Lecture 8 Notes

2024-07-15

Regression using mv.rds data

```
require(tidyverse)
```

```
## Loading required package: tidyverse
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts t
o become errors
```

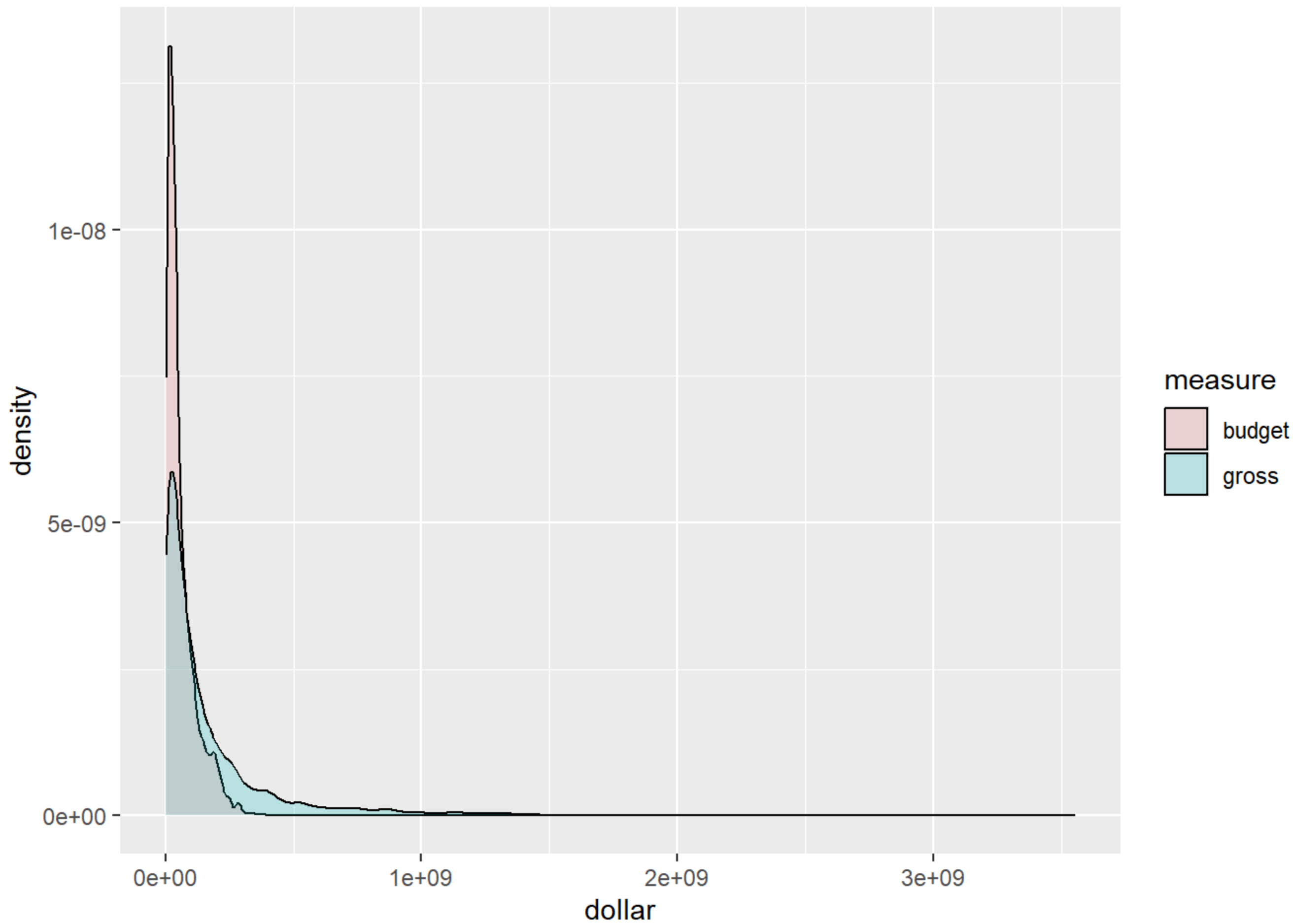
```
mv <- read_rds("https://github.com/jbisbee1/ISP_Data_Science_2024/raw/main/data/mv.Rds")
```

pivot_longer()

```
mv_long <- mv %>%
  select(title,budget,gross) %>%
  drop_na() %>%
  pivot_longer(cols = c("budget","gross"),
               names_to = "measure",
               values_to = "dollar")
```

