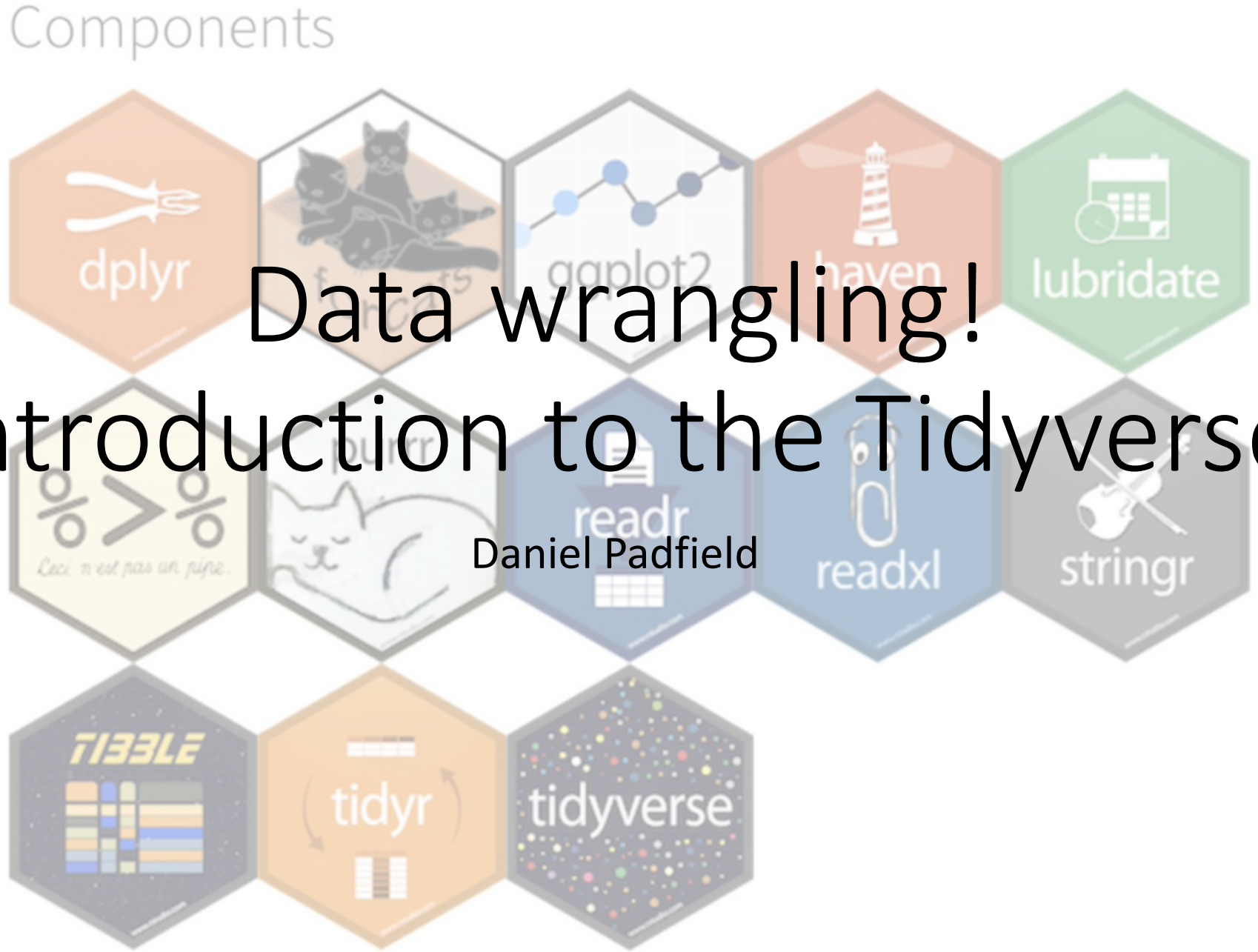
Data wrangling!
Introduction to the Tidyverse

