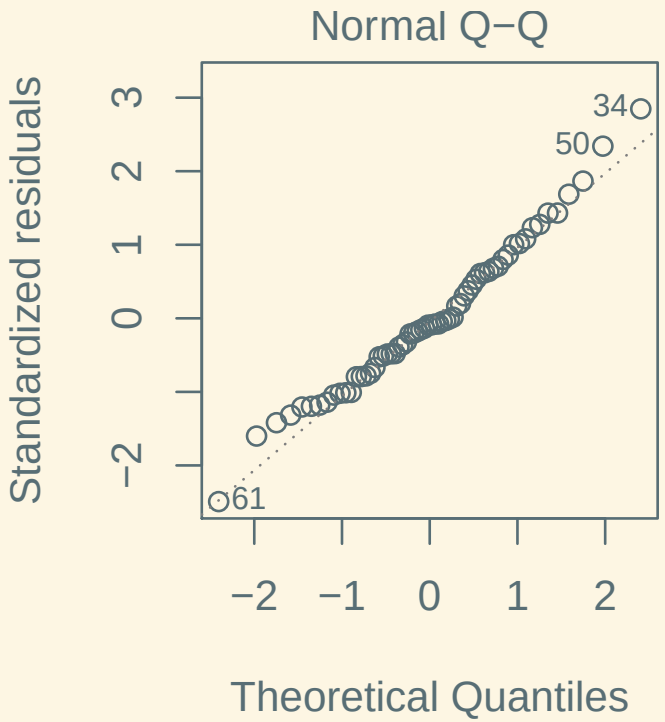
```
xyplot(
    log(brain.weight.grams) ~ log(body.weight.kilograms),
    data = BrainBodyWeight, type = c("p", "r"))
```



Log Brain and Log Body Weight

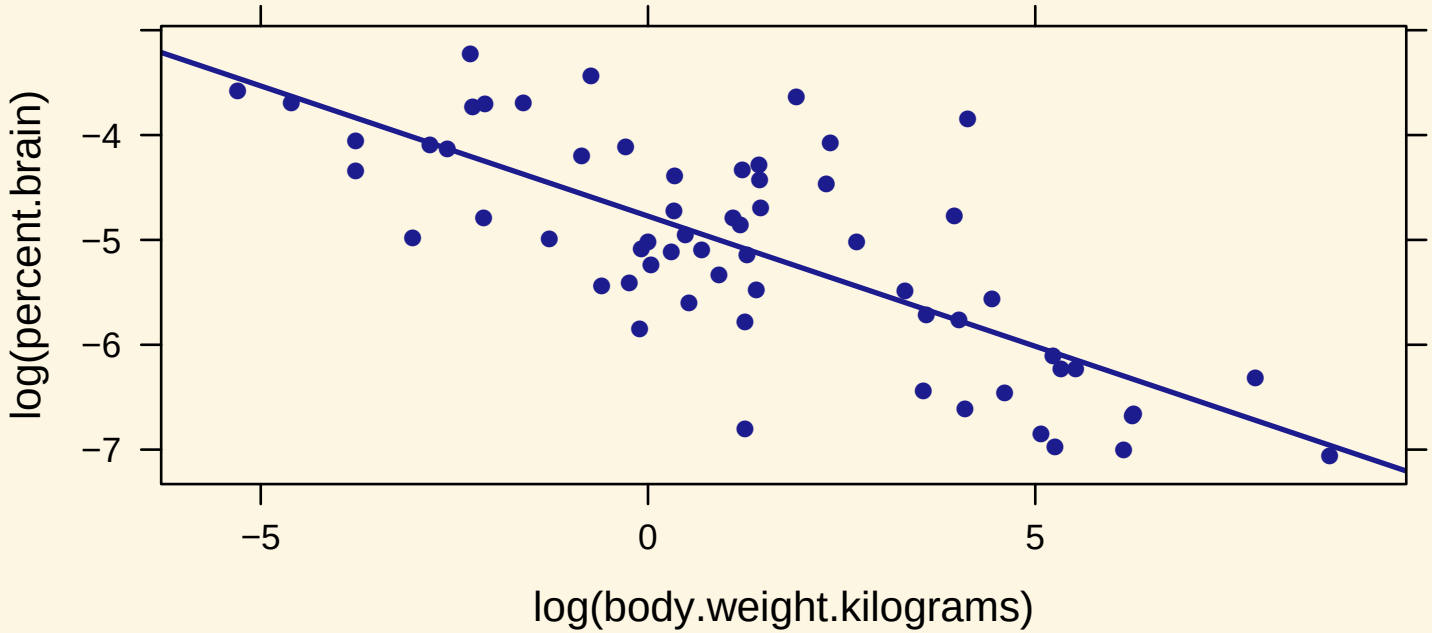
```
log.brain.model <-
    lm(log(brain.weight.grams) ~ log(body.weight.kilograms),
       data = BrainBodyWeight)
par(mfrow = c(1,2))
plot(log.brain.model, which = 1) #residuals by predicted
plot(log.brain.model, which = 2) #quantile-quantile
```



