

Lab 4 Solved: base R with vectors and data frames

Harris Coding Camp

Summer 2023

Using [and other base R tools for data analysis.

Warm-up

We'll use midwestern demographic data which is at this [link](#). The dataset includes county level data for a single year. We call data this type of data "cross-sectional" since it gives a point-in-time cross-section of the counties of the midwest.

```
# SOLUTION
midwest_data <- haven::read_dta("../Data/midwest.dta")
nrow(midwest_data)

## [1] 437

ncol(midwest_data)

## [1] 28

names(midwest_data)

##  [1] "PID"                  "county"                "state"
##  [4] "area"                 "poptotal"              "popdensity"
##  [7] "popwhite"              "popblack"               "popamerindian"
## [10] "popasian"              "popother"               "percwhite"
## [13] "percblack"              "percamerindian"        "percasian"
## [16] "percother"              "popadults"              "perchsd"
## [19] "percollege"             "percprof"               "poppovertyknown"
## [22] "percpovertyknown"       "percbelowpoverty"       "percchildbelowpovert"
## [25] "percadultpoverty"       "percelderlypoverty"     "inmetro"
## [28] "category"
```

1. What format¹ is the midwest data in? What function do you need to load it?
2. Load the package so you can read in the data and assign it to the name `midwest_data`. If you don't remember what package contains that function use `??` (as in `??read_xxx`)
`??` is a way to search through help for functions that are not currently loaded in R or if you forgot the exact name of a function.

¹i.e. is it a csv, dta, xlsx?

3. How many rows and columns does the data have?
4. Notice that row represents a county which is uniquely identified by a PID or using `county + state`.
5. Use `names()` to see the names of all the columns.

Using [and \$ with vectors.

Recall that columns are vectors and you can extract the vector using `$`.

1. Extract the `inmetro` and calculate the mean.

```
# SOLUTION
mean(midwest_data$inmetro)
```

```
## [1] 0.3432494
```

We might interpret the result as the proportion of counties that are urban ... but the data did not come with a codebook, so we are not sure! Let's explore.

2. Run the following code and explain in words what the two results are. Assign the two resulting vectors to names that reflect what they are.

```
midwest_data$poptotal[midwest_data$inmetro == 1]
midwest_data$poptotal[midwest_data$inmetro == 0]
```

3. What is the average (mean) population for urban midwest counties? How about non-urban counties? Do these numbers make sense?
4. What are the `max()` and `min()` for these vectors? Do these numbers make sense?
5. How many “urban counties” have fewer than 50000 residents. What proportion of the counties is this?

```
# SOLUTIONS
urban_pop <- midwest_data$poptotal[midwest_data$inmetro == 1]
rural_pop <- midwest_data$poptotal[midwest_data$inmetro == 0]

mean(urban_pop)
```

```
## [1] 223168.1
```

```
mean(rural_pop)
```

```
## [1] 29734.23
```

```
min(urban_pop)
```

```
## [1] 5315
```

```

max(urban_pop)

## [1] 5105067

min(rural_pop)

## [1] 1701

max(rural_pop)

## [1] 107066

length(urban_pop[urban_pop < 50000])

## [1] 37

length(urban_pop[urban_pop < 50000])/length(urban_pop)

## [1] 0.2466667

# this is a bit more sophisticated
sum(urban_pop < 50000)

## [1] 37

mean(urban_pop < 5000)

## [1] 0

```

1. You can use `sort()` to see the numbers in order.
2. What is the population of the 10th smallest urban county in the midwest? How about the 10th largest?
(Hint: Use `?sort` to learn how to change the order of your sort.)

```

# SOLUTION
# 10th smallest
sort(midwest_data$poptotal[midwest_data$inmetro == 1] )[10]

```

```

## [1] 25968

# 10th largest
sort(midwest_data$poptotal[midwest_data$inmetro == 1], decreasing = TRUE )[10]

## [1] 717400

```

3. What are the name of the 4 urban counties with population under 20000? (You can use `&` to combine conditional expressions.)
4. What are the PID of the 4 urban counties with population under 20000?

```

# SOLUTION
midwest_data$county[midwest_data$poptotal < 20000 & midwest_data$inmetro == 1]

## [1] "MENARD"      "OHIO"        "TIPTON"       "VERMILLION"

midwest_data$PID[midwest_data$poptotal < 20000 & midwest_data$inmetro == 1]

## [1] 625 720 742 745

# we could also use double indexing as we discuss below.
midwest_data[midwest_data$poptotal < 20000 & midwest_data$inmetro == 1, "PID"]

## # A tibble: 4 x 1
##   PID
##   <dbl>
## 1 625
## 2 720
## 3 742
## 4 745

midwest_data[midwest_data$poptotal < 20000 & midwest_data$inmetro == 1, "state"]

## # A tibble: 4 x 1
##   state
##   <chr>
## 1 IL
## 2 IN
## 3 IN
## 4 IN

