W4: Data Wrangling with Tidy Data, Part 1

```
Downloading webR package: generics
Downloading webR package: magrittr
Downloading webR package: fansi
Downloading webR package: utf8
Downloading webR package: vctrs
Downloading webR package: pillar
Downloading webR package: pkgconfig
Downloading webR package: tibble
Downloading webR package: withr
Downloading webR package: tidyselect
Downloading webR package: dplyr

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union
```

Where are we?

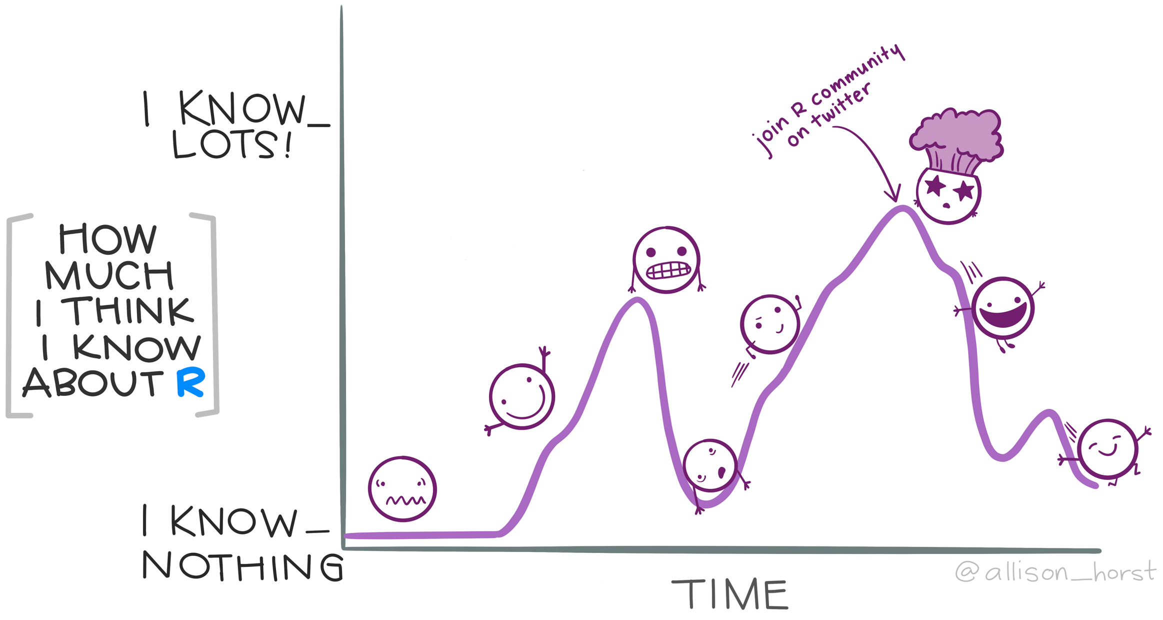
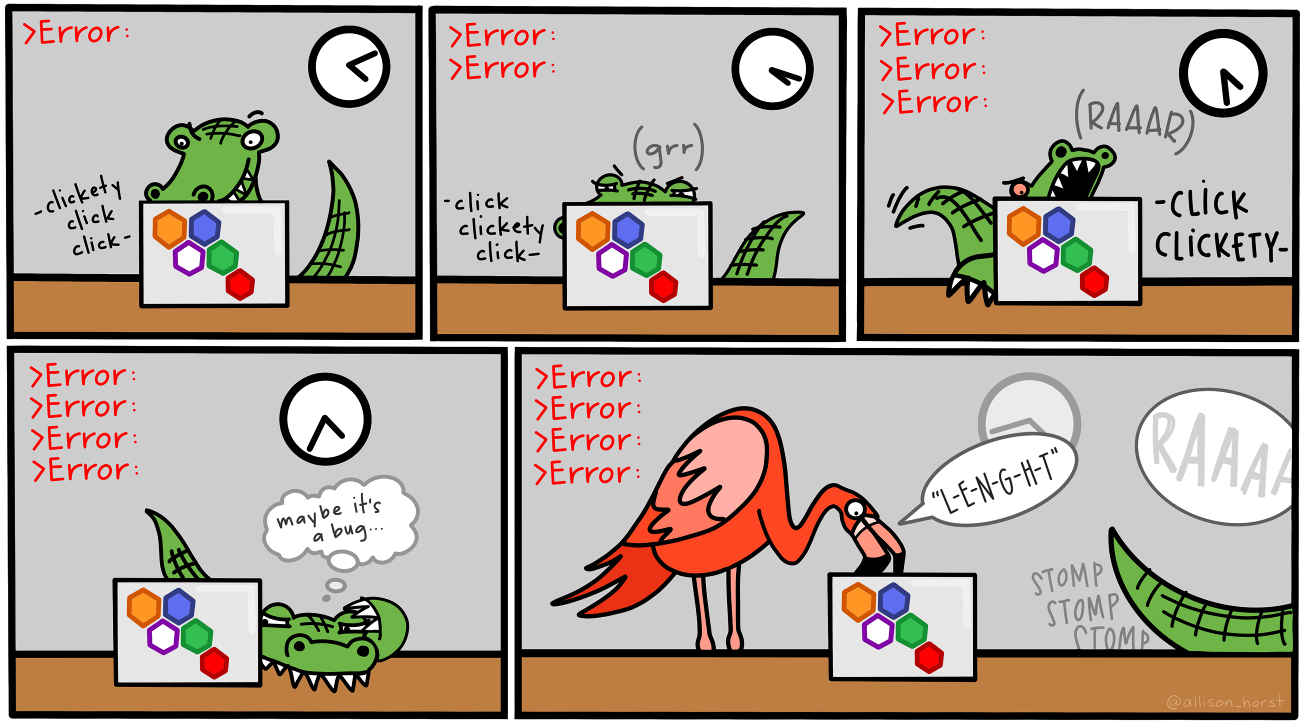
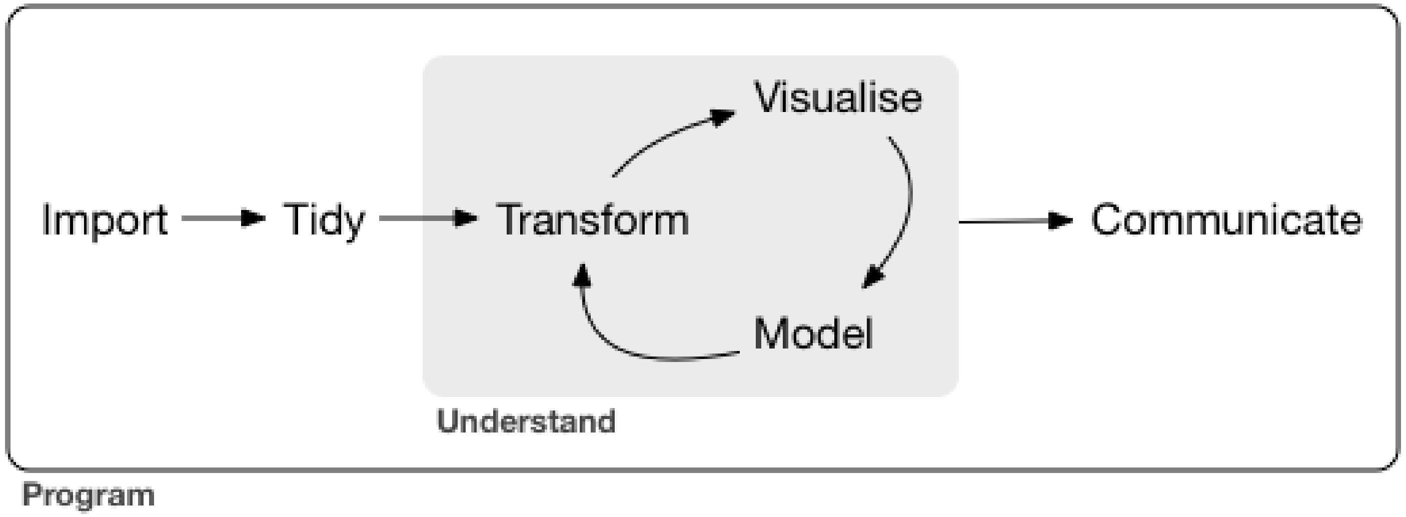


