Introduction to R for Finance

Eric Mazzola

6/9/2017

Disclaimer: The content of this RMarkdown note came from a course called Introduction to R for Finance in datacamp.

Introduction to R for Finance

Chapter 1: The Basics

1.1 Your first R script

Basic subtraction and addition.

```
# Addition!
3 + 5
## [1] 8

# Subtraction!
5 - 3
## [1] 2
```

1.2 Arithmetic in R (1)

Basic of all math.

```
# Addition
2 + 2
## [1] 4

# Subtraction
4 - 1
## [1] 3

# Multiplication
3 * 4
## [1] 12

# Division
4 / 2
## [1] 2

# Exponentiation
2^4
## [1] 16

# Modulo
7%%3
## [1] 1
```

1.3 Assignment and variables (1)

Assigning variable to a word.

```
# Assign 200 to savings
savings <- 200

# Print the value of savings to the console
savings
## [1] 200
```

1.4 Assignment and variables (2)

Assigning a variable then applying math to it.

```
# Assign 100 to my_money
my_money <- 100

# Assign 200 to dans_money
dans_money <- 200

# Add my_money and dans_money
my_money + dans_money
## [1] 300

# Add my_money and dans_money again, save the result to our_money
our_money <- my_money + dans_money
```

1.5 Financial returns (1)

Calculating returns.

```
# Variables for starting_cash and 5% return during January
starting_cash <- 200
jan_ret <- 5
jan_mult <- 1 + (jan_ret / 100)

# How much money do you have at the end of January?
post_jan_cash <- 210
starting_cash + (starting_cash*1.05)
## [1] 410
# Print post_jan_cash
post_jan_cash
## [1] 210

# January 10% return multiplier
jan_ret_10 <- 10
jan_mult_10 <- 1 + (jan_ret_10/100)

# How much money do you have at the end of January now?
post_jan_cash_10 <- starting_cash*jan_mult_10


print(post_jan_cash_10)
## [1] 220
```

1.6 Financial returns (2)

More detailed returns.

```
# Starting cash and returns
starting_cash <- 200
jan_ret <- 4
feb_ret <- 5

# Multipliers
jan_mult <- 1.04
feb_mult <- 1.05

# Total cash at the end of the two months
total_cash <- (starting_cash*jan_mult*feb_mult)

# Print total_cash
total_cash
## [1] 218.4
```

1.7 Data type exploration

how to identify variables.

```
# Apple's stock price is a numeric
apple_stock <- 150.45

# Bond credit ratings are characters
credit_rating <- "AAA"

# You like the stock market. TRUE or FALSE?
my_answer <- TRUE

# Print my_answer
my_answer
## [1] TRUE
```

Chapter 2: Vectors and Matrices

2.1 c()ombine

How to use the combine factor.

```
# Another numeric vector
ibm_stock <- c(159.82, 160.02, 159.84)

# Another character vector
finance <- c("stocks","bonds","investments")


# A logical vector
logic <- c(TRUE, FALSE, TRUE)
```

2.2 Vector names()

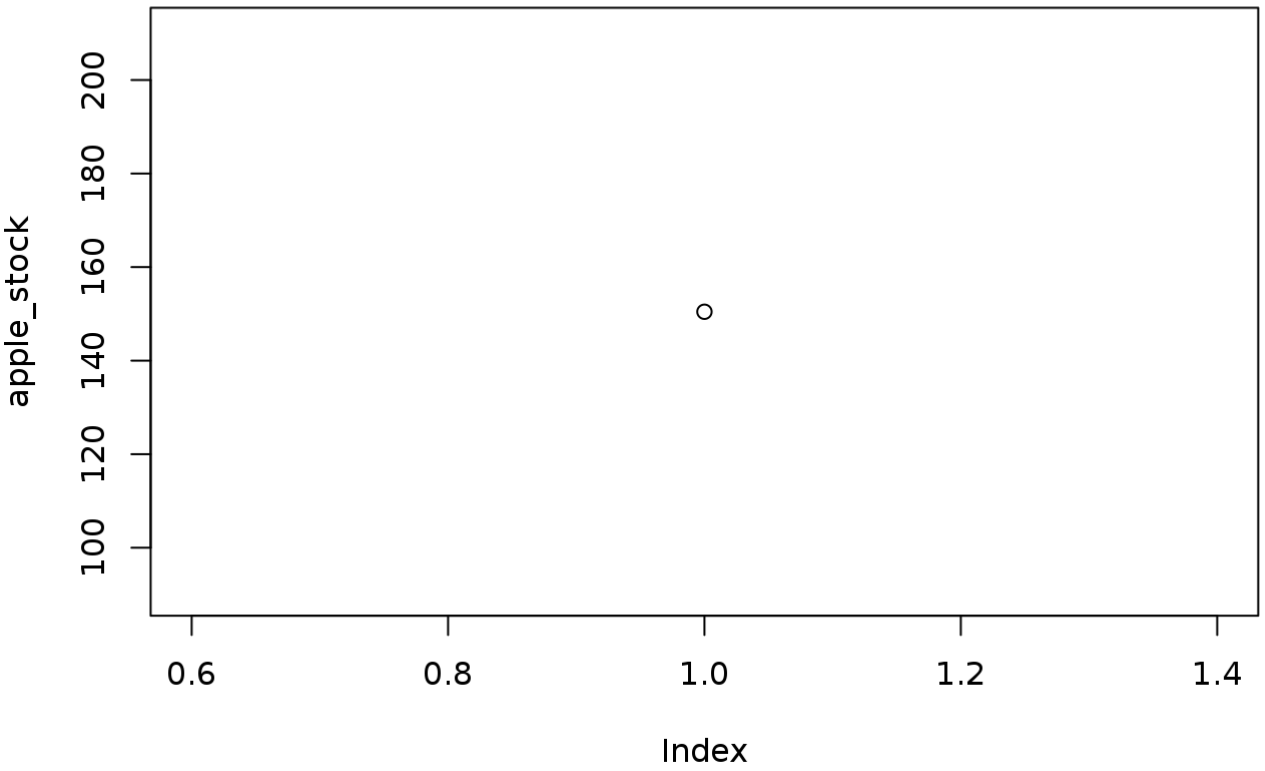
How to assign vector names.

```
# Vectors of 12 months of returns, and month names
ret <- c(5, 2, 3, 7, 8, 3, 5, 9, 1, 4, 6, 3)
months <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")

# Add names to ret
names(ret) <- months

# Print out ret to see the new names!
ret
## Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
##   5   2   3   7   8   3   5   9   1   4   6   3
```

