PSTAT 126

Lab 4

Roupen Khanjian

Spring 2021

```r
library(faraway) # Functions and Datasets for Books by Julian Faraway
library(alr4) # Data to Accompany Applied Linear Regression 4th Edition
library(tidyverse) # Easily Install and Load the 'Tidyverse'
library(GGally) # Extension to 'ggplot2'
library(palmerpenguins) # Palmer Archipelago (Antarctica) Penguin Data
```

Contents

Multiple Linear Regression

SLR in matrix form:

$$\begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

MLR with 2 predictor variables

$$\begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

MLR

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \varepsilon$$

- $\boldsymbol{X}$ = design matrix (n x p)

- β = coefficient vector (p x 1)
- $\mathbf{Y}$ = (n x 1)
- If n > p and the columns of $\boldsymbol{X}$ are linearly independent, then $\left(X^T X\right)^{-1}$ exists & the OLS estimator is:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X^T X})^{-1} \boldsymbol{X^T y}$$

Example in R

Data from faraway package. `gala` dataset is on species diversity on the Galapagos Islands

```r
data(gala)
head(gala[,-2])
```

```
##              Species  Area Elevation Nearest Scruz Adjacent
## Baltra            58 25.09       346     0.6   0.6     1.84
## Bartolome         31  1.24       109     0.6  26.3   572.33
## Caldwell           3  0.21       114     2.8  58.7     0.78
## Champion          25  0.10        46     1.9  47.4     0.18
## Coamano            2  0.05        77     1.9   1.9   903.82
## Daphne.Major      18  0.34       119     8.0   8.0     1.84
```

```r
glimpse(gala[,-2])
```

```
## Rows: 30
## Columns: 6
## $ Species   <dbl> 58, 31, 3, 25, 2, 18, 24, 10, 8, 2, 97, 93, 58, 5, 40, 347, ~
## $ Area      <dbl> 25.09, 1.24, 0.21, 0.10, 0.05, 0.34, 0.08, 2.33, 0.03, 0.18,~
## $ Elevation <dbl> 346, 109, 114, 46, 77, 119, 93, 168, 71, 112, 198, 1494, 49,~
## $ Nearest   <dbl> 0.6, 0.6, 2.8, 1.9, 1.9, 8.0, 6.0, 34.1, 0.4, 2.6, 1.1, 4.3,~
## $ Scruz     <dbl> 0.6, 26.3, 58.7, 47.4, 1.9, 8.0, 12.0, 290.2, 0.4, 50.2, 88.~
## $ Adjacent  <dbl> 1.84, 572.33, 0.78, 0.18, 903.82, 1.84, 0.34, 2.85, 17.95, 0~
```

```r
lmod <- lm(Species ~ Area + Elevation + Nearest + Scruz + Adjacent, data = gala)
summary(lmod)
```

```
##
## Call:
## lm(formula = Species ~ Area + Elevation + Nearest + Scruz + Adjacent,
##     data = gala)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -111.679  -34.898   -7.862   33.460  182.584
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.068221  19.154198   0.369 0.715351
## Area        -0.023938   0.022422  -1.068 0.296318
## Elevation    0.319465   0.053663   5.953 3.82e-06 ***
## Nearest      0.009144   1.054136   0.009 0.993151
## Scruz       -0.240524   0.215402  -1.117 0.275208
## Adjacent    -0.074805   0.017700  -4.226 0.000297 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 60.98 on 24 degrees of freedom
## Multiple R-squared:  0.7658, Adjusted R-squared:  0.7171
## F-statistic:  15.7 on 5 and 24 DF,  p-value: 6.838e-07
```

```
x <- model.matrix(~ Area + Elevation + Nearest + Scruz + Adjacent, gala)

head(x)
```

```
##              (Intercept)  Area Elevation Nearest Scruz Adjacent
## Baltra                 1 25.09       346     0.6   0.6     1.84
## Bartolome              1  1.24       109     0.6  26.3   572.33
## Caldwell               1  0.21       114     2.8  58.7     0.78
## Champion               1  0.10        46     1.9  47.4     0.18
## Coamano                1  0.05        77     1.9   1.9   903.82
## Daphne.Major           1  0.34       119     8.0   8.0     1.84
```

