Final Study Guide

Stat 133, Spring 2018, Prof. Sanchez

1) What function do you use to load a package?

`library()`

2) What command do you run to see the manual documentation of the function `"read.table"`?

`help(read.table)` or `?read.table`

3) Match the provided descriptions with the following commands.

a. `load(ggplot2)`
b. `help(ggplot)`
c. `install.packages("ggplot2")`
d. `g <- ggplot()`
e. `library(ggplot2)`
f. `man(ggplot)`
g. `obj <- ggplot2()`

- Creates an object of class `ggplot`. **d)**
- Downloads the package `ggplot2` from CRAN. **c)**
- Loads the package `ggplot2` to the current session. **e)**
- Displays the manual documentation of `ggplot`. **b)**

4) Write down a code example of the recycling rule in R.

```
x <- 1:5
x + 1
```

```
## [1] 2 3 4 5 6
```

5) What is the data type of each of the following vectors:

- `x`: where `x <- c(TRUE, FALSE)` **logical**
- `y`: where `y <- c(x, 10)` **numeric**
- `z`: where `z <- c(y, 10, "a")` **character**

6) In R, what is the typical symbol to represent missing values?

Missing values are represented by `NA`, which should be in capital letters, surrounded by no quotes.

7) What are the different (data) types of missing values that R provides?

`NA` (logical), `NA_integer_` (integer), `NA_real_` (real/double), `NA_character_` (character), and `NA_complex_` (complex).

8) What are the main differences between R matrices and R data frames?

A data frame can contain different types of data (is NOT atomic), and it is internally stored as an R list. In contrast, a matrix is an array and is atomic (all values are of the same type).

9) Which of the following commands reloads R objects from an R binary file `z.RData`:

- `library(z.RData)`
- `download.file("z.RData")`
- `load("z.RData")` **this one**
- `knit("z.RData")`

10) Write down the command to check if a variable `x` is greater than 5? (HINT: This command should return a `TRUE/FALSE` value)

```
x > 5
```

11) Suppose `y <- c(1, 4, 9, 16, 25)`. Write down the R command to return a vector `z`, in which each element of `z` is the square root of each element of the vector `y`.

```
y <- c(1, 4, 9, 16, 25)
z <- sqrt(y)
```

12) Write down 2 different R commands to return the first five elements of a vector `x` (assume `x` has more than 5 elements).

```
x <- letters[1:10]
x[1:5]
head(x, 5)
```

13) Consider a data.frame object `df`. You can use vectors of indices `index1` and `index2` to subscript the data frame like this: `df[index1, index2]`. Which of the following are NOT valid options for indexing (i.e. subscripting) `df`:

- Numeric vectors
- Missing values **this one**
- Logical vectors
- Blank spaces

14) What is the output of this command:

```
c(1, 2, 3, 4, 5) * 2
```

```
## [1]  2  4  6  8 10
```

15) What is the output of this command:

```
1:3^2
```

```
## [1] 1 2 3 4 5 6 7 8 9
```

16) What is the output of this command:

```
(1:5)*2
```

```
## [1]  2  4  6  8 10
```

17) What is the output of this command:

```
var<-3
Var*2
```

```
## Error in eval(expr, envir, enclos): object 'Var' not found
```

18) What is the output of this command:

```
x<-2
2x<-2*x
```

```
## Error: <text>:2:2: unexpected symbol
## 1: x<-2
## 2: 2x
##      ^
```

19) What is the output of this command:

```
sqrt4 <- sqrt(4)
sqrt4
```

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

20) What is the output of this command:

```
a number <- 16
```

```
## Error: <text>:1:3: unexpected symbol
## 1: a number
##       ^
```

21) Why the following comparisons return `TRUE`:

```
1 == TRUE
```

```
## [1] TRUE
```

```
0 == FALSE
```

```
## [1] TRUE
```

Because `TRUE` is coerced to 1, and `FALSE` is coerced to zero.

22) How do you use the function `seq()` to create the following vector?

```
 [1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
```

```
seq(from = 1, to = 2, by = 0.1)
```

23) Give an example using the function `rep()` to create the following vector:

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

```
rep(1:3, each = 2)
```

24) Give an example using the function `rep()` to create the following vector:

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

```
rep(1:3, 2)
```

25) Give an example of R code using the function `matrix()` that will give you the following output:

```
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

```r
x <- matrix(1:12, nrow = 4, ncol = 3)
```

26) Give an example of R code using the function `matrix()` that will give you the following output:

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
```

```r
y <- matrix(1:12,  byrow = TRUE, nrow = 4, ncol = 3)
```

27) You can use the colon operator `":"` to generate a sequence of numbers. For instance `1:3`. How can you get the help documentation for the colon operator?

Any of these options: `?":"`, `help(":")`, `?colon`, `help(colon)`

28) Every time you quit R, a message pops-up with the following question: `Save workspace image to /.RData?`.