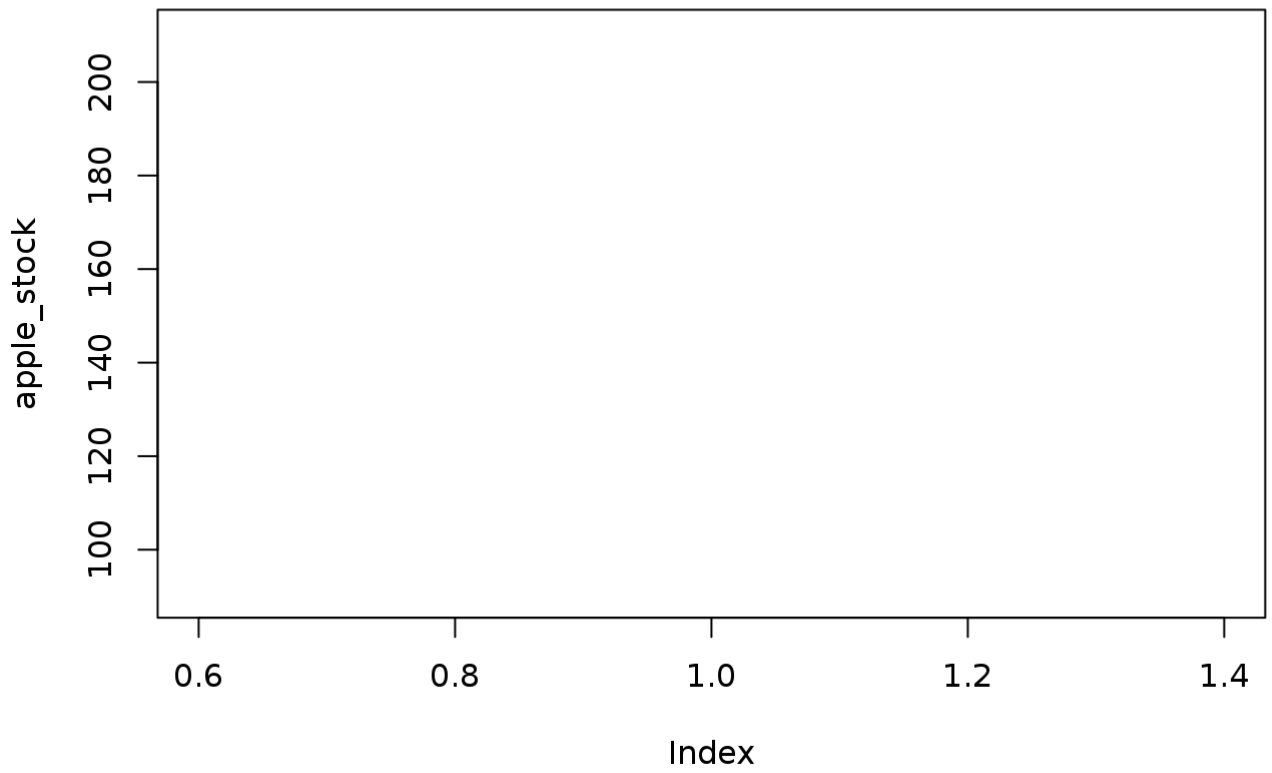
2.3 Visualize your vector

How to plot vectors to visualize them.

```r
# Look at the data
apple_stock
## [1] 150.45

# Plot the data points
plot(apple_stock)
```




```r

# Plot the data as a line graph
plot(apple_stock, type = "l")
```



2.4 Weighted average (1)

weighted variable attached to a vector.

```
# Weights and returns
micr_ret <- 7
sony_ret <- 9
micr_weight <- .2
sony_weight <- .8

# Portfolio return
portf_ret <-micr_ret*micr_weight+sony_ret*sony_weight
```

2.5 Weighted average (2)

More detailed weighted average.

```
# Weights, returns, and company names
ret <- c(7, 9)
weight <- c(.2, .8)
companies <- c("Microsoft", "Sony")

# Assign company names to your vectors
names(ret) <- companies
names(weight) <- companies

# Multiply the returns and weights together
ret_X_weight <- ret*weight

# Print ret_X_weight
ret_X_weight
## Microsoft      Sony
##       1.4       7.2

# Sum to get the total portfolio return
portf_ret <- sum(ret_X_weight)

# Print portf_ret
portf_ret
## [1] 8.6
```

2.6 Weighted average (3)

Much more detailed weighted average.

```
# Print ret
ret
## Microsoft      Sony
##         7         9

# Assign 1/3 to weight
weight <- 1/3

# Create ret_X_weight
ret_X_weight <- ret*weight

# Calculate your portfolio return
portf_ret <- sum(ret_X_weight)

# Vector of length 3 * Vector of length 2?
ret * c(.2, .6)
## Microsoft      Sony
##       1.4       5.4
```

2.7 Vector subsetting

Splitting data and assigning vector names.

```
# First 6 months of returns
ret[1:6]
## Microsoft      Sony      <NA>      <NA>      <NA>      <NA>
##         7         9        NA        NA        NA        NA

List <- c("Mar", "May")
# Just March and May
ret[List]
## <NA> <NA>
##   NA   NA

# Omit the first month of returns
ret[-1]
## Sony
##    9
class("May")
## [1] "character"
```

2.8 Create a matrix!

How to create a matrix.

```
# A vector of 9 numbers
my_vector <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

# 3x3 matrix
my_matrix <- matrix(data = my_vector, nrow = 3, ncol = 3)

# Print my_matrix
my_matrix
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9

# Filling across using byrow = TRUE
matrix(data = c(2, 3, 4, 5), nrow = 2, ncol = 2, byrow = TRUE)
##      [,1] [,2]
## [1,]    2    3
## [2,]    4    5
```

2.9 Matrix <- bind vectors

How to bind vectors within a matrix.

