



Welcome to QuACK!

Week 2

9/15/2025

Nina, Sophie, Alyson, and Victoria

Today's Agenda

- Warm-up activity
- Dataframes
- Packages
- Intro to Tidyverse
- Useful functions in Tidyverse
- Practice

Week 2 Warm-up – 10 minutes

1. Download this week's materials from this link: <https://tinyurl.com/quack-fall2025>
2. Make a New R script. Save it to your computer. Title it "S2_warmup"
3. Add a comment with the date, a title, and your name
4. Create three different vectors:
 - a. A vector of your favorite numbers
 - b. A vector of your favorite people
 - c. A vector of your favorite booleans
5. Run the code and check that everything looks correct in the global environment.
6. Open the starter script for this week.

Dataframes are a convenient way to view your data, and they can be configured in a number of different ways to best suit your needs

But there are some near-universal guidelines!

UC Berkeley



- Multiple data types in one column
- Multiple variables in a column (e.g., date + experimenter initials)
- Info outside of the dataframe (highlights, comments, etc.)
- Spaces in variable names



Data cleaning basics

- Sometimes, cleaning your data will involve changing capitalization, removing whitespace, or changing other details of your data formatting
- Useful functions for this are the `str_detect()` and `grep()` families
- They use RegEx to identify a pattern in your data, which you can then replace with something else

```
# Creating string vector
```

```
x <- c("ALYSON", "NINA")
```

```
# Calling grepl() function
```

```
grepl("N", x)
```

```
# Calling str_detect() function
```

```
str_detect("N", x)
```

Packages

- Base R is powerful on its own, but sometimes you'll need some added functionality
- Packages are collections of functions that other people have written and published for all to use
- R comes with some packages pre-loaded, but some you need to install/load yourself
- Here's how:

```
>> install.packages("packagename")
```

this will install the package and
add it to R on your computer

```
>> library(packagename)
```

this will load the package for
your current session

Tidyverse

- From their [website](#): “The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.”
- Each package has its own focus: reading in data, reshaping data, visualizing data, etc.
- You can install/load them all together using the name “tidyverse” or individually



%>%

Introducing: The Pipe Operator

And some useful tidyverse functions

- **select()** for (you guessed it) selecting things like columns that you want to include or drop from a dataframe
- **filter()** for including/removing rows with particular values
- **distinct()** for removing duplicate values
- **mutate()** for changing or adding columns with particular values (and much more)
- **case_when()** for using conditionals with mutate()
- **summarize()** for creating compact dataframes with summary statistics
- **group_by()** for collapsing your data by particular variables (usually for the purpose of summarizing)

And lastly, a cheat sheet!

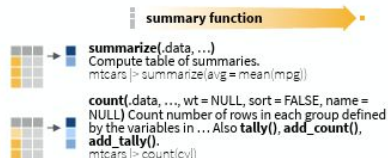
<https://rstudio.github.io/cheatsheets/html/data-transformation.html>

dplyr functions work with pipes and expect tidy data. In tidy data:



Summarize Cases

Apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).



Group Cases

Use **group_by(data, ..., add = FALSE, drop = TRUE)** to create a "grouped" copy of a table grouped by columns in ... dplyr functions will manipulate each "group" separately and combine the results.



Use **rowwise(data, ...)** to group data into individual rows. dplyr functions will compute results for each row. Also apply functions to list-columns. See tidy cheat sheet for list-column workflow.



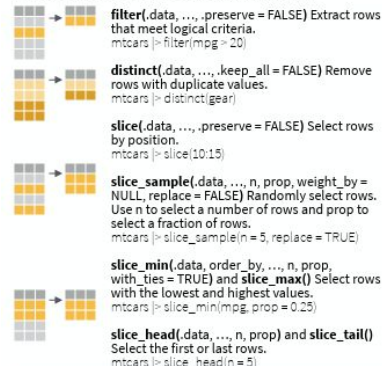
ungroup(x, ...) Returns ungrouped copy of table.

```
g_mtcars <- mtcars |> group_by(cyl)  
ungroup(g_mtcars)
```

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

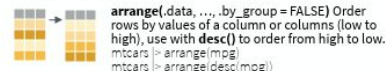


Logical and boolean operators to use with filter()

<code>==</code>	<code><</code>	<code><=</code>	<code>is.na()</code>	<code>%in%</code>	<code>xor()</code>
<code>!=</code>	<code>></code>	<code>>=</code>	<code>is.na()</code>	<code>!</code>	<code>&</code>

See ?base::Logic and ?Comparison for help.

