# Transformations

## Reminder of Linear Model Assumptions (and Why)

1. Relationship is linear
   - Critical if we're using a line, but…
   - If not, can fit a polynomial or use other methods discussed later in this class
2. Observations are independent
   - Necessary for inference (hypothesis test results and confidence intervals) to be correct
   - Predictions could still be OK: as $n \to \infty$, we still will recover the correct relationship between explanatory and response variables
3. Residuals follow a normal distribution
   - Necessary for hypothesis test results and confidence intervals to be correct
     - Mild skewness or short tails are OK if sample size is moderately large. Heavy tails or extreme skewness are problematic.
   - Predictions could still be OK: as $n \to \infty$, we still will recover the correct relationship between explanatory and response variables
   - If residual distribution is not normal, estimation methods other than least squares could have lower variance
4. Residuals have equal variance for all observations (homoskedastic)
   - Necessary for hypothesis test results and confidence intervals to be correct
   - Predictions could still be OK: as $n \to \infty$, we still will recover the correct relationship between explanatory and response variables
   - Estimation methods other than least squares could result in lower variance
5. No outliers/observations with high leverage
   - Could result in incorrect inferences and predictions, especially if $n$ is small.

Summary: Mostly, these problems result in…

- A loss of guarantees of correct Type I Error rates for hypothesis tests
- A loss of guarantees of correct coverage rates for confidence intervals
- Higher-than-necessary variance for parameter estimates and predictions

Our Goal:

- Fix problems with residuals (non-normal, heteroskedastic/unequal variance), and maybe also outliers.
- As a side effect, sometimes also make relationships more linear

Method: Transform the variables.

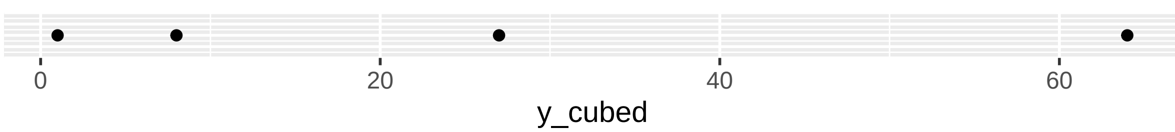
## The Ladder of Powers for Transformations

- Imagine a "ladder of powers" of $y$ (or $x$): We start at $y$ and go up or down the ladder.

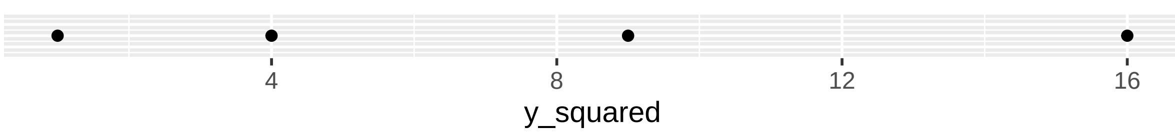
| Transformation | R Code | Comments |
|---|---|---|
| $\vdots$ | | |
| $e^y$ | `exp(y)` | Exactly where on the ladder the exponential transformation belongs depends on the magnitude of the data, but somewhere around here... |
| $y^2$ | `y^2` | |
| $y$ | | Start here (no transformation) |
| $\sqrt{y}$ | `sqrt(y)` | |
| $y^{\text{"}0\text{"}}$ | `log(y)` | We use $\log(y)$ here |
| $-1/\sqrt{y}$ | `-1/sqrt(y)` | The $-$ keeps the values of $y$ in order |
| $-1/y$ | `-1/y` | |
| $-1/y^2$ | `-1/y^2` | |
| $\vdots$ | | |

- Which direction?
  - If a variable is skewed right, move it down the ladder (pull down large values)
  - If a variable is skewed left, move it up the ladder (pull up small values)

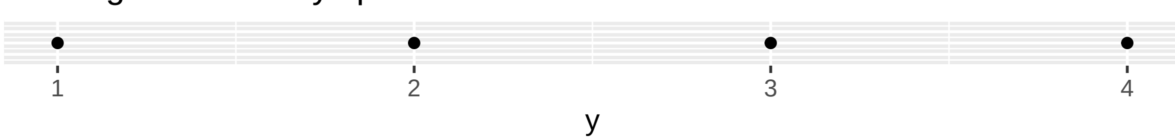
Moved Up 2 Steps: spread out points on the right side