```

1. What states are those counties in?

See above

1. *Common error:* Run the code below and get an error that says “Column County doesn’t exist”. Fix the error.

```

midwest_data[midwest_data$poptotal < 20000 & midwest_data$inmetro == 1,
             c("County", "state", "poptotal", "popdensity", "inmetro")]

```

This is a typo, the column name is country

1. Adjust the code above so we get the same rows, but see all the columns.

```

midwest_data[midwest_data$poptotal < 20000 & midwest_data$inmetro == 1, ]

```

```

## # A tibble: 4 x 28
##   PID county state area poptotal popdensity popwhite popblack popamerindian
##   <dbl> <chr>  <chr> <dbl>    <dbl>      <dbl>      <dbl>      <dbl>
## 1 625 MENARD IL    0.018    11164     620.    11101       9      29
## 2 720 OHIO   IN    0.005     5315     1063     5255      41       8
## 3 742 TIPTON IN    0.016    16119     1007.    15990      10      20
## 4 745 VERMILL~ IN    0.016    16773     1048.    16690      15      32
## # ... with 19 more variables: popasian <dbl>, popother <dbl>, percwhite <dbl>,
## #   percblack <dbl>, percamerindan <dbl>, perciasian <dbl>, percother <dbl>,
## #   popadults <dbl>, perchsd <dbl>, percollege <dbl>, percprof <dbl>,
## #   poppovertyknown <dbl>, percpovertyknown <dbl>, percbelowpoverty <dbl>,
## #   percchildbelowpovert <dbl>, percadultpoverty <dbl>,
## #   percelderlypoverty <dbl>, inmetro <dbl>, category <chr>

```

- These are the low population counties where `inmetro` is 1. Are the population densities low? Compare them to the population densities of similar population counties where `inmetro` is 0. If you have time, you can look at the locations on a map and get a better understanding of what `inmetro` captures.

```

# the popdensity are similar for these regions, though the inmetro == 1 counties
# are slightly bigger
midwest_data[midwest_data$inmetro == 0 & midwest_data$poptotal < 20000, ]

```

```

## # A tibble: 119 x 28
##   PID county state area poptotal popdensity popwhite popblack popamerindian
##   <dbl> <chr>  <chr> <dbl>    <dbl>      <dbl>      <dbl>      <dbl>
## 1 562 ALEXAN~ IL    0.014    10626     759     7054     3496      19
## 2 563 BOND    IL    0.022    14991     681.    14477     429       35
## 3 565 BROWN   IL    0.018    5836     324.     5264     547       14
## 4 567 CALHOUN IL    0.017    5322     313.     5298       1       8
## 5 568 CARROLL IL    0.027    16805     622.    16519     111       30
## 6 569 CASS    IL    0.024    13437     560.    13384      16       8
## 7 572 CLARK   IL    0.03     15921     531.    15842      10      26
## 8 573 CLAY    IL    0.028    14460     516.    14403       4      17
## 9 577 CRAWFO~ IL    0.026    19464     749.    19300      63      34
## 10 578 CUMBER~ IL    0.02     10670     534.    10627       5       6
## # ... with 109 more rows, and 19 more variables: popasian <dbl>,
## #   popother <dbl>, percwhite <dbl>, percblack <dbl>, percamerindan <dbl>,
## #   perciasian <dbl>, percother <dbl>, popadults <dbl>, perchsd <dbl>,
## #   percollege <dbl>, percprof <dbl>, poppovertyknown <dbl>,
## #   percpovertyknown <dbl>, percbelowpoverty <dbl>, percchildbelowpovert <dbl>,
## #   percadultpoverty <dbl>, percelderlypoverty <dbl>, inmetro <dbl>,
## #   category <chr>

```

```
midwest_data[midwest_data$inmetro == 0, "popdensity"]
```

```

## # A tibble: 1 x 1
##   popdensity
##   <dbl>
## 1 1271.
## 2 759
## 3 681.
## 4 324.