- `model.matrix` takes all your predictor values and adds a column of 1's to it to make the **design matrix**

```
n = dim(gala)[1]
p = 5 + 1
x_same <- cbind(intercept = rep(1, n), gala[,3:7])
head(x_same)
```

```
##              intercept  Area Elevation Nearest Scruz Adjacent
## Baltra               1 25.09       346     0.6   0.6     1.84
## Bartolome            1  1.24       109     0.6  26.3   572.33
## Caldwell             1  0.21       114     2.8  58.7     0.78
## Champion             1  0.10        46     1.9  47.4     0.18
## Coamano              1  0.05        77     1.9   1.9   903.82
## Daphne.Major         1  0.34       119     8.0   8.0     1.84
```

- Can also make the design matrix manually

```
y <- gala$Species
```

- response vector $\mathbf{Y}$ of length n

```
xtxi <- solve(t(x) %*% x)
```

- Computing $(X^T X)^{-1}$
- `solve(A)` computes A^{-1}

```
xtxi %*% t(x) %*% y
```

```
##                     [,1]
## (Intercept)  7.068220709
## Area        -0.023938338
## Elevation    0.319464761
## Nearest      0.009143961
## Scruz       -0.240524230
## Adjacent    -0.074804832
```

Above we obtain our $\hat{\beta} = (\boldsymbol{X^T X})^{-1}\boldsymbol{X^T y}$ values

```
solve(crossprod(x,x)) %*% crossprod(x,y)
```

```
##                     [,1]
## (Intercept)  7.068220709
## Area        -0.023938338
## Elevation    0.319464761
```

```
## Nearest      0.009143961
## Scruz       -0.240524230
## Adjacent    -0.074804832
```

Can also use the function `crossprod(a,b)` which computes $a^T b$

```r
coef(lmod)
```

```
##  (Intercept)         Area    Elevation      Nearest        Scruz     Adjacent
##  7.068220709 -0.023938338  0.319464761  0.009143961 -0.240524230 -0.074804832
```

Can see our coefficients from the summary output as well.

Dataset example

Dataset for today is on Horror Movies. Boo!

```r
# Data source
horror_movies <-
read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2019/2019-10-22/hor
```

```r
# Subset the data (In case you want to replicate, don't need to know these functions for this course)
horror <- horror_movies %>%
  drop_na(budget, movie_run_time, review_rating) %>% # remove missing values
  mutate(movie_run_time =
           as.numeric( # remove the min and convert to numeric
             str_remove_all(movie_run_time, "min"))) %>%
  filter(str_detect(budget, "(?<=\\$)\\d+")) %>% # filter for movies using $
  mutate(budget = str_remove_all(budget, ",")) %>% # remove commas
  mutate(budget =
           as.numeric( # get rid of $ and convert to numeric
             str_replace_all(budget, "[^[:alnum:]]", ""))) %>%
  filter(budget < 50000000 & budget > 5000000 &
         movie_run_time < 160 & str_detect(language, "English")) %>%
  select(c(budget, review_rating, movie_run_time))

head(horror)
```

```
## # A tibble: 6 x 3
##     budget review_rating movie_run_time
##      <dbl>         <dbl>          <dbl>
## 1 15000000           5.6            105
## 2  7000000           4.3             95
## 3  5200000           4.4            101
## 4  7750000           3.9             96
## 5 17000000           6.5             91
## 6 22000000           5.5             86
```

```r
horror$budget <- horror$budget * 1e-06 # Changed budget to in millions of dollars
head(horror)
```

```
## # A tibble: 6 x 3
##   budget review_rating movie_run_time
##    <dbl>         <dbl>          <dbl>
## 1  15              5.6            105
## 2   7              4.3             95
## 3   5.2            4.4            101
## 4   7.75           3.9             96
## 5  17              6.5             91
## 6  22              5.5             86
```

```r
budget <- horror$budget
movie_run_time <- horror$movie_run_time # Movie length in minutes
review_rating <- horror$review_rating # Number between 0 - 10
```

