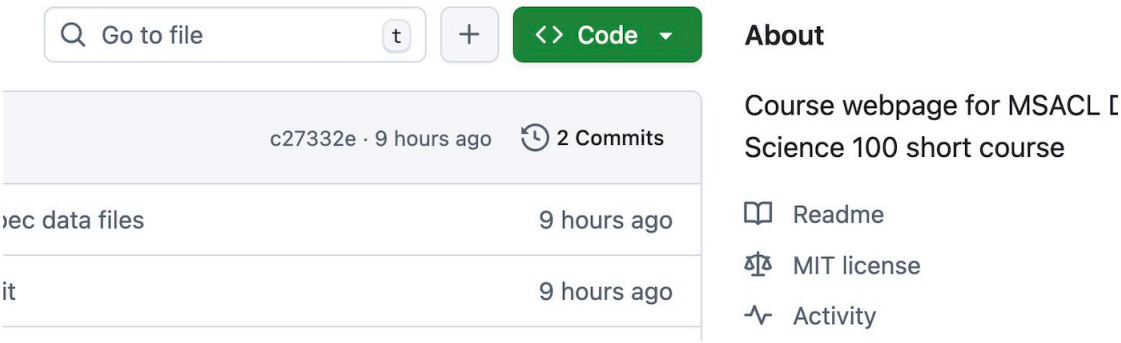
Lesson 1 Exercises

Exercises Setup

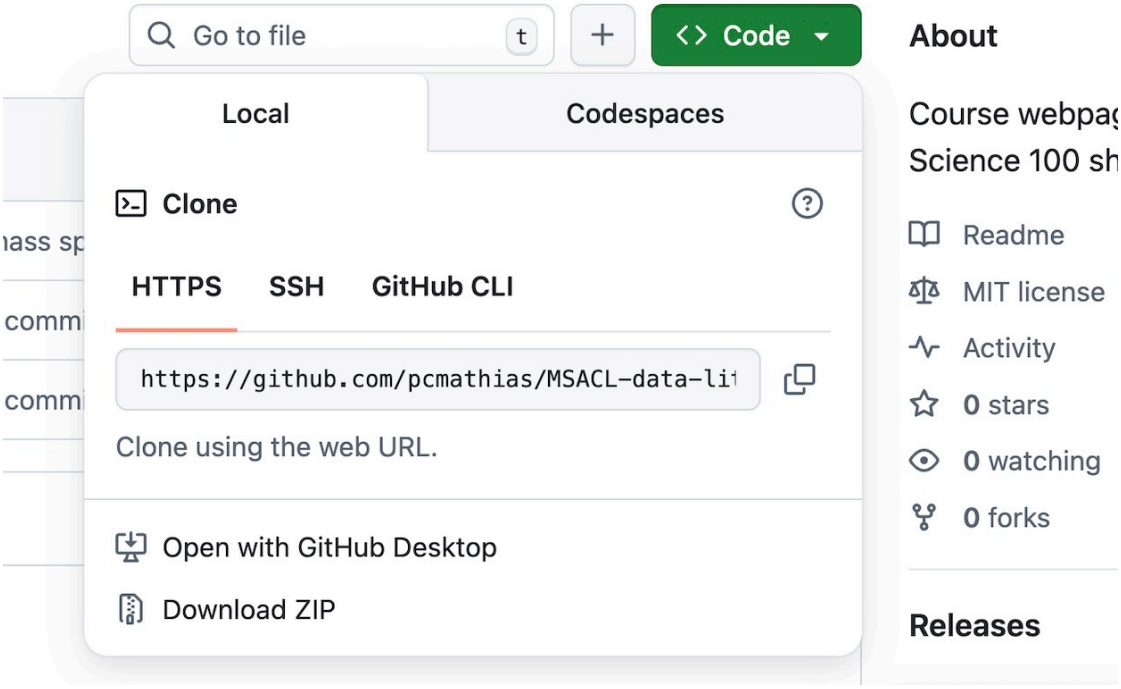
The materials for this course are hosted on a public Github website: https://github.com/pcmathias/MSACL-data-literacy. Github is a website that hosts repositories (repos), which are collections of files that are often organized into a project. You can think of each Github repository as a folder with files and subfolders. Github is a popular way to share code and projects with a broader community, who can contribute back to the repository. It also utilizes the git version control system, which allows users to track specific changes throughout the life cycle of a project.