- What is the so called *workspace image*? **The workspace is the workspace of your current session; it contains all the objects created during the session.**
- What type of file is `.RData`? **Any file with extension `.RData` is a file in R's binary format.**
- What happens if you choose the `Save` option? **R saves your *workspace image* in the file `.RData`.**

29) In RStudio, one of the panes has the tabs "Environment, History". What is the content of the "History" tab?

It shows the history of commands that you have been using.

30) In RStudio, one of the panes has the tabs "Files, Plots, Packages, Help, Viewer". If you click on the "Files" tab you will see the files of your home directory. There, you should be able to see a file called `.Rhistory`. What does this file contain?

The `.Rhistory` file contains the history of commands that you've typed in the current or older sessions.

31) When you start a new R session, a message with similar content to the text below appears on the console:

```
R version 3.3.1 (2016-06-21) -- "Bug in Your Hair"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale
```

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

- What happens when you type: `license()`? **The text of the GNU General Public License is displayed.**
- What happens when you type: `contributors()`? **Information about the authors of R and the list of contributors is displayed.**
- What happens when you type: `citation()`? **Indications on how to cite R in publications.**
- What happens when you type: `demo()`? **A series of demonstration plots are displayed.**

32) When knitting a file, what happens to those code chunks that use the option `echo = FALSE`?

The code in the chunk is not displayed in the knitted document.

33) When knitting a file, what happens to those code chunks that use the option `eval = FALSE`?

The code in the chunk is not evaluated.

34) When knitting a file, what happens to those code chunks that use the option `results = 'hide'`?

The output of the code is hidden (i.e. not displayed).

35) When knitting a file, what happens to those code chunks that use the option `comment = ""`?

The output of the code does not show the double hash ##.

36) A code chunk in an `.Rmd` file can be assigned a unique name by specifying a label inside the curly braces, like this: `{r some-label}`. When knitting a file, what happens if you have two different code chunks with the same label?

You get an error. You cannot have two different code chunks with the same name. You can, however, use the same label of another chunk in an empty chunk (i.e. no code in it).

37) What does the function `getwd()` do?

Returns the current working directory.

38) What does the function `setwd()` do?

Sets the working directory to the specified directory.

39) Say you have a data table in a CSV file supposedly called `dataset.csv`. When trying to read the file with `read.csv()` from your working directory you get the following error message:

```
Error in file(file, "rt") : cannot open the connection
In addition: Warning message:
In file(file, "rt") :
  cannot open file 'dataset.csv': No such file or directory
```

- What does the error indicate? **This error indicate that the file `dataset.csv` is not in the working directory.**

- What could be causing the error? **The file `dataset.csv` is not in your working directory. Or the name of the file is incorrect.**
- How could you try to solve the error? **Either: make sure that the file is indeed named `dataset.csv`; or change the working directory to where `dataset.csv` is located, or change the file path inside `read.csv()`.**

40) The reading-table functions use the following parameters: `header`, `sep`, `dec`, `row.names`, `colClasses`, `stringsAsFactors`. Explain what they do, and give an example of the value that each of these parameters can take.

- **`header`: whether the file contains the names of the variables as its first line.**
- **`sep`: the field separator character.**
- **`dec`: character used for decimal points.**
- **`row.names`: a vector of row names or a single number giving the column that contains the row names.**
- **`colClasses`: vector of classes to be assumed for the columns.**
- **`stringsAsFactors`: whether vectors should be converted to factors.**

41) What does `read.table()` and friends—e.g. `read.csv()`, `read.delim()`—do by default to columns with characters?

By default, all reading table functions in base R convert columns with characters into factors.

42) Explain the concept of *vectorization* a.k.a. vectorized operations.

An operation is applied to all the elements of a vector.

43) Explain the concept of atomic structures in R.

Atomic means that all the elements inside a data object must be of the same mode (i.e. or the same type).

44) The following question was posted on piazza in a previous edition of Stat 133. What would you answer to the student:

> read_csv() was working before, but now I am trying to restart my homework 2 and it says "Error: could not find function 'read_csv'. I installed the package readr and I am on the right directory. How do I get out of read_csv error?

The student should make sure to load the `"readr"` package: `library(readr)`

45) Explain the use of brackets `[ ]` and parentheses `( )` in R.

Brackets are used to manipulate objects: for slicing, subscripting, or extracting values. Parenthesis are used when invoking a function.

46) Consider a data frame `df` with two columns `x` and `y`. What happens if you try to extract a column that is not in `df`, e.g. `df$z`?

You get `NULL`

47) Name three functions for inspecting the contents of a data frame.