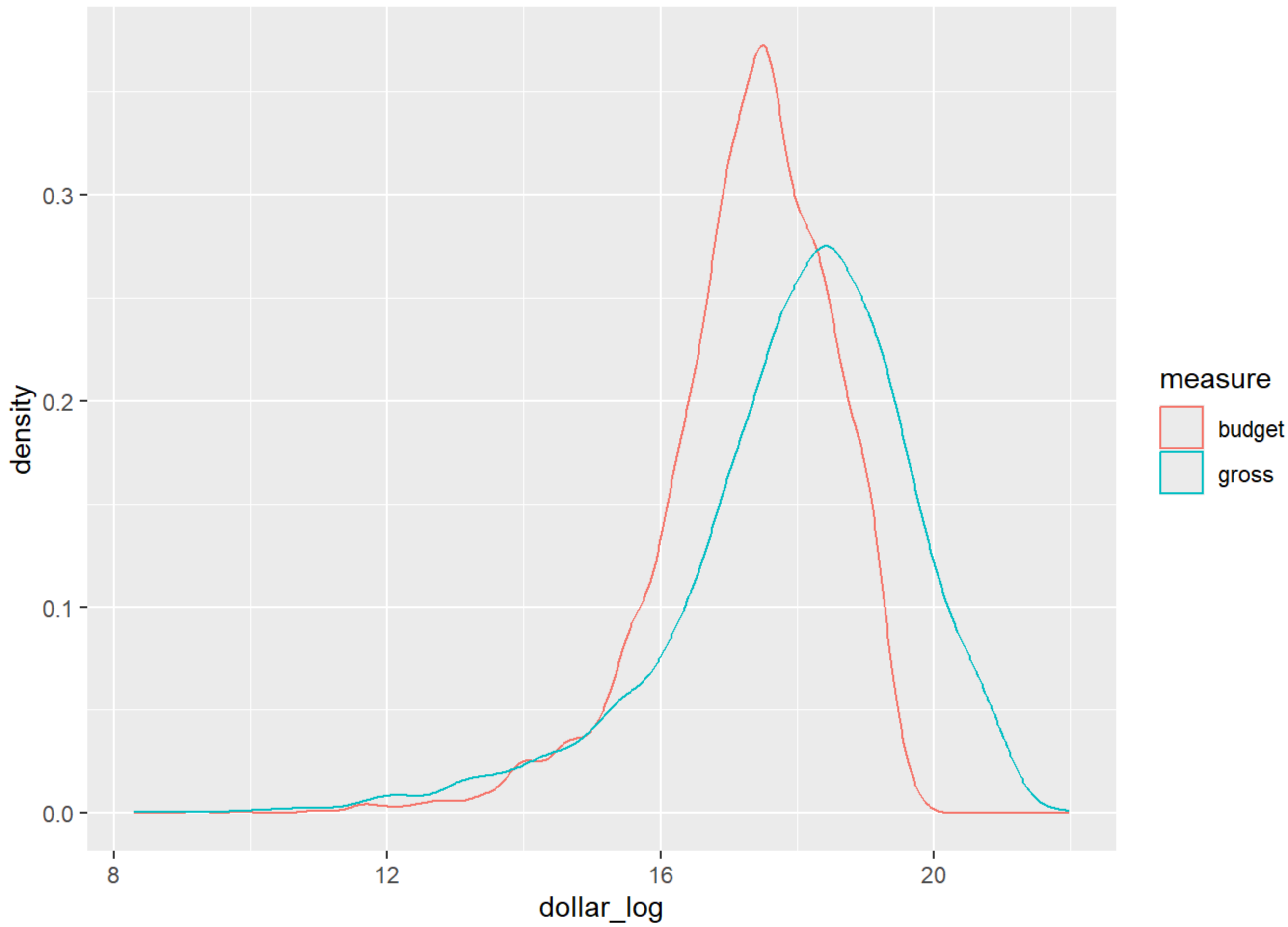
Create univariate visualization of both X and Y together

```
mv_long %>%
  ggplot(aes(x = dollar,
             fill = measure)) +
  geom_density(alpha = .2)
```