To make accessing the course materials as easy as possible, we recommend downloading the repository for this course to your computer by following these steps.

1. Navigate to the course website in your browser
2. Press the "Code" button near the top right of the page



3. Click "Download ZIP" to download a .zip file that contains the repository



4. Place the .zip file in a location you will remember on your computer and unzip it. The best practice is to create a "Projects" folder in which you create a new folder for each project.

Orientation to the Data Set

Throughout this short course we will refer to a mass spectrometry data set that includes three files. The batch data file includes data that capture information about six analytes that are measured on 20 batches per day over the course of a month. The sample data file includes information about each sample in each batch, which is composed of 52 samples. Each sample has 24 peaks with data on each peak captured in the peak data file.

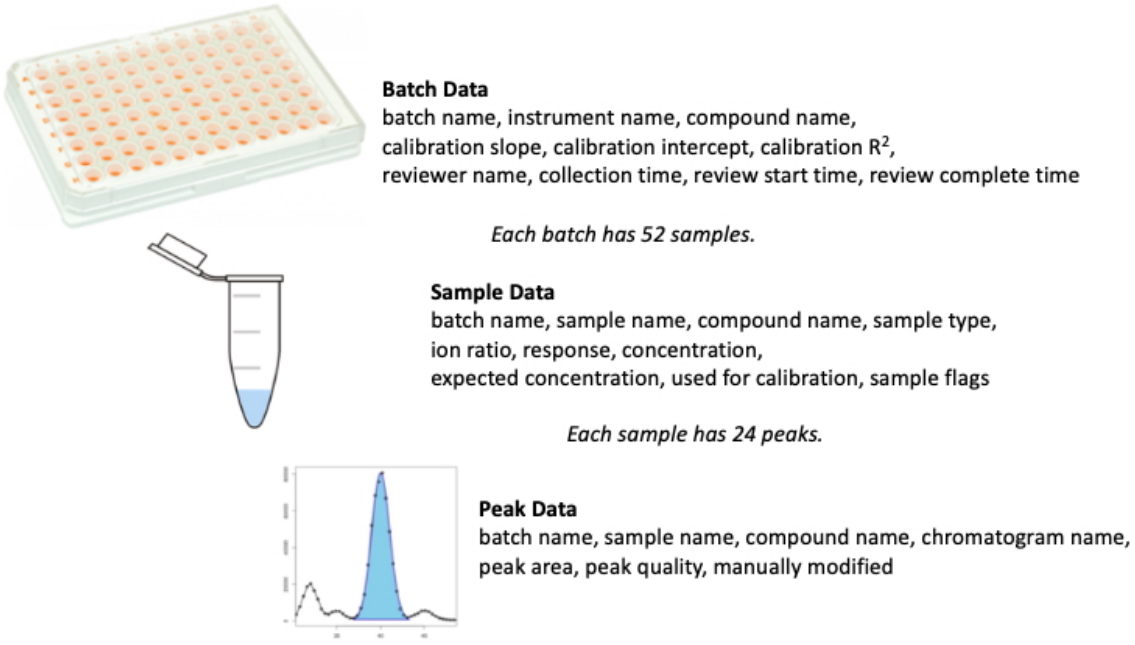


Figure 1: Summary of mass spec data set

Exercise 1: Data Types

We will explore data types in our mass spec data set and how different approaches to interacting with the data set may provide different views or perspectives of the data.