Daniel Padfield



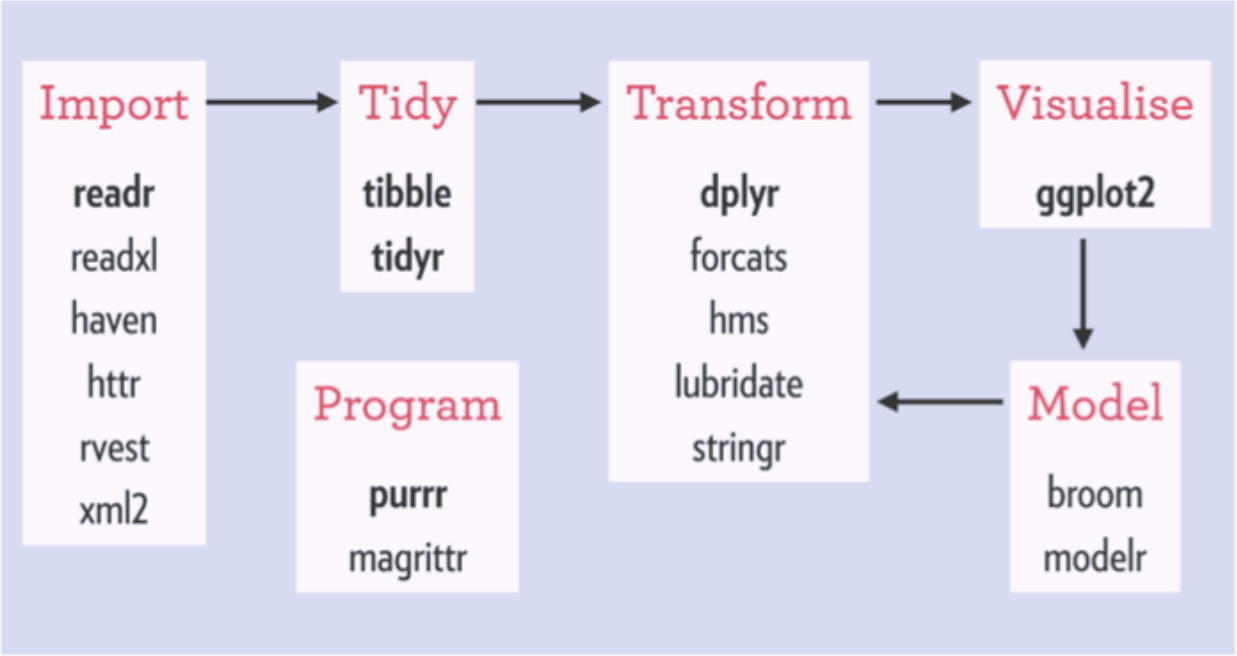
What is the tidyverse?

- Set of packages designed by Hadley Wickham and his team at Rstudio
- Try to create a set of tools for full-stack analysis that use the same syntax and set of rules
- Install the complete tidyverse using:
  - `install.packages("tidyverse")`

Components

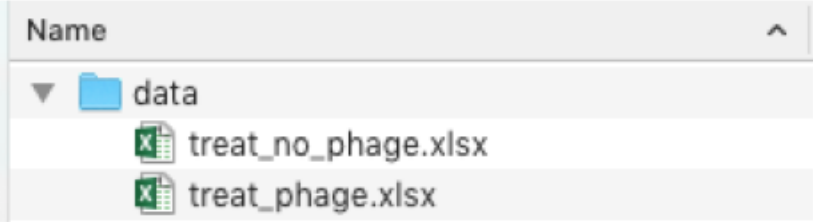
What is the tidyverse?

