R Packages II

The first two questions refer to the `kknn` package (https://github.com/KlausVigo/kknn), which implements a popular predictive modeling algorithm called k-nearest neighbors. Install it with `install.packages("kknn")` and load it with `library(kknn)` before you start.

1. Use `data()` to list the data sets stored in `kknn`. Then try typing the name of one of them, `glass`, at the console: unlike most packages you've used, it isn't there until you run `data(glass)`. What is this property of a package called, and which field of `DESCRIPTION` turns it on? Is that field in `kknn`'s `DESCRIPTION`?

2. What is the name of the file in the package that is edited to create the help files for these data sets?

The rest of the questions use `greetr`, the package you made in PS 5, rebuilt with `usethis` the way we rebuilt `mypkg` in lecture. Download `greetr.zip`, unzip it, and open the `greetr` folder in Positron (or RStudio). `hello()` is in `R/hello.R`, and `DESCRIPTION` is filled in.

3. Add the following tibble to your package as a data set called `greetings`. Write the command you used, and say which file it created.

```
library(tibble)
greetings <- tibble(
  greeting = c("Hello", "□□", "Hola", "Hei", "السلام عليكم"),
  language = c("English", "Chinese", "Spanish", "Norwegian", "Urdu")
)
```

4. Document this data set using `roxygen2` so that when a user types `?greetings` at the console they see a description of the data set, its format (the columns and their values), and its source (you can cite STAT 133). Please transcribe your documentation below.

5. Paste this Roxygen block into `R/hello.R`, directly above `hello()`, and fill in the three blanks: the argument, the return value, and an example. Run `devtools::document()` and preview the result with `?hello`. Write your three completed lines below.

```
#' Say hello
#'
#' Prints a greeting to whoever is named in `x`.
#'
#' @param ___
#' @returns ___
#' @examples
#' ___
#' @export
```

6. Run `devtools::check()`. How many errors, warnings, and notes does it report, and what does the note say needs fixing? Fix it: change the `License` field in `DESCRIPTION` to `MIT + file LICENSE`, and add a plain-text file called `LICENSE` at the top of the package with two lines, `YEAR: 2026` and `COPYRIGHT HOLDER:` followed by your name. Run `check()` again. What does it report now?

Testing

You'll need the `testthat` package for these problems: install it with `install.packages("testthat")` if you haven't already. To run tests at the console, outside a package, load it with `library(testthat)`.

7. Here is a function and three tests for it. Without running the code, predict the `?` values in the report that is returned by `test()`: `[ FAIL ? | WARN 0 | SKIP 0 | PASS ? ]`. For any test that fails, is the function wrong or the test?

```
celsius <- function(f) {
  (f - 32) * 5 / 9
}

test_that("celsius() converts the boiling point", {
  expect_equal(celsius(212), 100)
})

test_that("celsius() converts the freezing point", {
  expect_equal(celsius(0), 32)
})

test_that("celsius() converts a vector of temperatures", {
  temps <- celsius(c(32, 212))
  expect_length(temps, 2)
  expect_equal(temps, c(0, 100))
})
```

8. A classmate wrote this test for `celsius()` from the last problem, ran it, and it passed. Why did it pass? Change the test so it would catch the mistake.

```
test_that("celsius() rejects text", {
  expect_error(celsuis("hot"))
})
```

9. Here is part of a package with the `strsplit1()` from lecture and a new function that counts the words in a sentence, followed by the report from `test()`.

```
n_words <- function(x) {
  length(strsplit1(x, split = " "))
}
```

```
Failure ('test-n_words.R:6:3'): n_words() counts words with extra spaces
Expected `n_words("two  spaces")` to equal 2.
`actual`:   3
`expected`: 2
[ FAIL 1 | WARN 0 | SKIP 0 | PASS 4 ]
```

a. Which file and line holds the expectation that failed, and what is the test's description?

b. How many expectations passed?

c. Why does `n_words()` return 3? (Hint: look closely at the input, and run `strsplit1()` on it.)

d. Is this input a typical input, an edge case, or an error case?

The next four problems use your `greetr` package from the R Packages II problems. Open it in Positron (or RStudio), and run `library(devtools)` at the console.

10. Set `greetr` up for testing, with a test file for `hello()`. Write the two commands you used. What files did they create, and what changed in `DESCRIPTION`?