1. In the *exercises* folder within the main project folder, open the **Lesson 1 Exercise 1 Worksheet.xlsx** file.
2. Navigate to the *data* folder within the main project folder. Right click the **batch_data.csv** file and open the file with a text application such as Notepad for Windows or TextEdit for Macs.
3. Right click the **batch_data.csv** file and open with Excel. (Note that there is a similarly named **batch_data_ts.csv** file in the data folder as well. Do not open this yet.) If there is a pop-up message that prompts you to take action, read the message carefully and click "Convert."
4. For each of the column names (separated by commas in the text application), complete the **Lesson 1 Exercise 1 Worksheet.xlsx** based on the data types that you manually infer based on what we just learned about data types from the text file, and based on the format window in Excel.

5. Working the batch_data.csv file, click the File menu at the top left and click "Save As…" The file format to save should be defaulted to "Comma Separated Values (.csv)" - if it is not, select that and update the file name to "batch_data_excel.csv." Open the **batch_data_excel.csv** file using your text application. How is it different from the original **batch_data.csv** file that you previously opened within the text application?

6. Open the **batch_data_ts.csv** file in Excel. What is different in this file compared to the previous file?

7. Click on cell M2. In the formula box, enter the following:

   "=DATEVALUE(MID(I2,1,10))+TIMEVALUE(MID(I2,12,8))"

   The MID function extacts a specified number of characters from a text string, starting at a certain position. The formula will take the first 10 characters of the timestamp text and convert it into a date as a floating point value, take the 8 characters from the 12th position from the left and convert it into a time as a floating point value, and then sum the numeric values together.

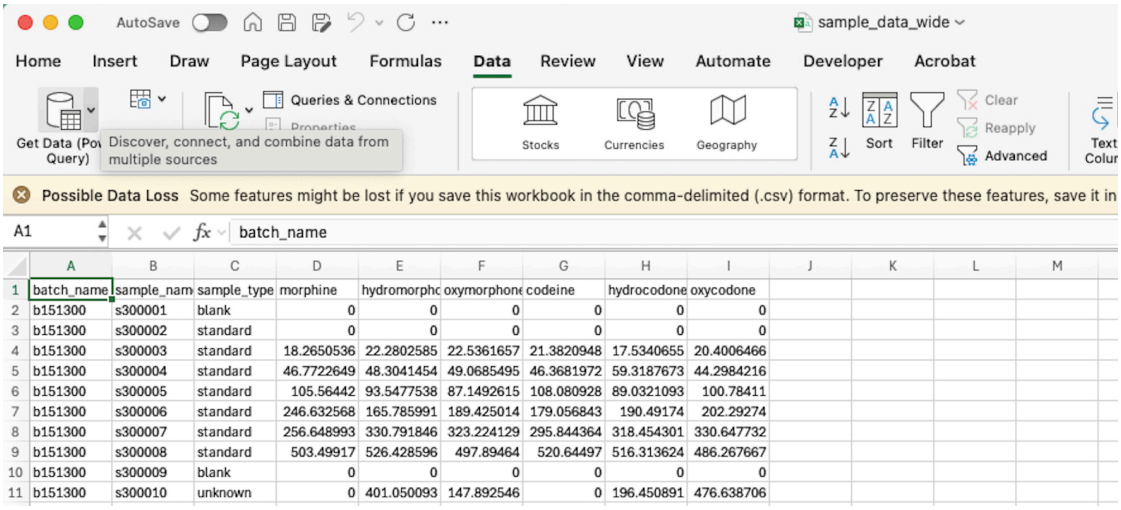
8. Convert the floating point number to a date and time recognized by Excel by right-clicking the cell, clicking on "Format Cells…", selecting "Time" from the Category list, and selecting the bottom most option in the Type window (3/14/12 13:30).

Exercise 2: Tidy Data

Many, but not all, systems and workflows will generate tidy data, where each variable is in its own column, each observation is in its own row, and each value is in its own cell. Transactional data that captures specific events and timestamps often is organized in this format, but sometimes the software for an instrument does not export data in this format.