- Can do almost all my programming with the same set of packages
- Makes my code more readable so future me can understand
- In general, drops the need for $ and “ ”
- Great documentation and support online



Read in files with purrr and readxl

- `read_excel()` handles excel files
- `map_df()` takes a vector a vector of file names and returns a dataframe



treat_phage.xlsx

Treatment	CLONE	10 degrees	15 degrees	20 degrees
phage	1	75	85	88
phage	2	64	64	74
phage	3	68	68	70
phage	4	89	94	99
phage	5	110	115	116
phage	6	48	52	57
phage	7	69	69	69
phage	8	140	141	151
phage	9	100	110	117
phage	10	99	107	107

Read in files with purrr and readxl

```
# list files
files <- list.files('data', full.names = TRUE)
# read in many files using map() and read_excel()
d <- map_df(files, read_excel)
```

	Treatment	CLONE	10 degrees	15 degrees	20 degrees
1	no phage	1	56	73	109
2	no phage	2	131	145	202
3	no phage	3	127	141	188
4	no phage	4	91	111	158
5	no phage	5	114	128	162
6	no phage	6	165	180	223
7	no phage	7	77	92	136
8	no phage	8	168	186	246
9	no phage	9	98	116	155
10	no phage	10	45	56	113
11	phage	1	75	78	82
12	phage	2	64	70	73
13	phage	3	68	71	71
14	phage	4	89	95	103
15	phage	5	110	111	115
16	phage	6	48	53	53
17	phage	7	69	78	82
18	phage	8	140	146	147
19	phage	9	100	100	105
20	phage	10	99	100	102

Tidy column names with janitor

- Writing code, odd column names can often be confusing
- No CAPITALS, numbers as first letter or spaces
- `janitor::clean_names()` automatically renames columns to be all lower case and replaces spaces with "_", puts an X in front of the first number
- Can do one at a time using:

```
dplyr::rename(d, new_column_name = old_column name)
```

Tidy column names with janitor

```
# change column names with janitor
d <- clean_names(d)

# or a single column with rename
d <- rename(d, clone = CLONE)
```

- From this…

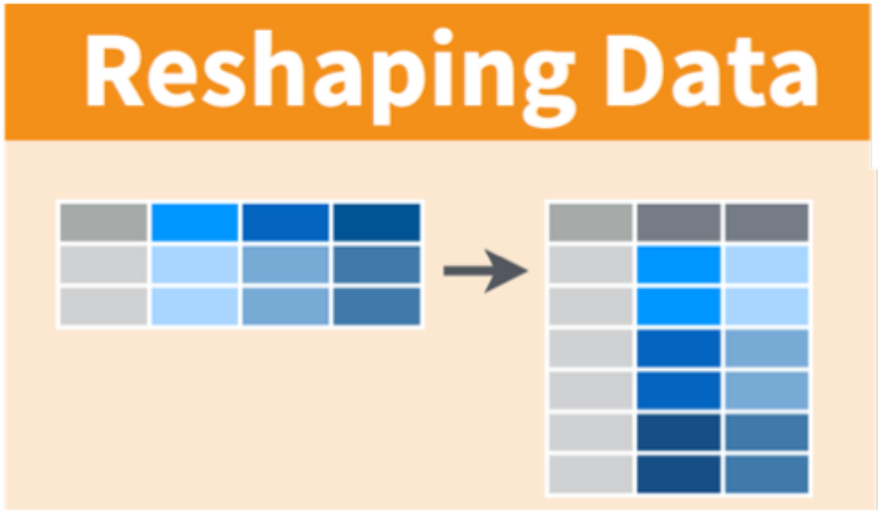
Treatment 165	CLONE	10 degrees	15 degrees	20 degrees
no phage	1	56	73	109
no phage	2	131	145	202
no phage	3	127	141	188
no phage	4	91	111	158
no phage	5	114	128	162
no phage	6	165	180	223
no phage	7	77	92	136
no phage	8	168	186	246
no phage	9	98	116	155
no phage	10	45	56	113

- To this…

treatment	clone	x10_degrees	x15_degrees	x20_degrees
no phage	1	56	73	109
no phage	2	131	145	202
no phage	3	127	141	188
no phage	4	91	111	158
no phage	5	114	128	162
no phage	6	165	180	223
no phage	7	77	92	136
no phage	8	168	186	246
no phage	9	98	116	155
no phage	10	45	56	113

Stack data (i.e. long format) using gather()

- Lots of packages (`ggplot2`, `lme4` etc) need data to be in long format
- `tidyr::gather()` combines multiple columns into key value pairs
- Similar to `reshape2::melt()`