Log to get rid of extreme skew

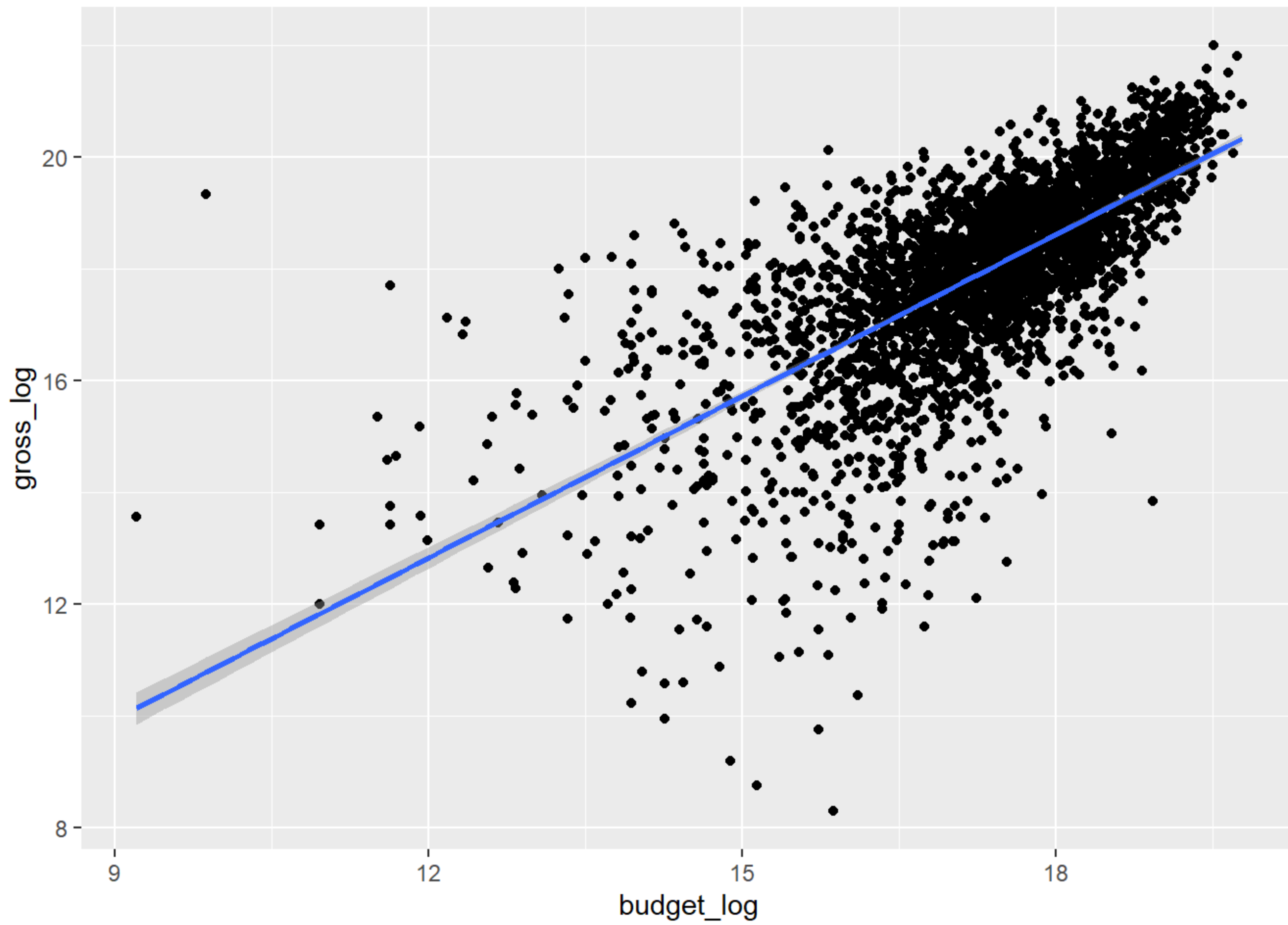
```
mv_long %>%
  mutate(dollar_log = log(dollar)) %>%
  ggplot(aes(x = dollar_log,
             color = measure)) +
  geom_density()
```



Running multivariate viz and regression

```
mv_analysis <- mv %>%
  drop_na(gross,budget) %>%
  mutate(gross_log = log(gross),
         budget_log = log(budget))

# Multivariate visualization
mv_analysis %>%
  ggplot(aes(x = budget_log,
             y = gross_log)) +
  geom_point() +
  geom_smooth(method = 'lm')
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Running regression with the lm() function

```
m1 <- lm(gross_log ~ budget_log,
   data = mv_analysis)