Illustration by Allison Horst

How's it going?



Data Science Workflow



We start with *Transform* and *Visualize* with the assumption that our data is in a nice, **“tidy”** state.

Our working Tidy Data: DepMap Project

https://depmap.org/

We will work with `metadata`, `mutation`, and `expression` `data.frame`s.

What do you want to do with this `data.frame`?

Remember that a major theme of the course is about: **How we organize ideas <-> Instructing a computer to do something.**

With Tidy data, we can ponder how we want to transform our data that satisfies our scientific question.

dplyr lets us do data wrangling

How is `dplyr` related to the `tidyverse`?

- `tidyverse` is a set of packages for working with data
- `dplyr` is one of them
- `ggplot2` is another
- `readr` loads data
- packages for dealing with data types

When you use:

```r
library(tidyverse)
```

That loads up the `tidyverse` packages

When do I use `library()`?

You should only have to load packages once in your session. So using `library(tidyverse)` will load most of everything you need.

Six main `dplyr` functions

Function Name	Purpose	When
`select()`	Selects sets of columns in df	This week
`filter()`	Filters rows in df	This week
`mutate()`	Calculate a New Column in df	Next Week
`group_by()/summarize()`	Calculate summary statistics across groups	Next Week
`arrange()`	Sorts a df by one or more columns	Next Week

And Some More!

