Weekly 5 Summary

Gaurang Kakade

Table of contents

Tuesday, Feb 7

! TIL

Today, I learnt the following concepts in class:

1. Integration of regression coefficients
2. Categorical covariates
3. Multiple regression
   - Extension from single regression
   - Other topics

Packages we will require this week

```
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.2 --
v ggplot2 3.4.0     v purrr   1.0.1
v tibble  3.1.8     v dplyr   1.1.0
v tidyr   1.3.0     v stringr 1.5.0
v readr   2.1.3     v forcats 1.0.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

```
library(ISLR2)
library(cowplot)
library(kableExtra)
```

```
Attaching package: 'kableExtra'

The following object is masked from 'package:dplyr':

    group_rows
```

Integration of regression coefficients

What is the interpretation of β_0 and β_1 ?

The regression model is given as follows:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

where:

1. y_i is the response
2. x_1 is the covariate
3. ϵ_i is the error (vertical black line in lecture 4 notes)
4. β_0 and β_1 are the regression coefficients
5. $i = 1, 2, ..., n$ are the indices for the observations

What is the interpretation for the regression coefficients ?

β_0 is the intercept and β_1 is the slope.

Let's consider the following example using `mtcars`

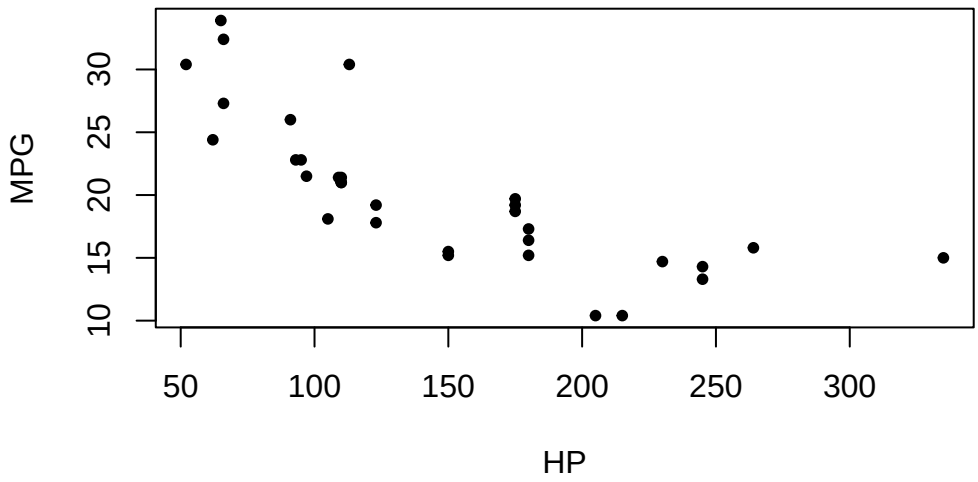
```
library(ggplot2)
mtcars %>% head() %>% kable()
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

The above code uses the mtcars dataset that comes pre-installed with R, the code is using the pipe operator %>% to pass the `mtcars` data to the function head(). This function returns the first 6 rows of the mtcars dataset. The result of head() is then passed to the kable() function from the knitr package. This function formats the data as a nice-looking table and outputs it in the R console.

Consider the following relationship

```
x <- mtcars$hp
y <- mtcars$mpg

plot(x, y, pch = 20, xlab = "HP", ylab = "MPG")
```



```r
model <- lm(y~x) # This line of code creates a linear
# regression model object in R. The response variable "y"
# is modeled as a linear function of the predictor
# variable "x". The syntax of the lm() function stands
# for "linear model". After running this code, the object
# model will contain information about the fit of the
# regression model, such as the coefficients, residuals,
# and other statistical properties.

summary(model)
```

```
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-5.7121 -2.1122 -0.8854  1.5819  8.2360

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 30.09886    1.63392  18.421  < 2e-16 ***
x           -0.06823    0.01012  -6.742 1.79e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.863 on 30 degrees of freedom
Multiple R-squared:  0.6024,    Adjusted R-squared:  0.5892
F-statistic: 45.46 on 1 and 30 DF,  p-value: 1.788e-07
```

