Lecture 14 Notes

2024-03-05

Opening the data

```
require(tidyverse)
```

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

```
## Warning: package 'tidyverse' was built under R version 4.3.2
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.2     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.4     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
```

```
## ── 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/DS1000_S2024/raw/main/data/mv.Rds')

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

Regression

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

RMSE

```
mv_analysis <- mv_analysis %>%
  mutate(preds = predict(model_gross_budget)) %>%
  mutate(errors = gross_log - preds)

rmse <- mv_analysis %>%
  mutate(se = errors^2) %>%
  summarise(mse = mean(se)) %>%
  mutate(rmse = sqrt(mse))
```

Cross Validation

```
set.seed(123)
cvRes <- NULL
for(i in 1:100) {
  # Generating a random list of rows
inds <- sample(x = 1:3179,
       size = round(3179/2),
       replace = FALSE)

train <- mv_analysis %>%
  slice(inds)

test <- mv_analysis %>%
  slice(-inds)

# Training on the training data
model_tmp <- lm(gross_log ~ budget_log,data = train)

# Predicting on the test data
test <- test %>%
  mutate(preds = predict(model_tmp,
                         newdata = test)) %>%
  mutate(errors = gross_log - preds)

rmse <- test %>%
  mutate(se = errors^2) %>%
  summarise(mse = mean(se)) %>%
  mutate(rmse = sqrt(mse))

cvRes <- cvRes %>%
  bind_rows(rmse)
}

cvRes %>%
  summarise(rmseOverall = mean(rmse))
```

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

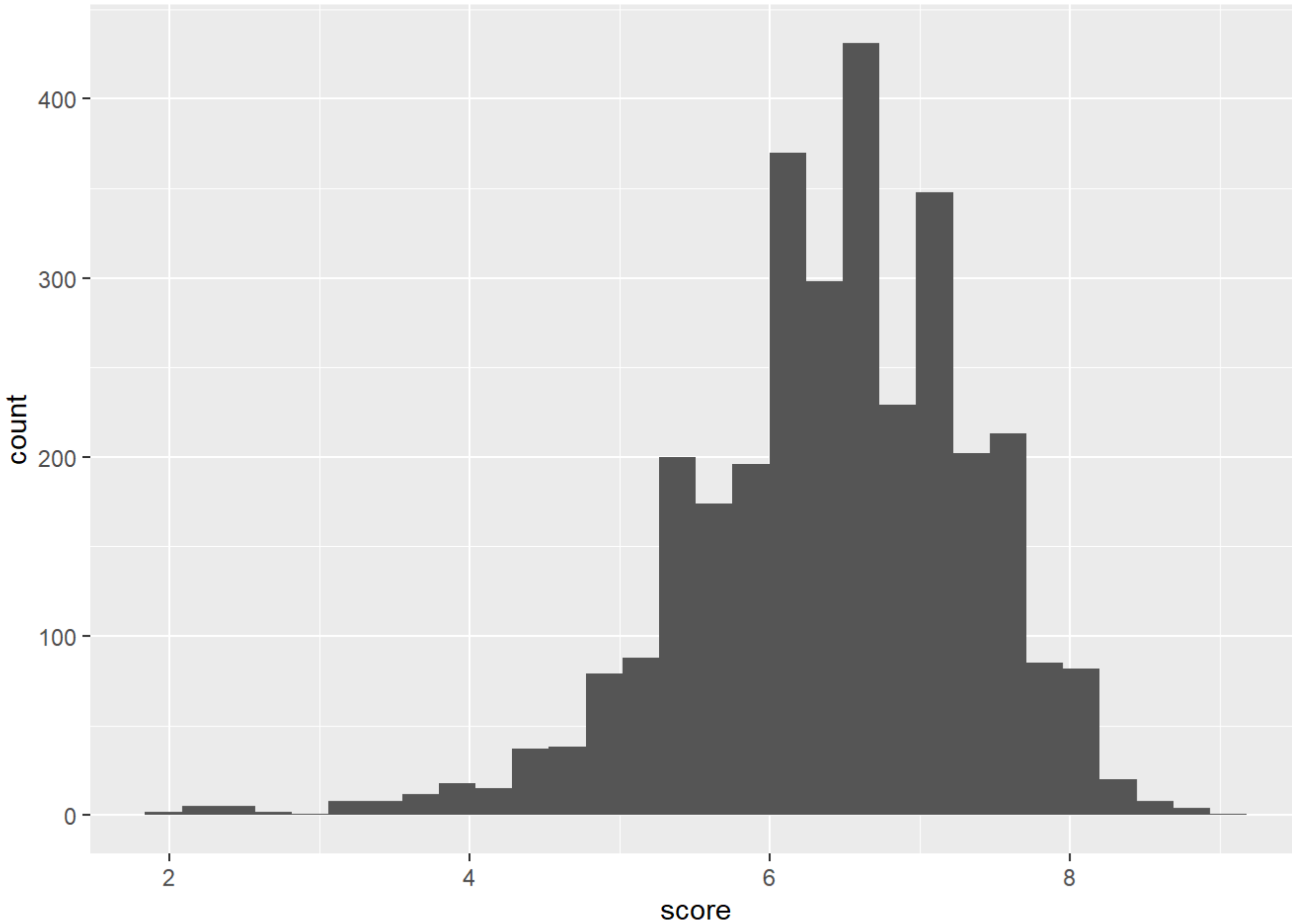
Alternative model