# Looking at result: two methods
# Method 1: use summary()
summary(m1)
```

```
## 
## Call:
## lm(formula = gross_log ~ budget_log, data = mv_analysis)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -8.2672 -0.6354  0.1648  0.7899  8.5599 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  1.26107    0.30953   4.074 4.73e-05 ***
## budget_log   0.96386    0.01786  53.971  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1.281 on 3177 degrees of freedom
## Multiple R-squared:  0.4783, Adjusted R-squared:  0.4782 
## F-statistic:  2913 on 1 and 3177 DF,  p-value: < 2.2e-16
```

```
# Method 2: use tidy() from broom package
require(broom)
```

```
## Loading required package: broom
```

```
tidy(m1)
```

```
## # A tibble: 2 × 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)    1.26     0.310       4.07 0.0000473
## 2 budget_log     0.964    0.0179     54.0  0
```

Tangent: log rules

```
(exp(0.96)-1)*100
```

```
## [1] 161.1696
```

Evaluating model performance

Step 1: Look at errors

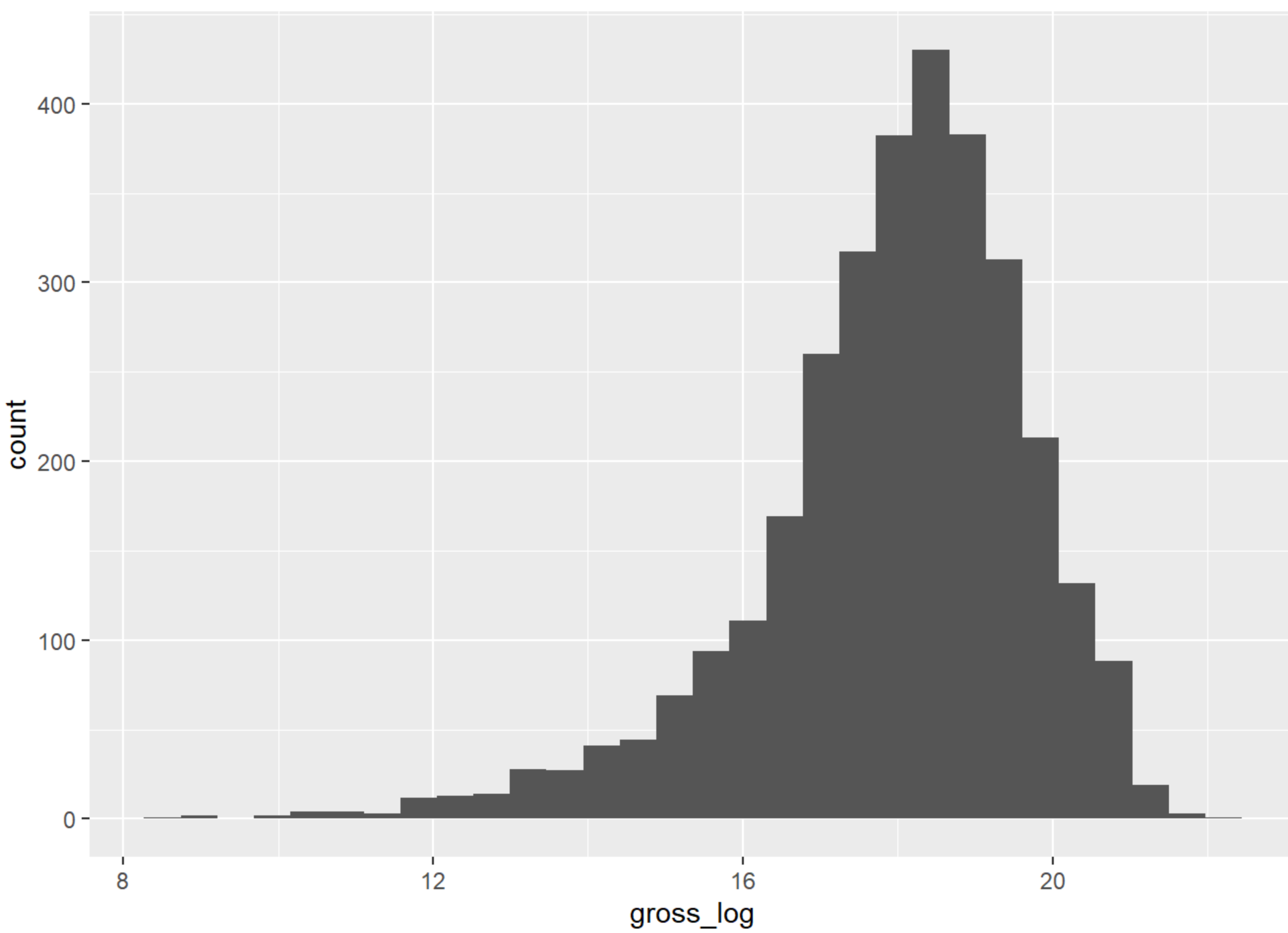
First, calculate the errors