For the intercept this means that :

> A "hypothetical" car with `hp = 0` will have `mpg = 30.09` $= \beta_0$

It's more interesting and instructive to consider the interpretation of the slope:

Let's say we have some covariate x_0 then the expected value for $y(x_0)$ is given by:

$$y(x_0) = \beta_0 + \beta_1 x_0$$

What's the expected value for $x_0 + 1$?

$$y(x_0 + 1) = \beta_0 + \beta_1 \times (x_0 + 1)$$

$$= \beta_0 + \beta_1 x_0 + \beta_1$$

$$= y(x_0) + \beta_1$$

$$\implies \beta_1 = y(x_0 + 1) - y(x_0)$$

Categorical covariates

Up until now, we have looked at _simple_ linear regression models where both x and y are quantitative.

Let's confirm that `cyl` is indeed categorical:

```
mtcars$cyl
```

```
 [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
```

Another example is with the iris dataset:

```
iris %>% head() %>% kable()
```

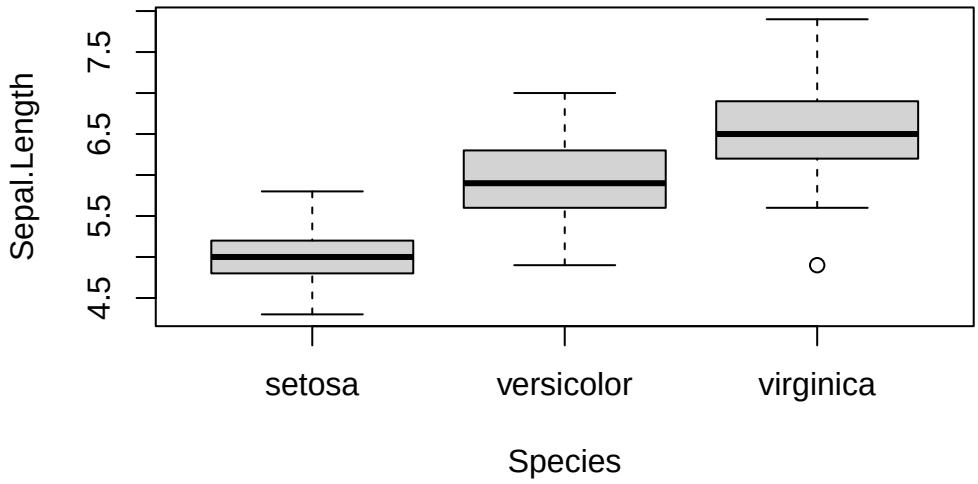
Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

Let's consider the following example:

We want to see if there is a relationship between `species` and `sepal length`. How would we start the EDA?

```
y <- iris$Sepal.Length
x <- iris$Species
```

```r
boxplot(Sepal.Length ~ Species, iris)
```



Let's look just run a linear regression model and see what the model output is going to look like:

```r
cat_model <- lm(Sepal.Length ~ Species, iris)
cat_model
```

```
Call:
lm(formula = Sepal.Length ~ Species, data = iris)

Coefficients:
      (Intercept)  Speciesversicolor   Speciesvirginica
            5.006              0.930              1.582
```

Even if x is categorical, we can still write down the regression model as follows:

$$y_i = \beta_0 + \beta_1 x_i$$

where $x_i \in \{setosa,\ versicolor,\ virginica\}$. This means that we end up with , (fundamentally) three different models.