1. Open **sample_data_wide.csv** in Excel. Does this look like tidy data? If not, which of the 3 principles of tidy data does it violate?

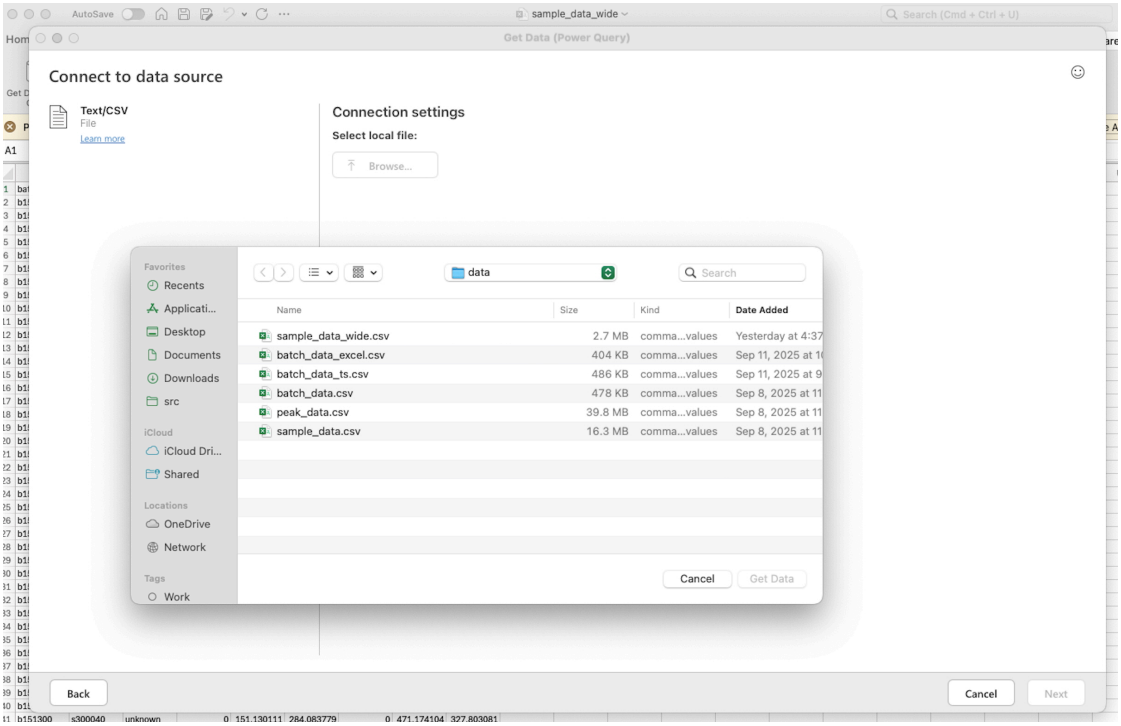
2. Navigate to the Data menu in Excel and click the down arrow beside the Get Data button.