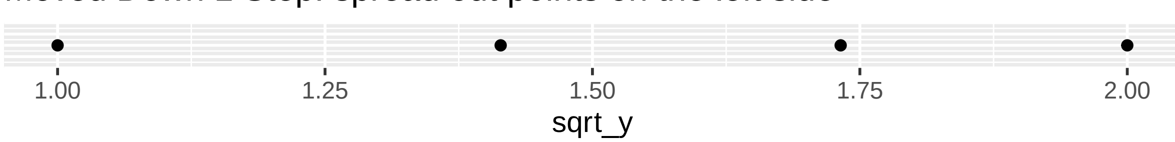

Moved Up 1 Step: spread out points on the right side



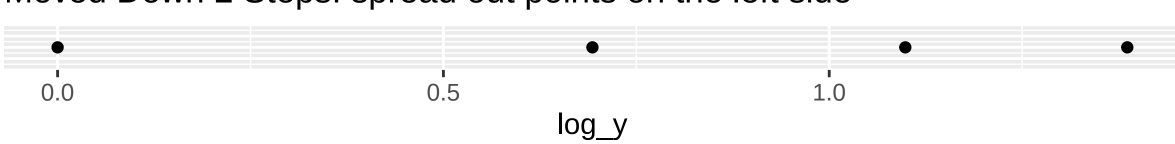

Starting Point: evenly spaced




Moved Down 1 Step: spread out points on the left side

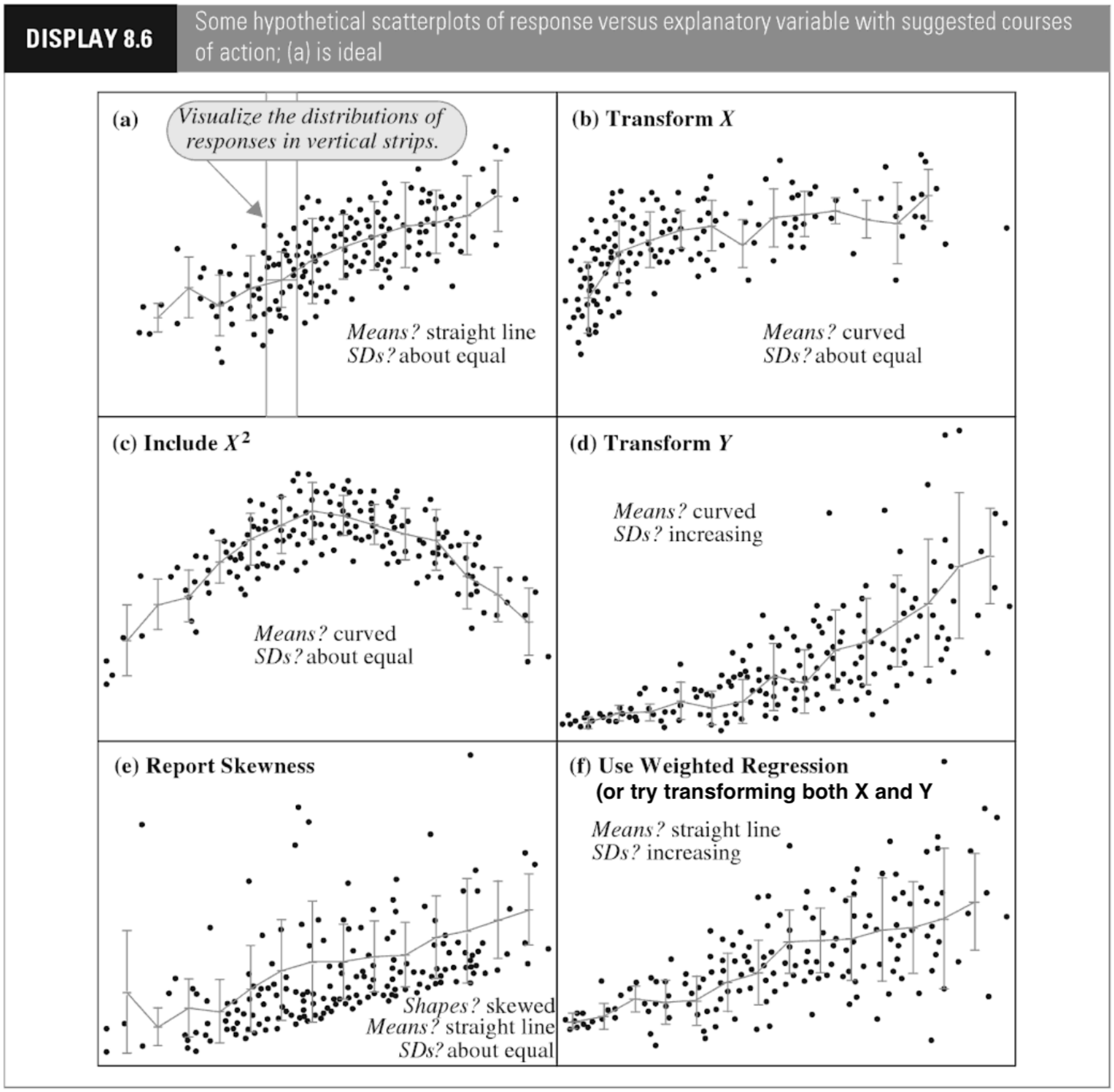



Moved Down 2 Steps: spread out points on the left side



**What to do is based on scatter plots**

Figure from The Statistical Sleuth.