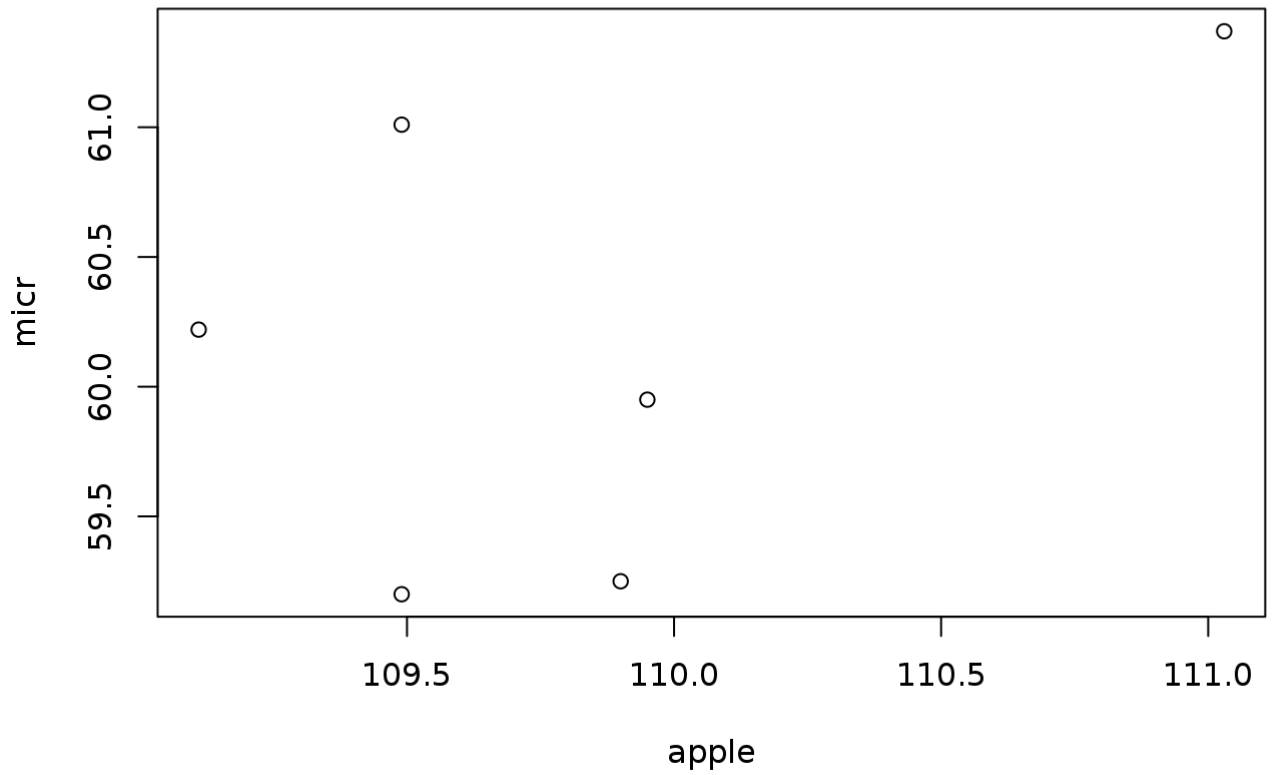
```
apple <- c(109.49, 109.90, 109.11, 109.95, 111.03)
ibm <- c(159.82, 160.02, 159.84, 160.35, 164.79)
micr <- c(59.20, 59.25, 60.22, 59.95, 61.37, 61.01)
# cbind the vectors together
cbind_stocks <- cbind(apple, ibm, micr)

# Print cbind_stocks
cbind_stocks
##       apple    ibm  micr
## [1,] 109.49 159.82 59.20
## [2,] 109.90 160.02 59.25
## [3,] 109.11 159.84 60.22
## [4,] 109.95 160.35 59.95
## [5,] 111.03 164.79 61.37
## [6,] 109.49 159.82 61.01

# rbind the vectors together
rbind_stocks <- rbind(apple, ibm, micr)

# Print rbind_stocks
rbind_stocks
##         [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
## apple 109.49 109.90 109.11 109.95 111.03 109.49
## ibm   159.82 160.02 159.84 160.35 164.79 159.82
## micr   59.20  59.25  60.22  59.95  61.37  61.01
```

2.10 Visualize your matrix

Plotting the matrix data.

```
# Define matrix
apple_micr_matrix <- cbind(apple, micr)
apple_micr_matrix
##       apple  micr
## [1,] 109.49 59.20
## [2,] 109.90 59.25
## [3,] 109.11 60.22
## [4,] 109.95 59.95
## [5,] 111.03 61.37
## [6,] 109.49 61.01
##      apple  micr
##[1,] 109.49 59.20
##[2,] 109.90 59.25
##[3,] 109.11 60.22
##[4,] 109.95 59.95
##[5,] 111.03 61.37
##[6,] 112.12 61.01
##[7,] 113.95 61.97
##[8,] 113.30 62.17
##[9,] 115.19 62.98
##[10,] 115.19 62.68
##[12,] 115.97 62.30
##[13,] 116.64 63.62
##[14,] 116.95 63.54
##[15,] 117.06 63.54
##[16,] 116.29 63.55
##[17,] 116.52 63.24
##[18,] 117.26 63.28
##[19,] 116.76 62.99
##[20,] 116.73 62.90
##[21,] 115.82 62.14
# View the data
apple_micr_matrix
##       apple  micr
## [1,] 109.49 59.20
## [2,] 109.90 59.25
## [3,] 109.11 60.22
## [4,] 109.95 59.95
```

```
## [5,] 111.03 61.37
## [6,] 109.49 61.01


# Scatter plot of Microsoft vs Apple
plot(apple_micr_matrix)
```



2.11 cor()relation

Visualizing corelations of data.

```
# Correlation of Apple and IBM
cor(apple, ibm)
## [1] 0.9131575

# stock matrix
stocks <- cbind(apple, micr, ibm)

# cor() of all three
cor(stocks)
##           apple      micr       ibm
## apple 1.0000000 0.4575786 0.9181842
## micr  0.4575786 1.0000000 0.6436983
## ibm   0.9181842 0.6436983 1.0000000
```

2.12 Matrix subsetting

Displaying certain rows or colunms of data.

```
# Third row
stocks[3,]
##  apple   micr    ibm
## 109.11  60.22 159.84

# Fourth and fifth row of the ibm column
stocks[4:5, "ibm"]
## [1] 160.35 164.79

# apple and micr columns
stocks [,c("apple","micr")]
##       apple  micr
## [1,] 109.49 59.20
## [2,] 109.90 59.25
## [3,] 109.11 60.22
## [4,] 109.95 59.95
## [5,] 111.03 61.37
## [6,] 109.49 61.01

stocks[]
##       apple  micr    ibm
## [1,] 109.49 59.20 159.82
## [2,] 109.90 59.25 160.02
## [3,] 109.11 60.22 159.84
## [4,] 109.95 59.95 160.35
## [5,] 111.03 61.37 164.79
## [6,] 109.49 61.01 159.82
```

Chapter 3: Data Frames

3.1 Create your first data.frame()

Simply how to create a data frame.

```
# Variables
company <- c("A", "A", "A", "B", "B", "B", "B")
cash_flow <- c(1000, 4000, 550, 1500, 1100, 750, 6000)
year <- c(1, 3, 4, 1, 2, 4, 5)

# Data frame
cash <- data.frame(company, cash_flow, year)

# Print cash
cash
##   company cash_flow year
## 1       A      1000    1
## 2       A      4000    3
## 3       A       550    4
## 4       B      1500    1
## 5       B      1100    2
## 6       B       750    4
## 7       B      6000    5
```

