Conducting t-tests in R

Avalon C.S. Owens, Eric R. Scott

10/5/2018

Setup for today

- Install the `tidyr` package
- Download `Example 1.csv` from Canvas
- Open a new notebook, save it, and load in these .csv files

```r
# For example:
example1 <- read.csv("Example 1.csv")
```

Overview

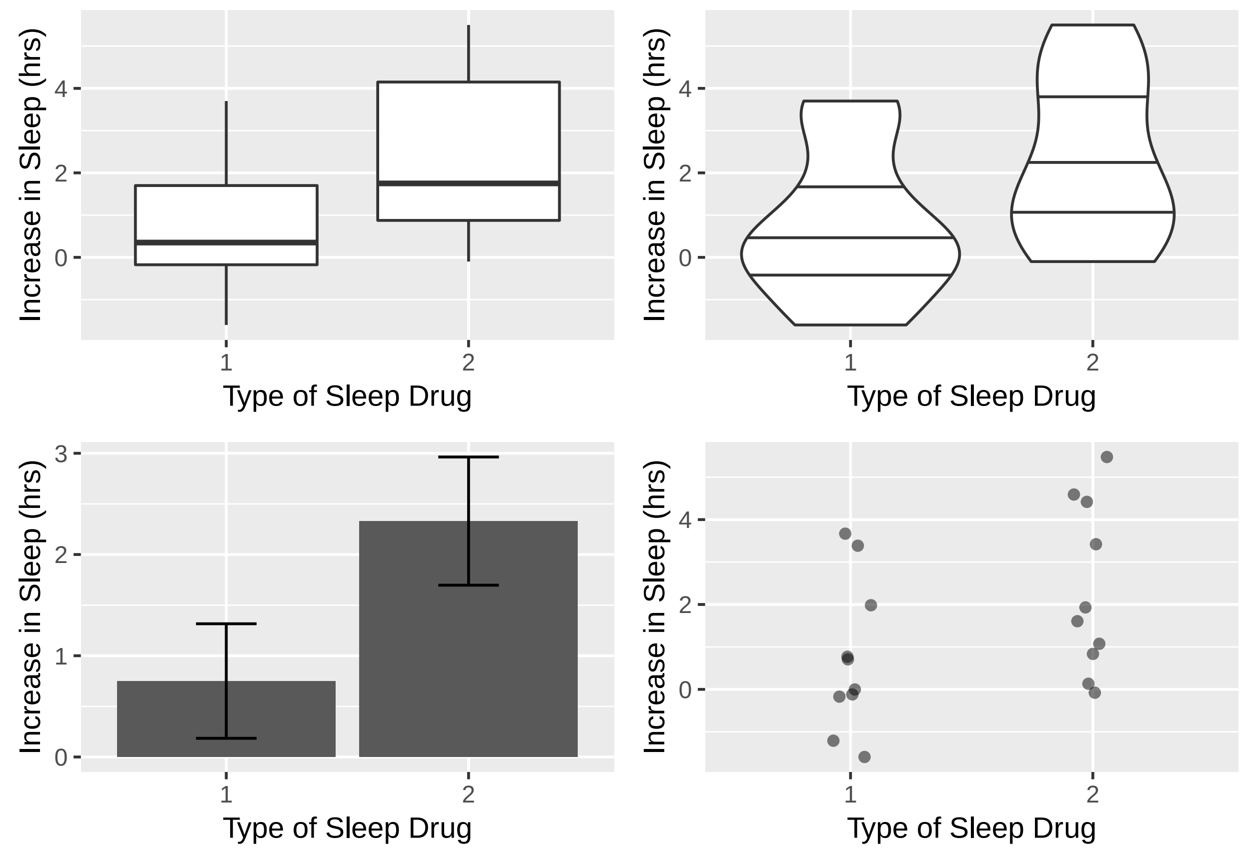
- More fun with `ggplot2`
- Tidying up your data
- One sample t-test
- Two sample t-test
    - x, y interface
    - formula interface
- Formulas in R

Plotting comparisons of group means

More fun with `ggplot2`

- Two-sample t-tests ask if two group means came from different distributions
- So, plots to go along with t-tests should show means and some measure of spread or distribution
- What plots have we learned so far that do this?