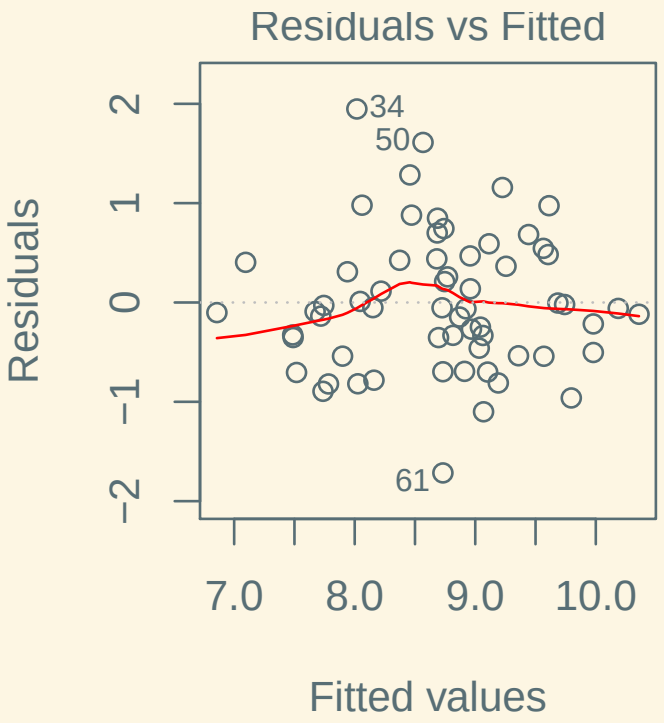


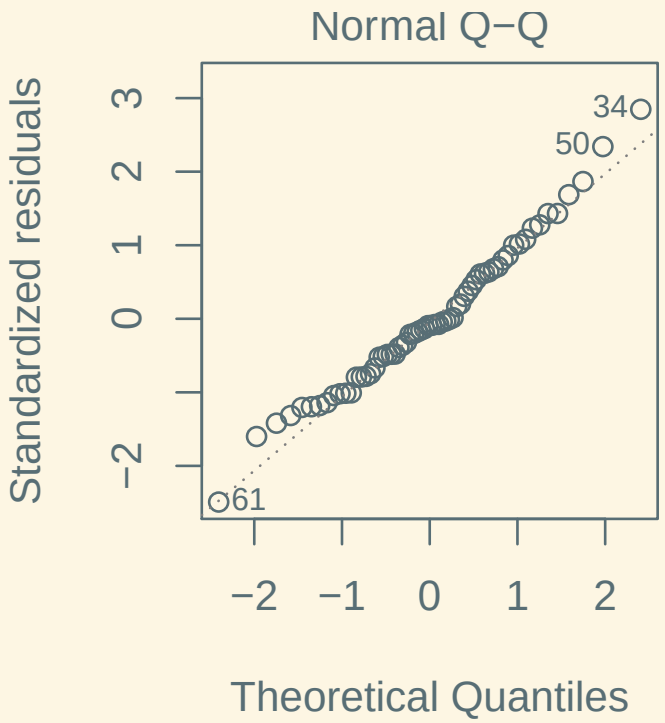
Percent Brain Weight by Body Weight

```
library(mosaic)
mutate(
    BrainBodyWeight,
    percent.brain = brain.weight.grams / (body.weight.kilograms * 1000)
    ) %>%
    xyplot(
        log(percent.brain) ~ log(body.weight.kilograms),
        data = ., type = c("p", "r"))
```



Percent Brain Weight By Body Weight





Influential Points

Handout

Unusual Cases

Influential point

An **influential point** is a data point that by itself has a large effect on the fitted regression line.

How do we measure "effect on the fit"?