Stack data (i.e. long format) using gather()

```
# convert data to long format using gather
d <- gather(d, key = 'temp',
               value = 'count',
               contains('degree'))
```

- From this…

treatment	clone	x10_degrees	x15_degrees	x20_degrees
86 phage	1	56	73	109
no phage	2	131	145	202
no phage	3	127	141	188
no phage	4	91	111	158
no phage	5	114	128	162
no phage	6	165	180	223
no phage	7	77	92	136
no phage	8	168	186	246
no phage	9	98	116	155
no phage	10	45	56	113
phage	1	75	78	82
phage	2	64	70	73
phage	3	68	71	71
phage	4	89	95	103
phage	5	110	111	115
phage	6	48	53	53
phage	7	69	78	82
phage	8	140	146	147
phage	9	100	100	105
phage	10	99	100	102

- To this…

treatment	clone	temp	count
p/Work/R_teaching/intro_2_tidyverse/ dyverse.R		ees	45
phage	1	x10_degrees	75
phage	2	x10_degrees	64
phage	3	x10_degrees	68
phage	4	x10_degrees	89
phage	5	x10_degrees	110
phage	6	x10_degrees	48
phage	7	x10_degrees	69
phage	8	x10_degrees	140
phage	9	x10_degrees	100
phage	10	x10_degrees	99
no phage	1	x15_degrees	73
no phage	2	x15_degrees	145
no phage	3	x15_degrees	141
no phage	4	x15_degrees	111
no phage	5	x15_degrees	128
no phage	6	x15_degrees	180
no phage	7	x15_degrees	92
no phage	8	x15_degrees	186

Select columns with select()

- Can select or deselect columns
- Via position

```
select(d, c(1,3,4))
```

- Via name

```
select(d, c(clone, treatment, temp, count))
```

- Drop columns

```
select(d, -c(clone, treatment))
```

- Helper functions

```
select(d, c(clone, everything()))
select(d, starts_with('t'))
```

`?select_helpers` for all functions

Add columns with mutate()

- Create new columns with mutate()

```
# change and add columns
d <- mutate(d, temp = readr::parse_number(temp),
               temp = as.numeric(temp),
               log_count = log10(count))
```

treatment	clone	temp	count	log_count
no phage	1	10	56	1.748188
no phage	2	10	131	2.117271
no phage	3	10	127	2.103804
no phage	4	10	91	1.959041
no phage	5	10	114	2.056905
no phage	6	10	165	2.217484
no phage	7	10	77	1.886491
no phage	8	10	168	2.225309
no phage	9	10	98	1.991226

Do the same function on many columns using mutate_at()

- We can do the same function on multiple columns using mutate_at()
- Change character or numeric columns to a factor

```
# change multiple columns using mutate_at()
d <- mutate_at(d, c('treatment', 'clone'), as.factor)
```

```
d                    60 obs. of 5 variables
  treatment: chr "no phage" "no phage" "no phage" "no phage" …
  clone : num 1 2 3 4 5 6 7 8 9 10 ...
```

Do the same function on many columns using mutate_at()

- Do the same function on multiple columns using mutate_at()
- Change character or numeric columns to a factor

```
# change multiple columns using mutate_at()
d <- mutate_at(d, c('treatment', 'clone'), as.factor)
```

```
d                    60 obs. of 5 variables
  treatment: Factor w/ 2 levels "no phage","phage": 1 1 1 1 1…
  clone : Factor w/ 10 levels "1","2","3","4",..: 1 2 3 4 5 6…
```

No more nested if_else() with case_when()

- Different dilution factors
  - 10 ºC = 10^{-5}, 15 ºC = 10^{-4}, 20 ºC = 10^{-6}
- Different bacteria
  - Clones 1-5 = LacZ, Clones 6-10 = wild type
- Common to want to add columns based on other conditions

No more nested if_else() with case_when()

- Different dilution factors
  - 10 ºC = 10^{-5}, 15 ºC = 10^{-4}, 20 ºC = 10^{-6}
- Different bacteria
  - Clones 1-5 = LacZ, Clones 6-10 = wild type

```
# using ifelse() and mutate()
d <- mutate(d, type = ifelse(clone <= 5, 'lacz', 'wt’),
            df = ifelse(temp == 10, 10^-5,
                        ifelse(temp == 15, 10^-4,
                               10^-6)))
```