**Start with the response**

Start exploring transformations by looking at the response variable, looking to fix: * Residuals skewed * Non-constant variance (heteroskedasticity)

### Example

Let's look at modeling a movie's international gross earnings in inflation-adjusted 2013 dollars (`intgross_2013`). For today, let's just think about using a single quantitative explanatory variable, `budget_2013`.

Here we read the data in and fit a simple linear regression model.

```
library(readr)
library(dplyr)
library(ggplot2) # general plotting functionality
library(GGally) # includes the ggpairs function, pairs plots via ggplot2
library(gridExtra) # for grid.arrange, which arranges the plots next to each other

options(na.action = na.exclude, digits = 7)

movies <- read_csv("http://www.evanlray.com/data/bechdel/bechdel.csv") %>%
  filter(mpaa_rating %in% c("G", "PG", "PG-13", "R"),
    !is.na(intgross_2013),
    !is.na(budget_2013))
```

### Function for Model Fitting and Plotting Diagnostics

We're about to fit a bunch of different models and look at residual diagnostic plots for them all. Since we want to do slight variations on the same thing a bunch of times, we should make a function!

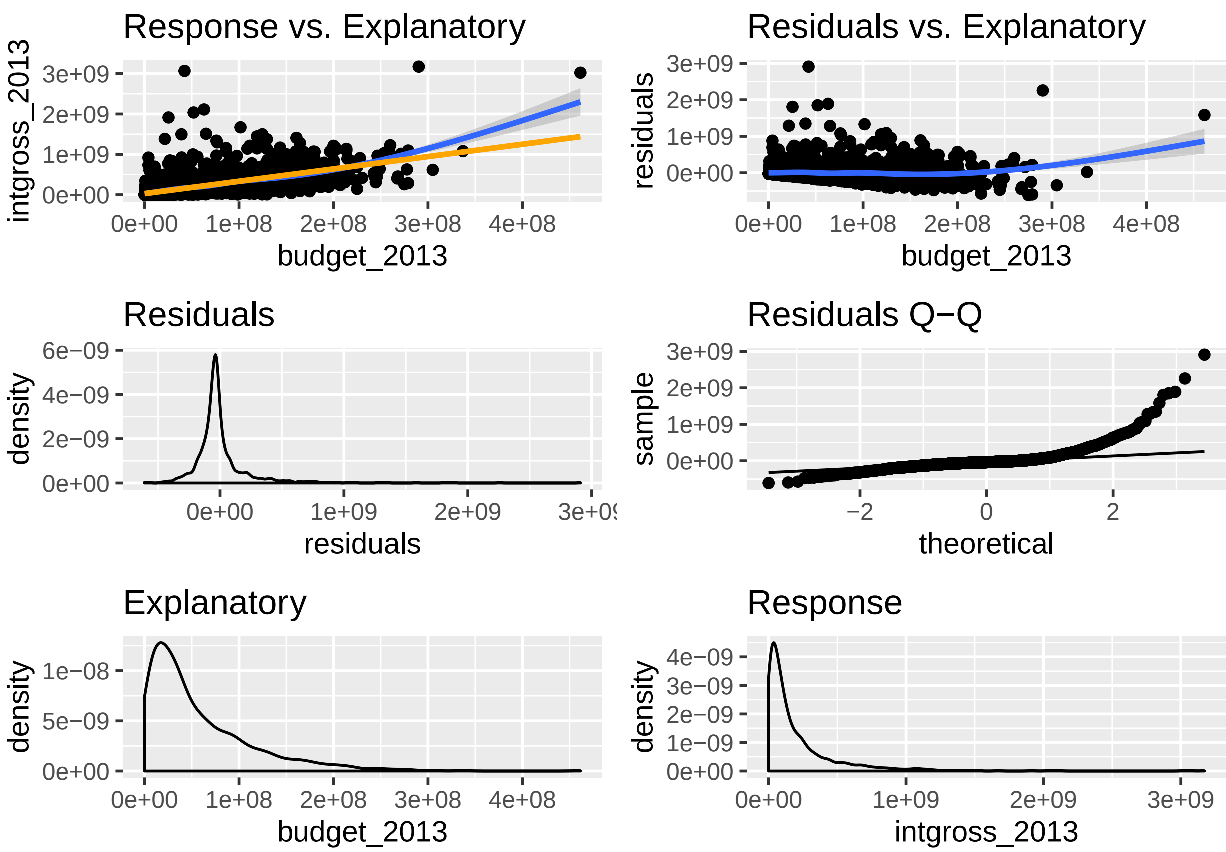
```
#' Fit a linear model with specified response and explanatory variables in the movies data set
#'
#' @param response character: response variable name
#' @param explanatory character: explanatory variable name
fit_model_and_make_plots <- function(response, explanatory) {
  fit_formula <- as.formula(paste0(response, " ~ ", explanatory))
  fit <- lm(fit_formula, data = movies)

  movies <- movies %>%
    mutate(
      residuals = residuals(fit),
      fitted = predict(fit)
    )

  p1 <- ggplot(data = movies, mapping = aes_string(x = explanatory, y = response)) +
    geom_point() +
    geom_smooth() +
    geom_smooth(method = "lm", color = "orange", se = FALSE) +
    ggtitle("Response vs. Explanatory")

  p2 <- ggplot(data = movies, mapping = aes_string(x = explanatory, y = "residuals")) +
    geom_point() +
    geom_smooth() +
    ggtitle("Residuals vs. Explanatory")

  p3 <- ggplot(data = movies, mapping = aes(x = residuals)) +
    geom_density() +
    ggtitle("Residuals")

  p4 <- ggplot(data = movies, mapping = aes(sample = residuals)) +
    stat_qq() +
    stat_qq_line() +
    ggtitle("Residuals Q-Q")
```

```r
  p5 <- ggplot(data = movies, mapping = aes_string(x = explanatory)) +
    geom_density() +
    ggtitle("Explanatory")

  p6 <- ggplot(data = movies, mapping = aes_string(x = response)) +
    geom_density() +
    ggtitle("Response")

  grid.arrange(p1, p2, p3, p4, p5, p6, ncol = 2)
}
```