Kinds of Influential Points

Outliers

An **outlier** is a data point that is unusually far from the trend line (i.e., it has an unusually large residual).

Leverage

A point has high **leverage** if it has an unusually extreme value on a predictor (explanatory variable). (Think of a see-saw)

Unusual Cases

Detecting Unusual Cases

- Residual plots
- Standardized/Studentized residuals
- Leverage measurement

Raw vs. Standardized Residuals

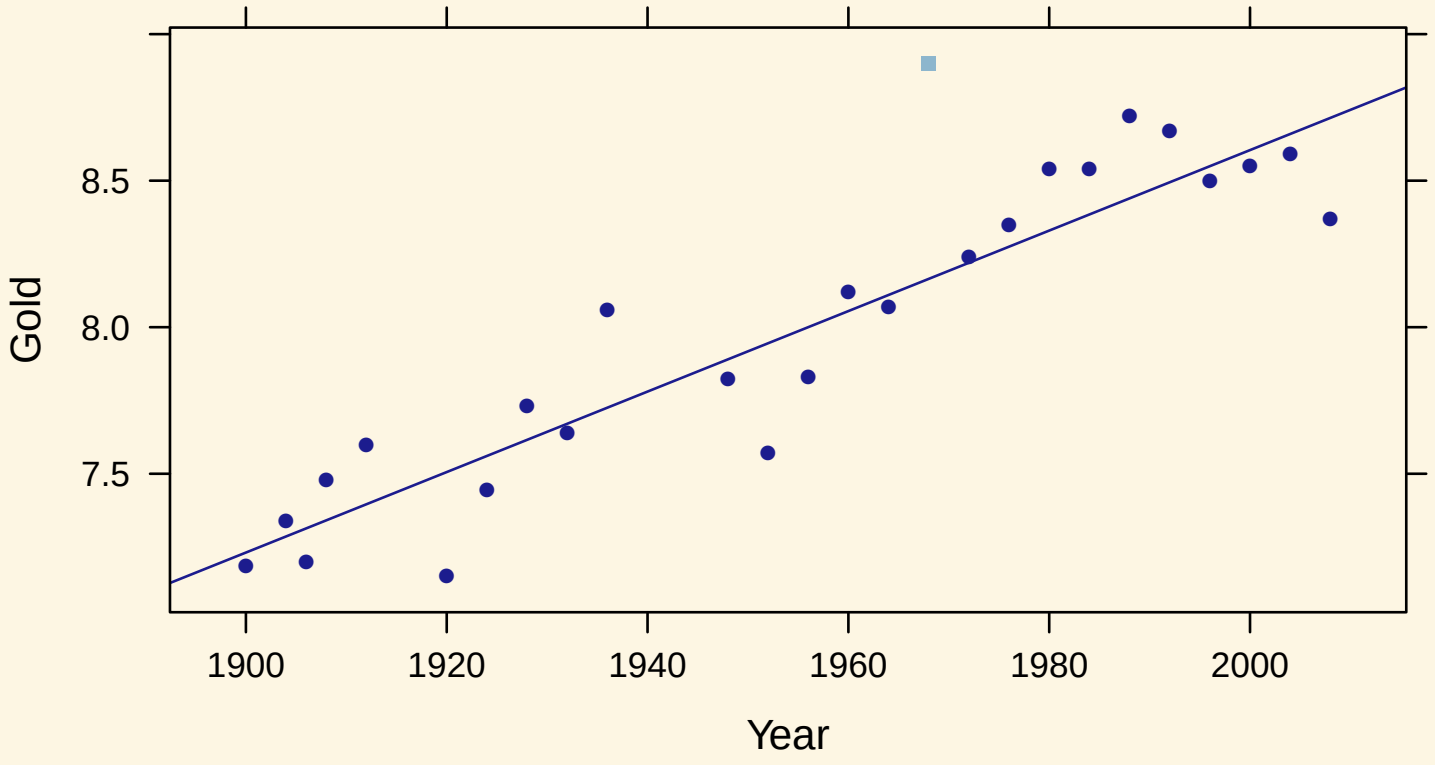
How do we tell if a residual is unusually large?