```
mv_analysis %>%
  select(gross_log,score)
```

```
## # A tibble: 3,179 × 2
##    gross_log score
##        <dbl> <dbl>
##  1      18.1   7.9
##  2      17.8   7.6
##  3      20.4   8.5
##  4      16.3   8.3
##  5      17.9   8.4
##  6      20.3   7.8
##  7      19.9   6.2
##  8      20.1   6.4
##  9      19.0   5.7
## 10      19.9   7.4
## # ℹ 3,169 more rows
```

```
# Univariate viz
mv_analysis %>%
  ggplot(aes(x = score)) +
  geom_histogram()
```

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

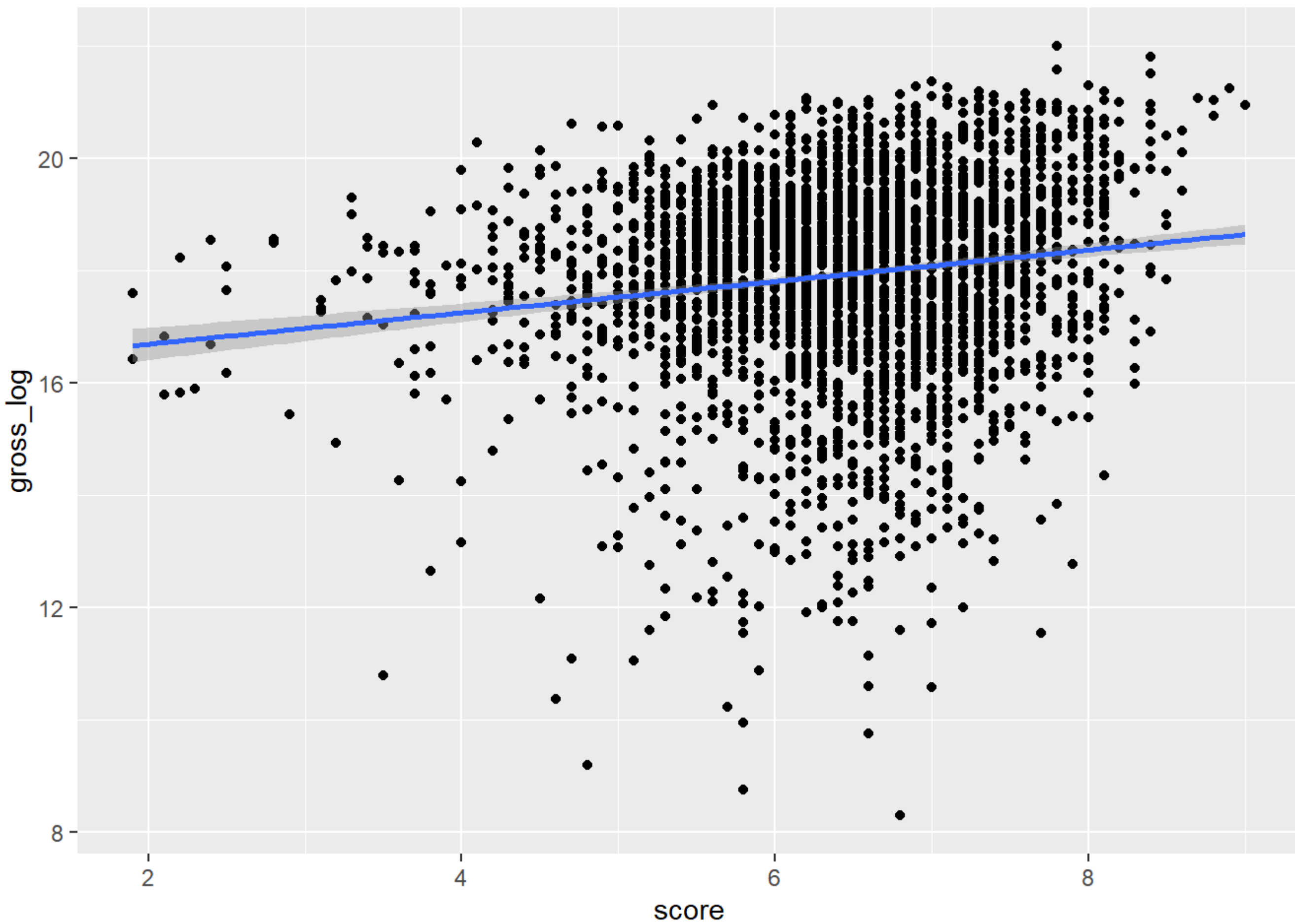