1. $y_i = \beta_0 + \beta_1 x_i =$ `setosa`
2. $y_i = \beta_0 + \beta_1 x_i =$ `versicolor`
3. $y_i = \beta_0 + \beta_1 x_i =$ `virginica`

This implies that:

1. $y_i = \beta_0 + \beta_1(x_i = c_1)$
2. $y_i = \beta_0 + \beta_1(x_i = c_2)$
3. $y_i = \beta_0 + \beta_1(x_i = c_3)$

This further implies that:

1. $y_i = \beta_0$
2. $y_i = \beta_0 + \beta_1(x_i = c_2)$
3. $y_i = \beta_0 + \beta_1(x_i = c_3)$

Now, the interpretation for the coefficients are as follows;

Intercept

β_0 is the expected y value when x belongs to the base category. This is what the intercept is capturing.

Slopes

β_1 with the name `Species.versicolor` represents the following:

`(Intercept)` $= y(x =$ `setosa`$)$

`Species.versicolor` $= y(x =$ `versicolor`$) - y(x =$ `setosa`$)$

`Species.virginica` $= y(x =$ `verginica`$) - y(x =$ `setosa`$)$

Reordering the factors

Let's say that we didn't want `setosa` to be the baseline level, and, instead we wanted `virginica` to be the baseline level. How would one do this?

First, we're going to reorder/relevel the categorical covariate

```
iris$Species
```

```
  [1] setosa     setosa     setosa     setosa     setosa     setosa
  [7] setosa     setosa     setosa     setosa     setosa     setosa
 [13] setosa     setosa     setosa     setosa     setosa     setosa
 [19] setosa     setosa     setosa     setosa     setosa     setosa
 [25] setosa     setosa     setosa     setosa     setosa     setosa
 [31] setosa     setosa     setosa     setosa     setosa     setosa
 [37] setosa     setosa     setosa     setosa     setosa     setosa
 [43] setosa     setosa     setosa     setosa     setosa     setosa
 [49] setosa     setosa     versicolor versicolor versicolor versicolor
 [55] versicolor versicolor versicolor versicolor versicolor versicolor
 [61] versicolor versicolor versicolor versicolor versicolor versicolor
 [67] versicolor versicolor versicolor versicolor versicolor versicolor
 [73] versicolor versicolor versicolor versicolor versicolor versicolor
 [79] versicolor versicolor versicolor versicolor versicolor versicolor
 [85] versicolor versicolor versicolor versicolor versicolor versicolor
 [91] versicolor versicolor versicolor versicolor versicolor versicolor
 [97] versicolor versicolor versicolor versicolor virginica  virginica
[103] virginica  virginica  virginica  virginica  virginica  virginica
[109] virginica  virginica  virginica  virginica  virginica  virginica
[115] virginica  virginica  virginica  virginica  virginica  virginica
[121] virginica  virginica  virginica  virginica  virginica  virginica
[127] virginica  virginica  virginica  virginica  virginica  virginica
[133] virginica  virginica  virginica  virginica  virginica  virginica
[139] virginica  virginica  virginica  virginica  virginica  virginica
[145] virginica  virginica  virginica  virginica  virginica  virginica
Levels: setosa versicolor virginica
```

```r
iris$Species <- relevel(iris$Species, "virginica")

iris$Species # After
```

```
  [1] setosa     setosa     setosa     setosa     setosa     setosa
  [7] setosa     setosa     setosa     setosa     setosa     setosa
 [13] setosa     setosa     setosa     setosa     setosa     setosa
 [19] setosa     setosa     setosa     setosa     setosa     setosa
 [25] setosa     setosa     setosa     setosa     setosa     setosa
 [31] setosa     setosa     setosa     setosa     setosa     setosa
 [37] setosa     setosa     setosa     setosa     setosa     setosa
 [43] setosa     setosa     setosa     setosa     setosa     setosa
 [49] setosa     setosa     versicolor versicolor versicolor versicolor
 [55] versicolor versicolor versicolor versicolor versicolor versicolor
```

```
 [61] versicolor versicolor versicolor versicolor versicolor versicolor
 [67] versicolor versicolor versicolor versicolor versicolor versicolor
 [73] versicolor versicolor versicolor versicolor versicolor versicolor
 [79] versicolor versicolor versicolor versicolor versicolor versicolor
 [85] versicolor versicolor versicolor versicolor versicolor versicolor
 [91] versicolor versicolor versicolor versicolor versicolor versicolor
 [97] versicolor versicolor versicolor versicolor virginica  virginica
[103] virginica  virginica  virginica  virginica  virginica  virginica
[109] virginica  virginica  virginica  virginica  virginica  virginica
[115] virginica  virginica  virginica  virginica  virginica  virginica
[121] virginica  virginica  virginica  virginica  virginica  virginica
[127] virginica  virginica  virginica  virginica  virginica  virginica
[133] virginica  virginica  virginica  virginica  virginica  virginica
[139] virginica  virginica  virginica  virginica  virginica  virginica
[145] virginica  virginica  virginica  virginica  virginica  virginica
Levels: virginica setosa versicolor
```