**Linear Fit**

```r
fit_model_and_make_plots(response = "intgross_2013", explanatory = "budget_2013")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

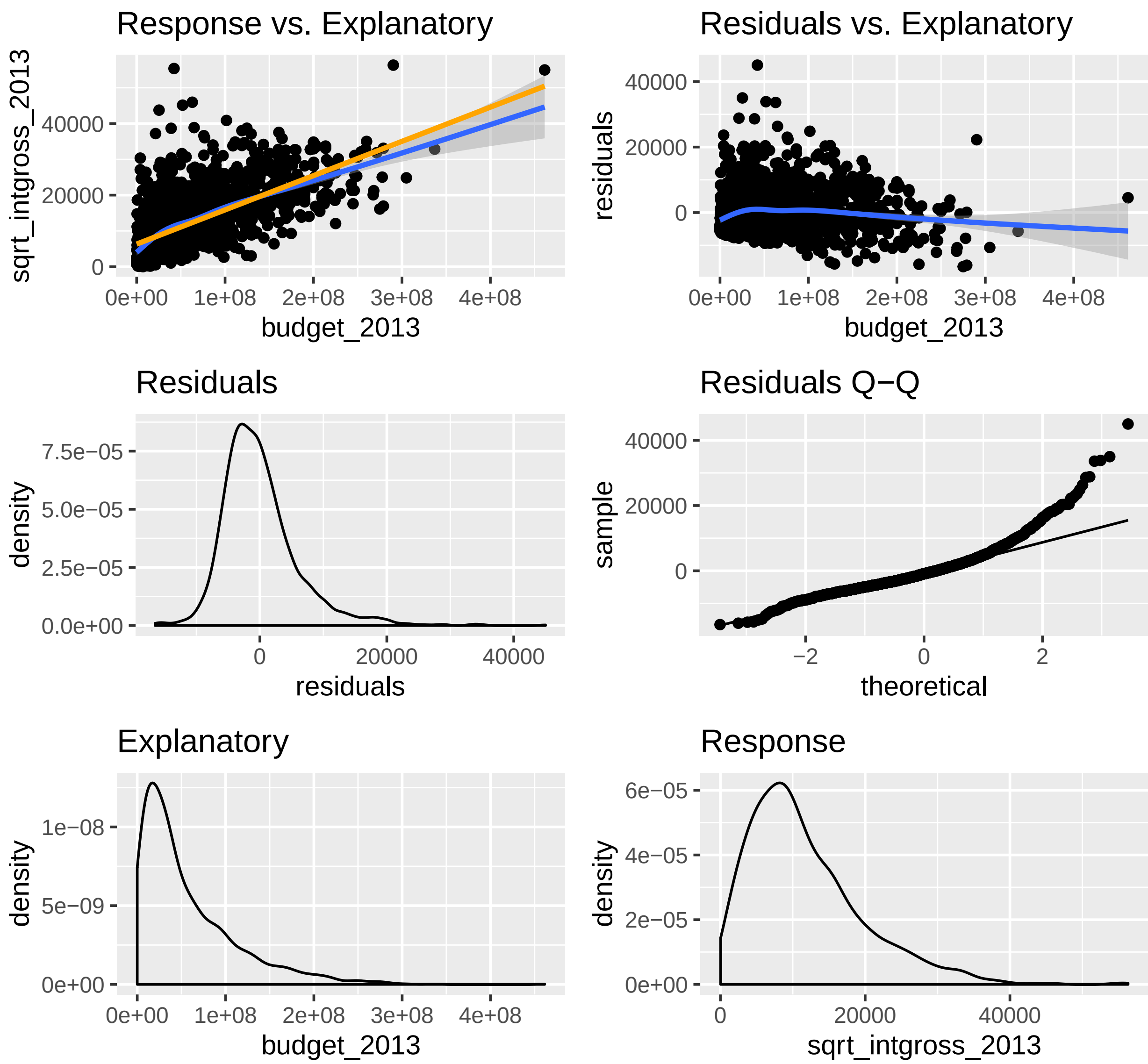



**In our example, what are the problems and how are we going to fix them?**

### Trying $\sqrt{\text{intgross\_2013}}$