ARRANGE CASES



ADD CASES



Summary Functions

TO USE WITH SUMMARIZE ()

summarize() applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output.

summary function

COUNT

`dplyr::n()` - number of values/rows
`dplyr::n_distinct()` - # of uniques
`sum(is.na())` - # of non-NA's

POSITION

`mean()` - mean, also `mean(is.na())`
`median()` - median

LOGICAL

`mean()` - proportion of TRUEs
`sum()` - # of TRUEs

ORDER

`dplyr::first()` - first value
`dplyr::last()` - last value
`dplyr::nth()` - value in nth location of vector

RANK

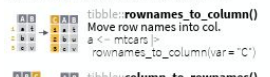
`quantile()` - nth quantile
`min()` - minimum value
`max()` - maximum value

SPREAD

`IQR()` - Inter-Quartile Range
`mad()` - median absolute deviation
`sd()` - standard deviation
`var()` - variance

Row Names

Tidy data does not use rownames, which store a variable outside of the columns. To work with the rownames, first move them into a column.



Also `tibble::has_rownames()` and `tibble::remove_rownames()`.

Combine Tables

COMBINE VARIABLES

bind_rows(..., id = NULL) Returns tables one on top of the other as a single table. Set `id` to a column name to add a column of the original table names (as pictured).

bind_cols(..., name_repair) Returns tables placed side by side as a single table. Column lengths must be equal. Columns will NOT be matched by `id` (to do that look at Relational Data below), so be sure to check that both tables are ordered the way you want before binding.

RELATIONAL DATA

Use a **"Mutating Join"** to join one table to columns from another, matching values with the rows that correspond to. Each join retains a different combination of values from the tables.

left_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ..., keep = FALSE, na_matches = "na") Join matching values from x to y.

right_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ..., keep = FALSE, na_matches = "na") Join matching values from y to x.

inner_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ..., keep = FALSE, na_matches = "na") Join data. Retain only rows with matches.

full_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"), ..., keep = FALSE, na_matches = "na") Join data. Retain all values, all rows.

COLUMN MATCHING FOR JOINS

Use `by = c("col1", "col2", ...)` to specify one or more common columns to match on.
`left_join(x, y, by = "A")`

Use a named vector, `by = c("C1" = "col2")`, to match on columns that have different names in each table.
`left_join(x, y, by = c("C" = "D"))`

Use `suffix` to specify the suffix to give to unmatched columns that have the same name in both tables.
`left_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))`

COMBINE CASES



Use a **"Filtering Join"** to filter one table against the rows of another.



anti_join(x, y, by = NULL, copy = FALSE, ..., na_matches = "na") Return rows of x that do not have a match in y. Use to see what will not be included in a join.

Use a **"Nest Join"** to inner join one table to another into a nested data frame.

nest_join(x, y, by = NULL, copy = FALSE, keep = FALSE, name = NULL, ...) Join data, nesting matches from y in a single new data frame column.

SET OPERATIONS

intersect(x, y, ...) Rows that appear in both x and y.

setdiff(x, y, ...) Rows that appear in x but not y.

union(x, y, ...) Rows that appear in x or y, duplicates removed. **union_all()** retains duplicates.

Use **setequal()** to test whether two data sets contain the exact same rows (in any order).



Time to get started in R!

- Go to <https://tinyurl.com/quack-fall2025>
- Download and unzip the course materials
- You will see many files:
 - Two R Markdown (.rmd) files
 - A .csv file
 - Two PDF files – slides and a practice doc
 - Some other folders – ignore these for now
- Open the R Markdown file marked “starter_code” – this is where we will code together for the first part of the session
- The .csv file contains the data we will be working with
- The practice PDF contains practice exercises for you to do during the second part of the session



Exit Ticket

Link: <https://tinyurl.com/quack-fall2025-exit-survey>