3.2 Making head()s and tail()s of your data with some str()ucture

Sorting data.

```
# Call head() for the first 4 rows
head(cash, 4)
##   company cash_flow year
## 1       A      1000    1
## 2       A      4000    3
## 3       A       550    4
## 4       B      1500    1

# Call tail() for the last 3 rows
tail(cash, 3)
##   company cash_flow year
## 5       B      1100    2
## 6       B       750    4
## 7       B      6000    5

# Call str()
str(cash)
## 'data.frame':    7 obs. of  3 variables:
##  $ company  : Factor w/ 2 levels "A","B": 1 1 1 2 2 2 2
##  $ cash_flow: num  1000 4000 550 1500 1100 750 6000
##  $ year     : num  1 3 4 1 2 4 5
```

3.3 Naming your columns / rows

Simply how to names rows and columns.

```
# Fix your column names
newNames <- c("company","cash_flow","year")
#colnames(cash) <- newNames
colnames(cash) <- c("company","cash_flow","year")
# Print out the column names of cash

#cash
colnames(cash)
## [1] "company"   "cash_flow" "year"
```

3.4 Accessing and subsetting data frames (1)

Accessing certain data sets and certain X,Y data.

```
# Third row, second column
cash[3,2]
## [1] 550

# Fifth row of the "year" column
cash[5,"year"]
## [1] 2
```

3.5 Accessing and subsetting data frames (2)

More detailed.

```
# Select the year column
cash[,"year"]
## [1] 1 3 4 1 2 4 5

# Select the cash_flow column and multiply by 2
cash[,"cash_flow"]*2
## [1]  2000  8000  1100  3000  2200  1500 12000

# Delete the company column
cash$company <- NULL

# Print cash again
cash
##   cash_flow year
## 1      1000    1
## 2      4000    3
## 3       550    4
## 4      1500    1
## 5      1100    2
## 6       750    4
## 7      6000    5
```

3.6 Accessing and subsetting data frames (3)

More detailed.

```
# Rows about company B
subset(cash,company == "B")
##   cash_flow year
## 4      1500    1
## 5      1100    2
## 6       750    4
## 7      6000    5

# Rows with cash flows due in 1 year
subset(cash,year == "1")
##   cash_flow year
## 1      1000    1
## 4      1500    1
```

3.7 Adding new columns

Adding new columns.

```
# Quarter cash flow scenario
cash$quarter_cash <- cash$cash_flow * .25

cash
##   cash_flow year quarter_cash
## 1      1000    1        250.0
## 2      4000    3       1000.0
## 3       550    4        137.5
## 4      1500    1        375.0
## 5      1100    2        275.0
## 6       750    4        187.5
## 7      6000    5       1500.0

# Double year scenario

cash$double_year <- cash$year * 2
cash$double_year <- 2
cash
##   cash_flow year quarter_cash double_year
## 1      1000    1        250.0           2
## 2      4000    3       1000.0           2
## 3       550    4        137.5           2
## 4      1500    1        375.0           2
## 5      1100    2        275.0           2
## 6       750    4        187.5           2
## 7      6000    5       1500.0           2
```

3.8 Present value of projected cash flows (1)

Plotting projected cash flow.

```
# Present value of $4000, in 3 years, at 5%
interest <- 5
year <- 3
#cash[, "cash_flow"]
cash_flow <- 4000

present_value_4k <- cash_flow  * (1 + interest / 100) ^ -year
present_value_4k
## [1] 3455.35

cash$present_value <- cash$cash_flow  * (1 + interest / 100) ^ -cash$year
# Present value of all cash flows
#cash$present_value <- present_value_4k

# Print out cash
cash
##   cash_flow year quarter_cash double_year present_value
## 1      1000    1        250.0           2      952.3810
## 2      4000    3       1000.0           2     3455.3504
## 3       550    4        137.5           2      452.4864
## 4      1500    1        375.0           2     1428.5714
## 5      1100    2        275.0           2      997.7324
## 6       750    4        187.5           2      617.0269
## 7      6000    5       1500.0           2     4701.1570
```

3.9 Present value of projected cash flows (2)

More detailed.

```
# Total present value of cash
total_pv <- sum(cash$present_value)

# Company B information
cash_B <- subset(cash, company == "B")

# Total present value of cash_B
total_pv_B <- sum(cash_B$present_value)
```

Chapter 4: Factors

4.1 Create a factor

How to create a factor.

```
# credit_rating character vector
credit_rating <- c("BB", "AAA", "AA", "CCC", "AA", "AAA", "B", "BB")

# Create a factor from credit_rating
credit_factor <- factor(credit_rating)

# Print out your new factor
credit_factor
## [1] BB  AAA AA  CCC AA  AAA B   BB
## Levels: AA AAA B BB CCC

# Call str() on credit_rating
str(credit_rating)
##  chr [1:8] "BB" "AAA" "AA" "CCC" "AA" "AAA" "B" "BB"

# Call str() on credit_factor
str(credit_factor)
##  Factor w/ 5 levels "AA","AAA","B",..: 4 2 1 5 1 2 3 4
```

4.2 Factor levels

Leanring about how to rename, add, drop and print factor levels.

```
# Identify unique levels
levels(credit_factor)
## [1] "AA"  "AAA" "B"   "BB"  "CCC"

# Rename the levels of credit_factor
levels(credit_factor) <- c("2A", "3A", "1B", "2B", "3C")

# Print credit_factor
credit_factor
## [1] 2B 3A 2A 3C 2A 3A 1B 2B
## Levels: 2A 3A 1B 2B 3C
```