`head()`, `tail()`, `str()`, `summary()`

48) Which command will fail to return the first five elements of a vector `x`? (assume `x` has more than 5 elements).

a. `x[1:5]`
b. `x[c(1,2,3,4,5)]`
c. `head(x, n = 5)`
d. `x[seq(1, 5)]`
e. `x(1:5)` **This one**

Consider the following data frame `df`:

```
     first     last gender born        spell
1    Harry   Potter   male 1980 sectumsempra
2 Hermione  Granger female 1979    alohomora
3      Ron  Weasley   male 1980   riddikulus
4     Luna Lovegood female 1981      episkey
```

49) Refer to the data frame `df`. What commands will fail to return the data of individuals born in 1980?

a. `df[c(TRUE, FALSE, TRUE, FALSE), ]`
b. `df[df[,4] == 1980, ]`
c. `df[df$born == 1980]` **This one**
d. `df[df$born == 1980, ]`
e. `df[ ,df$born == 1980]` **This one**

50) Refer to the data frame `df`. Your friend is trying to display the first three rows on columns 1 (`first`) and 2 (`last`), by unsuccessfully using the command: `df[1:3, 1 & 2]`

```
df[1:3, 1 & 2]
```

```
##      first    last gender born        spell
## 1    Harry  Potter   male 1980 sectumsempra
## 2 Hermione Granger female 1979    alohomora
## 3      Ron Weasley   male 1980   riddikulus
```

a. Why does the command above print all columns? **The command `df[1:3, 1 & 2]` displays all columns because `1 & 2` is a logical comparison that returns `TRUE`, and therefore all columns are selected.**

b. Write a command that would correctly display the first two columns. **`df[1:3, 1:2]`, or `df[1:3, c(1, 2)]`, `df[1:3, c('first', 'last')]`**

51) Refer to the data frame `df`. Write a command that would give you the following data from `df`:

```
         spell    first
1 sectumsempra    Harry
2    alohomora Hermione
3   riddikulus      Ron
4      episkey     Luna
```

`df[ ,c('spell', 'first')]` or also `df[ ,c(5,1)]`

52) Refer to the data frame `df`. Select the command that does NOT provide you information about the data frame `df`:

a) `head(df)`
b) `str(df)`
c) `tail(df)`

d) `rm(df)` **this one**
e) `summary(df)`

Consider the following tibble `sw`:

```
# star wars data frame
sw
```

```
# A tibble: 4 x 4
  name   gender height weight
  <chr>  <chr>   <dbl>  <int>
1 Anakin male     1.88     84
2 Padme  female   1.65     45
3 Luke   male     1.72     77
4 Leia   female   1.50     49
```

53) Refer to the data frame `sw`. Using `"dplyr"`, which of the following commands gives you the data of female individuals:

```
## # A tibble: 2 x 4
##   name  gender height weight
##   <chr> <chr>   <dbl>  <int>
## 1 Padme female   1.65     45
## 2 Leia  female   1.50     49
```

a) `filter(sw, gender == female)`
b) `select(sw, gender == 'female')`
c) `select(sw, gender == 'female')`
d) `filter(sw, gender == 'female')` **This one**

54) Refer to the data frame `sw`. Using the pipe operator `"%>%"` in `"dplyr"`, which of the following commands gives you the data of male individuals:

```
## # A tibble: 2 x 4
##   name   gender height weight
##   <chr>  <chr>   <dbl>  <int>
## 1 Anakin male     1.88     84
## 2 Luke   male     1.72     77
```

a) `sw %>% select(gender == 'male')`
b) `sw %>% group_by(gender == 'male')`
c) `sw %>% filter(gender == 'male')` **This one.**
d) `sw %>% filter(by == 'male')`

55) Refer to the data frame `sw`. Using `"dplyr"`, which of the following commands would give you `name` and `height` arranged by `height` as follows:

```
## # A tibble: 4 x 2
##   name   height
##   <chr>   <dbl>
## 1 Leia     1.50
## 2 Padme    1.65
## 3 Luke     1.72
## 4 Anakin   1.88
```

a) `sw %>% arrange(height) %>% select(name, height)` **This one.**

b) `sw %>% select(name, height) %>% arrange(height)` **This one.**
c) `sw %>% select(name, height) %>% arrange(desc(height))`
d) `sw %>% filter(name, height) %>% arrange(height)`

56) Refer to the data frame `sw`. Using the pipe operator `"%>%"` in `"dplyr"`, which of the following commands gives you average height:

```
## # A tibble: 1 x 1
##   avg_height
##        <dbl>
## 1       1.69
```

a) `sw %>% select(height) %>% summarise(avg_height = mean(height))` **This one.**
b) `sw %>% select(height) %>% avg_height = mean(height)`
c) `sw %>% select(height) %>% summarise(avg_height = mean(weight))`
d) `sw %>% select(height) %>% summarise(avg_height = median(height))`

57) Write a dplyr command that gives the average weight by gender:

```
## # A tibble: 2 x 2
##   gender avg_weight
##   <chr>       <dbl>
## 1 female       47.0
## 2 male         80.5
```

```
# average weight by gender
sw %>%
select(gender, weight) %>%
group_by(gender) %>%
summarise(avg_weight = mean(weight))
```

58) A student is trying to implement the following formula in R:

$$e^{\frac{-(X-\mu)^2}{2\sigma^2}}$$

However, the student gets unexpected results when using the code:

```
exp(-(x - mu)^2 / 2 * sigma^2)
```

Explain the problem and correct the code.