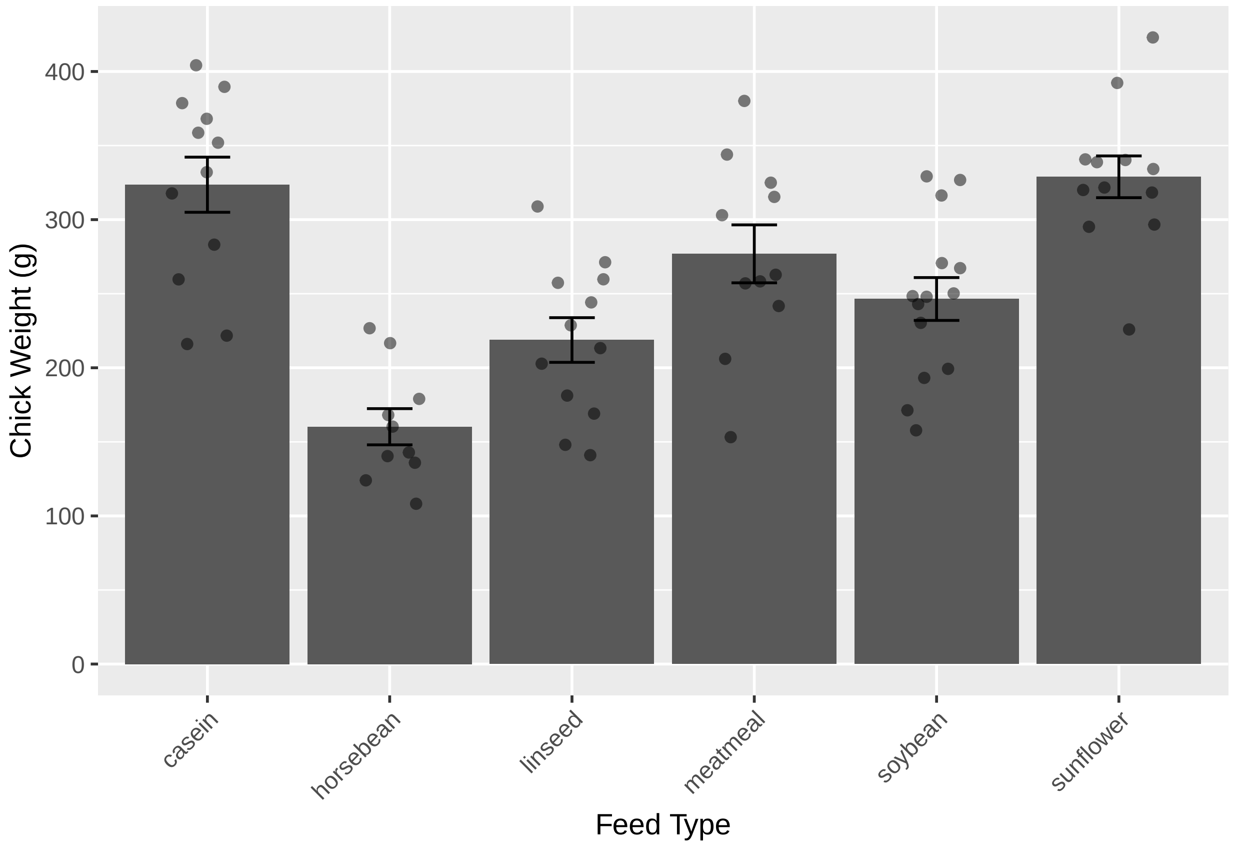
More fun with `ggplot2`



Making bar plots better

- Bar plots on their own do a poor job of displaying data, even with error bars
- Adding the actual data points on top can improve them