4.3 Factor summary

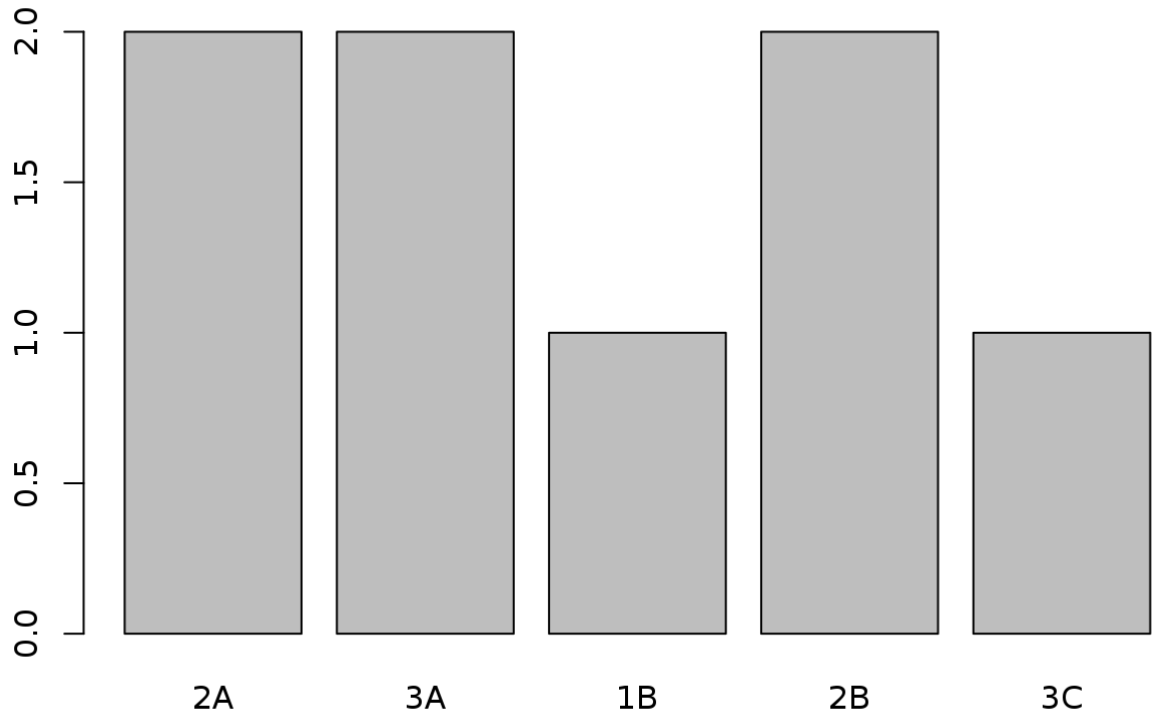
Summarizing factors.

```
# Summarize the character vector, credit_rating
summary(credit_rating)
##    Length     Class      Mode
##         8 character character

# Summarize the factor, credit_factor
summary(credit_factor)
## 2A 3A 1B 2B 3C
##  2  2  1  2  1
```

4.4 Visualize your factor

Plotting factors.

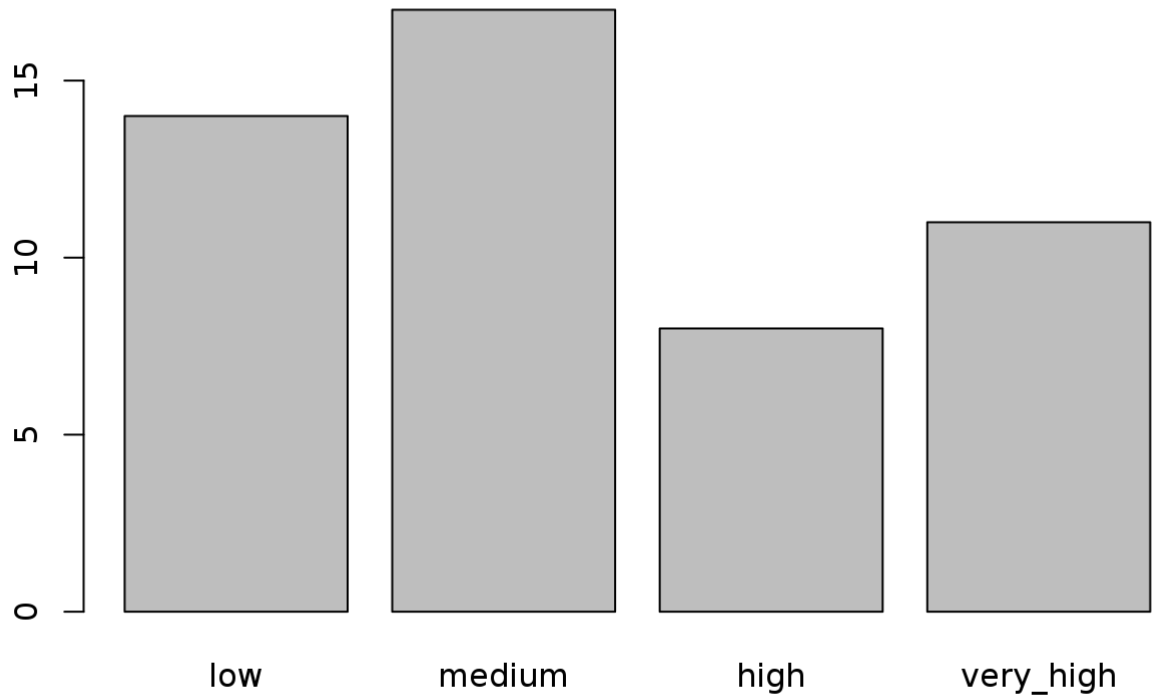
```
# Visualize your factor!
plot(credit_factor)
```



4.5 Bucketing a numeric variable into a factor

Sorting variables into buckets.