No more nested if_else() with case_when()

- Different dilution factors
  - 10 ºC = 10^{-4}, 15 ºC = 10^{-5}, 20 ºC = 10^{-6}
- Different bacteria
  - Clones 1-5 = LacZ, Clones 6-10 = wild type

```
# using ifelse() and mutate()
d <- mutate(d, type = ifelse(clone <= 5, 'lacz', 'wt’),
            df = ifelse(temp == 10, 10^-4,
                        ifelse(temp == 15, 10^-5,
                               10^-6)))

# using case_when() and mutate()
d <- mutate(d, type = ifelse(clone <= 5, 'lacz', 'wt’),
            df = case_when(temp == 10 ~ 10^-4,
                           temp == 15 ~ 10^-5,
                           temp == 20 ~ 10^-6))
```

Subset rows with filter()

- Similar to subset()
- removes rows of a dataframe based on a condition

```
# examples of using filter
# keep just 15 and 20 degrees
filter(d, temp > 10)
# keep just 10 degrees LacZ
filter(d, temp == 10 & type == ‘lacz’)
```

Piping with %>%

- Read a pipe as “and then”
- Allows your code to be read like a recipe
- Removes the need to constantly re-assign objects
- The “.” operator acts as an indicator for whatever is coming from the left hand side of the pipe

Put it all together! Pipe example!

```
# Example of using pipes %>%
d <- list.files('data', full.names = TRUE) %>%
     map_df(., read_excel) %>%
     clean_names(.) %>%
     gather(., key = 'temp', value = 'count',
               contains('degree')) %>%
     mutate(., temp = readr::parse_number(temp),
               temp = as.numeric(temp),
               log_count = log10(count)) %>%
     mutate_at(., c('treatment', 'clone'),
               as.factor) %>%
     …
```

The recipe version

d <- list all the files in the “data” directory (and then)
read them into a dataframe (and then)
change their names (and then)
make the data long format (and then)
add a bunch of columns (and then)
do the same function to treatment and clone …

Summary data using group_by() and summarise()

- Get summary stats based on groupings of your data
- summarise() - get means and standard deviations, or anything that returns a single value (i.e. max, min, number of rows, mode, median)

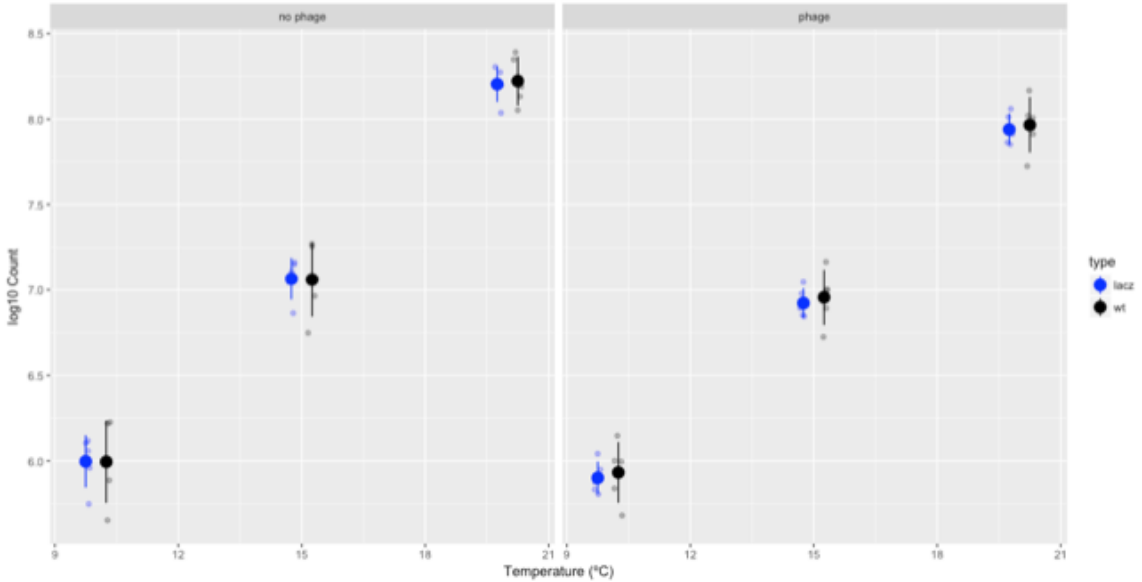
```
# get summary data
d_means <- group_by(d, type, treatment) %>%
           summarise(mean = mean(log_count),
                     sd = sd(log_count)) %>%
           ungroup()
```