```
mv_analysis <- mv_analysis %>%
  mutate(Yhat = predict(m1)) %>%
  mutate(error = gross_log - Yhat)

# Summarize the errors
summary(mv_analysis %>%
          select(error))
```

```
##      error
##  Min.   :-8.2672
##  1st Qu.:-0.6354
##  Median : 0.1648
##  Mean   : 0.0000
##  3rd Qu.: 0.7899
##  Max.   : 8.5599
```

```
# Reminder of what Y looks like
mv_analysis %>%
  ggplot(aes(x = gross_log)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

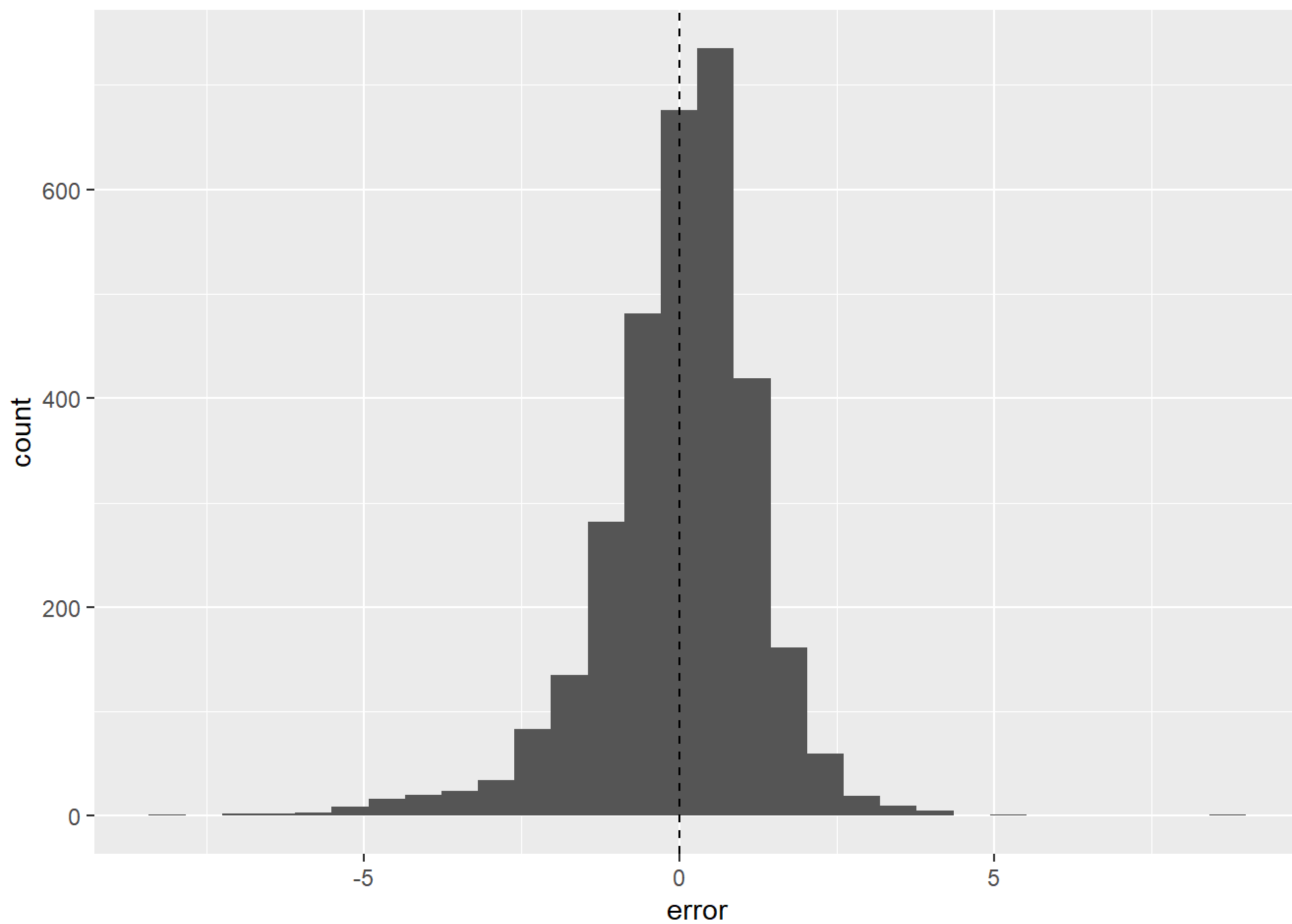


Looking at the shape of error