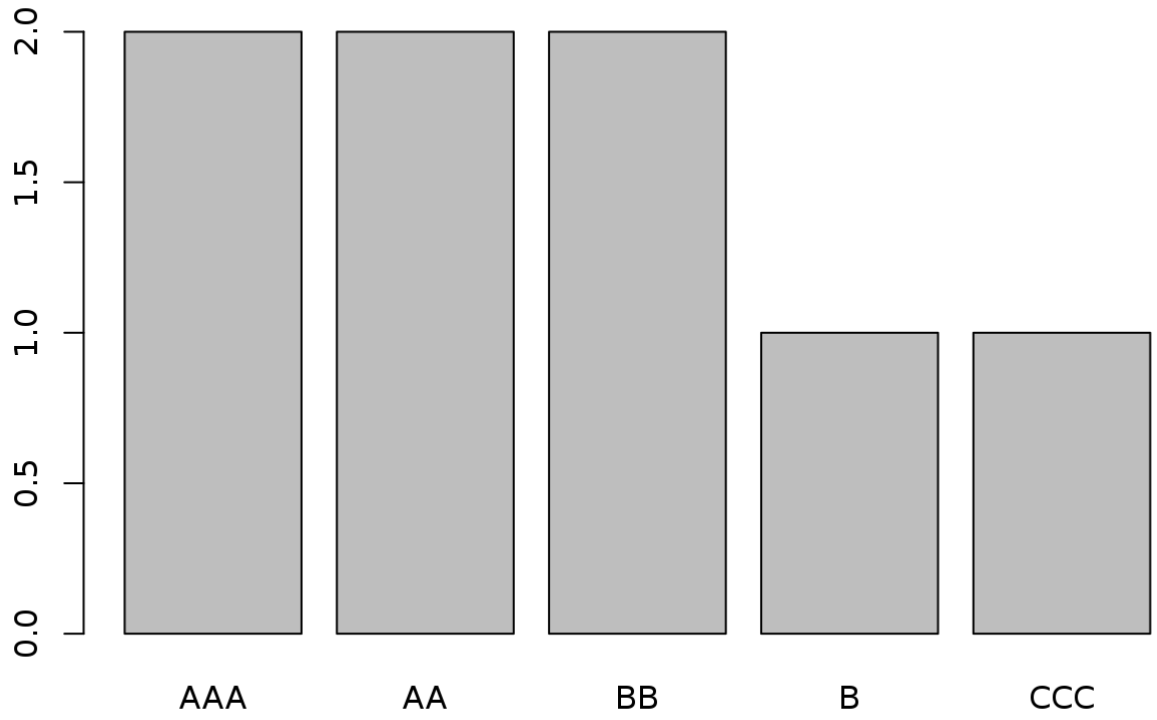
```r
# Define AAA_rank.
AAA_rank <- c(31,  48, 100, 53, 85, 73, 62, 74, 42, 38, 97, 61, 48, 86, 44, 9, 43, 18,  62,
              38, 23, 37, 54, 80, 78, 93, 47, 100, 22,  22, 18, 26, 81, 17, 98, 4,  83, 5,
              6,  52, 29, 44, 50, 2,  25, 19, 15, 42, 30, 27)
# Create 4 buckets for AAA_rank using cut()
AAA_factor <- cut(x = AAA_rank, breaks = c(0, 25, 50, 75, 100))

# Rename the levels
eric <- c("low", "medium", "high", "very_high")
levels(AAA_factor) <-eric
# Print AAA_factor
AAA_factor
##  [1] medium    medium    very_high high      very_high high      high
##  [8] high      medium    medium    very_high high      medium    very_high
## [15] medium    low       medium    low       high      medium    low
## [22] medium    high      very_high very_high very_high medium    very_high
## [29] low       low       low       medium    very_high low       very_high
## [36] low       very_high low       low       high      medium    medium
## [43] medium    low       low       low       low       medium    medium
## [50] medium
## Levels: low medium high very_high

# Plot AAA_factor
plot(AAA_factor)
```



4.6 Create an ordered factor

How to create an ordered factor.

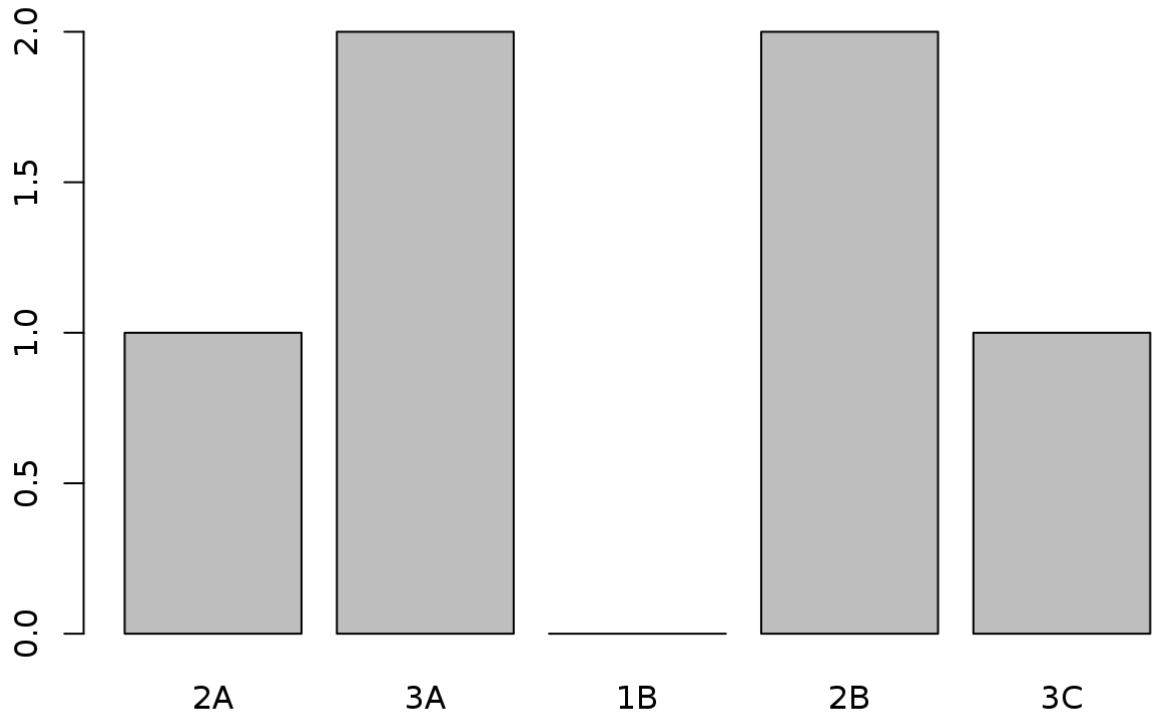
```
# Use unique() to find unique words
unique(credit_rating)
## [1] "BB"  "AAA" "AA"  "CCC" "B"

# Create an ordered factor
credit_factor_ordered <- factor(credit_rating, ordered = TRUE, levels = c("AAA", "AA", "BB", "B", "CCC"))

# Plot credit_factor_ordered
plot(credit_factor_ordered)
```



