Mock Midterm Exam — Answers
Introduction to Applied Data Science

Name: _____ Student No: _____

2026-05-17

Multiple Choice Answers

Question	Answer
1	B
2	B
3	D
4	B
5	C
6	C
7	D
8	C
9	C
10	C
11	B
12	C
13	C
14	C
15	D
16	C
17	C
18	C
19	D
20	B

Motivations:

1. **B** — The prediction objective focuses on minimising the gap between $\hat{y}$ and y; causal identification is the objective of *explanation*, not prediction.

2. **B** — Storing multiple values in a single cell (`"Math: 8, English: 7"`) violates tidy data: each cell must contain exactly one value.

3. **D** — `str()` displays the structure of an object, including column types and a short preview of values. `head()` shows rows but not types; `dim()` gives dimensions only; `class()` gives the top-level class.

4. **B** — `c()` applies implicit coercion and converts everything to the most general type (character). `list()` keeps each element's original type.

5. **C** — `length()` counts top-level elements of a list. The list has three elements (`a`, `b`, `c`), regardless of the length of `b`.

6. **C** — `getwd()` returns the current working directory. `setwd()` sets it; `list.files()` lists contents; `dir.exists()` checks existence.

7. **D** — Single-bracket `[` subsetting always returns the same class as the original object, so `countries["name"]` returns a one-column data frame. The double-bracket `[[` and `$` operators drop to a plain vector.

8. **C** — In `[rows, columns]`, the row-selection condition goes before the comma. Option A is missing the comma entirely; B filters columns instead of rows; D compares a string to a number.

9. **C** — `na.rm = TRUE` excludes `NA` values from the computation; it does not remove entire rows or impute zeros.

10. **C** — From `~/project/reports/`, `..` steps up to `~/project/`, then `data/unemployment.csv` navigates into the `data/` directory. Option A stays inside `reports/`; B uses an absolute `~`-path that is unrelated to the project; D uses an absolute root path.

11. **B** — `here()` always anchors to the project root, so the same call works whether the code is run from a script, a `.qmd` file, or an interactive console session.

12. **C** — `read_csv()` is the `readr` function for CSV files. `read.csv()` is base R; `read_excel()` is from `readxl`; `fromJSON()` is from `jsonlite`.

13. **C** — The `sheet` argument specifies which sheet to read. `page`, `tab`, and `range` are not valid argument names for `read_excel()`.

14. **C** — `GET()` performs an HTTP GET request, which retrieves data. POST uploads, PUT/PATCH updates, DELETE removes.

15. **D** — HTTP 200 means OK (success). 404 is Not Found; 401 is Unauthorised; 429 is Too Many Requests.

16. **C** — HTTP 401 (Unauthorised) means the request lacks valid authentication credentials.

17. **C** — The `?` separates the path from the query string. Each `key=value` pair after `?` is a query parameter; multiple parameters are separated by `&`.
18. **C** — `rvest` is R's primary web-scraping package. Visualisation is `ggplot2`; SQL interfaces use `DBI`/`dbplyr`; modelling uses packages like `lm` or `tidymodels`.
19. **D** — `html_attr("href")` extracts the value of a named HTML attribute. `html_text()` extracts text content; `html_name()` returns the tag name; `html_children()` returns child nodes.
20. **B** — The `>` combinator selects only *direct* children. A space (descendant combinator) would match at any nesting level.

Open Question Answers

Consider the following file structure:

```
my_project/
├── data/
│   └── gdp.csv
├── scripts/
│   └── analysis.R
└── report.qmd
```

1. (3pts) Relative path and `here()` equivalent.

- **Relative path** (working directory is `my_project/scripts/`):

  ```
  read_csv("../data/gdp.csv")
  ```

  `..` steps up one level from `scripts/` to `my_project/`, then navigates into `data/`.

- **Using `here()`**:

  ```
  read_csv(here("data", "gdp.csv"))
  ```

- **Key difference:** The relative path depends on the current working directory at run time. If the script is executed from a different location, the path breaks. `here()` always anchors to the project root (detected by the `.Rproj` or `.here` file), so the same call works correctly regardless of where the script is run from.