MLR model wusing lm()

```r
model_full <- lm(movie_run_time ~ budget +
                   review_rating) # MLR model

summary(model_full)
```

```
##
## Call:
## lm(formula = movie_run_time ~ budget + review_rating)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.037  -6.155  -0.981   5.829  34.266
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    68.2209     5.2273  13.051  < 2e-16 ***
## budget          0.3740     0.1123   3.331  0.00125 **
## review_rating   4.4613     0.9914   4.500 1.98e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.37 on 92 degrees of freedom
## Multiple R-squared:  0.3266, Adjusted R-squared:  0.312
## F-statistic: 22.32 on 2 and 92 DF,  p-value: 1.256e-08
```

```r
model_budget <- lm(movie_run_time ~
                     budget) # SLR model with budget

model_rating <- lm(movie_run_time ~
                     review_rating) # SLR model with review rating

## SLR Model with no intercept:

model_budget_no_intercept <- lm(movie_run_time ~ -1 + budget)

summary(model_budget_no_intercept)
```

```
##
## Call:
## lm(formula = movie_run_time ~ -1 + budget)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -93.184   5.958  36.799  59.194  90.659
##
## Coefficients:
##        Estimate Std. Error t value Pr(>|t|)
## budget   4.6201     0.2748   16.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 49.74 on 94 degrees of freedom
## Multiple R-squared:  0.7505, Adjusted R-squared:  0.7478
## F-statistic: 282.7 on 1 and 94 DF,  p-value: < 2.2e-16
```

```r
coef(model_budget)
```

```
## (Intercept)      budget
##  89.9793563   0.5280998
```

```r
coef(model_rating)
```

```
##   (Intercept) review_rating
##     68.632771      5.467923
```

```r
coef(model_full)
```

```
##   (Intercept)        budget review_rating
##     68.220855      0.374024      4.461262
```

If we are trying to predict movie run time by budget and review ratings, according to the above coefficients our model would be as follows.

Let x_{i1} = Budget and x_{i2} = Review rating.