First, univariate visualization of the errors

```
mv_analysis %>%
  ggplot(aes(x = error)) +
  geom_histogram() +
  geom_vline(xintercept = 0,
             linetype = 'dashed')
```

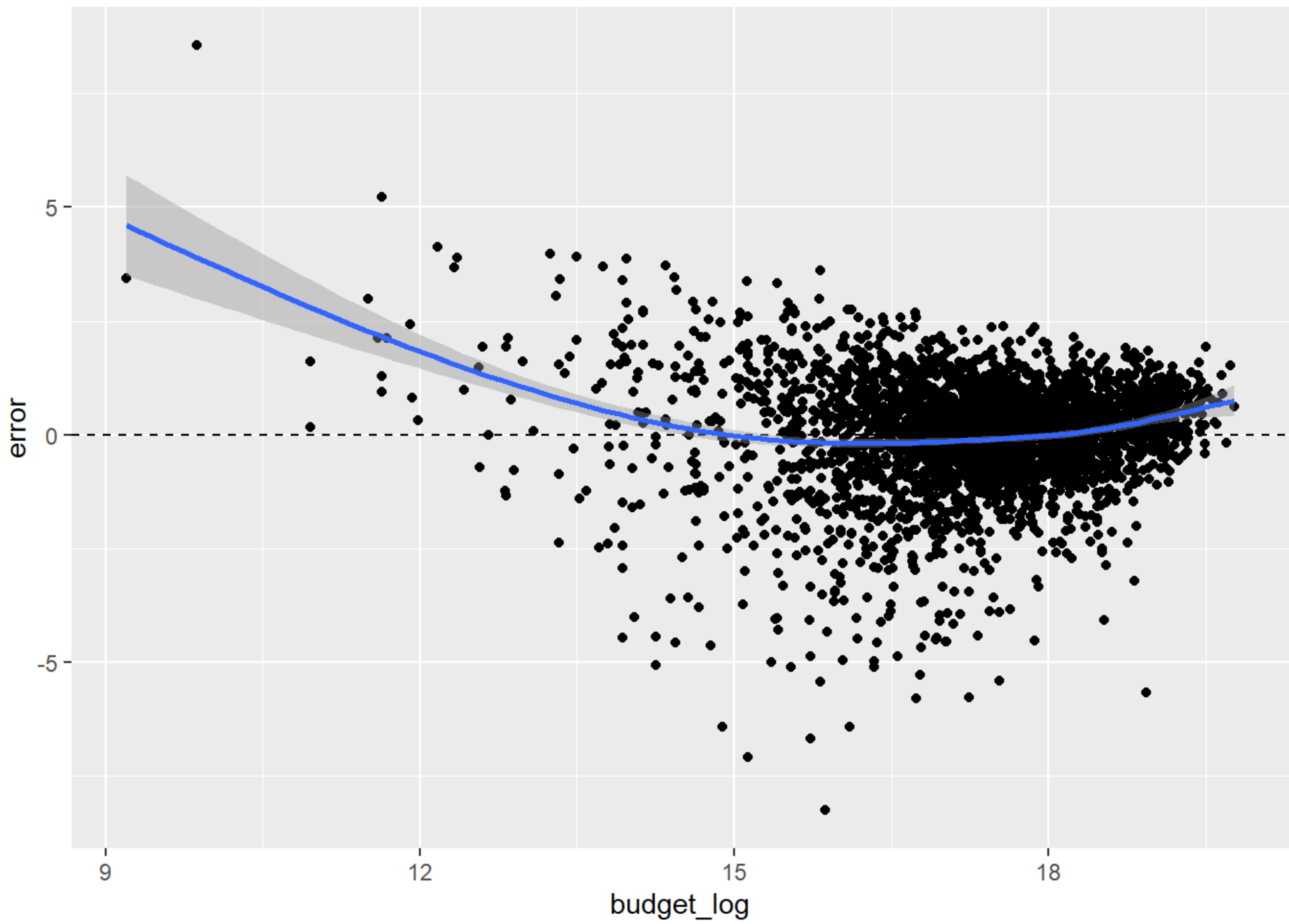
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Second, multivariate visualization of the errors

```
mv_analysis %>%
  ggplot(aes(x = budget_log,
             y = error)) +
  geom_point() +
  geom_hline(yintercept = 0,
             linetype = 'dashed') +
  geom_smooth()
```