type	treatment	temp	mean	sd
lacz	no phage	10	5.997042	0.15232289
lacz	no phage	15	7.065289	0.12167762
lacz	no phage	20	8.205022	0.10367582
lacz	phage	10	5.900907	0.09540223
lacz	phage	15	6.922300	0.08678401
lacz	phage	20	7.940386	0.09261839
wt	no phage	10	5.994745	0.24038944
wt	no phage	15	7.060244	0.21710026
wt	no phage	20	8.223238	0.14304463
wt	phage	10	5.932371	0.17753916
wt	phage	15	6.956145	0.16206695
wt	phage	20	7.967039	0.16320068

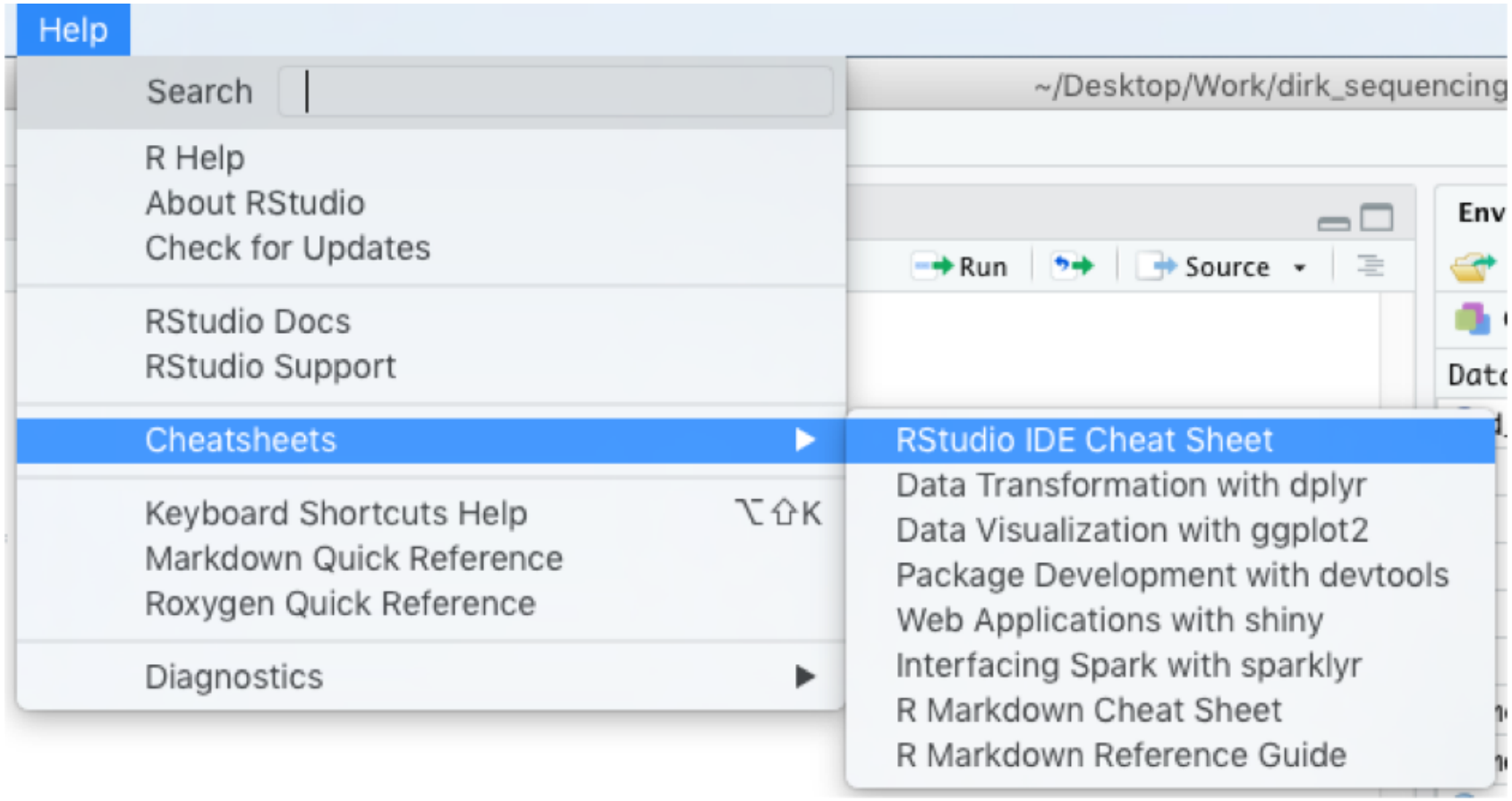
Plot with ggplot2

- Plot summary data with raw data

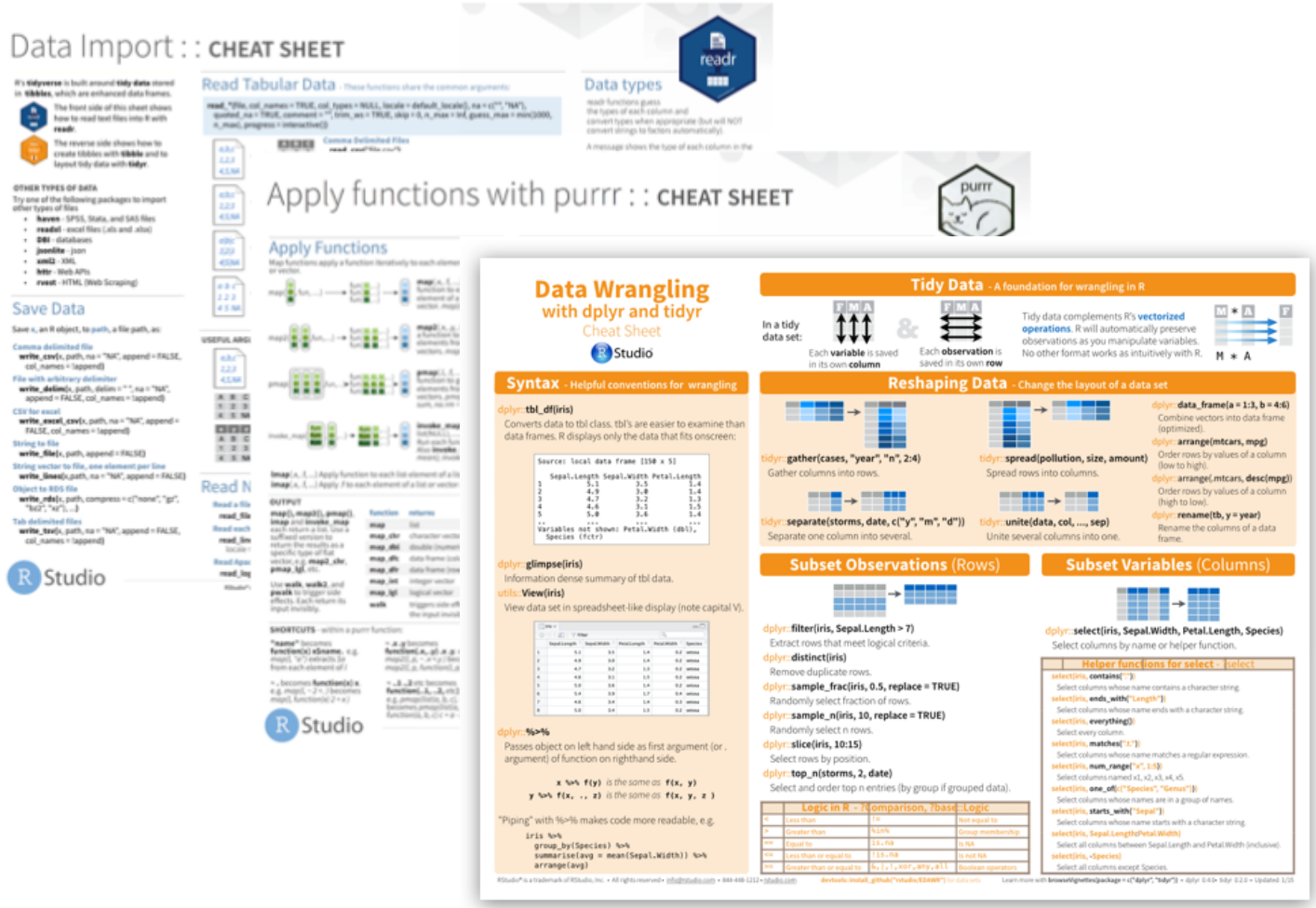
```
# plot with ggplot2
ggplot() +
    geom_linerange(aes(x = temp, ymin = mean - sd, ymax
= mean + sd, col = type, group = type), d_means,
position = position_dodge(width = 1)) +
    geom_point(aes(temp, mean, col = type), d_means,
size = 4, position = position_dodge(width = 1)) +
    geom_point(aes(temp, log_count, col = type),
position = position_jitterdodge(dodge.width = 1,
jitter.width = 0.2), alpha = 0.3, d) +
    facet_wrap(~ treatment) +
    scale_color_manual(values = c('blue', 'black')) +
    ylab('log10 Count') +
    xlab('Temperature (°C)')
```