Function Name	Purpose	When
`_join()`	Functions to merge two tables together	Next Week
`|>`	Operation to build pipelines	This Week

Subsetting a dataframe

In the dataframe you have here, which rows would you filter for and columns would you select that relate to a scientific question?

✅ Implicit: “I want to filter for rows such that the subtype is breast cancer and look at the Age and Sex.”

🚫 Explicit: “I want to filter for rows 20-50 and select columns 2 and 8”.

Notice that when we filter for rows in an implicitly way, we often formulate criteria about the columns.

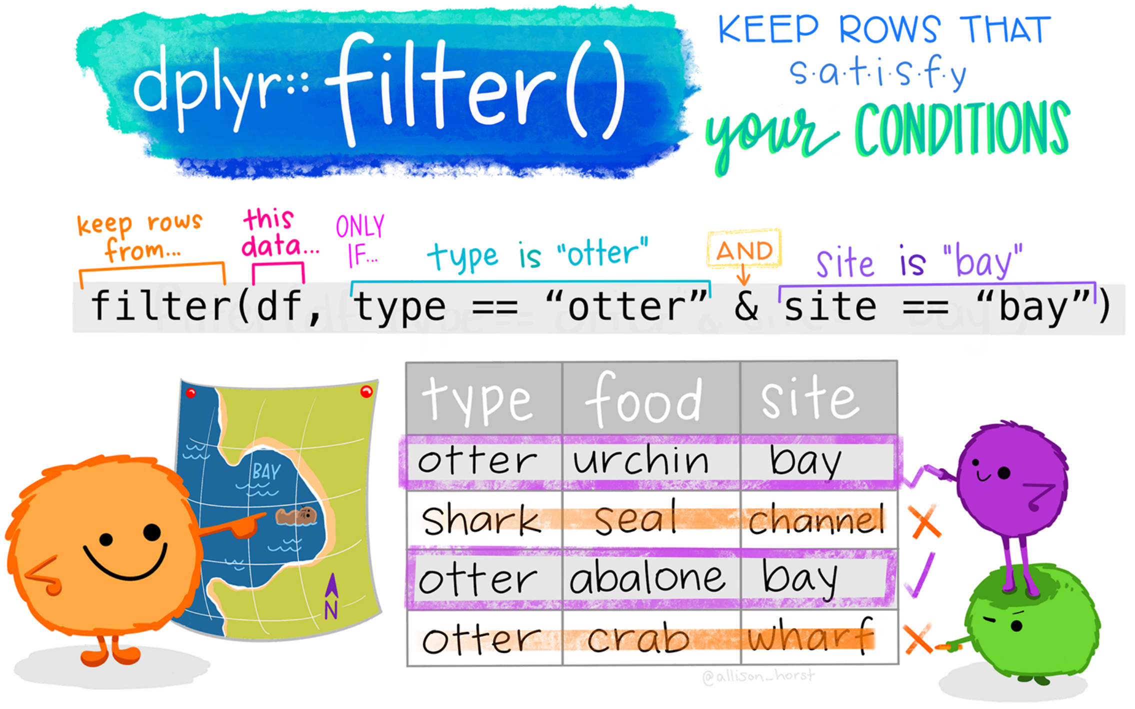
How we do it:

```r
library(tidyverse)

metadata_filtered = filter(metadata, OncotreeLineage == "Breast")
breast_metadata = select(metadata_filtered, ModelID, Age, Sex)

head(breast_metadata)
```

	ModelID <chr>	Age <dbl>	Sex <chr>
1	ACH-000017	43	Female
2	ACH-000019	69	Female
3	ACH-000028	69	Female
4	ACH-000044	47	Female
5	ACH-000097	63	Female
6	ACH-000111	41	Female

6 rows

Here, `filter()` and `select()` are functions from the `tidyverse` package.



```
filter(df, type == “otter” & site == “bay”)
```

type	food	site
otter	urchin	bay
shark	seal	channel
otter	abalone	bay
otter	crab	wharf

`filter()`

`metadata_filtered = filter(metadata, OncotreeLineage == "Breast")`:

The second argument: a logical indexing vector built from a comparison operator?

But the variable `OncotreeLineage` does not exist in our environment!

Rather, `OncotreeLineage` is a column from `metadata`, and we are referring to it as a **data variable**. We can directly refer to the column vector `metadata$OncotreeLineage` with just `OncotreeLineage`.