```
movies <- movies %>% mutate(
  sqrt_intgross_2013 = sqrt(intgross_2013)
)

fit_model_and_make_plots(response = "sqrt_intgross_2013", explanatory = "budget_2013")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

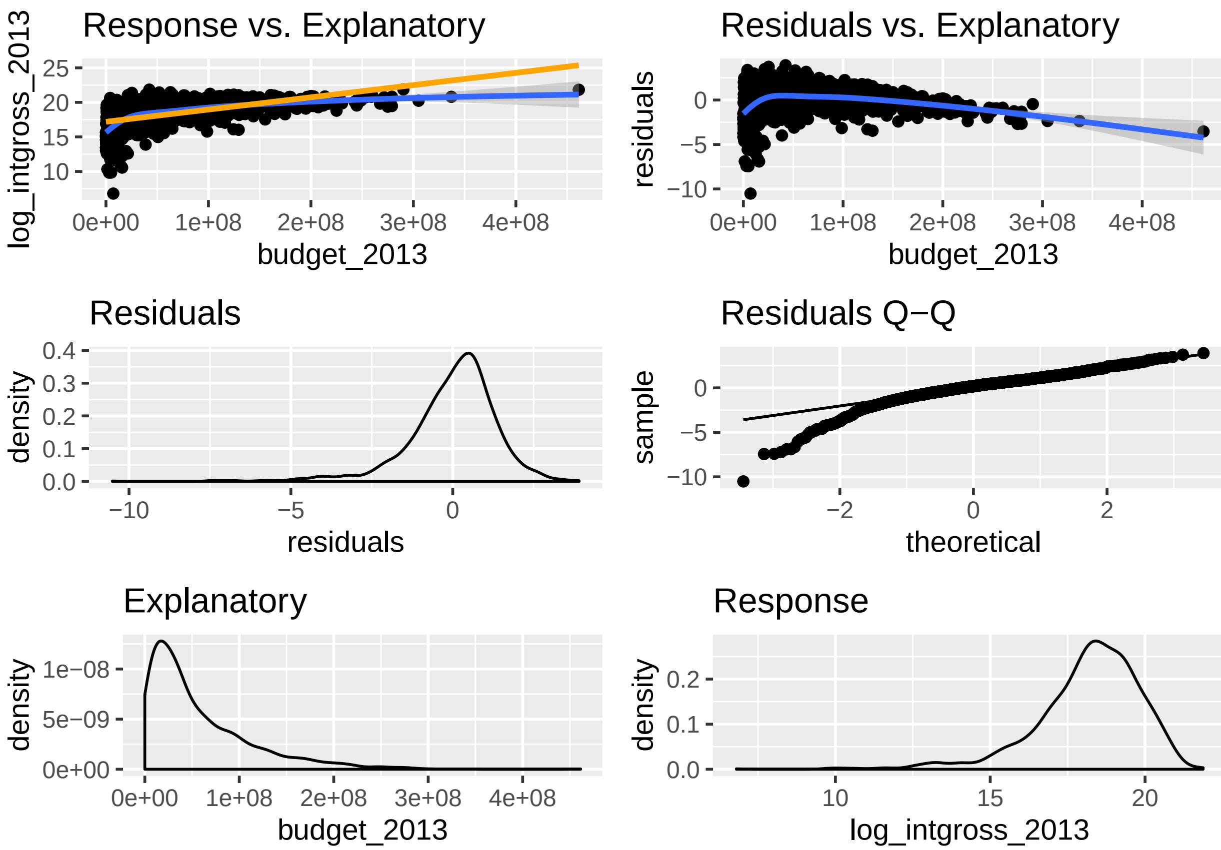



**What do we think?**

**Trying** log(**intgross_2013**)

```
movies <- movies %>% mutate(