$$\hat{Y}_i = 68.220855 + 0.374024x_{i1} + 4.461262x_{i2}$$

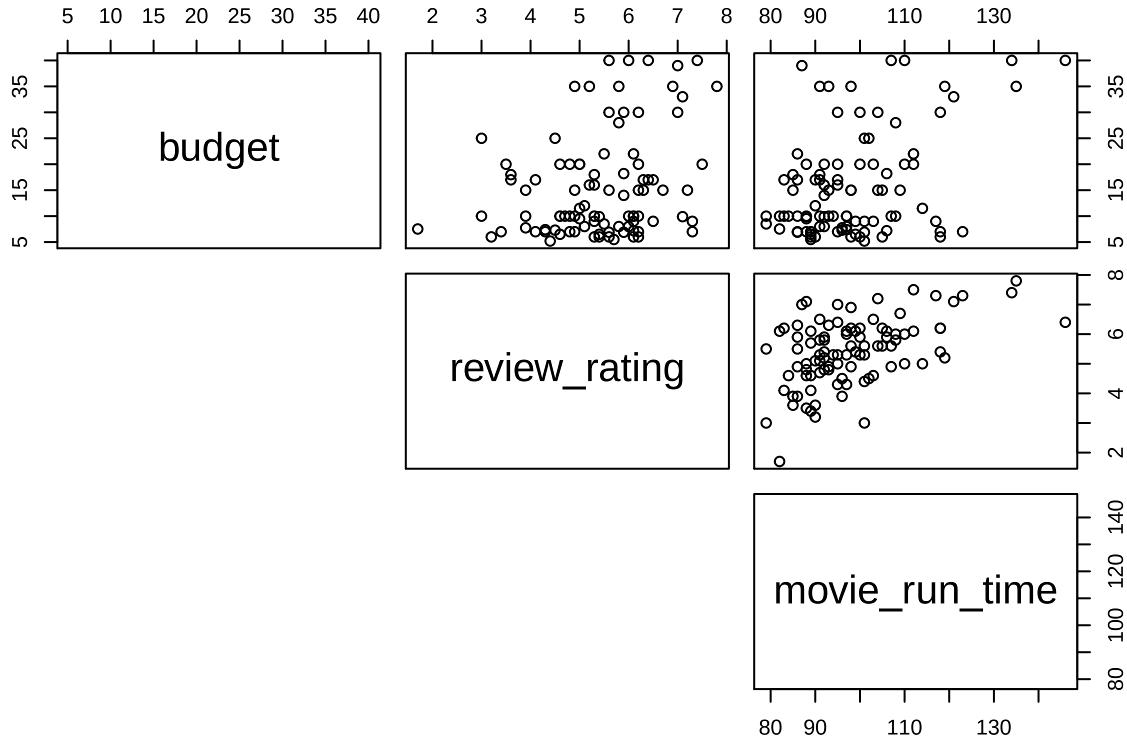
For example if we have a movie that has a budget of 1 million dollars and a review rating of 5, then we would expect this movie to be 90.901189 minutes long.

```r
68.220855 + 0.374024*1 + 4.461262*5
```