11. Replace the placeholder in `test-hello.R` with two tests: one that `hello("Oski")` returns `"Hello Oski!"`, and one that `hello()` given two names returns a character vector of length 2. Run `test()`. What does its last line say?

12. `hello()` prints its greeting as well as returning it, which clutters the output of `test()`. Refactor `hello()`: remove the `print()` so it just returns the string. Run `test()` again. Do the tests still pass, and what does that tell you? Does `hello("Oski")` still show its greeting when you run it at the console?

13. A user files a bug report: "`hello(NA)` says `Hello NA!`." Fix it with test-driven development:

a. **Red**: add a test to `test-hello.R` that `hello(NA)` throws an error whose message mentions `"missing"`. Run `test()` and confirm that it fails.

b. **Green**: change `hello()` so it throws that error, using `stop()`, whenever any element of `x` is `NA`. Run `test()` until it passes.

14. Testing isn't only for packages. Suppose you're about to analyze `storms` (from `dplyr`) and are relying on two facts about it:

- no row appears twice;
- every row with hurricane-force winds (`wind` of 64 knots or more) has a `category`.

Write each as its own `stopifnot()` call. Both fail. For each, look at the rows that break it and decide whether the data or the expectation is wrong.

15. *Optional Challenge*: Write the grade calculator from lecture, test first.

a. Create a test file for a function `letter_grade()` that holds every promise from the lecture's activity:

Score	`95`	`85`	`42`	`98`	`91`	`100`	`58`	`62`	`NA`	`0.85`	`0.91`	`86.5`	`89.5`	`0.865`
Grade	A	B	F	A+	A-	A+	F	D-	F	B	A-	B+	A-	B+

Add a test for a score of `1`, deciding what it should get.

b. Write `letter_grade()` so that every test passes, for a vector of scores. Two tools you haven't seen will help: `x %% 10` is the last digit of a whole number `x`, and `floor(x + 0.5)` rounds halves up (`round()` rounds halves to the even number, so `round(86.5)` is 86).

APIs I

The BART API serves live data, so some answers in this section (how many trains are about to depart, how long the wait is) depend on when you run your code. For those there is no single right answer.

Revisit the homepage for the BART Legacy API discussed in Lecture.

https://www.bart.gov/schedules/developers/api

16. What is the public key that can be used to access data through the legacy API?

Next visit the documentation site: https://api.bart.gov/docs/overview/index.aspx. Note that each category of data has its own endpoint / base URL, and all take a similar format (e.g. for estimated departure data use https://api.bart.gov/api/etd.aspx.)

17. Consider the following URL. Which part is the endpoint, and what are the parameters and their values? Using the documentation, what does this request ask for?

```
https://api.bart.gov/api/stn.aspx?cmd=stninfo&orig=12TH&key=MW9S-E7SL-26DU-VV8V&json=y
```

18. Provide the URL that will query the API for a list of all BART stations in JSON format.

19. Provide the corresponding shell/terminal command.

20. Provide the corresponding R code that will perform the query and parse the JSON response into an R list object. Use `simplifyVector = TRUE` in `resp_body_json()`, so that JSON arrays of records become data frames.

21. Subset the list to return a data frame that contains the station abbreviation, name, latitude, longitude, address, city, county, state, and zipcode for each station. Which county has the greatest number of stations?

---

22. Provide the R code that will query the API for estimated departure times (ETDs) from the Downtown Berkeley Station and parse the JSON response into an R list object.

23. Extract the element called `root` and then the `station` element within it. View the object using the data viewer. What type of object is it and what are its dimensions?

24. The `unnest()` function from the tidyr package expands a list-column into ordinary rows and columns. Run the code below, then explain what each `unnest()` step did to the rows and the columns of the data frame.

```
library(tidyr)
etd_unnested <- etd |>
  unnest(etd)
etd_fully_unnested <- etd_unnested |>
  unnest(estimate)
```

25. How long (in minutes) is the shortest estimated wait time for a train departing from Downtown Berkeley Station? Which destination does this train go to? For reference, what time is it now?

---

26. *Optional*: Query the BSA (BART System Advisories) endpoint to retrieve the latest system advisory information. You're welcome to use any method: web browser, command line, or R. What are the different bits of information that they provide through this endpoint?

27. *Optional Challenge*: Use the data provided by the BART API to construct a map of the BART system showing the locations of all stations. You could make it interactive with leaflet or static like the one below.