Solution. It divides by 2, then multiplies by `sigma^2` (instead of dividing by `2*sigma^2`). Use parentheses:

`exp((-(x - mu)^2) / (2 * sigma^2))` or also `exp(-(x - mu)^2 / (2 * sigma^2))`

59) Indicate whether the following statements are True of False.

- CSV format is a comma-delimited format. **TRUE**
- All data values in a plain text file are stored as a series of characters. **TRUE**
- The first row in a csv file is always used to indicate column names. **FALSE**
- Delimited format files allow you to define any sort of complex data structure. **FALSE**
- The ubiquity of spreadsheets, and its format of row-and-columns, make them the most efficient way to represent data. **FALSE**

- Spreadsheet data files stored in a riched format (e.g. Excel) can be opened in a text editor to inspect its contents. **FALSE**
- Spreadsheet software (e.g. Excel) can be used to view or explore field-delimited files. **TRUE**

60) Consider the following table (containing some "messy" data), and assume it is stored in a `csv` file:

name	gender	height	weight	status
Anakin	male	1.88m	84kg	1 = jedi
Padme	Female	1.65m	???	2 = queen
Luke	MALE	1.72m	77kg	1 = jedi
Leia	female	150cm	49kg	3 = princess

One of your friends has to do an exploratory analysis of the data table, and she asks for your help. Name four data cleaning aspects that you would discuss with your friend before doing any exploration.

Any four of the following comments is OK

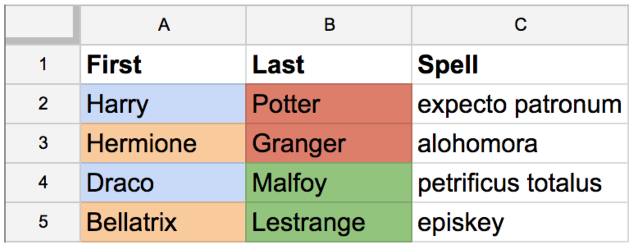
- **convert `gender` to either lower case or upper case.**
- **identify `height` values in meters, remove `"m"`, and convert to numeric.**
- **identify `height` values in centimeters, remove `"cm"`, and convert to numeric expressing value in meters (i.e. dividing by 100).**
- **remove characters `"kg"` in column `weight`, and then convert to numeric.**
- **replace `"???"` with missing value `NA`.**
- **clean the column `status`, preferably by removing numbers and equal sign, so that only the category names are left.**

61) Based on the previous data set, you suggest your friend to create a data dictionary (i.e. metadata). How would you create such dictionary? (verbal description, no code).

Give one point for each variable in the dictionary. These should be at least the name of the variable and a brief description of its meaning and/or its units. Optionally there should also be information about the type of data, and what are possible values for characters (or factor) variables. Here is one possible example of dictionary:

- `name`: character, indicating name of individual
- `gender`: character indicating sex: `male` and `female`
- `height`: numeric value indicating height in meters
- `weight`: numeric value indicating weight in kilograms
- `status`: character indicating status: `jedi`, `queen`, and `princess`

Consider a spreadsheet with the following content



	A	B	C
1	**First**	**Last**	**Spell**
2	Harry	Potter	expecto patronum
3	Hermione	Granger	alohomora
4	Draco	Malfoy	petrificus totalus
5	Bellatrix	Lestrange	episkey

- The cells `Harry` and `Draco` have a *blue* background indicating male gender.
- The cells `Hermione` and `Bellatrix` have an *orange* background indicating female gender.
- The cells `Potter` and `Granger` have a *maroon* background indicating house of Gryffindor.

- The cells `Malfoy` and `Lestrange` have a *green* background indicating house of Slytherin.

62) What is the issue with using highlighting features to codify information in a spreadsheet?

One of the following explanations is OK:

- **The highlighting is nice visually, but it's hard to grab that information for use in the later analysis.**
- **If the file is saved as a text file, the enriched formats of the colors won't be part of the resulting file.**

63) Describe how would you modify the content in the spreadsheet with Harry Potter's data to avoid highlighting cells. (verbal description, no code).

It would be better to include two more columns. One column for the Gender, and another column for the House of the individuals.

64) Below is a list of possible names for an R vector. Some names are valid and some are invalid. Identify the invalid names, and explain why they are invalid.

a. `3var_name`

- **Option a) is invalid because it starts with a number**.

b. `var_name4`

c. `_var_name`

- **Option c) is invalid because it starts with an underscore; underscores should not appear at the beginning of a name**.

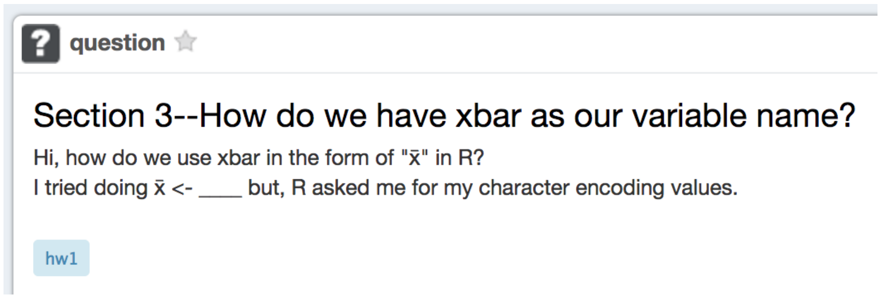
d. `.var_name`

e. `VarName`

f. `var-name`

- **Option f) is invalid because it contains a dash, which is reserved for the "minus" or subtraction operator**.