```
# Missingness
summary(mv_analysis %>%
          select(gross_log,score))
```

```
##    gross_log          score
##  Min.   : 8.287   Min.   :1.900
##  1st Qu.:17.068   1st Qu.:5.900
##  Median :18.189   Median :6.500
##  Mean   :17.922   Mean   :6.417
##  3rd Qu.:19.104   3rd Qu.:7.100
##  Max.   :21.991   Max.   :9.000
```

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

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



Cross Validation: Model 2

```r
set.seed(123)
cvRes <- NULL
for(i in 1:100) {
  # Generating a random list of rows
inds <- sample(x = 1:nrow(mv_analysis),
       size = round(nrow(mv_analysis)/2),
       replace = FALSE)

train <- mv_analysis %>%
  slice(inds)

test <- mv_analysis %>%
  slice(-inds)

# Training on the training data
model_tmp <- lm(gross_log ~ score,data = train)

# Predicting on the test data
test <- test %>%
  mutate(preds = predict(model_tmp,
                         newdata = test)) %>%
  mutate(errors = gross_log - preds)

rmse <- test %>%
  mutate(se = errors^2) %>%
  summarise(mse = mean(se)) %>%
  mutate(rmse = sqrt(mse))

cvRes <- cvRes %>%
  bind_rows(rmse)
}

cvRes %>%
  summarise(rmseOverall = mean(rmse))
```

```
## # A tibble: 1 × 1
##   rmseOverall
##         <dbl>
## 1        1.75
```

Multiple X Variables

```r
# Y = alpha + beta_1 X_1 + beta_2 X_2 + ... + e
model_budg_score <- lm(gross_log ~ budget_log + score,
                       mv_analysis)

summary(model_budg_score)
```

```
##
## Call:
## lm(formula = gross_log ~ budget_log + score, data = mv_analysis)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.3763 -0.5708  0.1501  0.7322  8.6189
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.70352    0.33839  -2.079   0.0377 *
## budget_log   0.96710    0.01742  55.527   <2e-16 ***
## score        0.29740    0.02316  12.843   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.249 on 3176 degrees of freedom
## Multiple R-squared:  0.5041, Adjusted R-squared:  0.5038
## F-statistic:  1614 on 2 and 3176 DF,  p-value: < 2.2e-16
```

```
(exp(0.29740)-1)*100
```

```
## [1] 34.63537
```

Cross Validation: Model 3

```r
set.seed(123)
cvRes <- NULL
for(i in 1:100) {
  # Generating a random list of rows
inds <- sample(x = 1:nrow(mv_analysis),
       size = round(nrow(mv_analysis)/2),
       replace = FALSE)

train <- mv_analysis %>%
  slice(inds)

test <- mv_analysis %>%
  slice(-inds)

# Training on the training data
model_tmp <- lm(gross_log ~ budget_log + score,data = train)

# Predicting on the test data
test <- test %>%
  mutate(preds = predict(model_tmp,
                         newdata = test)) %>%
  mutate(errors = gross_log - preds)

rmse <- test %>%
  mutate(se = errors^2) %>%
  summarise(mse = mean(se)) %>%
  mutate(rmse = sqrt(mse))

cvRes <- cvRes %>%
  bind_rows(rmse)
}

cvRes %>%
  summarise(rmseOverall = mean(rmse))
```

```
## # A tibble: 1 × 1
##   rmseOverall
##         <dbl>
## 1        1.25
```

Movie Ratings

