Lab 2: Matrices and Cleaning Data

2025-04-11

Matrices

How do we handle matrices in R?

Vectors Let's start with vectors (which are matrices where one of the dimensions is 1)

Suppose I want the following vectors:

$$a = \begin{pmatrix} 1 \\ 5 \\ 8 \end{pmatrix}, \; b = \begin{pmatrix} 4 & 9 & 0 \end{pmatrix}, \; b * a = ?$$

```
# Use c()

a = c(1, 5, 8)

b = c(4, 9, 0)


# What if I multiply it?

b*a # That does element wise (4*1  9*5  0*8)
```

```
## [1]  4 45  0
```

```
# For vector and matrix multiplication: %*%
b%*%a
```

```
##      [,1]
## [1,]   49
```

```
a%*%b
```

```
##      [,1]
## [1,]   49
```

Uh oh! $b * a = a * b$? That's not true! What's going on? R thinks it's being smart

- $\% * \%$ performs the *inner* product of any two vectors and will make them conform
- If you want $a * b$ (the *outer* product of a and b) use $outer(b, a)$
- Basically: R will make your vectors conform. Which is usually nice, but **be careful!**

Matrices How to write one, how to transpose, how to invert, multiply?

Suppose we want these matrices:

$$X = \begin{pmatrix} 3 & 1 & 2 \\ 0 & 1 & 5 \\ 4 & 0 & 2 \end{pmatrix}, \; Y = \begin{pmatrix} 9 & 3 & 0 \\ 4 & 2 & 3 \end{pmatrix}$$

The *matrix*() function works like this: **matrix(c(list numbers), nrow = ?, ncol = ?)**

- **c(list numbers)**: list all the numbers in your matrix starting from the top right and moving *down*
  - For X: **c(3, 0, 4, 1, 1, 0, 2, 5, 2)**
- **nrow** = number of rows: of the list given, how many rows should this matrix have?
- **ncol** = number of columns: of the list, how many columns should this matrix have? (sufficient to only have 1 of **nrows** or **ncols**)

```
# Write Matrices: matrix(c(numbers), nrow = number of rows, ncol = number of columns)
X = matrix(c(3, 0, 4, 1, 1, 0, 2, 5, 2), ncol = 3)

X
```

```
##      [,1] [,2] [,3]
## [1,]    3    1    2
## [2,]    0    1    5
## [3,]    4    0    2
```

```
Y = matrix(c(9, 4, 3, 2, 0, 3), ncol = 3)

Y
```

```
##      [,1] [,2] [,3]
## [1,]    9    3    0
## [2,]    4    2    3
```

Matrix multiplication: just like with vectors: $\% * \%$

$$\text{Try:} \quad Y * X, \quad b * X, \quad X * a$$

```
# Y*X (2 x 3)
Y %*% X
```

```
##      [,1] [,2] [,3]
## [1,]   27   12   33
## [2,]   24    6   24
```

```
# b*X (1 x 3)
b %*% X
```

```
##      [,1] [,2] [,3]
## [1,]   12   13   53
```

```r
# X*a (3 x 1)
X %*% a
```

```
##      [,1]
## [1,]   24
## [2,]   45
## [3,]   20
```

As you can see, it makes the vectors a and b conformable to the matrixes!

More operations:

- Transpose: $t(matrix)$
- Invert: $solve(matrix)$
- Determinant: $det(matrix)$
- Eigenvalues and vectors: $eigen(matrix)\$values$, $eigen(matrix)\$vectors$
- Diagonal: $diag(matrix)$

```r
# Transpose X
X_T = t(X)
X_T
```

```
##      [,1] [,2] [,3]
## [1,]    3    0    4
## [2,]    1    1    0
## [3,]    2    5    2
```

```r
# Invert X and prove it's the inverse
X_inv = solve(X)

X_inv
```

```
##            [,1]       [,2]       [,3]
## [1,]  0.1111111 -0.1111111  0.1666667
## [2,]  1.1111111 -0.1111111 -0.8333333
## [3,] -0.2222222  0.2222222  0.1666667
```

```r
X %*% X_inv # not perfect since X_inv rounds to decimals
```

```
##      [,1]          [,2]         [,3]
## [1,]    1 -5.551115e-17 5.551115e-17
## [2,]    0  1.000000e+00 0.000000e+00
## [3,]    0  0.000000e+00 1.000000e+00
```

```r
# Determinant of X
det_X = det(X)

det_X
```

```
## [1] 18
```

```
# Eigenvalues and Vectors
X_eig = eigen(X)

X_eig$values
```

```
## [1]  6.000000e+00+0.000000i -2.498002e-16+1.732051i -2.498002e-16-1.732051i
```

```
X_eig$vectors
```

```
##                [,1]                     [,2]                     [,3]
## [1,] -0.5773503+0i -0.04508348-0.2342606i -0.04508348+0.2342606i
## [2,] -0.5773503+0i  0.90166963+0.0000000i  0.90166963+0.0000000i
## [3,] -0.5773503+0i -0.18033393+0.3123475i -0.18033393-0.3123475i
```

```
# Give diagonal elements of X
X_diag = diag(X)

X_diag
```

```
## [1] 3 1 2
```

Suppose I want to merge X and Y in these two ways:

$$A = \begin{pmatrix} X \\ Y \end{pmatrix}, \quad B = \begin{pmatrix} X \; Y' \end{pmatrix}$$

We're going to use *rbind*() (which stands for **row bind**) and *cbind*() (**column bind**)

```
# A is a row bind
A = rbind(X, Y)
A
```

```
##      [,1] [,2] [,3]
## [1,]    3    1    2
## [2,]    0    1    5
## [3,]    4    0    2
## [4,]    9    3    0
## [5,]    4    2    3
```

```
# B is a column bind
B = cbind(X, t(Y))
B
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    3    1    2    9    4
## [2,]    0    1    5    3    2
## [3,]    4    0    2    0    3
```

Data Frames and Matrices How to turn data table into matrix, and vice versa

Turn the dataset *cars*, into a matrix

```r
# Get data
data = cars

# Turn into matrix
cars_matrix = as.matrix(cars)


# We can find the dimensions using dim()
dim(cars_matrix)
```

```
## [1] 50  2
```

Turn that matrix A from earlier into a dataset

```r
A_data = as.data.frame(A)

A_data
```

```
##   V1 V2 V3
## 1  3  1  2
## 2  0  1  5
## 3  4  0  2
## 4  9  3  0
## 5  4  2  3
```

This data is pretty boring though. Let's make it look better with…

Data Cleaning

We are going to learn how to clean and wrangle data with a package you're already familiar with: *dplyr*

```r
# Load dplyr
library(pacman)
p_load(dplyr)
```

Suppose you are sent this dataset about schools in Oregon's three most populous cities: Portland, Eugene, and Salem. Download it and take a look:

- Get the url by going to the Lab GitHub page => README => Week 2 => Muddy data to clean

```r
# get data
p_load(readr)

urlfile = "https://raw.githubusercontent.com/cmulholland217/Metrics_Lab_Spring2025/refs/heads/main/dirty

dirty_data = read_csv(url(urlfile))
```

```
## Rows: 180 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): City
## dbl (5): Number, gender, NEIGHBORHOOD, Teachers, GPA
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```r
summary(dirty_data)
```

```
##      City               Number          gender        NEIGHBORHOOD
##  Length:180         Min.   :115.0   Min.   :0.4500   Min.   : 592
##  Class :character   1st Qu.:326.0   1st Qu.:0.4900   1st Qu.:1649
##  Mode  :character   Median :397.0   Median :0.5000   Median :1988
##                     Mean   :406.1   Mean   :0.4998   Mean   :2055
##                     3rd Qu.:520.0   3rd Qu.:0.5100   3rd Qu.:2594
##                     Max.   :651.0   Max.   :0.5500   Max.   :3269
##                     NA’s   :19      NA’s   :10       NA’s   :10
##     Teachers          GPA
##  Min.   : 2.00   Min.   :-2.993
##  1st Qu.:17.00   1st Qu.: 2.131
##  Median :23.00   Median : 2.401
##  Mean   :22.55   Mean   : 2.355
##  3rd Qu.:27.00   3rd Qu.: 2.713
##  Max.   :44.00   Max.   : 3.563
##  NA’s   :15
```