65) Consider the following question posted in piazza, associated with the formula to calculate the mean (or average): $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.



Section 3--How do we have xbar as our variable name?

Hi, how do we use xbar in the form of "x̄" in R?
I tried doing x̄ <- ____ but, R asked me for my character encoding values.

What is the issue with the attempted code, and how would you help to solve it?

The problem is the use of the character: $\bar{x}$, which is not a valid name for an R object. The solution would be to use variable names like `x_mean`, or `x_avg`, or `x_bar`, for instance.

66) This question is based on a post from piazza (slitghtly adapted).

Consider the previous tibble `sw` and the four commands below:

```
# star wars data frame
sw
```

```
# A tibble: 4 x 4
  name   gender height weight
  <chr>  <chr>   <dbl>  <int>
1 Anakin male     1.88     84
2 Padme  female   1.65     45
3 Luke   male     1.72     77
4 Leia   female   1.50     49
```

```
typeof(sw[["weight"]])
```

```
## [1] "integer"
```

```
typeof(sw$weight)
```

```
## [1] "integer"
```

```
typeof(sw["weight"])
```

```
## [1] "list"
```

```
typeof(sw[,"weight"])
```

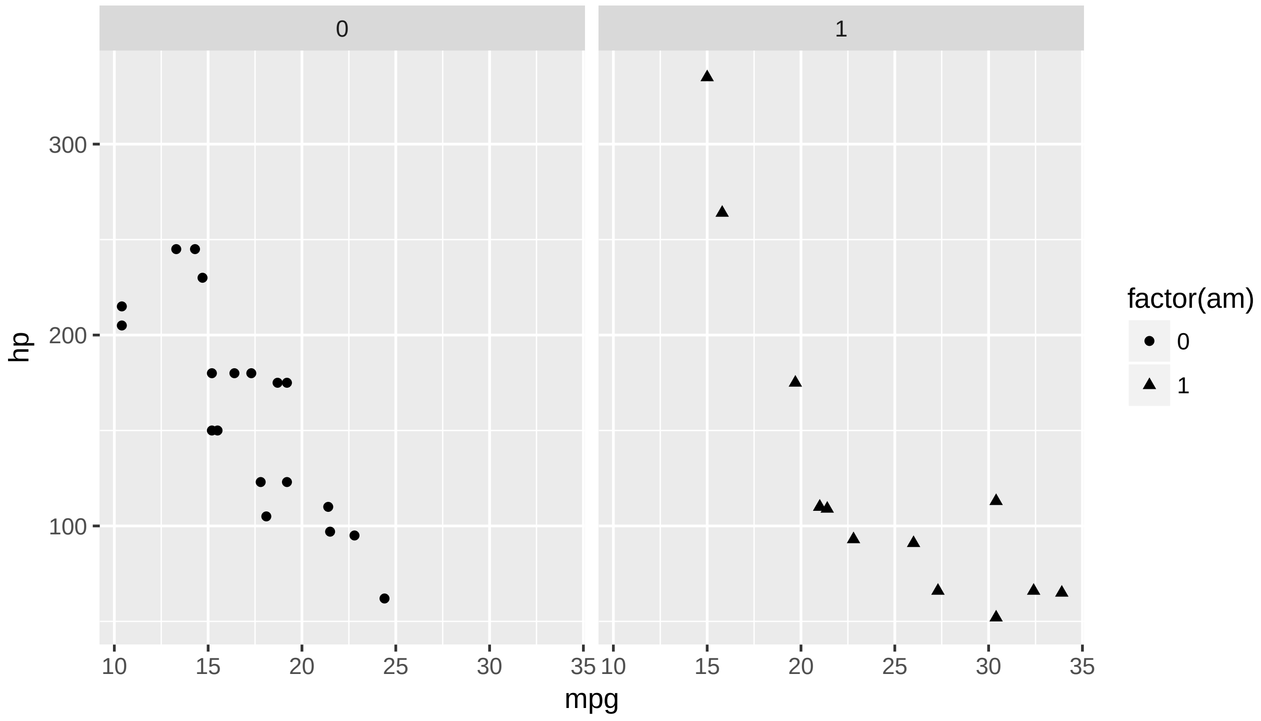
```
## [1] "list"
```

Why do `typeof(sw[["weight"]])` and `typeof(sw$weight)` return `"integer"`, while `typeof(sw["weight"])` and `typeof(sw[,"weight"])` return `"list"`?

The first and second commands use `[[ ]]` and `$` which are the operators that *extract* the vector that forms the `weight` column.