Once we do the releveling, we can now run the regression model:

```
new_cat_model <- lm(Sepal.Length ~ Species, iris)
new_cat_model
```

```
Call:
lm(formula = Sepal.Length ~ Species, data = iris)

Coefficients:
      (Intercept)      Speciessetosa  Speciesversicolor
            6.588             -1.582             -0.652
```

Thursday, Jan 19

> **!** TIL
>
> Today, I learnt the following concepts in class:
>
> 1. Multiple regression
> 2. Interpretation of coefficients of multiple regression
> 3. Significance of Interpretation of coefficients of multiple regression

Three plotting libraries

1. base plot (plot)
2. ggplot
3. plotly

```
library(plotly)
```

```
Attaching package: 'plotly'
```

```
The following object is masked from 'package:ggplot2':

    last_plot
```

```
The following object is masked from 'package:stats':

    filter
```

```
The following object is masked from 'package:graphics':

    layout
```

Multiple Regression

This is the extension of simple linear regression to multiple covariates $X = [x_1|x_2|...|x_p]$, i.e.,

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... \beta_p x_p + \epsilon$$

In particular, the data looks like the following:

$\mathbf{y}$	$\mathbf{x}_1$	$\mathbf{x}_2$	...	$\mathbf{x}_p$
y_1	$x_{1,1}$	$x_{2,1}$	...	$x_{p,1}$
y_2	$x_{1,2}$	$x_{2,2}$	...	$x_{p,2}$
y_3	$x_{1,3}$	$x_{2,3}$	...	$x_{p,3}$
⋮	⋮	⋮	...	⋮
y_n	$x_{1,n}$	$x_{2,n}$	...	$x_{p,n}$

and, the full description of the model is as follows:

$$y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \cdots + \beta_p x_{p,i} + \epsilon_i,$$

knitr::kable()

In this, knitr is the package and kable() is the function.

plotly::last_plot()

In this, plotly is the package and last_plot is the function.

ggplot2::plot()

Similarly, in this ggplot2 is the package and plot() is the function.

Consider the `Credit` dataset

```
library(ISLR2)
attach(Credit)

df <- Credit %>% tibble()
colnames(df) <- tolower(colnames(df))
df
```

```
# A tibble: 400 x 11
   income limit rating cards   age educat~1 own   student married region balance
    <dbl> <dbl>  <dbl> <dbl> <dbl>    <dbl> <fct> <fct>   <fct>   <fct>    <dbl>
 1   14.9  3606    283     2    34       11 No    No      Yes     South      333
 2  106.   6645    483     3    82       15 Yes   Yes     Yes     West       903
 3  105.   7075    514     4    71       11 No    No      No      West       580
 4  149.   9504    681     3    36       11 Yes   No      No      West       964
 5   55.9  4897    357     2    68       16 No    No      Yes     South      331
 6   80.2  8047    569     4    77       10 No    No      No      South     1151
 7   21.0  3388    259     2    37       12 Yes   No      No      East       203
 8   71.4  7114    512     2    87        9 No    No      No      West       872
 9   15.1  3300    266     5    66       13 Yes   No      No      South      279
10   71.1  6819    491     3    41       19 Yes   Yes     Yes     East      1350
# ... with 390 more rows, and abbreviated variable name 1: education
```