The variables are:

- **City**: What city the school is in
- **Number**: Number of students
- **GPA**: Average GPA at the school
- **gender**: What portion of the school is male
- **NEIGHBORHOOD**: Rough population estimate of the neighborhood
- **Teachers**: The number of teachers

Your boss wants to know the effect of the number of students per teacher on GPA (controlling for confounding variables) and wants to run this regression and WILL NOT change their code:

```r
reg = lm(data = school_data,
         formula = gpa ~ student_teacher_ratio + neighborhood_pop + percent_male + factor(city))
```

You are told the variables mean:

- **school_data**: the dataset of schools with more than 100 students
- **gpa**: Average GPA at the school (same as **GPA**)
- **student_teacher_ratio**: Number of students / Number of teachers (how many students per teacher)
- **neighborhood_pop**: Population for the neighborhood the school is in

- **percent_male**: the *percent* of the school that is male
- **factor(city)**: recall this creates the fixed effects variables for each city

Let's use functions in the dplyr package to clean the data to our specification:

```
# First, let's rename our variables:


new_names <- c("city", "students", "percent_male", "neighborhood_pop", "teachers", "gpa")

school_data <- dirty_data %>%
  # Rename columns using `setNames()`
  setNames(new_names) %>%

  # Drop rows with missing values
  na.omit() %>%

  # Filter for schools with more than 100 students
  filter(students > 100) %>%

  # Convert percent_male to actual percentage
  mutate(percent_male = 100 * percent_male) %>%

  # Create a student-teacher ratio
  mutate(student_teacher_ratio = students / teachers)
```

Now run your boss' code!

```
reg = lm(data = school_data,
         formula = gpa ~ student_teacher_ratio + neighborhood_pop + percent_male + factor(city))

summary(reg)
```

```
##
## Call:
## lm(formula = gpa ~ student_teacher_ratio + neighborhood_pop +
##     percent_male + factor(city), data = school_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7027 -0.1635 -0.0236  0.1776  0.6207
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            4.872e+00  6.151e-01   7.921  1.1e-12 ***
## student_teacher_ratio -9.955e-02  4.057e-03 -24.538  < 2e-16 ***
## neighborhood_pop       3.216e-05  3.675e-05   0.875 0.383289
## percent_male          -1.605e-02  1.197e-02  -1.340 0.182557
## factor(city)Portland   2.082e-01  5.656e-02   3.680 0.000345 ***
## factor(city)Salem      1.098e-01  6.570e-02   1.672 0.097097 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.2566 on 125 degrees of freedom
## Multiple R-squared:  0.8495, Adjusted R-squared:  0.8435
## F-statistic: 141.1 on 5 and 125 DF,  p-value: < 2.2e-16
```

This means for every student added per teacher, the average GPA for the school decreases by **0.09**