The third and fourth commands use `[ ]` and `[ , ]` which are the operators that select the element but don't *get access* (inside) the vectors.

67) Consider the following plot obtained using the data `mtcars`:



What call generates the previous figure:

```r
# option 1
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(aes(shape = factor(am))) +
  facet_wrap(~ am)
```

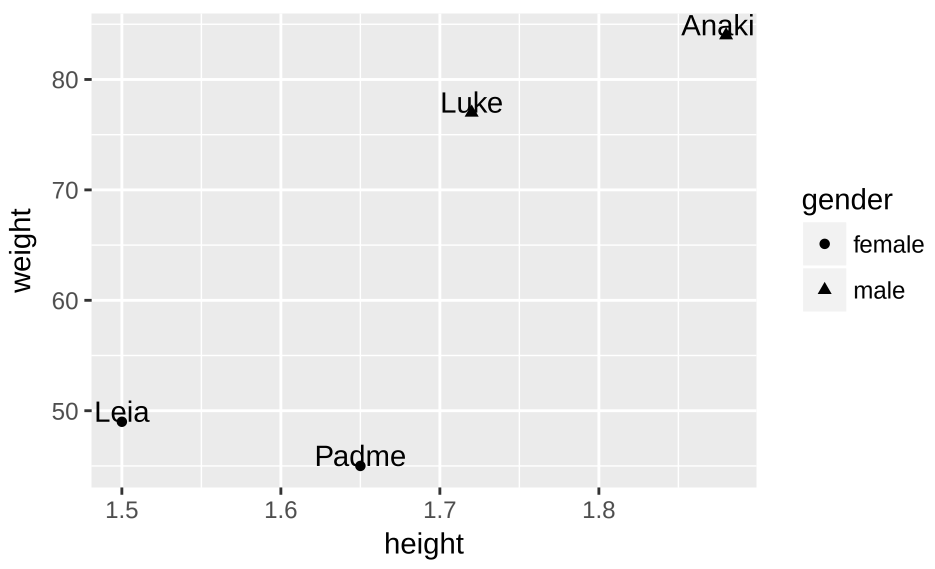
```r
# option 2
ggplot(mtcars, aes(x = mpg, y = hp)) +
  geom_text(size = factor(am)) +
  facet_wrap(~ am)
```

```r
# option 3
ggplot(mtcars, aes(x = mpg, y = hp)) +
  geom_point(aes(shape = factor(am))) +
  facet_frames(~ am)
```

```r
# option 4
ggplot(mtcars, aes(x = mpg, y = hp)) +
  geom_point(aes(shape = factor(am))) +
  facet_wrap(~ am)
```

Solution: option 4

68) Consider the following ggplot based on the tibble `sw`:

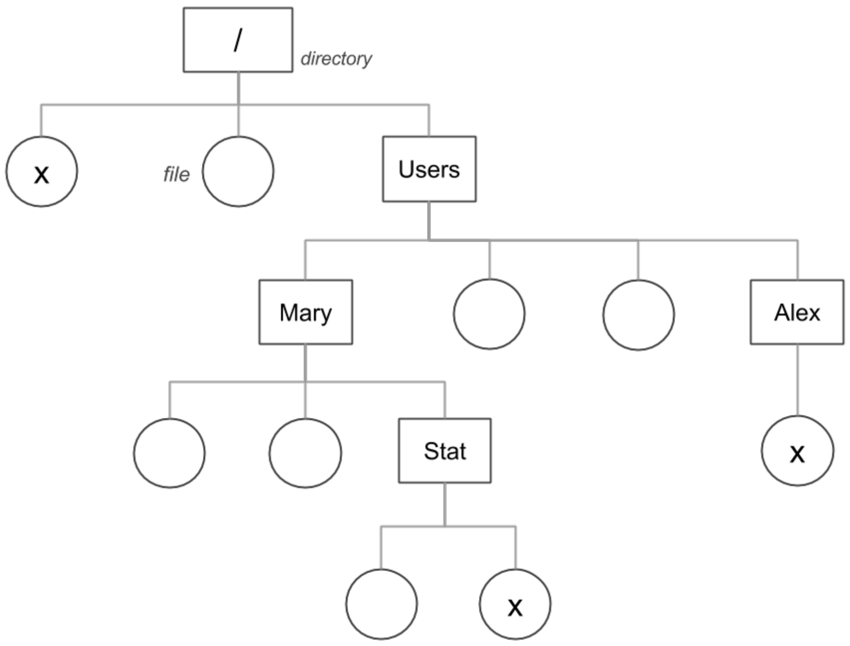


Fill in the blanks:

```
ggplot(data = ______, aes(x = _________, y = _________))  ___
  geom_________(______(shape = gender))  ___
  geom_________(aes(label = name))
```

```
ggplot(data = sw, aes(x = height, y = weight)) +
  geom_point(aes(shape = gender)) +
  geom_text(aes(label = name), size = 4, vjust = 0)
```

Consider the filesystem illustrated in the scheme below:



69) Write the absolute path name to the `x` file in directory `Alex`.