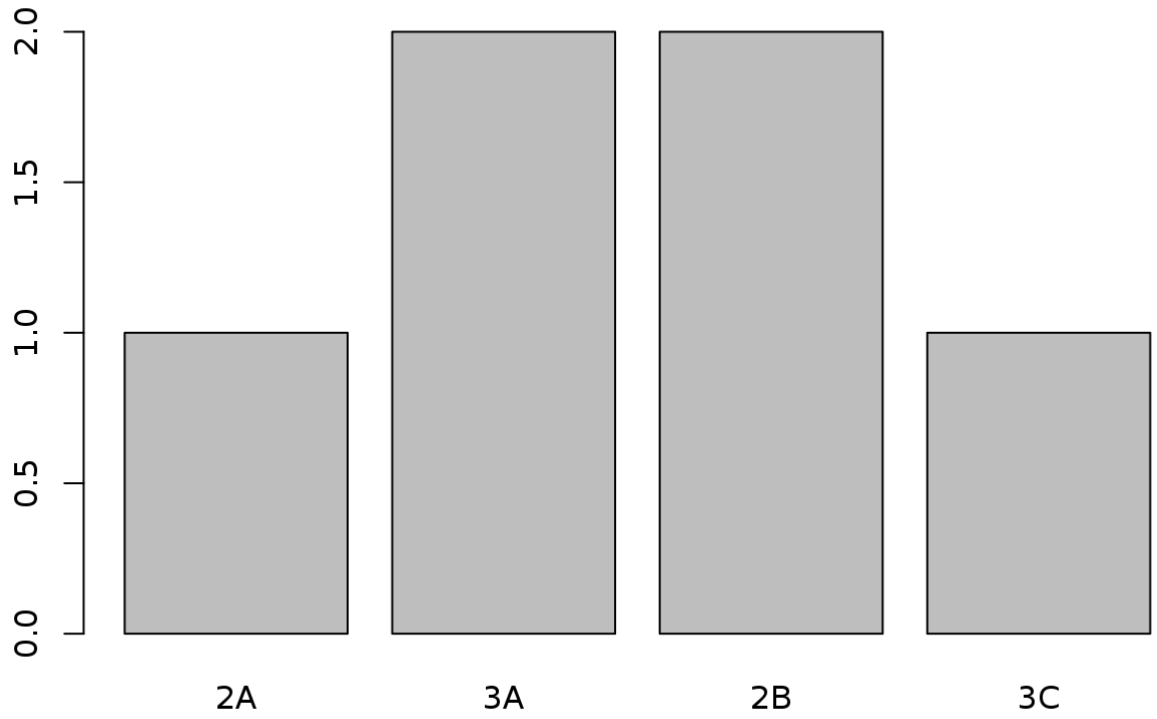
4.7 Subsetting a factor

Learning how to subset a factor.

```
# Remove the A bonds at positions 3 and 7. Don't drop the A level.
keep_level <- credit_factor[-3]
credit_factor
## [1] 2B 3A 2A 3C 2A 3A 1B 2B
## Levels: 2A 3A 1B 2B 3C
keep_level <- keep_level[-6]
keep_level
## [1] 2B 3A 3C 2A 3A 2B
## Levels: 2A 3A 1B 2B 3C
# Plot keep_level
plot(keep_level)
```




```
# Remove the A bonds at positions 3 and 7. Drop the A level.
drop_level <- credit_factor[c(-3,-7), drop = TRUE]
drop_level
## [1] 2B 3A 3C 2A 3A 2B
## Levels: 2A 3A 2B 3C
# Plot drop_level
plot(drop_level)
```



4.8 stringsAsFactors

A more in depth organization of data.

```
# Variables
credit_rating <- c("AAA", "A", "BB")
bond_owners <- c("Dan", "Tom", "Joe")

# Create the data frame of character vectors, bonds
bonds <- data.frame(credit_rating, bond_owners, stringsAsFactors = FALSE)

# Use str() on bonds
str(bonds)
## 'data.frame':    3 obs. of  2 variables:
##  $ credit_rating: chr  "AAA" "A" "BB"
##  $ bond_owners  : chr  "Dan" "Tom" "Joe"

# Create a factor column in bonds called credit_factor from credit_rating
bonds$credit_factor <- factor(bonds$credit_rating, ordered = TRUE, levels = c("AAA","A","BB"))

# Use str() on bonds again
str(bonds)
## 'data.frame':    3 obs. of  3 variables:
##  $ credit_rating: chr  "AAA" "A" "BB"
##  $ bond_owners  : chr  "Dan" "Tom" "Joe"
##  $ credit_factor: Ord.factor w/ 3 levels "AAA"<"A"<"BB": 1 2 3
```

Chapter 5: Lists

5.1 Create a list

How to create a list.

```
# List components
name <- "Apple and IBM"
apple <- c(109.49, 109.90, 109.11, 109.95, 111.03)
ibm <- c(159.82, 160.02, 159.84, 160.35, 164.79)
cor_matrix <- cor(cbind(apple, ibm))

# Create a list
portfolio <- list(name, apple, ibm, cor_matrix)

# View your first list
portfolio
## [[1]]
## [1] "Apple and IBM"
##
## [[2]]
## [1] 109.49 109.90 109.11 109.95 111.03
##
## [[3]]
## [1] 159.82 160.02 159.84 160.35 164.79
##
## [[4]]
##           apple       ibm
## apple 1.0000000 0.9131575
## ibm   0.9131575 1.0000000
```

5.2 Named lists

How to name lists.

```
# Add names to your portfolio
names(portfolio) <- c("portfolio_name", "apple", "ibm", "correlation")

# Print portfolio
portfolio
## $portfolio_name
## [1] "Apple and IBM"
##
## $apple
## [1] 109.49 109.90 109.11 109.95 111.03
##
## $ibm
## [1] 159.82 160.02 159.84 160.35 164.79
##
## $correlation
##           apple       ibm
## apple 1.0000000 0.9131575
## ibm   0.9131575 1.0000000
```

5.3 Access elements in a list

How to access elements in a list.

```
# Second and third elements of portfolio
portfolio[c(2,3)]
## $apple
## [1] 109.49 109.90 109.11 109.95 111.03
##
## $ibm
## [1] 159.82 160.02 159.84 160.35 164.79

# Use $ to get the correlation data
portfolio$correlation
##           apple       ibm
## apple 1.0000000 0.9131575
## ibm   0.9131575 1.0000000
```

5.4 Adding to a list

How to add a list.

```
# Add weight: 20% Apple, 80% IBM
portfolio$weight <- c(apple = .20, ibm = .80)

# Print portfolio
portfolio
## $portfolio_name
## [1] "Apple and IBM"
##
## $apple
## [1] 109.49 109.90 109.11 109.95 111.03
##
## $ibm
## [1] 159.82 160.02 159.84 160.35 164.79
##
## $correlation
##           apple       ibm
## apple 1.0000000 0.9131575
## ibm   0.9131575 1.0000000
##
## $weight
## apple   ibm
##   0.2   0.8

# Change the weight variable: 30% Apple, 70% IBM
portfolio$weight <- c(apple = .30, ibm = .70)

# Print portfolio to see the changes
portfolio
## $portfolio_name
## [1] "Apple and IBM"
##
## $apple
## [1] 109.49 109.90 109.11 109.95 111.03
##
## $ibm
## [1] 159.82 160.02 159.84 160.35 164.79
##
## $correlation
```

```
##           apple       ibm
## apple 1.0000000 0.9131575
## ibm   0.9131575 1.0000000
##
## $weight
## apple   ibm
##   0.3   0.7
```