**Bonus Stuff Lets quickly go through an example of using lapply since it will be useful for upcoming homeworks. What if you wanted to run this regression for different bins of school size?

```
# Define the group breaks and labels
school_data$size_group <- cut(
  school_data$students,
  breaks = c(100, 200, 300, 400, 500, 600, Inf),
  labels = c("100-200", "201-300", "301-400", "401-500", "501-600", "600+"),
  right = TRUE
)

# Split the data by group
grouped_data <- split(school_data, school_data$size_group)

# Run the regression within each group checking if the group has more than one city (for city fixed eff
reg_list <- lapply(grouped_data, function(df) {
  if (length(unique(df$city)) > 1) {
    lm(gpa ~ student_teacher_ratio + neighborhood_pop + percent_male + factor(city), data = df)
  } else {
    lm(gpa ~ student_teacher_ratio + neighborhood_pop + percent_male, data = df)
  }
})


# View summaries (or just one)
lapply(reg_list, summary)
```

```
## $‘100-200‘
##
## Call:
## lm(formula = gpa ~ student_teacher_ratio + neighborhood_pop +
##     percent_male + factor(city), data = df)
##
## Residuals:
##          1          2          3          4          5          6          7
##  6.350e-02 -3.449e-03 -1.325e-01  4.272e-02 -1.379e-01  1.676e-01  1.908e-17
##          8          9         10
##  1.477e-01 -5.739e-03 -1.419e-01
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            0.0540828  3.6707668   0.015    0.989
## student_teacher_ratio -0.0952693  0.0044434 -21.441  2.8e-05 ***
## neighborhood_pop       0.0010486  0.0008071   1.299    0.264
## percent_male           0.0612193  0.0634176   0.965    0.389
## factor(city)Portland   0.2573052  0.1794424   1.434    0.225
## factor(city)Salem      0.3671969  0.2959818   1.241    0.283
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.1677 on 4 degrees of freedom
## Multiple R-squared:  0.9954, Adjusted R-squared:  0.9897
## F-statistic: 174.4 on 5 and 4 DF,  p-value: 9.077e-05
##
##
## $‘201-300‘
##
## Call:
## lm(formula = gpa ~ student_teacher_ratio + neighborhood_pop +
##     percent_male + factor(city), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.47837 -0.13573 -0.01335  0.09217  0.42764
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            4.3691359  2.7510388   1.588   0.1633
## student_teacher_ratio -0.0701689  0.0272921  -2.571   0.0423 *
## neighborhood_pop      -0.0006092  0.0009935  -0.613   0.5623
## percent_male          -0.0009116  0.0517158  -0.018   0.9865
## factor(city)Portland  -0.0041720  0.2320236  -0.018   0.9862
## factor(city)Salem      0.3294066  0.2973124   1.108   0.3103
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.323 on 6 degrees of freedom
## Multiple R-squared:  0.8119, Adjusted R-squared:  0.6552
## F-statistic: 5.181 on 5 and 6 DF,  p-value: 0.03477
##
##
## $‘301-400‘
##
## Call:
## lm(formula = gpa ~ student_teacher_ratio + neighborhood_pop +
##     percent_male + factor(city), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.39732 -0.18196 -0.03992  0.10564  0.63774
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            4.0852185  1.3321784   3.067  0.00423 **
## student_teacher_ratio -0.1141041  0.0134987  -8.453 7.16e-10 ***
## neighborhood_pop       0.0004450  0.0003957   1.125  0.26864
## percent_male          -0.0090204  0.0249428  -0.362  0.71986
## factor(city)Portland   0.1223225  0.1040653   1.175  0.24798
## factor(city)Salem      0.0664969  0.1428058   0.466  0.64444
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