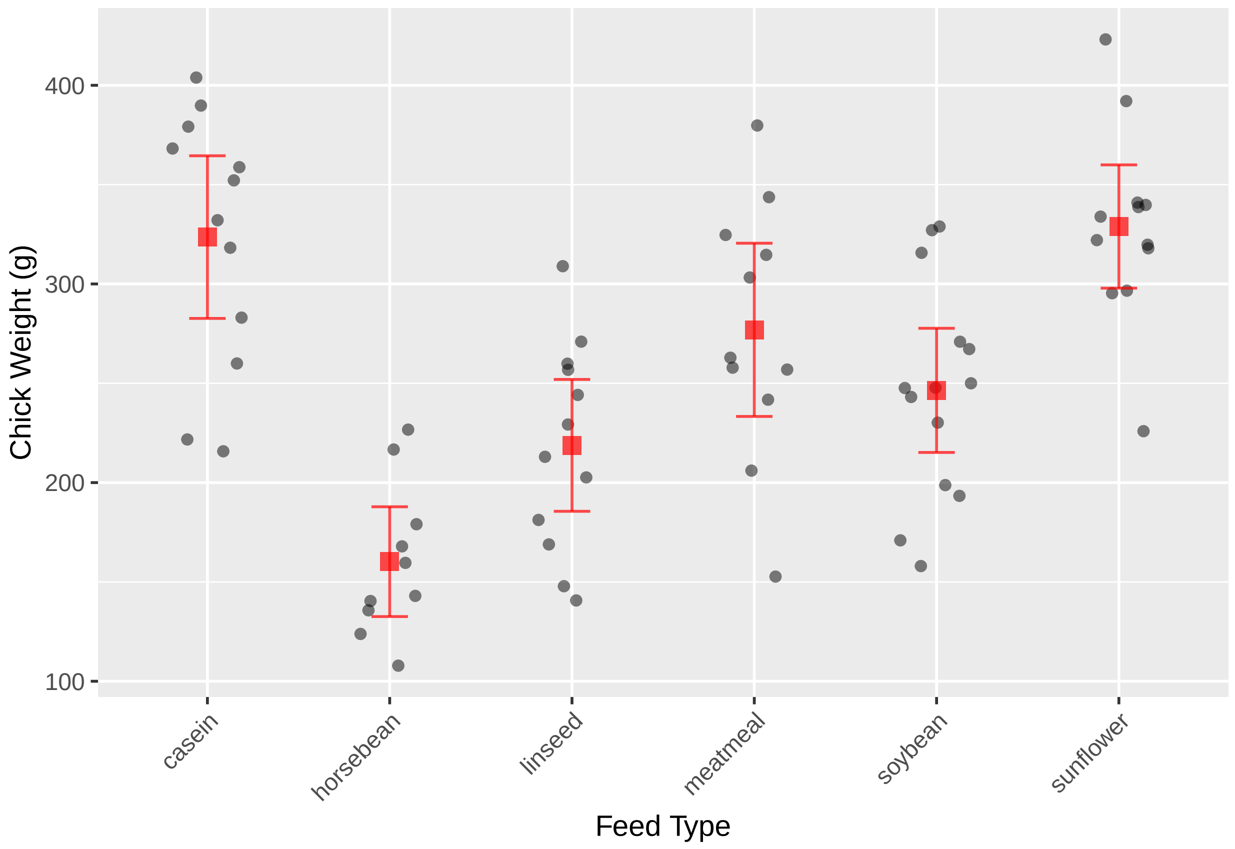
LIVE CODING



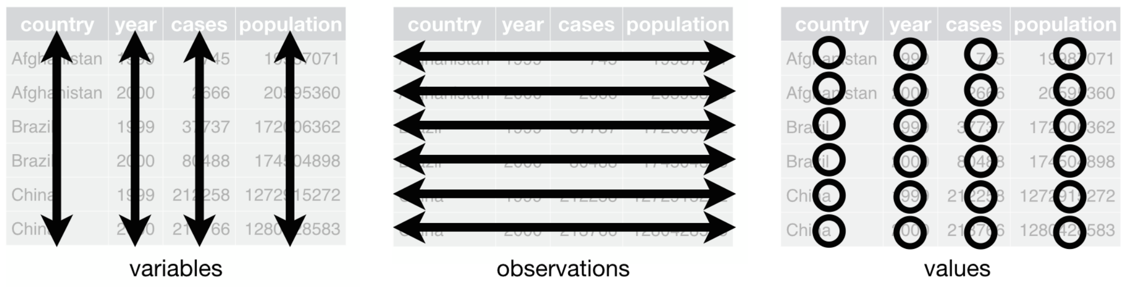
Ditch the bar!