`/Users/Alex/x`

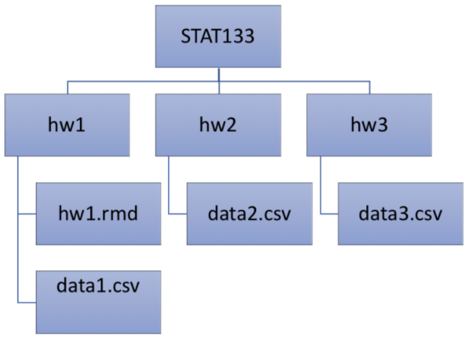
70) Write the relative path from `Alex` to `Stat`.

`../Mary/Stat`

71) Select the option that indicates the relative path to file `x` at the top from within the directory `Stat`:

a) `../Mary/x`
b) `../../../x` **this one**
c) `/x`
d) `/Users/Mary/Stat/x`

72) Consider the filesystem illustrated in the scheme below:



a. Suppose you are at `STAT133` directory, write the shell command to navigate to directory `hw1`.

```
cd hw1
```

b. Suppose you are at `hw1` directory, write the shell command to list contents in your current directory in long format.

```
ls -l
```

c. Suppose you are at `hw1` directory, write the shell command to create a new subdirectory `image` in your working directory.

```
mkdir image
```

d. Suppose you are at `hw1` directory, write the shell command to rename `data2.csv` as `newdata.csv`

```
mv ../hw2/data2.csv  ../hw2/newdata.csv
```

e. Suppose you are at `hw1` directory. Explain what the command `rm *.csv` does?

Removes all the csv file inside current directory

f. Suppose you are at `hw1` directory. Explain what the command `cp ./data1.csv ../hw2/data2.csv` does?

It copies the `data1.csv` file to the `hw2` directory as `data2.csv`. Because `hw2` already has a `data2.csv`, this will be replaced with the copied file.

73) Suupose you are working at the command line (e.g. terminal, gitbash) and your working directory contains a CSV file `data.csv`. Explain what the follwoing commands are doing:

a. `wc data.csv`

count lines, words, and bytes

b. `wc -l data.csv`

count number of lines

c. `head data.csv`

inspect first 10 rows

d. `tail data.csv`

inspect last 10 rows

e. `less data.csv`

see contents with a paginator

f. `head -n 11 data.csv > newdata.csv`

rediction of 11 rows of `data.csv` to `newfile.csv`

g. `cut -d "," -f 3 data.csv`

select the 3rd column in `data.csv`

h. `cut -d "," -f 3 data.csv | tail +2 > newfile.csv`

select 3rd column of `data.csv`, excluding column name, and redirect output to `newfile.csv`

74) Assume that `a` and `b` are two numbers. Indicate whether the following commands are valid or invalid.

a) `if a > b {z <- a + b}` **Invalid.**

b) `if (a > b) z <- a + b else y <- a * b` **Valid.**

c) `if (a > b) then z <- a + b` **Invalid.**

d) `if (a > b) z <- a + b` **Valid.**

e) `if (a > b) {z <- a + b} else {y <- a * b}` **Valid.**

75) For a set of numbers $x_1, x_2, x_3, \dots, x_n$, the power mean with exponent p is defined as:

$$\left(\frac{1}{n} \sum_{i}^{n} x_i^p \right)^{1/p} = \frac{1}{n} (x_1^p + \cdots + x_n^p)^p$$

Assume that the values for $x_1, x_2, x_3, \dots, x_n$ are given by the following vector `x`:

```
# vector of numbers
x <- 1:10
```

One of your friends wrote a function in R to calculate the power mean as:

```
# function to compute power mean
# input: a numeric vector
# output: power mean
power-mean <- function(y, p) {
  n <- length(x)
  summation <- 0
  for (i in 1:n) {
    summation + (x[i]^p)
  }
  summation <- (1/n) * summation
  return(summation^(1/p)
}
```

Your friend asks you to review the code and look for any bugs. Help your friend find any errors, and write code that fixes the function.

the name of the function must not contain the dash `-`

the input parameter is `y` but the code uses `x`

the for loop does not update `summation`

there is a missing `)` in the `return()` statement

76) The formula of the weighted average is:

$$avg = \frac{w_1 x_1 + w_2 x_2 + \cdots + w_n x_n}{w_1 + w_2 + \cdots + w_n}$$

Assume that the values for x and w are given by the following vectors:

```r
# vectors of values and weights
x <- 1:10
w <- seq(from = 0.1, to = 1, by = 0.1)
```

One of your friends wrote R code to implement such formula in the following way:

```r
# computation of the numerator
numerator <- 0
for (j in 1:10) {
  numerator <- numerator + (w[j] * x[j])
}

# computation of the denominator
denominator <- 0
for (i in 1:10) {
  denominator <- denominator + w[j]
}

# computation of weighted average
avg <- denominator / numerator
```

Your friend asks you to review the code and look for any bugs. What would you tell your friend?