```

```

## 5      714.
## 6      313.
## 7      622.
## 8      560.
## 9      819.
## 10     531.
## # ... with 277 more rows

```

Rapid fire:

Using [and \$ complete the following challenges:

1. What states have an Adams County?
2. How many counties are in Indiana?
3. What county has the highest percent Asian in this data?
4. Make a data frame that includes the 10 largest counties (by total population) and shows the county, state, total population, and percent with college degree. Assign the output to the name `top_ten`.
5. A student tried to answer “How many counties are in Indiana?” with the following code:

```
length(midwest_data[midwest_data$state == "IN", ])
```

```
## [1] 28
```

What does `length()` count for tibbles and `data.frames`?²

Length provides the number of columns! This makes sense if we keep in mind that the tibble is a list and the relevant length is the number of “pointers” need to store the list.

Ordering columns

It would be nice to sort the table with the 10 largest counties by population. How do we do that?

Let’s start with a simpler example – this is a good way to get intuition for what the code does!

1. Create the following test data set. Your random numbers will be different!

```

test_data <- tibble(
  id = c(1, 4, 2, 3, 5),
  gpa = 4 * runif(5)
)

test_data

```

```

## # A tibble: 5 x 2
##       id     gpa
##   <dbl> <dbl>
## 1     1    2.77
## 2     4    1.88
## 3     2    1.60
## 4     3    1.11
## 5     5    3.01

```

²Use `nrow()` instead!

2. Explain what the following code does.

```
test_data[c(1, 2),]
```

```
## # A tibble: 2 x 2
##       id     gpa
##   <dbl> <dbl>
## 1     1    2.77
## 2     4    1.88
```

```
test_data[c(2, 1),]
```

```
## # A tibble: 2 x 2
##       id     gpa
##   <dbl> <dbl>
## 1     4    1.88
## 2     1    2.77
```

Did you notice that the order of the rows changed! This suggests that we could re-order or sort the data frame, if we knew the correct **order** of the rows.

3. Pull out the rows that correspond to ids 1, 2, 3. (e.g. the **3rd** row corresponds to **id** number 2). To make a data frame like this:

Solution

```
test_data[c(1,3,4), ]
```

```
## # A tibble: 3 x 2
##       id     gpa
##   <dbl> <dbl>
## 1     1    2.77
## 2     2    1.60
## 3     3    1.11
```

4. base R has a function called **order()** which tells you the order of the rows (in increasing order). Use **order()** on the test_data ids. Plug this into your **[** to sort the data.

Solution

```
test_data[order(test_data$id), ]
```

```
## # A tibble: 5 x 2
##       id     gpa
##   <dbl> <dbl>
## 1     1    2.77
## 2     2    1.60
## 3     3    1.11
## 4     4    1.88
## 5     5    3.01
```

5. Now sort **top_ten** in **decreasing** order. The expected output is:

```
# Solution
top_ten[order(top_ten$poptotal, decreasing = TRUE), ]
```

```
## # A tibble: 10 x 4
##   county    state poptotal percollege
##   <chr>     <chr>    <dbl>      <dbl>
## 1 COOK      IL      5105067     28.0
## 2 WAYNE     MI      2111687     19.4
## 3 CUYAHOGA  OH      1412140     25.1
## 4 OAKLAND    MI      1083592     37.0
## 5 FRANKLIN  OH      961437      32.2
## 6 MILWAUKEE WI      959275      25.4
## 7 HAMILTON  OH      866228      29.8
## 8 MARION     IN      797159      26.7
## 9 DU PAGE   IL      781666      42.8
## 10 MACOMB    MI      717400      20.7
```

II. Investigate the diamonds dataset

Throughout this exercise, we will be working with the `diamonds` dataset (comes with `tidyverse`), which contains the prices and other attributes of almost 54,000 diamonds. (use `?diamonds` to see the codebook.)

- Run the following command to familiarize ourselves with this dataset. How many observations and variables are included in `diamonds`?

SOLUTION: 53,940 observations (rows) and 10 variables (columns)

```
# tidyverse
glimpse(diamonds)
# base R (utils)
str(diamonds)
```

- Try to describe the shape and center of the `price` distribution by observing the summary statistics like the mean, median and quartiles.

SOLUTION: We see that the mean is much larger than the median, which suggests there are some very expensive diamonds in the right tail of the distribution.

```
summary(diamonds$price)
```

- How many diamonds cost less than \$500? less than \$250? How many diamonds cost \$15000 or more?