- The height of the bar in a bar plot **just represents the mean**.
- Bar area can be **misleading** because it **doesn't** represent spread in any way.
- You *could* get rid of bars entirely and just plot means as a point!
- This is great for when your data values are not all > 0 and you have few data points.

LIVE CODING



Tidying up your data



There are many ways to represent data, but only some are useful

Data formats

- All the data we've given you so far has been *curated* to make it friendly
- There are many ways to record data—not always in the right format for R!

Example 1:

You want to know if two common tea cultivars, Tie Guan Yin (TGY) and Long Jing (LJ), are equally susceptible to attack by the tea green leafhopper (*Empoasca onukii*).

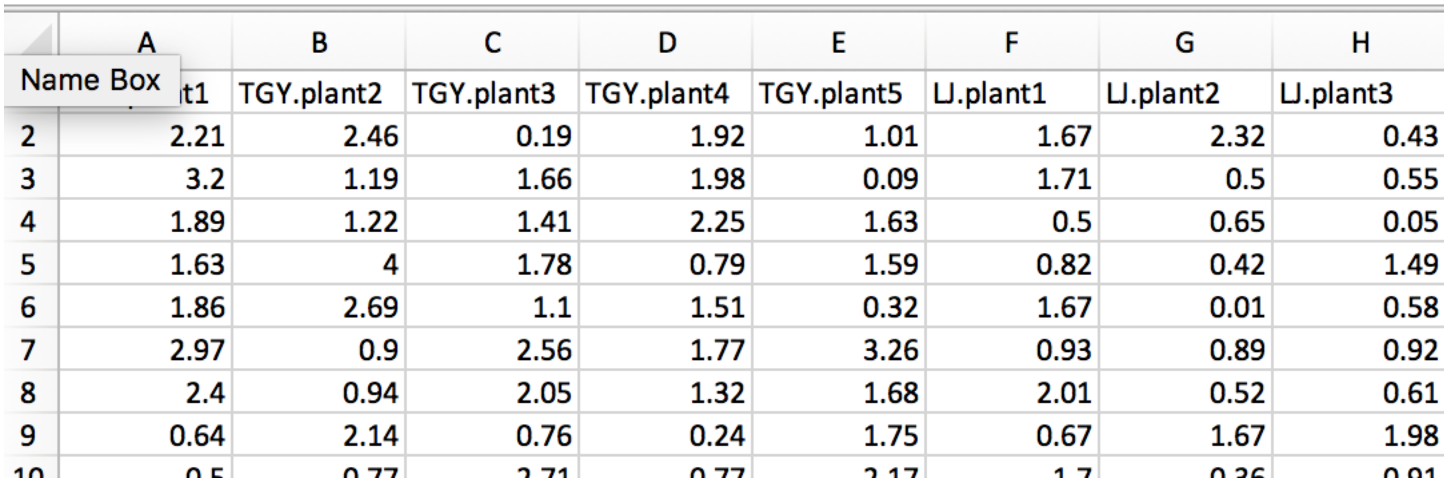
Example 1:

You randomly sample 5 plants of each cultivar and measure the amount of damage by collecting 20 leaves on each plant, scanning them, and using image analysis software to count the percentage of pixels with brown spots.

Many possible data formats

There are many ways you might have recorded the data

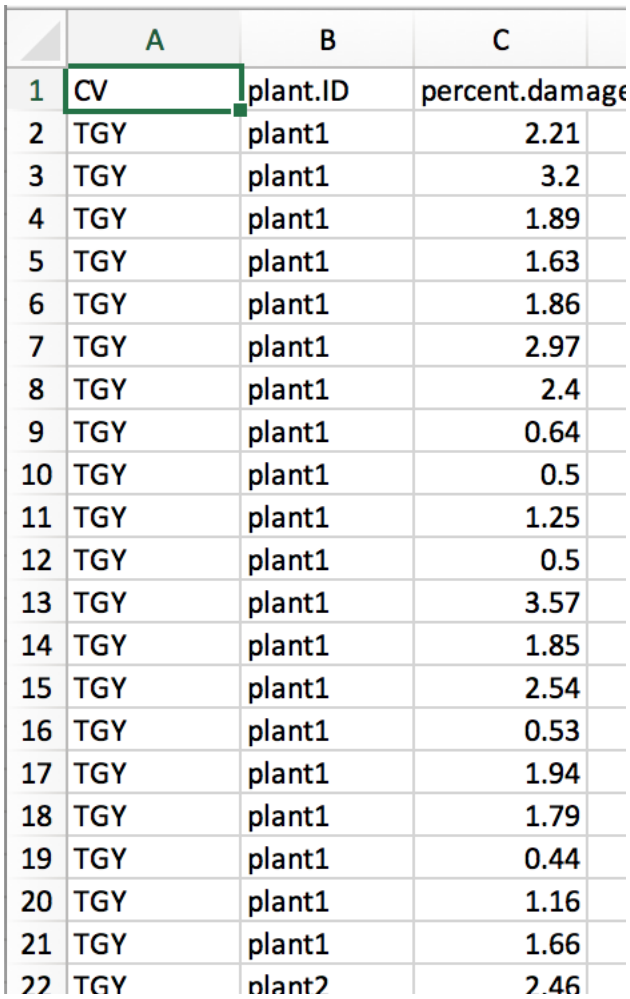
For example, with a separate column for every plant:



	A	B	C	D	E	F	G	H
	t1	TGY.plant2	TGY.plant3	TGY.plant4	TGY.plant5	LJ.plant1	LJ.plant2	LJ.plant3
2	2.21	2.46	0.19	1.92	1.01	1.67	2.32	0.43
3	3.2	1.19	1.66	1.98	0.09	1.71	0.5	0.55
4	1.89	1.22	1.41	2.25	1.63	0.5	0.65	0.05
5	1.63	4	1.78	0.79	1.59	0.82	0.42	1.49
6	1.86	2.69	1.1	1.51	0.32	1.67	0.01	0.58
7	2.97	0.9	2.56	1.77	3.26	0.93	0.89	0.92
8	2.4	0.94	2.05	1.32	1.68	2.01	0.52	0.61
9	0.64	2.14	0.76	0.24	1.75	0.67	1.67	1.98
10	0.5	0.77	2.71	0.77	2.17	1.7	0.36	0.91

Many possible data formats

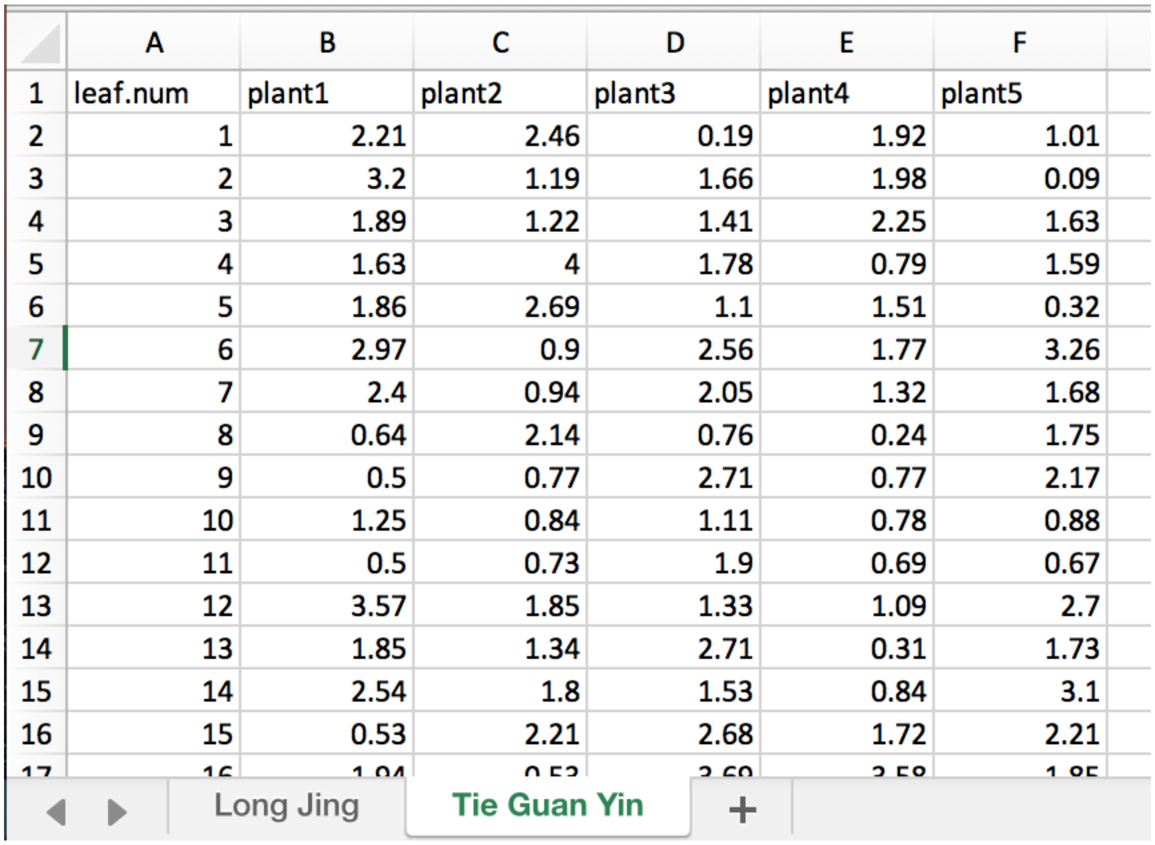
Or with just three columns:



	A	B	C
1	CV	plant.ID	percent.damage
2	TGY	plant1	2.21
3	TGY	plant1	3.2
4	TGY	plant1	1.89
5	TGY	plant1	1.63
6	TGY	plant1	1.86
7	TGY	plant1	2.97
8	TGY	plant1	2.4
9	TGY	plant1	0.64
10	TGY	plant1	0.5
11	TGY	plant1	1.25
12	TGY	plant1	0.5
13	TGY	plant1	3.57
14	TGY	plant1	1.85
15	TGY	plant1	2.54
16	TGY	plant1	0.53
17	TGY	plant1	1.94
18	TGY	plant1	1.79
19	TGY	plant1	0.44
20	TGY	plant1	1.16
21	TGY	plant1	1.66
22	TGY	plant2	2.46

Many possible data formats

Or using separate spreadsheets or pages for each cultivar:



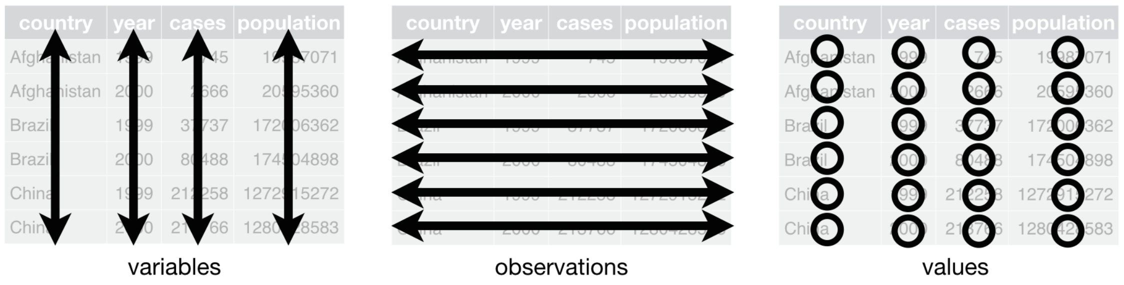
	A	B	C	D	E	F
1	leaf.num	plant1	plant2	plant3	plant4	plant5
2	1	2.21	2.46	0.19	1.92	1.01
3	2	3.2	1.19	1.66	1.98	0.09
4	3	1.89	1.22	1.41	2.25	1.63
5	4	1.63	4	1.78	0.79	1.59
6	5	1.86	2.69	1.1	1.51	0.32
7	6	2.97	0.9	2.56	1.77	3.26
8	7	2.4	0.94	2.05	1.32	1.68
9	8	0.64	2.14	0.76	0.24	1.75
10	9	0.5	0.77	2.71	0.77	2.17
11	10	1.25	0.84	1.11	0.78	0.88
12	11	0.5	0.73	1.9	0.69	0.67
13	12	3.57	1.85	1.33	1.09	2.7
14	13	1.85	1.34	2.71	0.31	1.73
15	14	2.54	1.8	1.53	0.84	3.1
16	15	0.53	2.21	2.68	1.72	2.21
17	16	1.04	0.53	3.69	3.58	1.85

Tidy data

Not all of these data frames are easy to use in R.