There are two bugs. The first bug has to do with the index `j` used `w[j]` in the second loop. The second bug has to do with the calculation of `avg`, the division should be `numerator / denominator`.

77) The Compound Interest Formula is given by:

$$A = P\left(1 + \frac{r}{n}\right)^{nt}$$

where:

- P = principal amount (the initial amount you borrow or deposit)
- r = annual interest rate (as a decimal)
- t = number of years the amount is deposited or borrowed for.
- n = number of times the interest is compounded per year
- A = amount of money accumulated after n years, including interest.

Write a function `amount()` to implement the formula of the amount with compound interest. Use descriptive names for the arguments. In other words, don't use `P` or `r`, for principal or interest; instead choose a more descriptive name. Also, give default values to all the arguments. This function should return the corresponding amount of money.

```
# one possible implementation is:
# (students can choose different default values)
amount <- function(principal = 1, rate = 0.01, years = 1, times = 12) {
  principal * (1 + rate / times)^(times * years)
}
```

78) What does the following code print out at each iteration?

```
f <- 1
g <- 1

for (i in 1:5) {
  print(g)
  g <- f - g
  f <- f + g
}
```

```
## [1] 1
## [1] 0
## [1] 1
## [1] 1
## [1] 2
```

79) Consider the gaussian (Normal) function, given in the equation below, and the code that implements such equation.

$$f(x) = \frac{1}{\sqrt{2\pi}s} exp\left\{-\frac{1}{2}\left(\frac{x-m}{s}\right)^2\right\}$$

```r
# gaussian function
f<-function(x=1,m=0,s=1){
a<-1/(sqrt(2*pi))
b<-exp(-0.5*((x-m)/s)^2)
a*(1/s)*b
}
```

The function works and it has no bugs… but it is hard to review the code at first glance. Name at least four aspects that you would change to improve readability.

Any four of the following comments is OK:

- **add documentation for what the function does.**
- **add documentation for what the inputs are.**
- **add documentation for what the output is.**
- **add spaces between operators.**
- **use indentation inside braces.**
- **give more descriptive names.**
- **add comments.**

80) The code below takes each element in the vector `x` and transforms it by calling `floor()` and adding an integer `K`. The transformed values are stored in the vector `y`. Rewrite the code below without using a loop or an apply function:

```r
x <- rnorm(50, 0, 2)
y <- x
for (i in 1:length(y)) {
  K <- 2
  y[i] <- floor(x[i]) + K
}
```

```r
# use vectorized code and recycling
y <- floor(x) + K
```

81) Use logical subsetting to rewrite the following code, eliminating the need for any loops or if statement (Don't write a function).

```r
x <- rnorm(20)
y <- x

for (i in 1:length(x)) {
   if (x[i] < -1) {
     y[i] <- 0
   } else {
     if (x[i] > 1) {
       y[i] <- 1
     } else y[i] <- 3 * x[i]^2
   }
}
```

```r
# one option is:
y <- 3 * (x^2)
y[x < -1] <- 0
y[x > 1] <- 1

# another option:
y[x < -1] <- 0
y[x > 1] <- 1
y[x >= -1 & x <= 1] <- 3 * x[x >= -1 & x <= 1]^2
```

82) Match the provided concepts with the corresponding descriptions:

```
- dot                        - caret: ^
- regular expression         - [:xdigit:]
- metacharacters             - asterisk: *
- character class            - \\
```

a) _____ characters with special meaning, that do not match themselves literally **metacharacters**

b) _____ match hexadecimal digits **`[:xdigit:]`**

c) _____ match only one out of several characters **character class**

d) _____ pattern describing a certain amount of text **regular expression**

e) _____ used for escaping characters in R \\

f) _____ match the preceding token zero or more times **asterisk: ***

g) _____ typing it after an opening bracket negates the character class

h) _____ matches a single character **`dot`**

83) Consider the following character vector:

```r
pns <- c("pan", "pen", "pin", "pon", "pun", "p.n", "p1n")
```

What elements are matched by the following regular expressions (write the name of the elements, NOT their positions). *Note*: matched wither by `grep()` or `str_detect()`

a. `"[ei]"` **pen, pin**

b. `"p.n"` **pan, pen, pin, pon, pun, p.n, p1n**

c. `"p[[:alpha:]]n"` **pan, pen, pin, pon, pun**

d. `"p[0-9]n"` **p1n**

e. `"p..n"` **no matches**

f. `"\\d"` **p1n**

g. `"\\."` **p.n**

84) Consider the following character vector:

```r
food <- c("burrito", "burger", "pizza", "salad")
```

What elements are matched by the following regular expressions (write the name of the elements, NOT their positions):

a. `"[ei]"` **burrito, burger, pizza**

b. `"r[r]"` **burrito**

c. `"[alpha]"` **pizza, salad**

d. `"[[:alpha:]]"` **burrito, burger, pizza, salad**

e. `"\\w+"` **burrito, burger, pizza, salad**