$Y = GPA$ $e_i = 2.6$ is very large

$Y = SAT$ $e_i = 2.6$ is very small

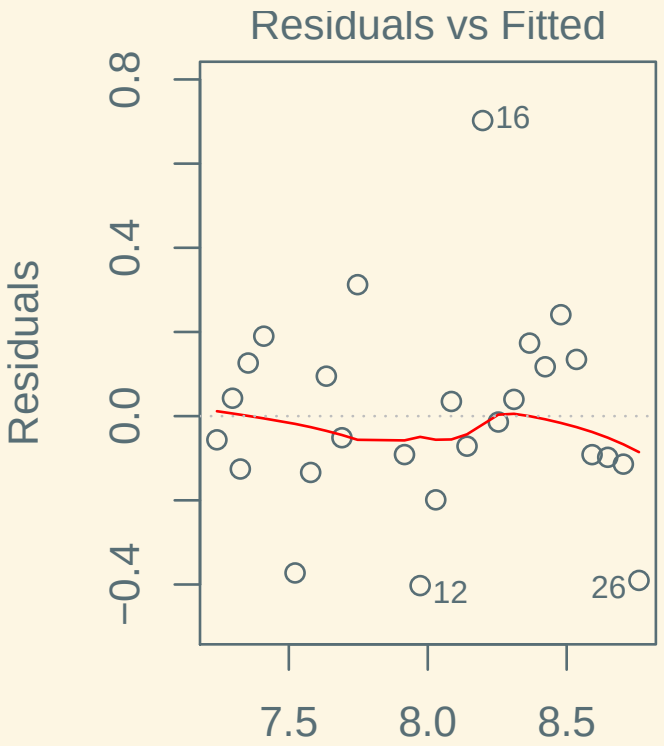
Men's Long Jump

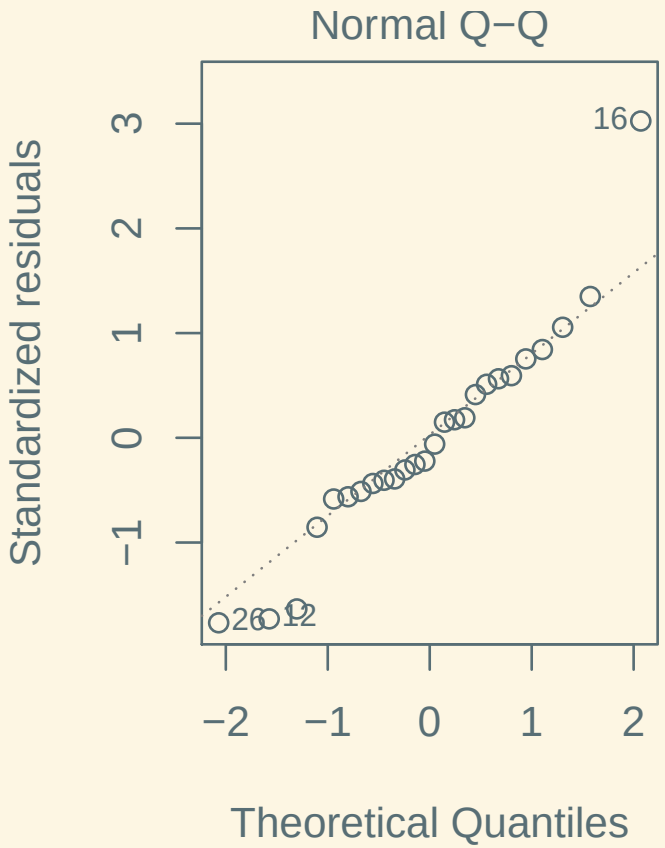
```
library(Stat2Data)
data(LongJumpOlympics)
xyplot(
    Gold ~ Year, data = LongJumpOlympics, type = c("p", "r"),
    groups = (Year == 1968)  ## highlight the outlier
    )
```