TIDY DATA RULES:

1. Each variable (e.g. measurement, treatment) must have its own column
2. Each observation (e.g. individual) must have its own row
3. Each value must have its own cell



Why be tidy?

- R, specifically `dplyr` and `ggplot2`, are designed to work with tidy data
- It is good practice to have a consistent format for all your data frames

BUT

Sometimes data is **untidy** because:

- It's convenient to enter it into a spreadsheet a different way
- Someone who doesn't know R or tidy data gives it to you
- An instrument outputs its data in an odd format
- A particular type of data has a different convention for formatting

Tidying untidy data

You *could* do it in Excel, but…

- It's easy to make errors
- It can be time-consuming (lots of copying and pasting!)
- It doesn't scale well (hard to do if you have thousands of rows)
- `tidyr` provides functions to do data tidying in R!

Intro to data tidying

Step 1: figure out what the variables and observations are

```
df1 <- read.csv("Example 1.csv")
head(df1, 3)
```

```
##   TGY.plant1 TGY.plant2 TGY.plant3 TGY.plant4 TGY.plant5 LJ.plant1
## 1       2.21       2.46       0.19       1.92       1.01      1.67
## 2       3.20       1.19       1.66       1.98       0.09      1.71
## 3       1.89       1.22       1.41       2.25       1.63      0.50
##   LJ.plant2 LJ.plant3 LJ.plant4 LJ.plant5
## 1      2.32      0.43      0.96      0.51
## 2      0.50      0.55      0.86      0.03
## 3      0.65      0.05      0.05      1.66
```

- How many variables? How many observations?

Converting "wide" data to tidy data with `gather()`

- The column names in the previous slide represent *values*, **not** the names of variables
- `gather()` turns column names into a variable called `key` and values into a variable called `value`

```
gather(df1) %>% head()
```

```
##          key value
## 1 TGY.plant1  2.21
## 2 TGY.plant1  3.20
## 3 TGY.plant1  1.89
## 4 TGY.plant1  1.63
## 5 TGY.plant1  1.86
## 6 TGY.plant1  2.97
```

Gathering

- Give the new columns names with `key =` and `value =`

```
df1.a <- df1 %>%
  gather(key = "cultivar.plant", value = "percent_damage")

head(df1.a)
```

```
##   cultivar.plant percent_damage
## 1     TGY.plant1           2.21
## 2     TGY.plant1           3.20
## 3     TGY.plant1           1.89
## 4     TGY.plant1           1.63
## 5     TGY.plant1           1.86
## 6     TGY.plant1           2.97
```

Separating

- Better, but this still isn't tidy. Why?

```
head(df1.a)
```

```
##   cultivar.plant percent_damage
## 1     TGY.plant1           2.21
## 2     TGY.plant1           3.20
## 3     TGY.plant1           1.89
## 4     TGY.plant1           1.63
## 5     TGY.plant1           1.86
## 6     TGY.plant1           2.97
```

Separating

- Our first column was two variables together. Let's `separate()` them!

```
df.fin <- df1.a %>%
  separate(cultivar.plant, into = c("cultivar", "plantID"))

head(df.fin)
```

```
##   cultivar plantID percent_damage
## 1      TGY  plant1           2.21
## 2      TGY  plant1           3.20
## 3      TGY  plant1           1.89
## 4      TGY  plant1           1.63
## 5      TGY  plant1           1.86
## 6      TGY  plant1           2.97
```

Tidy yet?

- Now the data are tidy!

- Can I now use these data to do a t-test? Why or why not?

```r
sample_n(df.fin, 6)
```

```
##     cultivar plantID percent_damage
## 189       LJ  plant5           1.93
## 34       TGY  plant2           1.80
## 192       LJ  plant5           0.07
## 197       LJ  plant5           1.11
## 36       TGY  plant2           0.53
## 38       TGY  plant2           1.52
```

Conducting t-tests with `t.test()`

One-sample `t.test()`

- Is the distribution of `WolfTeeth` lengths from a population with a mean of 12 cm?

```r
library(abd)
t.test(WolfTeeth$length, mu = 12)
##
##  One Sample t-test
##
## data:  WolfTeeth$length
## t = -29.909, df = 34, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 12
## 95 percent confidence interval:
##  10.20890 10.43681
## sample estimates:
## mean of x
##  10.32286
```

- Displays H_A, t, degrees of freedom (df), and a p-value
- Hey, look at that! It also calculates a 95% confidence interval!