```
##
## Residual standard error: 0.2832 on 34 degrees of freedom
## Multiple R-squared:  0.7165, Adjusted R-squared:  0.6748
## F-statistic: 17.19 on 5 and 34 DF,  p-value: 1.813e-08
##
##
## $‘401-500‘
##
## Call:
## lm(formula = gpa ~ student_teacher_ratio + neighborhood_pop +
##     percent_male + factor(city), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.54615 -0.14327 -0.01183  0.18222  0.47437
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            5.573e+00  1.666e+00   3.345   0.0027 **
## student_teacher_ratio -9.467e-02  2.831e-02  -3.344   0.0027 **
## neighborhood_pop      -9.021e-05  3.065e-04  -0.294   0.7710
## percent_male          -2.597e-02  2.806e-02  -0.926   0.3639
## factor(city)Portland   2.798e-01  1.299e-01   2.154   0.0415 *
## factor(city)Salem      7.397e-02  1.324e-01   0.559   0.5815
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.2625 on 24 degrees of freedom
## Multiple R-squared:  0.377,  Adjusted R-squared:  0.2472
## F-statistic: 2.904 on 5 and 24 DF,  p-value: 0.03449
##
##
## $‘501-600‘
##
## Call:
## lm(formula = gpa ~ student_teacher_ratio + neighborhood_pop +
##     percent_male + factor(city), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5141 -0.1456 -0.0064  0.1469  0.3975
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            4.6393230  1.3641610   3.401  0.00198 **
## student_teacher_ratio -0.1033180  0.0197537  -5.230 1.34e-05 ***
## neighborhood_pop       0.0002279  0.0003340   0.682  0.50045
## percent_male          -0.0225916  0.0221805  -1.019  0.31685
## factor(city)Portland   0.2716835  0.1462689   1.857  0.07343 .
## factor(city)Salem      0.1891720  0.1343749   1.408  0.16982
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.2451 on 29 degrees of freedom
## Multiple R-squared:  0.6934, Adjusted R-squared:  0.6406
## F-statistic: 13.12 on 5 and 29 DF,  p-value: 1.015e-06
##
##
## $`600+`
##
## Call:
## lm(formula = gpa ~ student_teacher_ratio + neighborhood_pop +
##     percent_male, data = df)
##
## Residuals:
## ALL 4 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            2.255445        NaN     NaN      NaN
## student_teacher_ratio -0.131321        NaN     NaN      NaN
## neighborhood_pop       0.002506        NaN     NaN      NaN
## percent_male          -0.100003        NaN     NaN      NaN
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 3 and 0 DF,  p-value: NA
```

What If I just want the coeffecients or R squared?

```
lapply(reg_list, function(model) coef(model))
```

```
## $`100-200`
##           (Intercept) student_teacher_ratio      neighborhood_pop
##           0.054082771          -0.095269336           0.001048598
##          percent_male  factor(city)Portland     factor(city)Salem
##           0.061219330           0.257305208           0.367196948
##
## $`201-300`
##           (Intercept) student_teacher_ratio      neighborhood_pop
##          4.3691358808         -0.0701688771         -0.0006092457
##          percent_male  factor(city)Portland     factor(city)Salem
##         -0.0009116321         -0.0041720254          0.3294065684
##
## $`301-400`
##           (Intercept) student_teacher_ratio      neighborhood_pop
##          4.0852185339         -0.1141040607          0.0004449501
##          percent_male  factor(city)Portland     factor(city)Salem
##         -0.0090203557          0.1223225159          0.0664968601
##
## $`401-500`
##           (Intercept) student_teacher_ratio      neighborhood_pop
##          5.573105e+00         -9.466614e-02         -9.020949e-05
##          percent_male  factor(city)Portland     factor(city)Salem
##         -2.596672e-02          2.798479e-01          7.397484e-02
##
## $`501-600`
##           (Intercept) student_teacher_ratio      neighborhood_pop
##          4.6393229599         -0.1033180385          0.0002279025
##          percent_male  factor(city)Portland     factor(city)Salem
##         -0.0225916021          0.2716835405          0.1891719718
##
## $‘600+‘
##           (Intercept) student_teacher_ratio      neighborhood_pop
##            2.25544515           -0.13132064            0.00250644
##          percent_male
##           -0.10000285
```

```
lapply(reg_list, function(model) summary(model)$r.squared)
```

```
## $‘100-200‘
## [1] 0.9954346
##
## $‘201-300‘
## [1] 0.8119378
##
## $‘301-400‘
## [1] 0.716506
##
## $‘401-500‘
## [1] 0.3769789
##
## $‘501-600‘
## [1] 0.6934445
##
## $‘600+‘
## [1] 1
```