5.5 Removing from a list

How to remove a list.

```
# Take a look at portfolio
portfolio
## $portfolio_name
## [1] "Apple and IBM"
##
## $apple
## [1] 109.49 109.90 109.11 109.95 111.03
##
## $ibm
## [1] 159.82 160.02 159.84 160.35 164.79
##
## $correlation
##           apple       ibm
## apple 1.0000000 0.9131575
## ibm   0.9131575 1.0000000
##
## $weight
## apple   ibm
##   0.3   0.7

# Remove the microsoft stock prices from your portfolio
portfolio$microsoft <- NULL
```

5.6 Split it

How to split data for display.

```
# Define grouping from year
grouping <- cash$year

# Split cash on your new grouping
split_cash <- split(cash, grouping)

# Look at your split_cash list
split_cash
## $`1`
##   cash_flow year quarter_cash double_year present_value
## 1      1000    1          250           2       952.381
## 4      1500    1          375           2      1428.571
##
## $`2`
##   cash_flow year quarter_cash double_year present_value
## 5      1100    2          275           2      997.7324
##
## $`3`
##   cash_flow year quarter_cash double_year present_value
## 2      4000    3         1000           2       3455.35
##
## $`4`
##   cash_flow year quarter_cash double_year present_value
## 3       550    4        137.5           2      452.4864
## 6       750    4        187.5           2      617.0269
##
## $`5`
##   cash_flow year quarter_cash double_year present_value
## 7      6000    5         1500           2      4701.157

# Unsplit split_cash to get the original data back.
original_cash <- unsplit(split_cash, grouping)

# Print original_cash
original_cash
##   cash_flow year quarter_cash double_year present_value
## 1      1000    1        250.0           2      952.3810
## 2      4000    3       1000.0           2     3455.3504
```

```
## 3       550    4        137.5           2      452.4864
## 4      1500    1        375.0           2     1428.5714
## 5      1100    2        275.0           2      997.7324
## 6       750    4        187.5           2      617.0269
## 7      6000    5       1500.0           2     4701.1570
```

5.7 Split-Apply-Combine

How to split then re combine data.

```
# Print split_cash
split_cash
## $`1`
##   cash_flow year quarter_cash double_year present_value
## 1      1000    1          250           2       952.381
## 4      1500    1          375           2      1428.571
##
## $`2`
##   cash_flow year quarter_cash double_year present_value
## 5      1100    2          275           2      997.7324
##
## $`3`
##   cash_flow year quarter_cash double_year present_value
## 2      4000    3         1000           2       3455.35
##
## $`4`
##   cash_flow year quarter_cash double_year present_value
## 3       550    4        137.5           2      452.4864
## 6       750    4        187.5           2      617.0269
##
## $`5`
##   cash_flow year quarter_cash double_year present_value
## 7      6000    5         1500           2      4701.157

# Print the cash_flow column of B in split_cash
split_cash$B$cash_flow
## NULL

# Set the cash_flow column of company A in split_cash to 0
split_cash$A$cash_flow <- 0

# Use the grouping to unsplit split_cash
cash_no_A <- unsplit(split_cash, grouping)

# Print cash_no_A
cash_no_A
##   cash_flow year quarter_cash double_year present_value
## 1      1000    1        250.0           2      952.3810
```

```
## 2      4000    3       1000.0          2     3455.3504
## 3       550    4        137.5          2      452.4864
## 4      1500    1        375.0          2     1428.5714
## 5      1100    2        275.0          2      997.7324
## 6       750    4        187.5          2      617.0269
## 7      6000    5       1500.0          2     4701.1570
```

5.8 Attributes

How to assign attributes to data.

```
# my_matrix and my_factor
my_matrix <- matrix(c(1,2,3,4,5,6), nrow = 2, ncol = 3)
rownames(my_matrix) <- c("Row1", "Row2")
colnames(my_matrix) <- c("Col1", "Col2", "Col3")

my_factor <- factor(c("A", "A", "B"), ordered = T, levels = c("A", "B"))

# attributes of my_matrix
attributes(my_matrix)
## $dim
## [1] 2 3
##
## $dimnames
## $dimnames[[1]]
## [1] "Row1" "Row2"
##
## $dimnames[[2]]
## [1] "Col1" "Col2" "Col3"

# Just the dim attribute of my_matrix
attr(my_matrix, "dim")
## [1] 2 3

# attributes of my_factor
attributes(my_factor)
## $levels
## [1] "A" "B"
##
## $class
## [1] "ordered" "factor"
```

Quiz 1

1. Compare vectors and matrices. What are similarities and differences? Hypothetically, vectors are like folders on a PC. In the sense that they have a title and contain information that is related to the title. As for matrices, they have the same principal as spreadsheets. In the sense that it has columns and within those columns are organized data. They are both similar in the sense they can both store data, but they are

different due to the type of data that is stored within them, or at least organized differently.

2. Compare matrices and data frames. What are similarities and differences? Matrices and data frames are the same because they both contain rows and columns and they are different because you can have multiple data types in a single data frame.

3. Create your first vector, matrix, data frame, factor, and list. Do this within a R code chunk.

Vector:

```
eric <- 21
```

Matrix:

```
my_family <- c(21, 21, 23, 25)
my_matrix <- matrix(data = my_family, nrow = 4, ncol = 1)
my_matrix
##      [,1]
## [1,]   21
## [2,]   21
## [3,]   23
## [4,]   25
```

Data Frame:

```
family_letter <- c("E", "M", "B", "K")
cash_flow <- c(10, 9, 22, 32)
year <- c(1, 3, 2, 4)

cash <- data.frame(family_letter, cash_flow, year)
cash
##   family_letter cash_flow year
## 1             E        10    1
## 2             M         9    3
## 3             B        22    2
## 4             K        32    4
```

Factor:

```
family_rating <- c("EE", "E", "M", "MM", "BBB", "B", "KK", "K")
family_factor <- factor(family_rating)
family_factor
## [1] EE  E   M   MM  BBB B   KK  K
## Levels: B BBB E EE K KK M MM
str(family_rating)
##  chr [1:8] "EE" "E" "M" "MM" "BBB" "B" "KK" "K"
str(family_factor)
##  Factor w/ 8 levels "B","BBB","E",..: 4 3 7 8 2 1 6 5
```

List:

```
name <- "Eric and Brian"
eric <- c(1, 2, 3, 4, 5)
brian <- c(6, 7, 8, 9, 10)
cor_matrix <- cor(cbind(eric,brian))

portfolio <- list(name, eric, brian, cor_matrix)

portfolio
## [[1]]
## [1] "Eric and Brian"
##
## [[2]]
## [1] 1 2 3 4 5
##
## [[3]]
## [1]  6  7  8  9 10
##
## [[4]]
##       eric brian
## eric     1     1
## brian    1     1
```