Interpret the output to answer the biological question!

Two-sample `t.test()`

There are two ways to use `t.test()` to do a two-sample t-test

1. "x, y" interface

```r
t.test(<<group 1 data>>, <<group 2 data>>, <<other arguments>>)
```

2. "formula" interface

```r
t.test(response_var ~ grouping_var,
       data = <<data frame object>>, <<other arguments>>)
```

Tidy two-sample `t.test()`

- If your data are tidy, it will probably be easiest to use the **formula interface**.
- In R, formulas always have a "~" in them.

- These are **not** mathematical formulas, but a way of telling R how variables are related.

Formulas in R

```r
my.formula <- percent.damage ~ cultivar
str(my.formula)
```

```
## Class 'formula'  language percent.damage ~ cultivar
##   ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
```

Read as "`percent.damage` as explained by `cultivar`" or "`percent.damage` distributed as `cultivar`"

Tidy two-sample `t.test()`

`ToothGrowth` is a built-in, tidy dataset measuring the effect of vitamin C supplements on guinea pig tooth growth

```r
str(ToothGrowth)
```

```
## 'data.frame':    60 obs. of  3 variables:
##  $ len : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
##  $ supp: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
##  $ dose: num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
```

Tidy two-sample `t.test()`

Does the type of vitamin C supplement affect tooth length?

```r
t.test(len ~ supp, data = ToothGrowth)
```

```
## 
##  Welch Two Sample t-test
## 
## data:  len by supp
## t = 1.9153, df = 55.309, p-value = 0.06063
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1710156  7.5710156
## sample estimates:
## mean in group OJ mean in group VC
##         20.66333         16.96333
```

What does this confidence interval mean?

Un-tidy two-sample `t.test`

- Takes two vectors (and no `data=` argument)
- This way of using `t.test()` will also be important for paired t-tests (coming next week!)

```r
groupA <- rnorm(n = 10, mean = 1, sd = 1)
groupB <- rnorm(10, 1.3, 1)
```

```r
t.test(groupA, groupB)
```

```
##
##  Welch Two Sample t-test
##
## data:  groupA and groupB
## t = -1.3967, df = 17.933, p-value = 0.1795
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.330579  0.268052
## sample estimates:
## mean of x mean of y
## 0.9940239 1.5252877
```

Un-tidy two-sample `t.test`

If you have an un-tidy data frame, you can use the `$` operator to access individual columns

```
t.df <- data.frame(groupA, groupB)
head(t.df, 3)
```

```
##       groupA    groupB
## 1  2.5234902 1.9891441
## 2  0.6437127 2.0950239
## 3 -0.3925080 0.9392797
```

```
t.test(t.df$groupA, t.df$groupB)
```

```
##
##  Welch Two Sample t-test
##
## data:  t.df$groupA and t.df$groupB
## t = -1.3967, df = 17.933, p-value = 0.1795
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.330579  0.268052
## sample estimates:
## mean of x mean of y
## 0.9940239 1.5252877
```

Equal and unequal variance t-tests

- By default, `t.test` assumes unequal variances.
- To do an equal variance t-test, add the argument `var.equal = TRUE`

```
t.test(len ~ supp, data = ToothGrowth, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  len by supp
## t = 1.9153, df = 58, p-value = 0.06039
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1670064  7.5670064
## sample estimates:
```

```
## mean in group OJ mean in group VC
##         20.66333         16.96333
```

Alternative hypotheses

- By default, the alternative hypothesis is that $\delta \neq 0$. You can change this with `mu =`
- Default is a two-tailed test, change with `alternative = "greater"` or `alternative = "less"`
- What hypothesis is this testing?

```
t.test(len ~ supp, data = ToothGrowth,
       mu = 1,
       alternative = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  len by supp
## t = 1.3976, df = 55.309, p-value = 0.0839
## alternative hypothesis: true difference in means is greater than 1
## 95 percent confidence interval:
##  0.4682687       Inf
## sample estimates:
## mean in group OJ mean in group VC
##         20.66333         16.96333
```