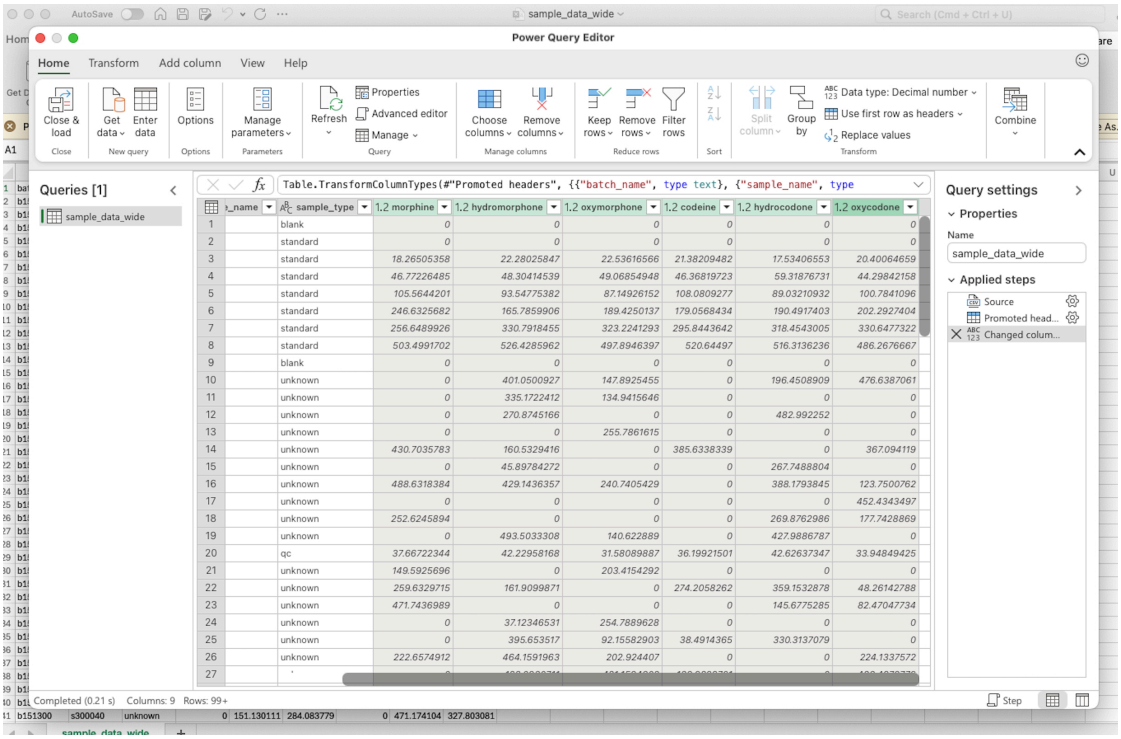
	A	B	C	D	E	F	G	H	I
1	batch_name	sample_nam	sample_type	morphine	hydromorphc	oxymorphone	codeine	hydrocodone	oxycodone
2	b151300	s300001	blank	0	0	0	0	0	0
3	b151300	s300002	standard	0	0	0	0	0	0
4	b151300	s300003	standard	18.2650536	22.2802585	22.5361657	21.3820948	17.5340655	20.4006466
5	b151300	s300004	standard	46.7722649	48.3041454	49.0685495	46.3681972	59.3187673	44.2984216
6	b151300	s300005	standard	105.56442	93.5477538	87.1492615	108.080928	89.0321093	100.78411
7	b151300	s300006	standard	246.632568	165.785991	189.425014	179.056843	190.49174	202.29274
8	b151300	s300007	standard	256.648993	330.791846	323.224129	295.844364	318.454301	330.647732
9	b151300	s300008	standard	503.49917	526.428596	497.89464	520.64497	516.313624	486.267667
10	b151300	s300009	blank	0	0	0	0	0	0
11	b151300	s300010	unknown	0	401.050093	147.892546	0	196.450891	476.638706