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



#RMSE

- How bad is our model on average?

```
# Method 1: Step-by-step
# Error: already calculated above
mv_analysis %>%
  select(title,error)
```

```
## # A tibble: 3,179 × 2
##    title                error
##    <chr>                <dbl>
##  1 Almost Famous       -0.834
##  2 American Psycho      0.913
##  3 Gladiator            0.930
##  4 Requiem for a Dream -0.195
##  5 Memento              0.826
##  6 Cast Away            0.980
##  7 Scary Movie          2.04
##  8 The Perfect Storm    0.286
##  9 Coyote Ugly          0.321
## 10 X-Men                0.784
## # ℹ 3,169 more rows
```

```
# Squared Error (SE)
mv_analysis <- mv_analysis %>%
  mutate(se = error^2)

# Mean Squared Error (MSE)
rmse <- mv_analysis %>%
  summarise(mse = mean(se))

# Root Mean Squared Error (RMSE)
rmse <- rmse %>%
  mutate(rmse = sqrt(mse))

# Messy code
mv_analysis %>%
  summarise(rmse = sqrt(mean((error)^2)))
```

```
## # A tibble: 1 × 1
##    rmse
##   <dbl>
## 1  1.28
```

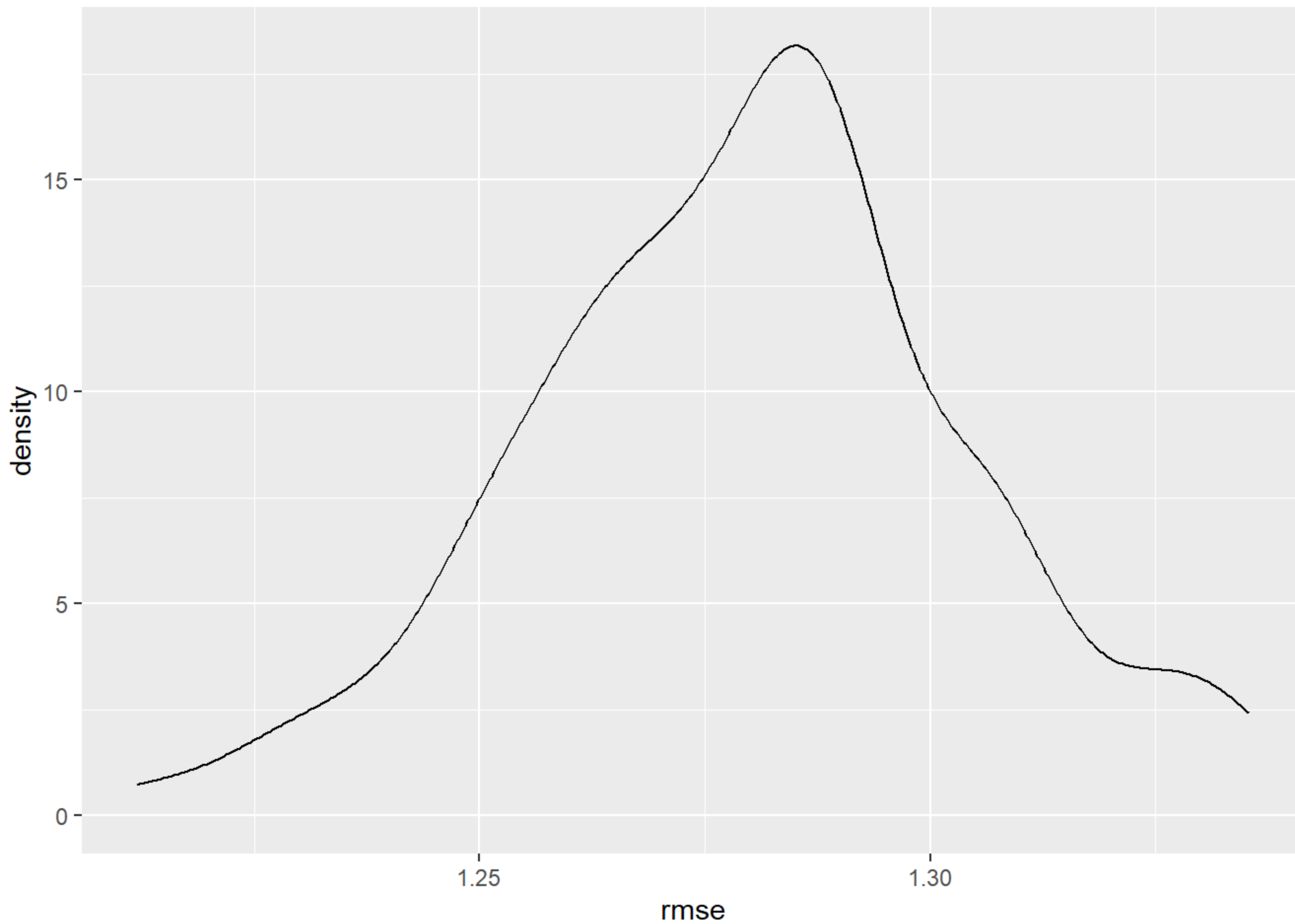
Cross validation

- Very similar to bootstrapping

```
set.seed(123)
cv_result <- NULL
for(i in 1:100) {
  # Step 1: Divide data
  train <- mv_analysis %>%
    sample_n(size = round(nrow(mv_analysis)*.5),
             replace = F)

  test <- mv_analysis %>%
    anti_join(train)

  # Step 2: Train the model
  mTmp <- lm(formula = gross_log ~ budget_log,
             data = train)

  # Step 3: Evaluate the model
  test <- test %>%
    mutate(YHat = predict(mTmp,newdata = test)) %>%
    mutate(error = gross_log - YHat)

  # RMSE
  answer <- test %>%
    summarise(rmse = sqrt(mean((error^2)))) %>%
    mutate(cvInd = i)

  # Save result
  cv_result <- cv_result %>%
    bind_rows(answer)
}

# Summary 1: just calculate the mean
mean(cv_result$rmse)
```

```
## [1] 1.279899
```

```