  log_intgross_2013 = log(intgross_2013)
)

fit_model_and_make_plots(response = "log_intgross_2013", explanatory = "budget_2013")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

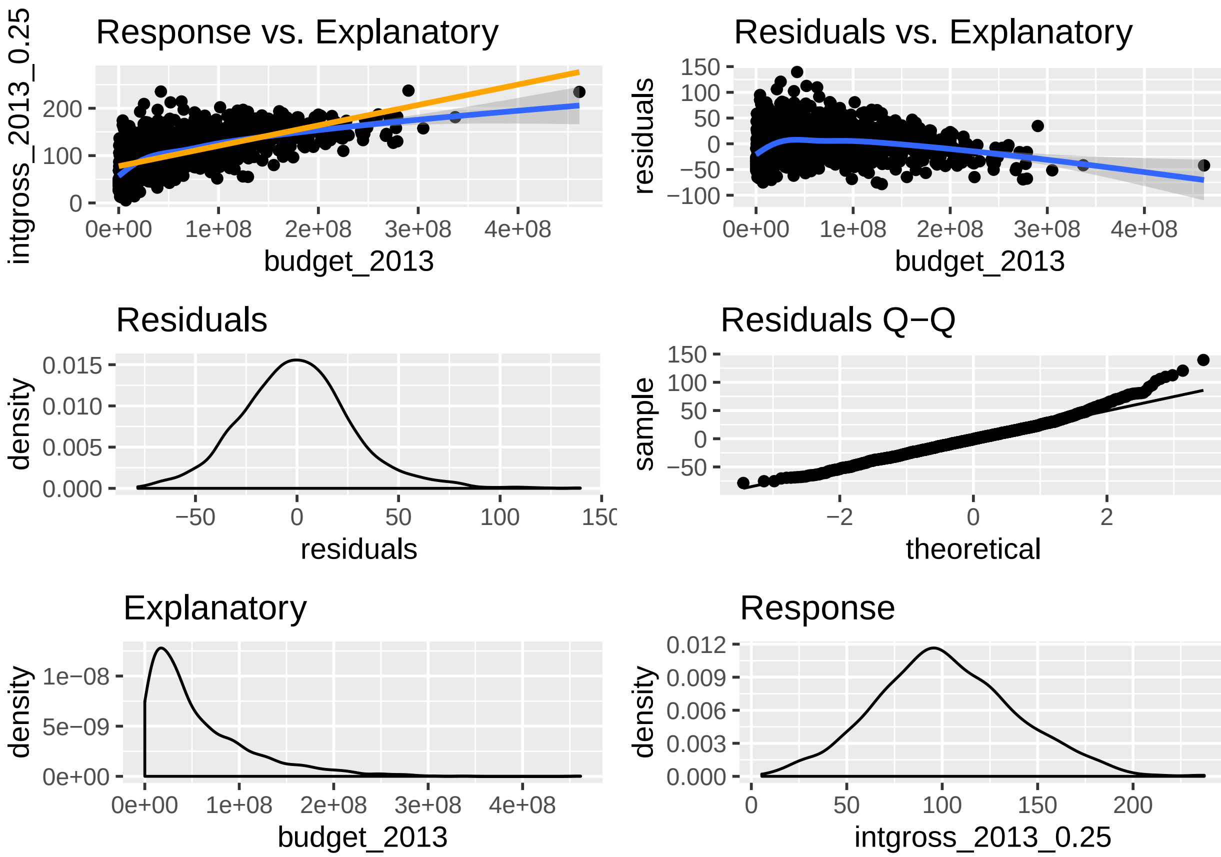



**What do we think?**

**Trying intgross_2013$^{0.25}$**

```
movies <- movies %>% mutate(
  intgross_2013_0.25 = intgross_2013^{0.25}
)