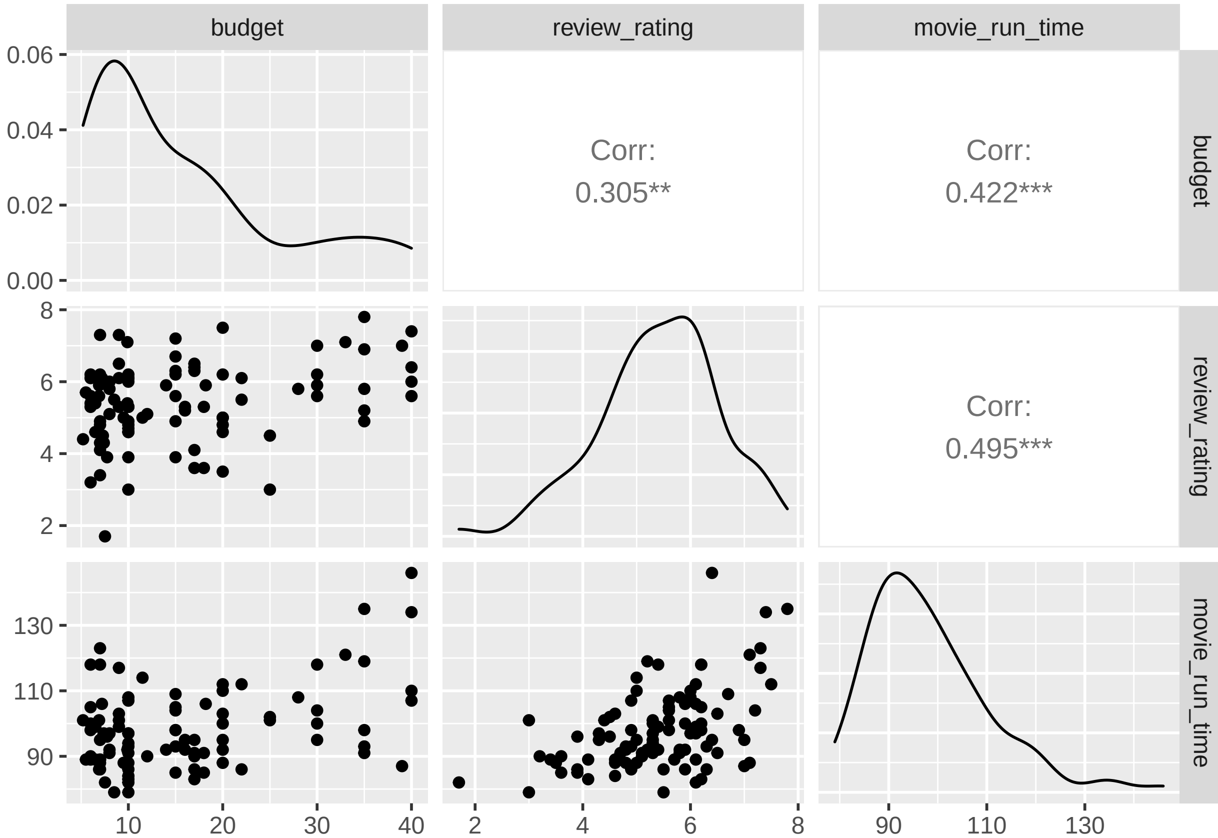
```
## [1] 90.90119
```

Visualizing MLR

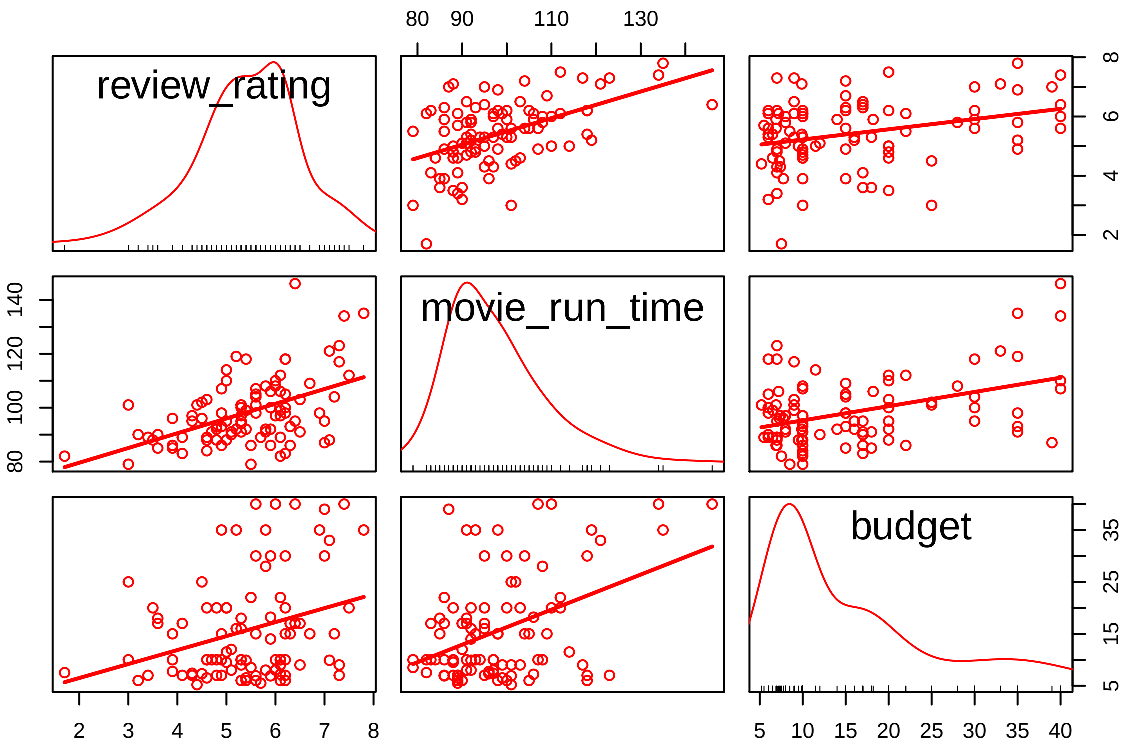
```r
pairs(horror, lower.panel = NULL)
```



```r
ggpairs(horror)
```



```r
# from car package which is loaded when you load the package alr4
scatterplotMatrix(~ review_rating+ movie_run_time + budget, col = "red", smooth = F)
```