and, we'll know at the following three columns: `income`, `rating`, `limit`.

```
df3 <- df %>% select(income, rating, limit)
df3
```

```
# A tibble: 400 x 3
   income rating limit
    <dbl>  <dbl> <dbl>
 1   14.9    283  3606
 2  106.     483  6645
 3  105.     514  7075
 4  149.     681  9504
 5   55.9    357  4897
 6   80.2    569  8047
 7   21.0    259  3388
 8   71.4    512  7114
 9   15.1    266  3300
10   71.1    491  6819
# ... with 390 more rows
```

If we want to see how the credit limit is related to income and credit rating, we can visualize the following plot

```
fig <- plot_ly(df3, x=~income, y=~rating, z=~limit)
fig %>% add_markers()
```

The regression mode is as follows:

```
model <- lm(limit ~ income + rating, df3)
model
```

```
Call:
lm(formula = limit ~ income + rating, data = df3)

Coefficients:
(Intercept)       income       rating
  -532.4711       0.5573      14.7711
```

What does the regression model look like here?

```
ranges <- df3 %>%
    select(income, rating) %>%
    colnames() %>%
    map(\(x) seq(0.1 * min(df3[x]), 1.1 * max(df3[x]),
    length.out = 50))
```

```r
# The code is generating a sequence of numbers that
# represent the range of values for two variables income
# and rating in the data frame df3. This sequence of
# numbers is being stored in the ranges object.
# The code uses the select function to select only the
# columns income and rating from the data frame df3. Then
# it uses the colnames function to extract the names of
# these columns. Next, the code uses the map function
# from the purrr package to apply a function to each
# column name. The function generates a sequence of
# numbers that covers the range of values in each column,
# with a step of 0.1 times the minimum value and 1.1
# times the maximum value. The number of numbers in the
# sequence is specified by the length.out argument, which
# is set to 50.


b <- model$coefficients
z <- outer(
    ranges[[1]],
    ranges[[2]],
    Vectorize(function(x2, x3) {
        b[1] + b[2] * x2 + b[3] * x3
    })
)

# This code is defining a matrix z that contains the
# predicted values for a multiple linear regression model
# with two predictors (income and rating) based on the
# coefficients b from the model. The outer function
# calculates the Cartesian product of the two ranges
# (ranges[[1]] and ranges[[2]]), i.e., the values of x2
# and x3 for all possible combinations of these two
# variables. For each combination, the linear equation
# represented by the coefficients b is calculated using
# b[1] + b[2] * x2 + b[3] * x3 and the result is stored
# in the matrix z.


fig %>%
    add_surface(x = ranges[[1]], y = ranges[[2]], z = t(z),
```

```
    alpha = 0.3) %>%
    add_markers()
```

What is the interpretation for this coefficients?

1. β_0 is the expected value of y when $income = 0$ and $rating = 0$
2. β_1 is saying that if *rating* is held constant and *income* changes by 1 unit, then the corresponding change in the `limit` is 0.5573.
3. β_2 is saying that if `income` is held constant and `rating` changes by 1 unit, then the corresponding change in `limit` is 14.771.