fit_model_and_make_plots(response = "intgross_2013_0.25", explanatory = "budget_2013")
```

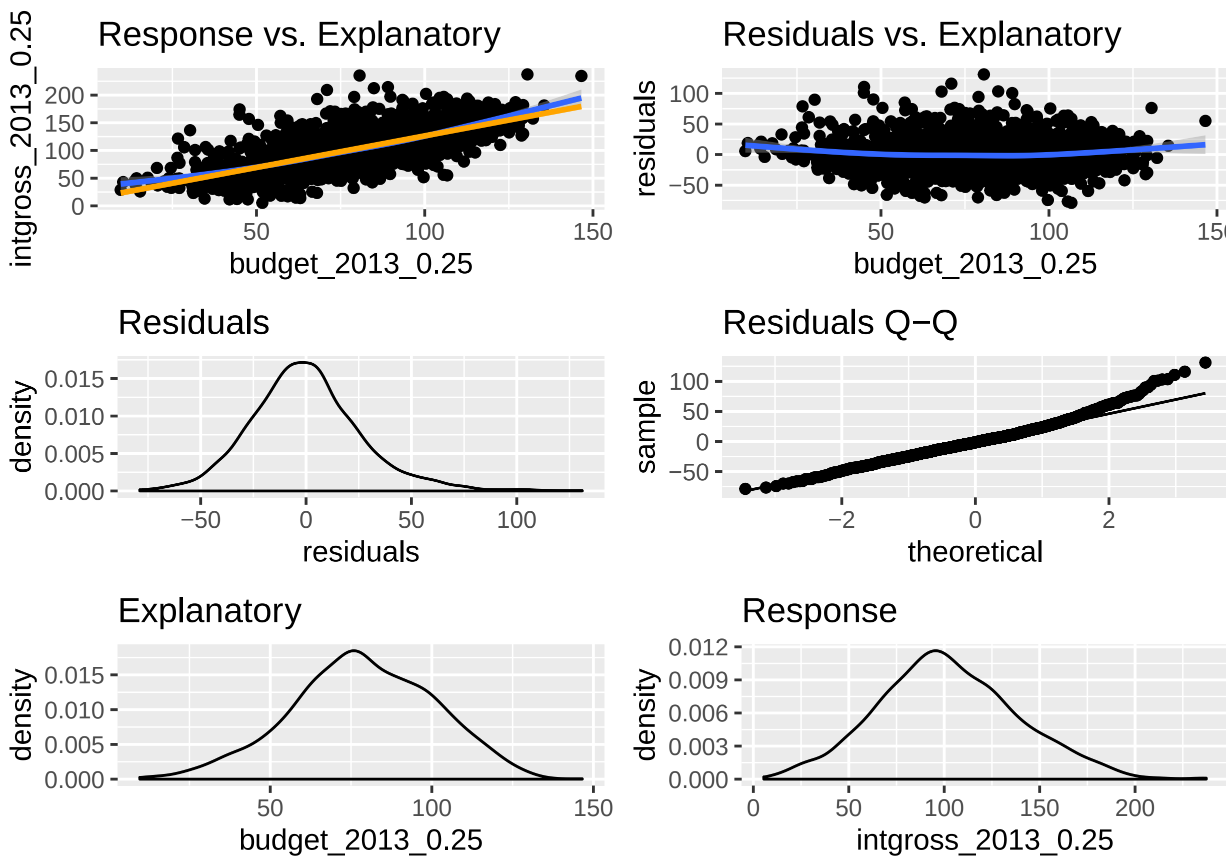
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



### Transformations of both variables...

```
movies <- movies %>% mutate(
  intgross_2013_0.25 = intgross_2013^{0.25},
  budget_2013_0.25 = budget_2013^{0.25}
)

fit_model_and_make_plots(response = "intgross_2013_0.25", explanatory = "budget_2013_0.25")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



## Making Predictions in Models with Transformed Variables

- You need to give your model transformed x's to generate predictions
- You usually want predictions for the response on the original (untransformed) scale.

Here's an example of making predictions for test set observations and finding MSE on original scale:

```
# train/test split
set.seed(29347)
train_inds <- caret::createDataPartition(movies$intgross_2013, p = 0.8)
train_movies <- movies %>% slice(train_inds[[1]])
test_movies <- movies %>% slice(-train_inds[[1]])

# transformation for train data
train_movies <- train_movies %>%
  mutate(
    intgross_2013_0.25 = intgross_2013^{0.25},
    budget_2013_0.25 = budget_2013^{0.25}
  )

# note: for the test set I only need to apply transformations to explanatory variables
# since I will evaluate predictions for the response on the original data scale.
test_movies <- test_movies %>%
  mutate(
    budget_2013_0.25 = budget_2013^{0.25}
  )

# fit to transformed data on training set
fit <- lm(intgross_2013_0.25 ~ budget_2013_0.25, data = train_movies)

# predictions based on transformed budget for the test set
# the result is a prediction of (intgross_2013)^0.25
predicted_intgross_2013_0.25 <- predict(fit, newdata = test_movies)

# undo the transformation of the response to get predictions of intgross_2013
predicted_intgross_2013 <- predicted_intgross_2013_0.25^4

# calculate MSE
mean((test_movies$intgross_2013 - predicted_intgross_2013)^2)
```

```
## [1] 6.524786e+16
```