Men's Long Jump: Residuals

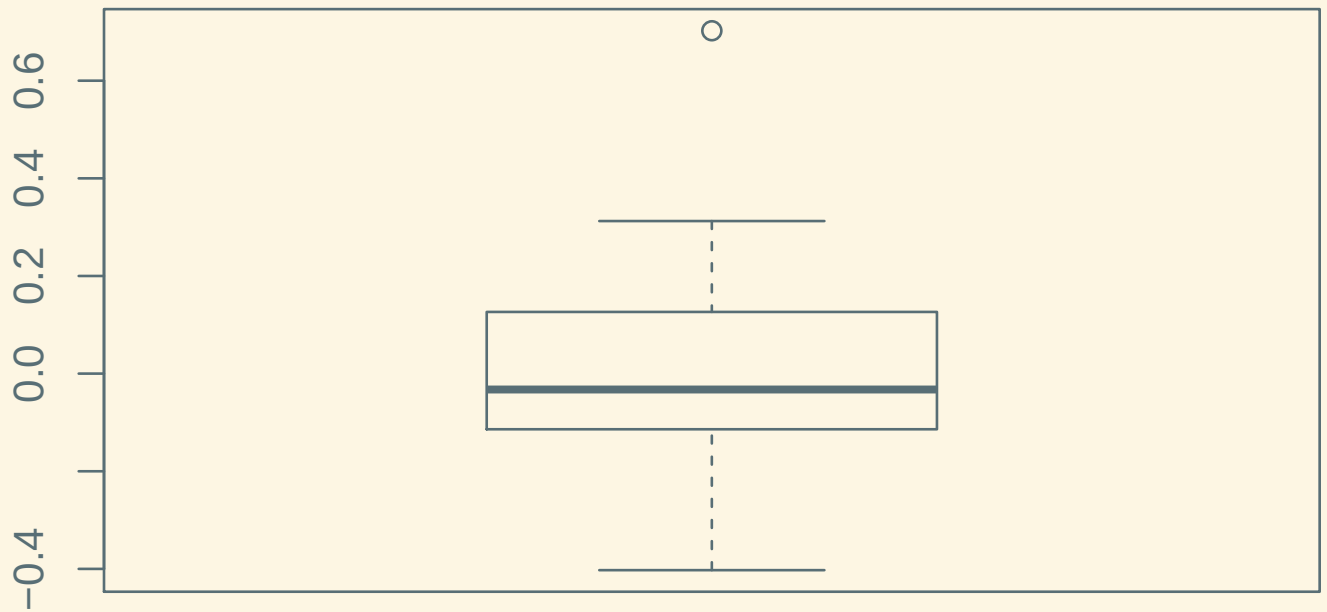
```
long.jump.model <- lm(Gold ~ Year, data = LongJumpOlympics)
par(mfrow = c(1,2))
plot(long.jump.model, which = 1)
plot(long.jump.model, which = 2)
```





Men's Long Jump: Box Plot of Residuals

```
boxplot(residuals(long.jump.model))
```



The outlier has a residual which is more than one and a half times the IQR beyond the upper quartile.