Cheatsheets for the tidyverse



Cheatsheets for the tidyverse



Moving from base R to the tidyverse

- [Blog post link!]

Base R command	Tidyverse Command	What it does and why you should use the tidyverse version	Comment
read.csv()	read_csv()	reads in a csv file, but its much faster, shows progress bar for large files, can automatically parse data types	also see read_delim(), read_tsv() and readxl::read_xlsx()
sort(), order()	arrange()	sort column(n) within a data frame	see also order_by()
mtcars$mpg = ...	mutate()	modify a column	see also transmute() which drops existing variables
mtcars[,c("mpg", "am")], subset()	select(), rename()	select or rename columns	see also pull()

MicrobioUoE

- Set of functions I commonly use, wrote into a package

Installation

Installation of packages from GitHub is relatively straightforward.

```
# firstly need to make sure devtools is installed
install.packages('devtools')

# install MicrobioUoE
devtools::install_github('padpadpadpad/MicrobioUoE')
```

Functions

- bind_biolog_sheet()
  - cleans and binds biolog data on a single sheet of an excel spreadsheet. Returns a dataframe where each substrate is a column and each row is a separate plate. The values in each column are the OD readings of that plate in that substrate.
- bind_biolog_all()
  - cleans and binds biolog data across multiple sheets of an excel spreadsheet. Returns a dataframe where each substrate is a column and each row is a separate plate. The values in each column are the OD readings of that plate in that substrate. Also includes columns for the sheet and plate of each row.

```
filename <- '~/Desktop/biolog_data.xlsx'
sheets <- paste0('S', 1:10, sep = '')


MicrobioUoE::bind_biolog_all(filename, sheets)
```

- stock_sol_vol()
  - calculates the amount of volume needed to dilute a stock solution to a required concentration and volume

```
MicrobioUoE::stock_sol_vol(stock_sol_conc = 0.75, new_sol_conc = 0.2, new_sol_vol = 1000)
#> [1] 266.6667
```

- calc_norm_OD()
  - takes an csv of OD readings and writes another csv of the required volumes to normalise the concentrations across samples

```
MicrobioUoE::calc_norm_OD('first_OD_readings.csv',
                          'df_volumes.csv',
                          new_sol_conc = 0.1,
                          new_sol_vol = 1000,
                          control = 0.035)
```

Cheers!

Components