Try `filter` Out

Try `filter()` for `Sex == "Female"`:

R Code Start Over Run Code

```r
metadata_filtered = filter(metadata, ----------)
metadata_filtered
```

`select()`

The input arguments for `select()` are also **data variables**.

```r
select(metadata_filtered,     # Our dataset
       ModelID, Age, Sex)    # Our columns
```

ModelID <chr>	Age <dbl>	Sex <chr>
ACH-000017	43	Female
ACH-000019	69	Female
ACH-000028	69	Female
ACH-000044	47	Female
ACH-000097	63	Female
ACH-000111	41	Female
ACH-000117	46	Female
ACH-000147	54	Female
ACH-000148	74	Female
ACH-000196	44	Female

1-10 of 92 rows

Previous **1** 2 3 4 5 6 ... 10 Next

Try `select()` out

Add `OncotreeLineage` to the `select()` statement:

R Code Start Over Run Code

```r
select(metadata_filtered,     # Our dataset
       ModelID, Age, ----)    # Our columns
```

Keep In Mind

- `select()` works on columns
- `filter()` works on rows

Combining Operations into a Pipeline

The Common Thing about `tidyverse` functions

Both `filter()` and `select()`:

- Take a `data.frame` as input
- Return a `data.frame` as output

Why Pipes?

When combining multiple functions in one expression, it gets harder to read:

```
breast_metadata = select(filter(metadata, OncotreeLineage == "Breast"), ModelID, Age, Sex)
```

Or, this: 🤨

```
result2 = function1(function2(function3(dataframe)))
```

Or… 🤕

```
result = function1(function2(function3(dataframe, df_col4, df_col2), arg2), df_col5, arg1)
```

R style guide

Pipes to make nested functions readable

```
result2 = dataframe |>
    function1 |>
    function2 |>
    function3
```

```
result = function1(df_col5, arg1) |>
         function2(arg2) |>
         function3(df_col4, df_col2)
```

Applying our knowledge

Rewrite the `select()` and `filter()` function composition example using the pipe metaphor and syntax.

```r
breast_metadata = metadata |>
  filter(OncotreeLineage == "Breast") |>
  select(ModelID, Age, Sex)

breast_metadata
```

ModelID <chr>	Age <dbl>	Sex <chr>
ACH-000017	43	Female
ACH-000019	69	Female
ACH-000028	69	Female
ACH-000044	47	Female
ACH-000097	63	Female
ACH-000111	41	Female
ACH-000117	46	Female
ACH-000147	54	Female
ACH-000148	74	Female
ACH-000196	44	Female

1-10 of 92 rows   Previous **1** 2 3 4 5 6 ... 10 Next

Reading Code with Pipes

When I see pipes, I read them as `AND THEN`:

```r
metadata |>                                      ## I took the metadata data.frame AND THEN
   filter(OncotreeLineage == "Breast") |>        ## I filtered it AND THEN
   select(ModelID, Age, Sex)                     ## I selected columns from it
```

Why does this work?

- Pipes work by assuming the first argument is the dataset
- We input our `data.frame` into the first function:

```
metadata |>
   filter(OncotreeLineage == "Breast")
```

The output at this point is a `data.frame`, which means we can feed it into our next function:

```
metadata |>
   filter(OncotreeLineage == "Breast") |>
   select(ModelID, Age, Sex)
```

The output at this point is also a `data.frame`.

Tip for building pipelines

Look at the output at each step using `head()` before you move on!

```r
metadata |>
  filter(OncotreeLineage == "Breast") |>
  head()
```

	ModelID <chr>	PatientID <chr>	CellLineName <chr>	StrippedCellLineName <chr>	Age <dbl>
1	ACH-000017	PT-8CE6ah	SK-BR-3	SKBR3	43
2	ACH-000019	PT-viJKnw	MCF7	MCF7	69
3	ACH-000028	PT-viJKnw	KPL-1	KPL1	69
4	ACH-000044	PT-HMBfbj	MDA-MB-134-VI	MDAMB134VI	47
5	ACH-000097	PT-k1TO7o	ZR-75-1	ZR751	63
6	ACH-000111	PT-yKJqsn	HCC1187	HCC1187	41

6 rows | 1-6 of 31 columns

Step 2

```r
metadata |>
  filter(OncotreeLineage == "Breast") |>
  select(ModelID, Age, Sex) |>
  head()
```

	ModelID <chr>	Age <dbl>	Sex <chr>
1	ACH-000017	43	Female
2	ACH-000019	69	Female
3	ACH-000028	69	Female
4	ACH-000044	47	Female
5	ACH-000097	63	Female
6	ACH-000111	41	Female

6 rows

🤠

Try it Out

Build a pipeline that

- `filter(OncotreeLineage == "Lung")`
- `select(ModelID, OncotreeLineage, Age)`

Try piping the output into `head()` as you build it up

R Code Start Over Run Code

```r
metadata |>
  filter(-------=="") |>
  head()
```

What’s Next

- Making new columns in data with `mutate()`
- Make summaries with `group_by()`/`summarize()`
- Merging datasets with `_join()` functions