Have a look at the following `data.frame`:

```
  student_id  name grade passed
1          1  Anna   7.5   TRUE
2          2 Boris   8.8   TRUE
```

```
3          3  Chen   6.2  FALSE
4          4 Diana   9.1   TRUE
```

2. (3pts) Difference between `students["grade"]` and `students[["grade"]]`.

- `students["grade"]` uses single-bracket subsetting, which always returns an object of the **same class** as the original. Here it returns a **one-column `data.frame`** with the column name preserved.
- `students[["grade"]]` uses double-bracket subsetting, which *drops* the wrapping structure and returns the **underlying `numeric` vector** of grade values.
- **Why it matters:** Many functions (e.g., `mean()`, `sd()`, `hist()`) expect a plain vector. Passing a data frame instead causes an error. Conversely, functions that expect a data frame (e.g., `ggplot()`) will fail if given a bare vector. Using the correct operator avoids hard-to-debug type errors.

Have a look at the following URL:

```
https://api.worldbank.org/v2/country/nl/indicator/NY.GDP.MKTP.CD?format=json&
per_page=10
```

3. (3pts) URL components.

Component	Value	Explanation
Protocol	`https`	Secure HTTP; encrypts the connection
Domain	`api.worldbank.org`	The server hosting the World Bank API
Endpoint	`/v2/country/ nl/indicator/ NY.GDP.MKTP.CD`	`v2` = API version 2; `nl` = Netherlands country code; `NY.GDP.MKTP.CD` = World Bank indicator for GDP at current USD
Query parameter 1	`format=json`	Requests the response in JSON format (default is XML)
Query parameter 2	`per_page=10`	Limits the number of records returned per page to 10

Have a look at the following HTML snippet:

```
<div class="article">
  <h2 class="headline">EU Economy Grows by 2.5%</h2>
  <p class="summary">The European economy showed strong growth...</p>
  <a href="https://news.example.com/eu-growth">Read more</a>
</div>
<div class="article">
  <h2 class="headline">Inflation Remains Stable</h2>
  <p class="summary">Consumer prices held steady in October...</p>
  <a href="https://news.example.com/inflation">Read more</a>
</div>
```

4. (3pts) CSS selectors and `rvest`.

- **(a)** Selector for all elements with class `"headline"`:

  ```
  .headline
  ```

  The leading dot denotes a class selector; it matches any element regardless of tag name.

- **(b)** Selector for `<a>` elements inside a `<div class="article">`, and extracting `href` values:

  ```
  div.article a
  ```

  The space (descendant combinator) matches `<a>` tags anywhere inside the `<div>`. In `rvest`:

  ```
  page |>
    html_elements("div.article a") |>
    html_attr("href")
  ```

  This returns a character vector containing all `href` attribute values.

Have a look at the following `rvest` code:

```
library(rvest)

page     <-  read_html("https://en.wikipedia.org/wiki/List_of_countries_by_GDP_
(PPP)")
result <- page |>
  html_element("table.wikitable") |>
  html_table()
```

5. (3pts) Step-by-step explanation.

1. `page <- read_html(...)` — downloads the raw HTML of the Wikipedia page at the given URL, parses it into a structured XML/HTML tree, and stores it in `page`.
2. `html_element("table.wikitable")` — searches `page` for the **first** HTML element matching the CSS selector `table.wikitable` (a `<table>` tag with class `wikitable`) and returns that element.
3. `html_table()` — converts the selected HTML `<table>` element into an R **data frame** (tibble), mapping table rows to rows and columns to columns.

- **`result`** is a **data frame** (tibble) containing the contents of the first wikitable on the page.
- **If `html_element()` were replaced with `html_elements()`:** `html_elements()` returns *all* matching elements, not just the first. `html_table()` applied to the resulting node set would return a **list of data frames**, one for each matching wikitable found on the page.

End of Answers