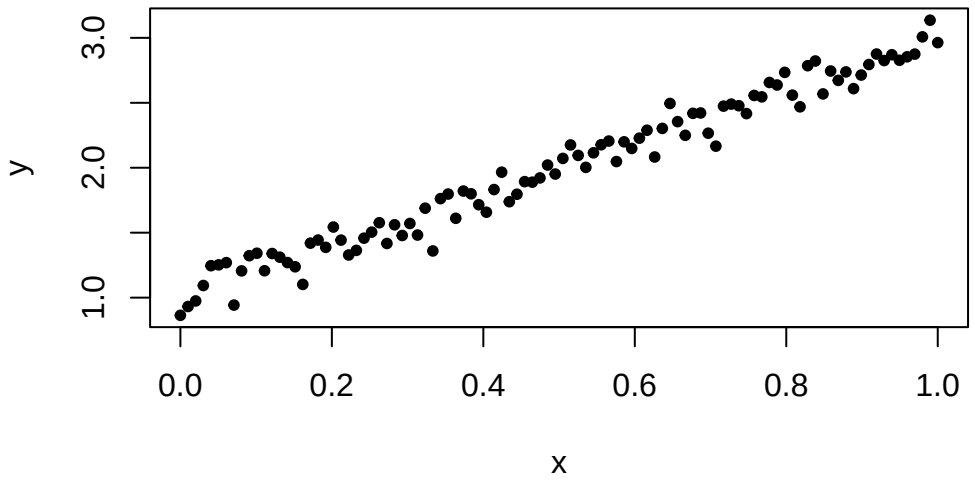
What about the significance ?

```
summary(model)
```

```
Call:
lm(formula = limit ~ income + rating, data = df3)

Residuals:
    Min      1Q  Median      3Q     Max
-420.97 -121.77   14.97  126.72  485.48

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -532.47115   24.17283 -22.028   <2e-16 ***
income         0.55727    0.42349   1.316    0.189
rating        14.77115    0.09647 153.124   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 182.3 on 397 degrees of freedom
Multiple R-squared:  0.9938,    Adjusted R-squared:  0.9938
F-statistic: 3.18e+04 on 2 and 397 DF,  p-value: < 2.2e-16
```

Multiple regression

```r
x <- seq(0, 1, length.out = 100)
b0 <- 1.0
b1 <- 2.0
y <- b0 + b1 * x + rnorm(100) * 0.1
plot(x,y, pch = 20)
```




```r
model <- lm(y~x)
summary(model)
```

```
Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-0.30218 -0.06497  0.00815  0.06555  0.21026

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.99864    0.02001   49.92   <2e-16 ***
x            1.98907    0.03456   57.55   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1008 on 98 degrees of freedom
```

```
Multiple R-squared:  0.9713,    Adjusted R-squared:  0.971
F-statistic:  3312 on 1 and 98 DF,  p-value: < 2.2e-16
```

This code generates a linear model in R. It starts by generating an **x** sequence of 100 equally spaced values between 0 and 1. Then it creates two variables, **b0** and **b1**, with values of 1.0 and 2.0, respectively. It uses these values to generate **y** with a formula **b0 + b1 * x + rnorm(100) * 0.7**. **rnorm(100)** generates random normal deviates with a mean of 0 and standard deviation of 0.7. The code then plots **x** against **y** using the **plot()** function and fits a linear regression model to the data using the **lm()** function with the formula **y ~ x**. Finally, it summarizes the results of the linear regression model using the **summary()** function.

The value **0.1** in the code is the standard deviation of the normal distribution used to generate random noise to add to the **y** variable.

As the standard deviation increases, the Adjusted R-squared decreases

```
library(ISLR2)
library(dplyr)
library(ggplot2)

attach(ISLR2::Credit)
```

```
The following objects are masked from Credit:

    Age, Balance, Cards, Education, Income, Limit, Married, Own,
    Rating, Region, Student
```

```
df <- Credit %>% tibble() %>% rename_all(tolower)
df
```