```r
model_gross_rating <- lm(gross_log ~ rating,
                         mv_analysis)

summary(model_gross_rating)
```

```
##
## Call:
## lm(formula = gross_log ~ rating, data = mv_analysis)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.6749 -0.8191  0.1630  1.1082  5.2339
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      19.1818     0.2182  87.914  < 2e-16 ***
## ratingNC-17      -2.4483     0.6900  -3.548 0.000393 ***
## ratingNot Rated  -4.4322     0.3510 -12.626  < 2e-16 ***
## ratingPG         -0.3905     0.2314  -1.688 0.091517 .
## ratingPG-13      -0.7633     0.2229  -3.425 0.000623 ***
## ratingR          -1.9123     0.2224  -8.599  < 2e-16 ***
## ratingTV-MA      -3.2064     1.1546  -2.777 0.005515 **
## ratingUnrated    -4.6564     0.6441  -7.229 6.05e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.603 on 3166 degrees of freedom
##   (5 observations deleted due to missingness)
## Multiple R-squared:  0.1771, Adjusted R-squared:  0.1753
## F-statistic: 97.33 on 7 and 3166 DF,  p-value: < 2.2e-16
```

Movie Genre

```
model_gross_genre <- lm(gross_log ~ genre,
                         mv_analysis)

summary(model_gross_genre)
```

```
##
## Call:
## lm(formula = gross_log ~ genre, data = mv_analysis)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8819 -0.7452  0.2469  1.0992  3.6548
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    18.58333    0.05560 334.235  < 2e-16 ***
## genreAdventure -0.18471    0.13449  -1.373 0.169712
## genreAnimation  0.67982    0.12467   5.453 5.33e-08 ***
## genreBiography -1.14278    0.12421  -9.200  < 2e-16 ***
## genreComedy    -1.00044    0.08101 -12.350  < 2e-16 ***
## genreCrime     -1.39688    0.12729 -10.974  < 2e-16 ***
## genreDrama     -1.41465    0.09161 -15.442  < 2e-16 ***
## genreFamily     0.37140    1.16627   0.318 0.750166
## genreFantasy   -1.69110    0.46030  -3.674 0.000243 ***
## genreHorror    -0.91636    0.14597  -6.278 3.90e-10 ***
## genreMystery    0.03469    0.62516   0.055 0.955749
## genreRomance   -2.68759    0.82561  -3.255 0.001145 **
## genreSci-Fi    -0.16302    1.16627  -0.140 0.888846
## genreThriller  -0.58577    0.82561  -0.709 0.478068
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.647 on 3165 degrees of freedom
## Multiple R-squared:  0.1407, Adjusted R-squared:  0.1372
## F-statistic: 39.87 on 13 and 3165 DF,  p-value: < 2.2e-16
```

```
mv_analysis %>%
  count(genre)
```

```
## # A tibble: 14 × 2
##    genre         n
##    <chr>     <int>
##  1 Action      878
##  2 Adventure   181
##  3 Animation   218
##  4 Biography   220
##  5 Comedy      782
##  6 Crime       207
##  7 Drama       512
##  8 Family        2
##  9 Fantasy      13
## 10 Horror      149
## 11 Mystery       7
## 12 Romance       4
## 13 Sci-Fi        2
## 14 Thriller      4
```

Bechdel Score

```
model_gross_BS <- lm(gross_log ~ bechdel_score,
                         mv_analysis)

summary(model_gross_BS)
```

```
## 
## Call:
## lm(formula = gross_log ~ bechdel_score, data = mv_analysis)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -9.9000 -0.7755  0.2216  1.0792  3.8042 
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)   18.36809    0.08565 214.443   <2e-16 ***
## bechdel_score -0.06042    0.03545  -1.704   0.0885 .  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1.665 on 2056 degrees of freedom
##   (1121 observations deleted due to missingness)
## Multiple R-squared:  0.001411,   Adjusted R-squared:  0.0009253 
## F-statistic: 2.905 on 1 and 2056 DF,  p-value: 0.08845
```

```
mv_analysis %>%
  drop_na(bechdel_score) %>%
  ggplot(aes(x = factor(bechdel_score),
             y = gross_log)) +
  geom_violin()
```