3. In the Choose data source menu select Text/CSV.

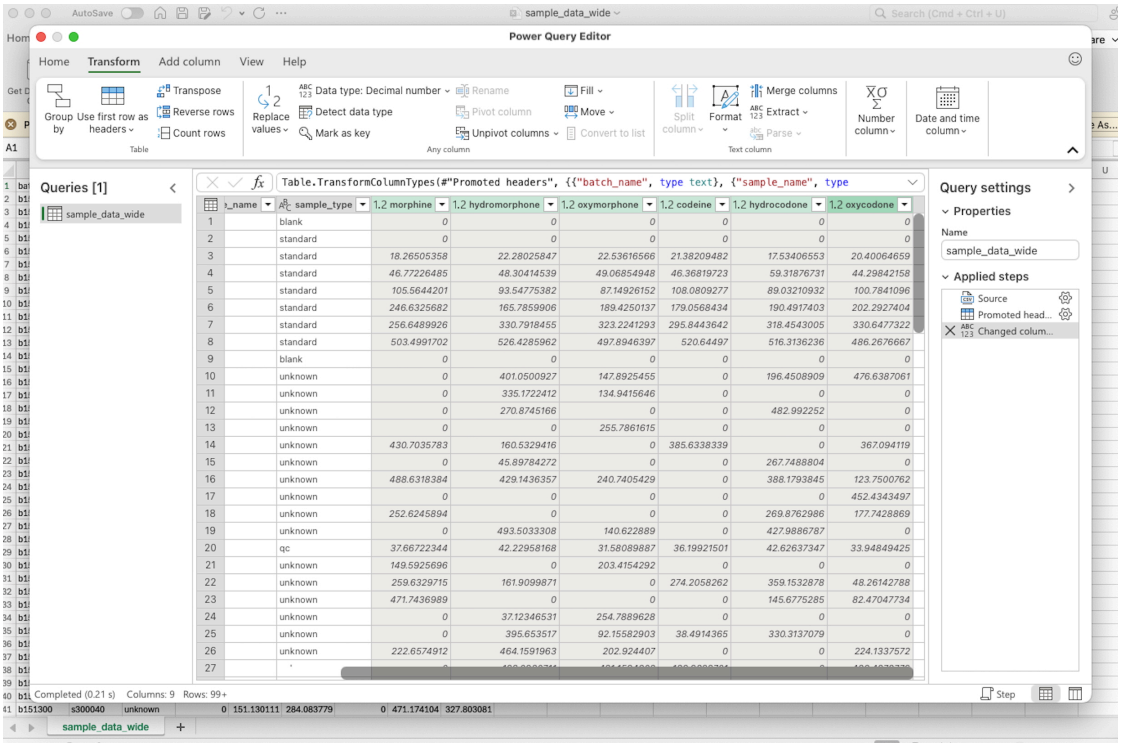
4. Click the Browse button and select **sample_data_wide.csv**, then click the Get Data button.



5. Hit the Next button and you should see your data set on the next page. Excel provides some information about the file and contents above and shows an indicator of the data types for each column by the column name (text vs. numeric).