Solution: from the summary, we see there are no diamonds less than \$250. It's also clear that <\$500 and >\$15000 are a small portion of the data as both values are close to the extremes and outside of the middle 50 percent of the data (aka the interquartile range). Specifically:

```
nrow(diamonds[diamonds$price < 500, ])
```

```
## [1] 1729
```

```
nrow(diamonds[diamonds$price > 15000, ])
```

```
## [1] 1655
```

4. Which cut has the highest priced diamond? What is the price?

```
# bracket and dollar sign solution
diamonds[diamonds$price == max(diamonds$price), c("cut", "price")]
```

```
## # A tibble: 1 x 2
##   cut      price
##   <ord>    <int>
## 1 Premium 18823
```

5. Redo part 4 with the lowest priced diamond.

```
diamonds[diamonds$price == min(diamonds$price), c("cut", "price")]
```

```
## # A tibble: 2 x 2
##   cut      price
##   <ord>    <int>
## 1 Ideal    326
## 2 Premium  326
```

6. Is there any relationship between `price` and `carat` of a diamond? What might explain that pattern?
Run the code below and comment on the result.

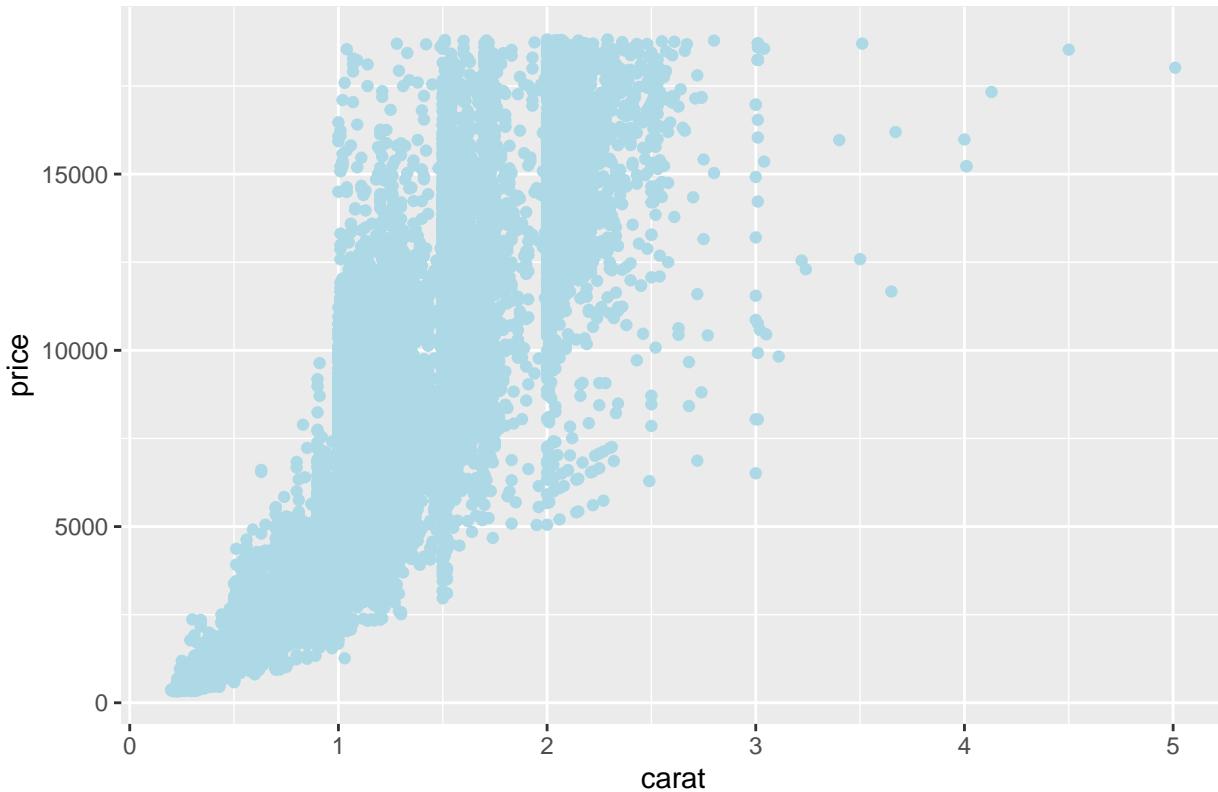
```
ggplot(diamonds) +
  geom_point(aes(x = log(carat), y = log(price)),
             color = "lightblue") +
  labs(title = "Price vs Carat")
```

7. What does the graph look like if we don't take logs? Is the relationship still linear?

Solution: When we remove the logs, the relationship is no longer linear. We see a large increase in price that is quite dispersed once the diamond is over 1 carat. We also notice bunching around round numbers of carats

```
ggplot(diamonds) +
  geom_point(aes(x = carat, y = price),
             color = "lightblue") +
  labs(title = "Price vs Carat")
```

Price vs Carat



8. We might want the specific statistical relationship. How much does $\log(\text{price})$ increase on average when the $\log(\text{carat})$ size increases by one unit? Enter statistics!