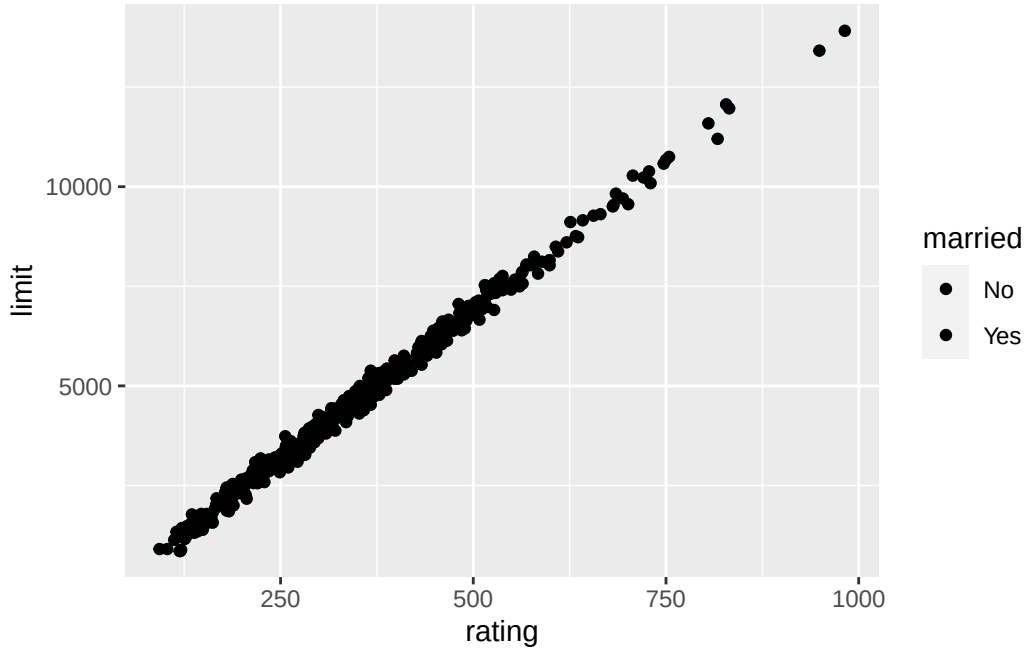
```
# A tibble: 400 x 11
   income limit rating cards   age educat~1 own   student married region balance
    <dbl> <dbl>  <dbl> <dbl> <dbl>    <dbl> <fct> <fct>   <fct>   <fct>    <dbl>
 1   14.9  3606    283     2    34       11 No    No      Yes     South      333
 2  106.   6645    483     3    82       15 Yes   Yes     Yes     West       903
 3  105.   7075    514     4    71       11 No    No      No      West       580
 4  149.   9504    681     3    36       11 Yes   No      No      West       964
 5   55.9  4897    357     2    68       16 No    No      Yes     South      331
 6   80.2  8047    569     4    77       10 No    No      No      South     1151
 7   21.0  3388    259     2    37       12 Yes   No      No      East       203
 8   71.4  7114    512     2    87        9 No    No      No      West       872
```

```
 9   15.1  3300    266     5    66       13 Yes   No      No      South      279
10   71.1  6819    491     3    41       19 Yes   Yes     Yes     East      1350
# ... with 390 more rows, and abbreviated variable name 1: education
```

```
model <- lm(limit ~ rating + married, df)
model
```

```
Call:
lm(formula = limit ~ rating + married, data = df)

Coefficients:
(Intercept)       rating   marriedYes
    -528.09        14.87       -25.97
```

```
ggplot(df) +
      geom_point(aes(x = rating, y = limit, color = married)) +
      geom_point(aes(x = rating, y = limit, fill= married))
```



This code is loading the **ISLR2** and **dplyr** packages, as well as the **ggplot2** library. The **Credit** data set is loaded from the **ISLR2** package and saved as a tibble named **df**. **attach** is a function in R that temporarily loads a data set into the search path, so that its objects

can be accessed directly by name, rather than having to reference the data set each time with its full name. In other words, you can use the variables in the data set as if they are in the work space. However, it is generally not recommended to use **`attach`** as it can lead to naming conflicts and make the code harder to maintain.The **`rename_all`** function is used to convert the names of all columns in the tibble to lowercase. A linear regression model is then fit to the data using the formula **`limit ~ rating + married`**. The response variable **`limit`** is modeled as a linear function of the predictor variables **`rating`** and **`married`**. Finally, a scatterplot is created using **`ggplot2`**, with the points colored and filled according to the value of the **`married`** variable.