Standardized and Studentized Residuals

Standardized Residuals

$$\frac{y_i - \hat{y}_i}{\hat{\sigma}_\varepsilon \sqrt{1 - h_i}} \tag{1}$$

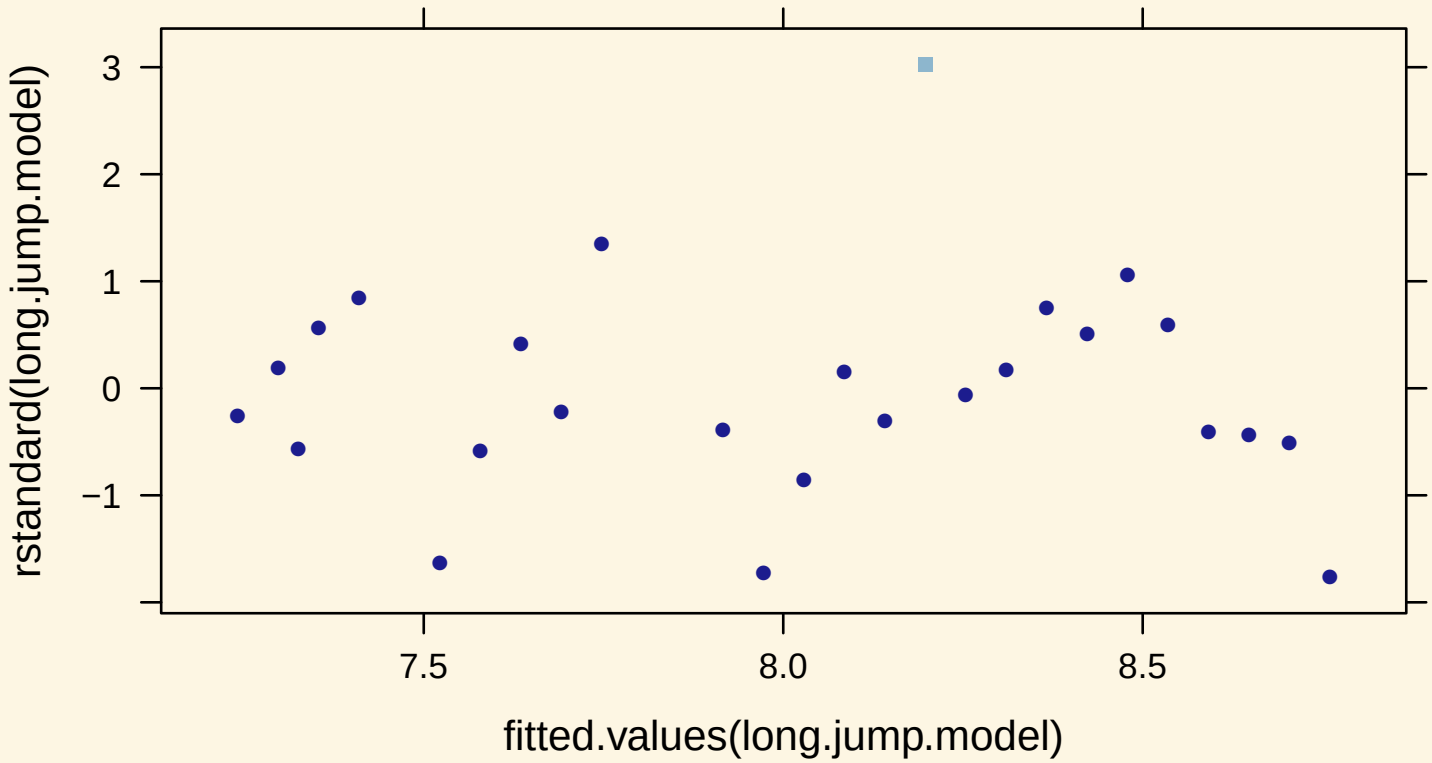
"Studentized" Residuals

$$\frac{y_i - \hat{y}_i}{\hat{\sigma}_\varepsilon^{(i)} \sqrt{1 - h_i}} \tag{2}$$

where $\hat{\sigma}_\varepsilon^{(i)}$ is standard deviation of all residuals *other than* i, and h_i is called the "leverage" of point i (more on this soon)

Standardized Residuals

```
xyplot(rstandard(long.jump.model) ~ fitted.values(long.jump.model),
       groups = abs(rstandard(long.jump.model)) > 2)
```



Standardized Residuals

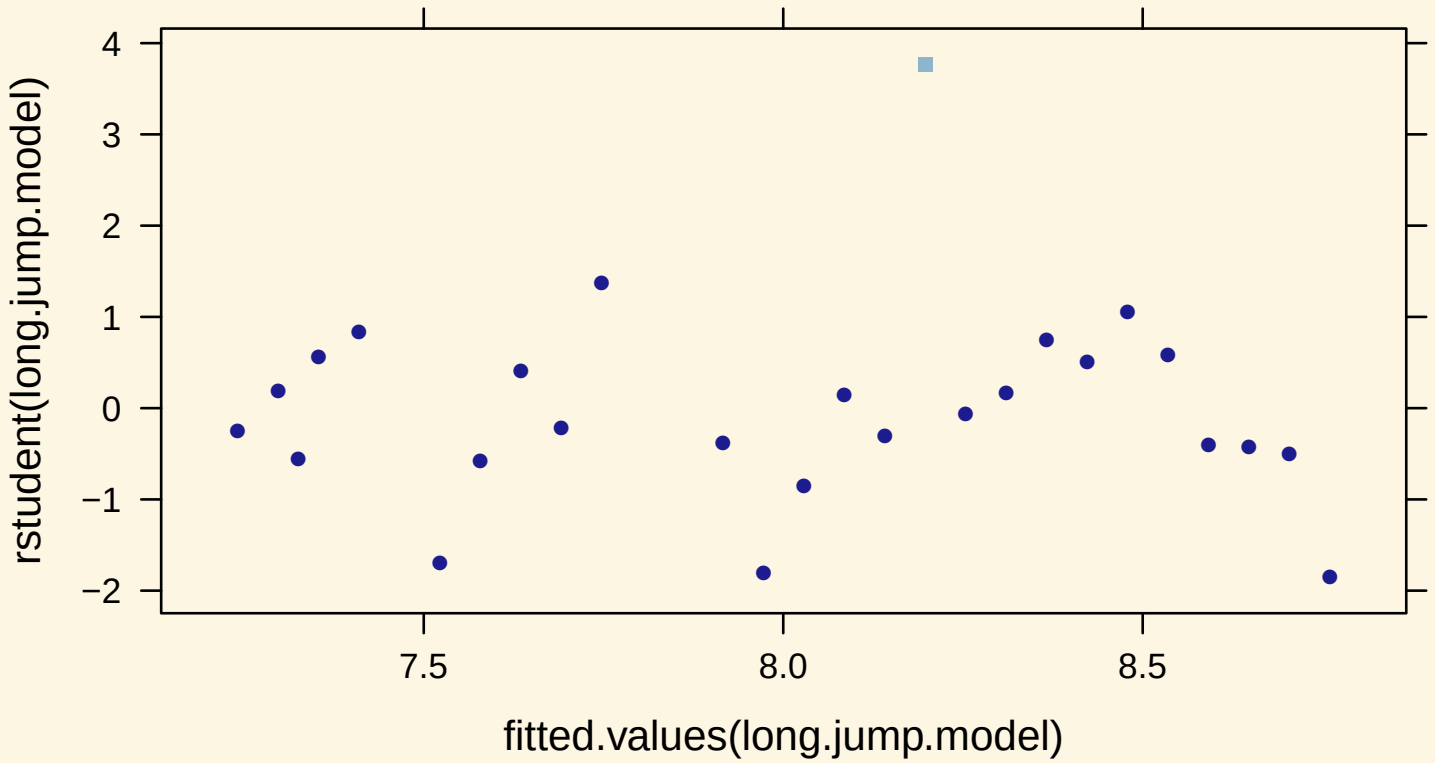
Keep an eye out for standardized residuals in excess of ± 2 or ± 3. Under normality, only 5% will exceed ± 2, and less than 0.2% exceed ± 3.