# Summary 2: Univariate visualization
cv_result %>%
  ggplot(aes(x = rmse)) +
  geom_density()
```



New RQ: What is the relationship between a movie's gross and it's IMDB score?

- Theory (boring Teacher's theory): The score informs consumers which movies are good, and they then go out to watch those movies, increasing the gross.
- Univariate visualization of `score`
- Multivariate visualization of `score` and `gross_log`.
- Regression result

```
m2 <- lm(gross_log ~ score,
         data = mv_analysis)

tidy(m2)
```

```
## # A tibble: 2 × 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   16.1      0.211      76.5  0       
## 2 score          0.279    0.0325      8.58 1.49e-17
```

```
(exp(0.279)-1)*100
```

```
## [1] 32.18073
```

Cross validation for RMSE

```
set.seed(123)
cv_result <- NULL
for(i in 1:100) {
  # Step 1: Divide data
  train <- mv_analysis %>%
    sample_n(size = round(nrow(mv_analysis)*.5),
             replace = F)

  test <- mv_analysis %>%
    anti_join(train)

  # Step 2: Train the model
  mTmp_score <- lm(formula = gross_log ~ score,
             data = train)

  mTmp_budget <- lm(formula = gross_log ~ budget_log,
             data = train)

  # Step 3: Evaluate the model
  test <- test %>%
    mutate(YHat_score = predict(mTmp_score,newdata = test),
           YHat_budget = predict(mTmp_budget,newdata = test)) %>%
    mutate(error_score = gross_log - YHat_score,
           error_budget = gross_log - YHat_budget)

  # RMSE
  answer <- test %>%
    summarise(rmse_score = sqrt(mean(error_score^2)),
              rmse_budget = sqrt(mean((error_budget^2)))) %>%
    mutate(cvInd = i)

  # Save result
  cv_result <- cv_result %>%
    bind_rows(answer)
}

# Summary 1: just calculate the mean
mean(cv_result$rmse_score)
```

```
## [1] 1.748458
```

```r
mean(cv_result$rmse_budget)
```

```
## [1] 1.279899
```