The function `lm()` runs a “linear regression” and answers our question: the $\log(\text{price})$ increases by 1.676 log units for each log-carat.³

```
lm(log(price) ~ log(carat), data = diamonds)
```

9. We can visualize this using the following code:

```
ggplot(diamonds) +  
  geom_point(aes(x = log(carat), y = log(price)),  
             color = "lightblue") +  
  # we get the estimates from linear regression output  
  geom_abline(aes(slope = 1.676, intercept = 8.449)) +  
  labs(title = "Price vs Carat")
```

10. Is the relationship stable across cut? Run a regression to find out how much does $\log(\text{price})$ increase on average when the $\log(\text{carat})$ size increases by one unit for **good cut diamonds**? Make a plot showing the relationship.

³This is a preview of your stats sequence!

```

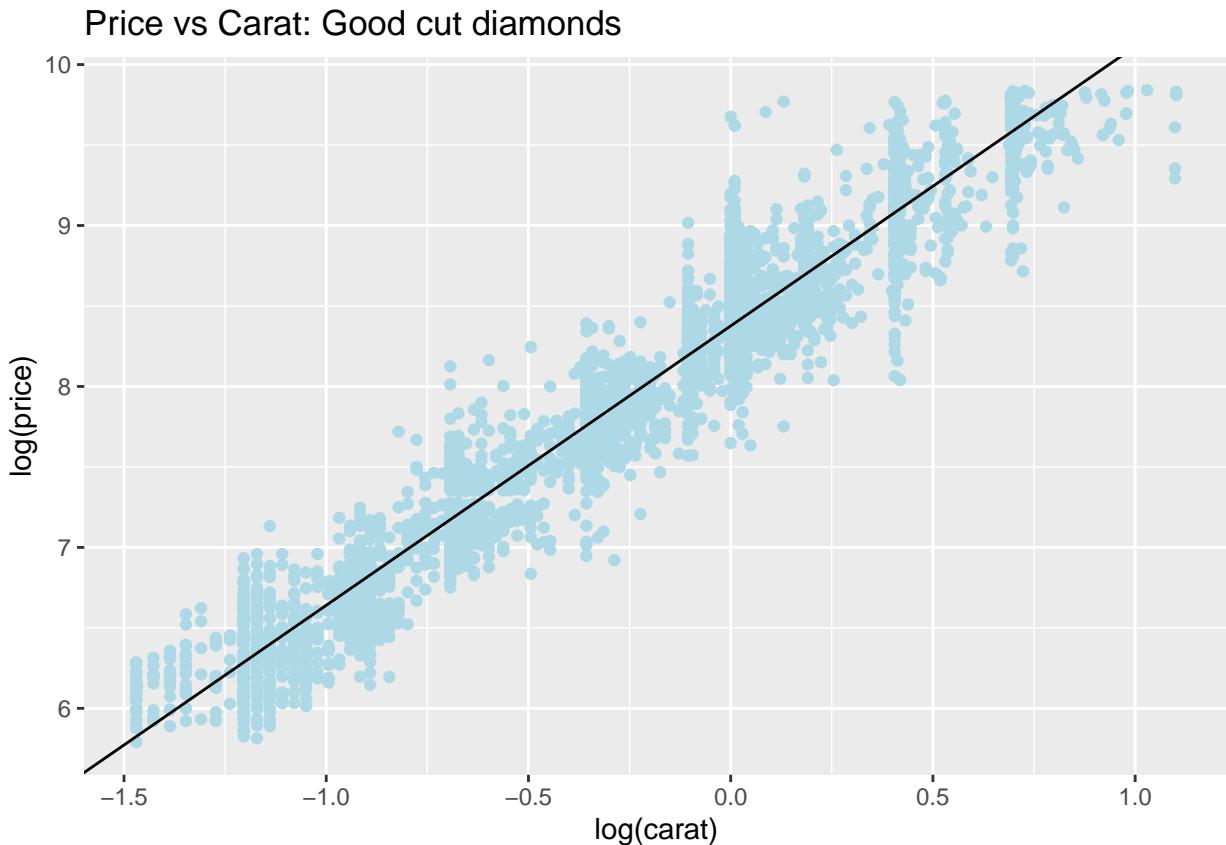
good_cut_diamonds <- diamonds[diamonds$cut == "Good",]

lm(log(price) ~ log(carat), data = good_cut_diamonds)

##
## Call:
## lm(formula = log(price) ~ log(carat), data = good_cut_diamonds)
##
## Coefficients:
## (Intercept)  log(carat)
##           8.376        1.736

ggplot(good_cut_diamonds) +
  geom_point(aes(x = log(carat), y = log(price)),
             color = "lightblue") +
  geom_abline(aes(slope = 1.736, intercept = 8.376)) +
  labs(title = "Price vs Carat: Good cut diamonds")