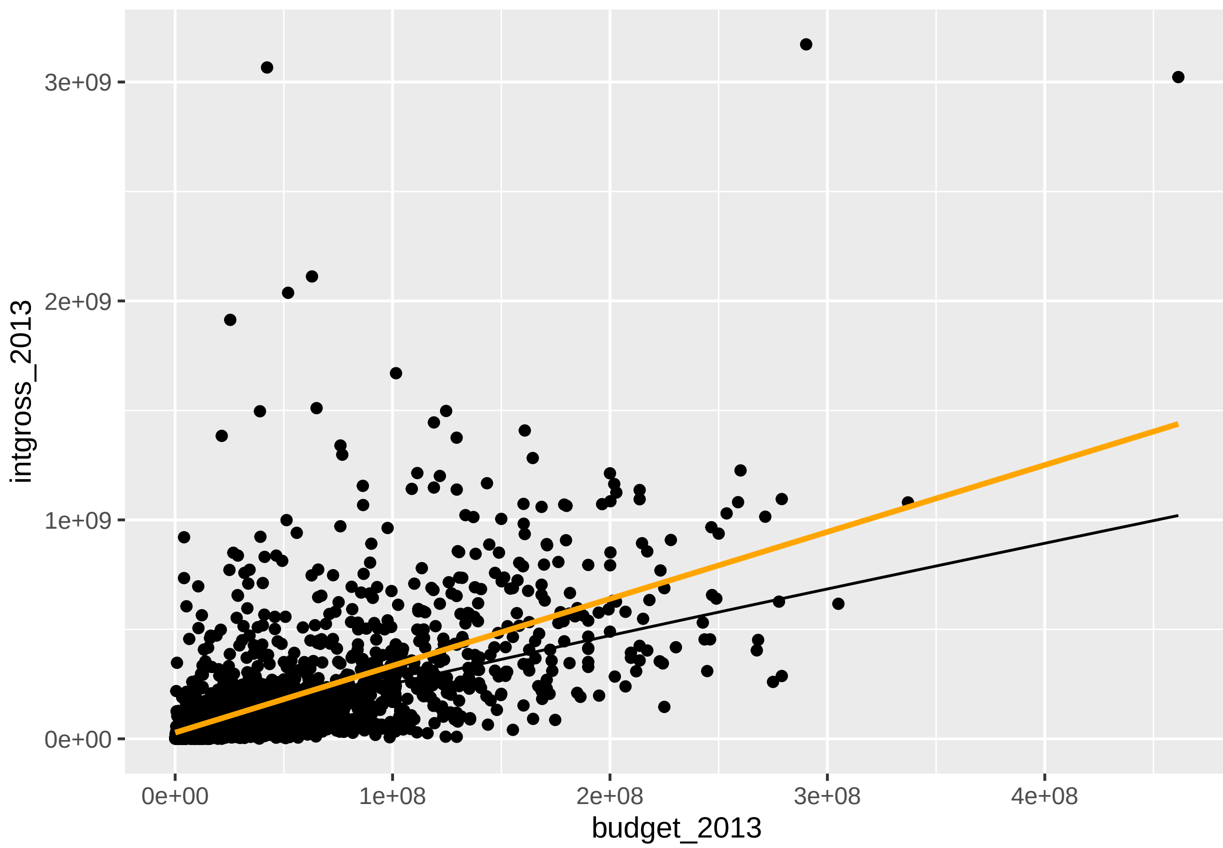
```
# That's so big, how about its square root (RMSE)
sqrt(mean((test_movies$intgross_2013 - predicted_intgross_2013)^2))
```

```
## [1] 255436612
```

Rough interpretation: on average, test set predictions are off by about $255 million.

You also have to take care when making plots:

```
predict_transformed_scale <- function(x) {
  pred_0.25 <- predict(fit, data.frame(budget_2013_0.25 = x^{0.25}))
  return(pred_0.25^4)
}

ggplot(data = movies, mapping = aes(y = intgross_2013, x = budget_2013)) +
  geom_point() +
  stat_function(fun = predict_transformed_scale) +
  geom_smooth(method = "lm", color = "orange", se = FALSE)
```




An effect of fitting to transformed data was to reduce the influence of those outlying observations on the line.

**Transformations may or may not help test set predictive performance**

Here we fit a linear regression model without transformations and get lower test set (R)MSE.

```
# fit to transformed data on training set
fit <- lm(intgross_2013 ~ budget_2013, data = train_movies)

# predictions based on transformed budget for the test set
# the result is a prediction of (intgross_2013)^0.25
predicted_intgross_2013 <- predict(fit, newdata = test_movies)

# calculate MSE
mean((test_movies$intgross_2013 - predicted_intgross_2013)^2)
```

```
## [1] 5.723862e+16
```

```
# That's so big, how about its square root (RMSE)
sqrt(mean((test_movies$intgross_2013 - predicted_intgross_2013)^2))
```

```
## [1] 239245949
```