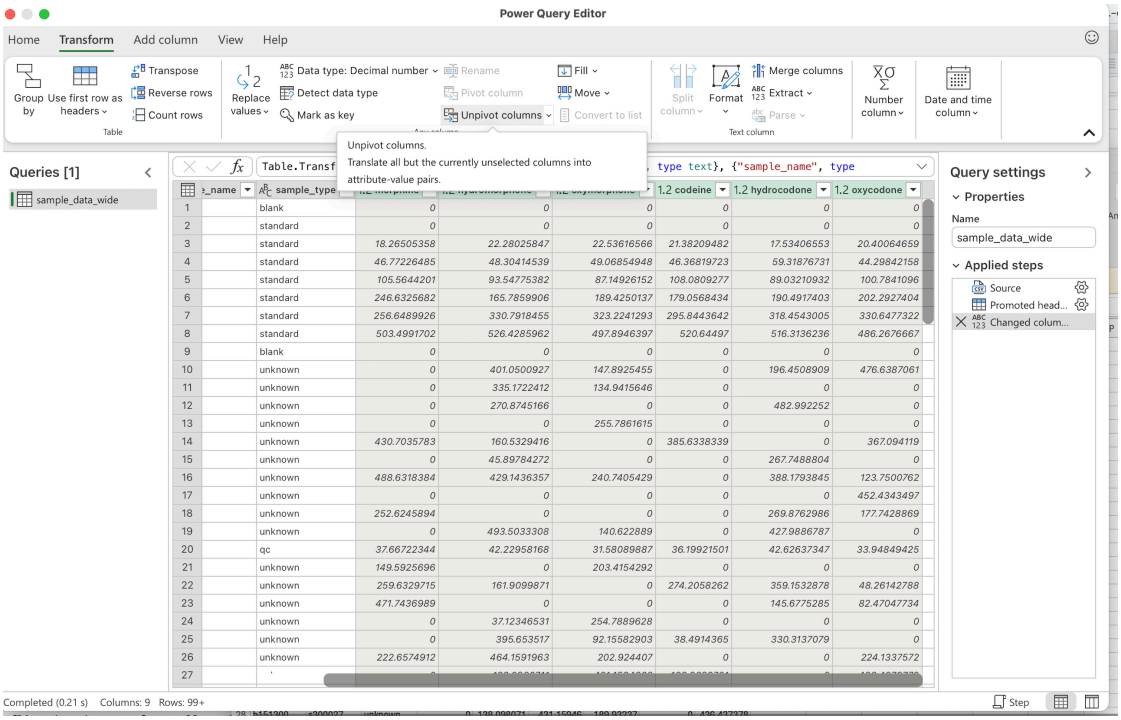


6. Hit the Transform data button to bring up a different menu. You are now within the Power Query Editor, which allows for some transformations.



7. Click the morphine column, hold the shift button, and then click the final oxycodone column. All 6 analyte columns should be highlighted.

8. Click the Unpivot columns button. The data in the window should change and you should now see one concentration value for a single analyte per row.



9. Navigate to the Home menu and hit the Close & load button to load the data into an Excel worksheet. Note how long this takes to complete.

Data Transformation in R

R and other programming languages such as Python that are intended to transform data are efficient at these types of operations. The equivalent operations of loading and viewing the data can be performed with the following code:

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
# loads "packages" that allow the user to use functions that others have written
sample_wide <- read_csv("../data/sample_data_wide.csv")
```

```
Rows: 31148 Columns: 9
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (3): batch_name, sample_name, sample_type
dbl (6): morphine, hydromorphone, oxymorphone, codeine, hydrocodone, oxycodone

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# reads the sample_data_wide.csv file into R
head(sample_wide, 10) # displays the first 10 rows
```

```
# A tibble: 10 x 9
   batch_name sample_name sample_type morphine hydromorphone oxymorphone codeine
   <chr>      <chr>       <chr>          <dbl>         <dbl>       <dbl>   <dbl>
 1 b151300    s300001     blank            0             0           0       0
 2 b151300    s300002     standard         0             0           0       0
 3 b151300    s300003     standard        18.3          22.3        22.5    21.4
 4 b151300    s300004     standard        46.8          48.3        49.1    46.4
 5 b151300    s300005     standard       106.           93.5        87.1   108.
 6 b151300    s300006     standard       247.          166.        189.    179.
 7 b151300    s300007     standard       257.          331.        323.    296.
 8 b151300    s300008     standard       503.          526.        498.    521.
 9 b151300    s300009     blank            0             0           0       0
10 b151300    s300010     unknown          0           401.        148.      0
# i 2 more variables: hydrocodone <dbl>, oxycodone <dbl>
```

Data transformation into a tidy data set requires only one function:

```
sample <- pivot_longer(sample_wide,
                       cols = morphine:oxycodone,
                       names_to = "analyte",
                       values_to = "concentration")
head(sample, 10) # view the first 10 lines of the new dataframe
```

```
# A tibble: 10 x 5
   batch_name sample_name sample_type analyte       concentration
```

	<chr>	<chr>	<chr>	<chr>	<dbl>
1	b151300	s300001	blank	morphine	0
2	b151300	s300001	blank	hydromorphone	0
3	b151300	s300001	blank	oxymorphone	0
4	b151300	s300001	blank	codeine	0
5	b151300	s300001	blank	hydrocodone	0
6	b151300	s300001	blank	oxycodone	0
7	b151300	s300002	standard	morphine	0
8	b151300	s300002	standard	hydromorphone	0
9	b151300	s300002	standard	oxymorphone	0
10	b151300	s300002	standard	codeine	0