```



We can make the above plots with base R as well.

```

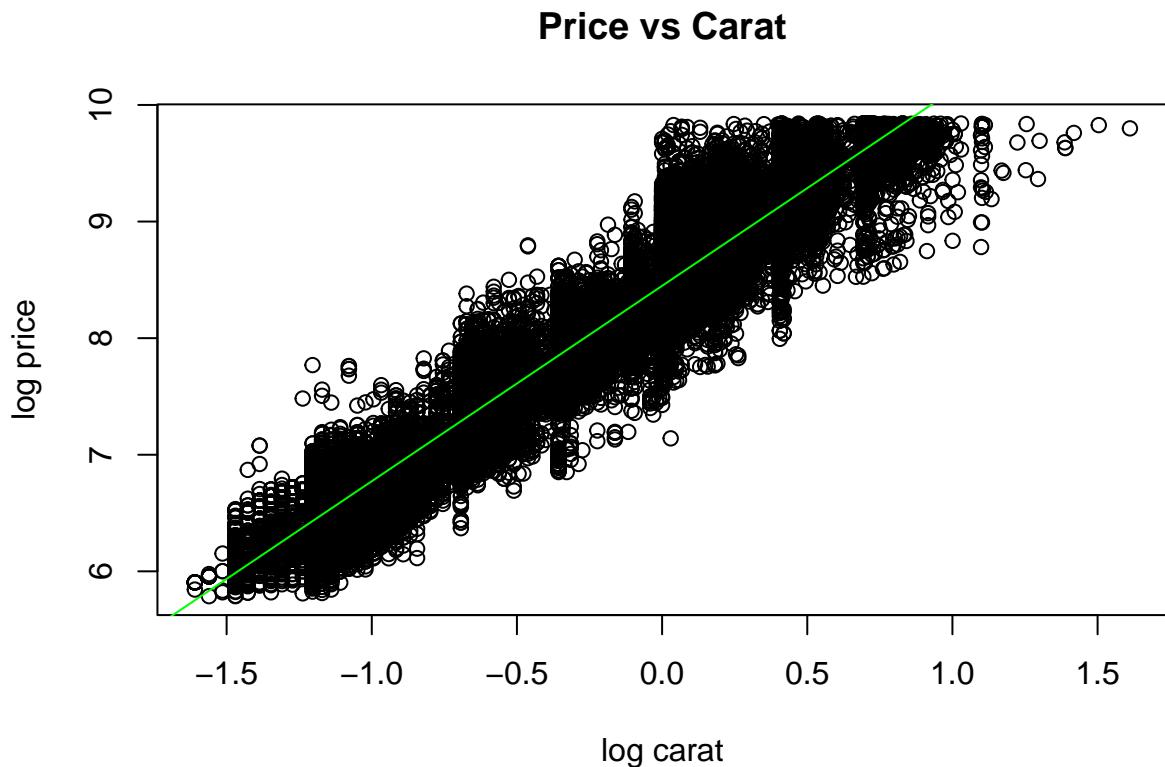
reg_model <- lm(log(price) ~ log(carat), data = diamonds)

```

```

plot(log(price) ~ log(carat), # plot log(price) by log(carat)
     type = "p", # plot type "p" (points)
     data = diamonds, # data from diamonds
     ylab = "log price", # add y-axis label
     xlab = "log carat", # add x-axis label
     main = "Price vs Carat" # add title
   )
abline(reg_model, col="green")

```



```

plot(price ~ carat, # plot log(price) by log(carat)
     type = "p", # plot type "p" (points)
     data = diamonds, # data from diamonds
     ylab = "price", # add y-axis label
     xlab = "carat", # add x-axis label
     main = "Price vs Carat" # add title
   )

```

Price vs Carat