Global F-test found in summary() output

$H_0 : \beta_1 = \beta_2 = ... = \beta_p = 0$ vs $H_1 : \beta_j \neq 0$ for some $j = 1, 2, ..., p$

```
summary(model_full)
```

```
##
## Call:
## lm(formula = movie_run_time ~ budget + review_rating)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.037  -6.155  -0.981   5.829  34.266
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    68.2209     5.2273  13.051  < 2e-16 ***
## budget          0.3740     0.1123   3.331  0.00125 **
## review_rating   4.4613     0.9914   4.500 1.98e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.37 on 92 degrees of freedom
## Multiple R-squared:  0.3266, Adjusted R-squared:  0.312
## F-statistic: 22.32 on 2 and 92 DF,  p-value: 1.256e-08
```

Confidence intervals for mean response

- Here we use $x_0 = 50$ bill length (mm)

95% Confidence interval for Mean response

```
penguins_noChinstrap <- penguins %>%
  filter(species != "Chinstrap") %>%
  drop_na(bill_length_mm, body_mass_g)

model <- lm(body_mass_g ~ bill_length_mm , data = penguins_noChinstrap)

n <- nrow(penguins_noChinstrap) # number of observations
x <- penguins_noChinstrap$bill_length_mm # predictor variable
y <- penguins_noChinstrap$body_mass_g # response variable
x_bar <- mean(x) # mean of bill_length_mm
y_bar <- mean(y) # mean of body_mass_g
Sxx <- sum((x - x_bar) ^ 2)
sigma_hat <- summary(model)$sigma # Residual Standard Error (RSE)
Yhat_50 <- # predicted body mass when bill length is 50 mm
  as.numeric(coef(model)[1] + coef(model)[2] * 50)
y_hat <- fitted(model) # fitted values

se_50 <-  sigma_hat*sqrt(1/n + (50 - x_bar)^2/Sxx) # se of y_hat(x_0)
t_pct <-  qt(p = 0.975, df = n - 2) # t-statistic
CI_95 <-  c(Yhat_50 - se_50*t_pct, Yhat_50 + se_50*t_pct)
CI_95
```

```
## [1] 5264.955 5430.243
```

```r
predict(model, newdata = data.frame(bill_length_mm = 50),
        level = 0.95, interval = 'confidence')
```

```
##        fit      lwr      upr
## 1 5347.599 5264.955 5430.243
```

```r
# Can look at multiple values for x0.
predict(model, newdata = data.frame(bill_length_mm = c(50, 55)),
        level = 0.95, interval = 'confidence')
```

```
##        fit      lwr      upr
## 1 5347.599 5264.955 5430.243
## 2 6053.041 5929.836 6176.246
```

Visualizing confidence interval bands

```r
ngrid <-  274
grid <-  seq(min(x), max(x), length = ngrid)
new <- data.frame(bill_length_mm = grid)
p1 <- predict(model, new, se.fit = TRUE, interval = "confidence", level = 0.95)

conf_pred_tib <- tibble(x,y, y_hat, new = new$bill_length_mm,
                        UL_c = p1$fit[,3], LL_c = p1$fit[,2])
```

```r
ggplot(data = conf_pred_tib) +
  geom_point(aes(x = x, y = y), color = "darkgreen", alpha = 0.8, size = 2) +
  geom_line(aes(x = x, y = y_hat), color = "blue") +
  geom_line(aes(x = new, y = UL_c), color = "purple", linetype = "twodash") +
  geom_line(aes(x = new, y = LL_c), color = "purple", linetype = "twodash") +
  scale_x_continuous(breaks = seq(30, 60, by = 10)) +
  scale_y_continuous(breaks = seq(2000, 7000, by = 1000)) +
  labs(x = "bill length (mm)",
       y = "body mass (grams)",
       title = "95% Confidence bands") +
  theme_minimal()
```