Studentized Residuals

- Worry: a large residual will inflate the estimated standard deviation of the residuals, and therefore lead to understimating the standardized residual.
- Solution: "Studentize" the residuals by fitting the model with that case removed, then find new $\hat{\sigma}_\varepsilon$.

Studentized Residuals

```
xyplot(rstudent(long.jump.model) ~ fitted.values(long.jump.model),
       groups = abs(rstudent(long.jump.model)) > 2)
```



Influence

Two characteristics contribute to influence of a data point on regression line:

1. Distance in Y from trend (think: residual for line fit w/o that point)
2. Distance of X from $\bar{X}$ (think: distance from center on a see-saw)

Leverage

Leverage

$$h_i = \frac{1}{n}\left(1 + \frac{(x_i - \bar{x})^2}{\frac{1}{n}\sum_{i'=1}^{n}(x_{i'} - \bar{x})^2}\right) \approx \frac{1}{n}\left(1 + z_{x,i}^2\right)$$

where $z_{x,i}$ is the z-score of the ith data point in the x direciton. h_i measures influence of a point with value x_i on regression line.

Typical leverage: $2/n$. Keep an eye out for leverage in excess of $4/n$ or $6/n$.

Standardized and Studentized Residuals

Standardized Residuals

$$\frac{y_i - \hat{y}_i}{\hat{\sigma}_\varepsilon \sqrt{1 - h_i}} \tag{3}$$

"Studentized" Residuals

$$\frac{y_i - \hat{y}_i}{\hat{\sigma}_\varepsilon^{(i)} \sqrt{1 - h_i}} \tag{4}$$

where $\hat{\sigma}_\varepsilon^{(i)}$ is standard deviation of all residuals *other than* i, and h_i is the leverage of point i (more on this soon).

Cook's Distance

Annother measure of influence that combines leverage and size of residual is **Cook's Distance**.

Cook's Distance

$$D_i := \frac{\sum_{j=1}^{J} (\hat{y}_j - \hat{y}_{j(i)})^2}{\hat{\sigma}^2}$$

Equivalently

$$D_i := \frac{\hat{\varepsilon}_i}{\hat{\sigma}^2} \left[\frac{h_i}{(1-h_i)^2} \right]$$

where $\hat{y}_j$ is the predicted value of data point j with all data, $\hat{y}_{j(i)}$ is the predicted value if data point i is removed, $\hat{\varepsilon}_i$ is the estimated residual of data point i, and h_i is the leverage.

Cook's Distance for Long Jumps

```
plot(long.jump.model, which = 4)
```