Summer R

Session 5: Joins, factors, and pivoting

Dominic Bordelon, Research Data Librarian, ULS

Agenda

Joins

Factors

Pivoting

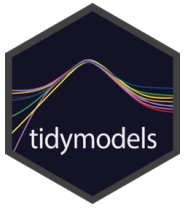
Inference and modeling

Introducing forcats, tidyr, and tidymodels

We've used dplyr for many data frame operations. In our last session, we're trying out three new packages:

- **forcats**: working with factors (categorical variables); tidyverse

- **tidyr**: restructure (pivot) a data frame; tidyverse
- **tidymodels**: inference and modeling



```r
library(tidyverse)
# includes dplyr, forcats, tidyr
library(medicaldata)

# install.packages("tidymodels")
library(tidymodels)
```

Joins

Relational data and table joins

- Much of the world's data lives in *relational database management systems* (RDBMSes), or simply "relational databases"
- Sometimes you might work with one of these directly; other times you might receive CSV files which originated in a RDBMS
- "Relational" == Table A has a relationship to Table B, via column X which appears in both tables ("foreign key relationship")
- Why relational?
  - One big table gets to be unmanageable, but we also don't want to duplicate info in the multiple places that we need it
  - Often-repeated values (e.g., a drug name) can be more efficiently stored/retrieved if they are replaced by a number, which points to the human-readable name.

Joining synthesizes a new a table from two or more parent tables.

Relationship types

- **One-to-one relationships**: one row in Table A corresponds to one row in Table B
- **One-to-many relationships**: “lookup” function; controlled vocabulary; Table B describes categories that are referenced in Table A
- **Many-to-many relationships**: there are many subjects (Table A), and many genetic markers (Table B); a patient may have an unspecified number of markers and vice versa, a relationship described in Table C

It can be helpful to have these relationships in mind when thinking about different ways we might want to join tables together.

Types of joins

Join type asks, “if I have unpaired rows when I join Tables A and B, what do I do with those rows?”

- Full join: keep all rows from both tables, filling NA as needed
- Inner join: keep only rows that appear in *both* tables
- Left join: inner join + keep all of the rows from the *left* table, filling NA as needed
- Right join: inner join + keep all of the rows from the *right* table, filling NA as needed
- Filtering joins: filter rows of Table A based on presence/absence of the row’s ID in Table B

Each of the above joins might produce a different number of rows.

```r
lg_sample <- read_csv("data/lg_sample.csv")
asa_status <- read_csv("data/asa_status.csv")
```

Consider these two tables:

```r
# 6-row sample of licorice_gargle

lg_sample %>%
  select(1:3)
```

```
# A tibble: 6 × 3
  preOp_gender preOp_asa
preOp_calcBMI
         <dbl>     <dbl>
<dbl>
1            0         2
31.3
2            0         1
32.3
3            1         3
30.5
4            0         3
32.2
5            1        NA
36.3
6            0         2
```

```r
# listing of ASA statuses

asa_status
```

```
# A tibble: 4 × 2
  asa_status_id asa_status
          <dbl> <chr>
1             1 a normal healthy
patient
2             2 a patient with mild
systemic disease
3             3 a patient with severe
systemic disease
4             4 UNUSED CATEGORY
```

full_join(): combine all data

```
1  lg_sample %>%
2    select(1:4) %>%
3    full_join(y = asa_status, by = join_by(preOp_asa == asa_status_id))
```

```
# A tibble: 7 × 5
  preOp_gender preOp_asa preOp_calcBMI preOp_age asa_status
         <dbl>     <dbl>          <dbl>     <dbl> <chr>
1            0         2           31.3        80 a patient with mild systemic
d…
2            0         1           32.3        58 a normal healthy patient
3            1         3           30.5        65 a patient with severe
systemic…
4            0         3           32.2        54 a patient with severe
systemic…
5            1        NA           36.3        56 <NA>
6            0         2           28.0        68 a patient with mild systemic
d…
7           NA         4           NA          NA UNUSED CATEGORY
```

Note 7 rows: 5 with matches in both tables; one Table A patient with no pre-op ASA status reported; and one Table B unused

`inner_join()`: keep only rows from both tables

```
lg_sample %>%
  select(1:4) %>%
  inner_join(y = asa_status, by = join_by(preOp_asa == asa_status_id))
```

```
# A tibble: 5 × 5
  preOp_gender preOp_asa preOp_calcBMI preOp_age asa_status
         <dbl>     <dbl>          <dbl>     <dbl> <chr>
1            0         2           31.3        80 a patient with mild systemic
d…
2            0         1           32.3        58 a normal healthy patient
3            1         3           30.5        65 a patient with severe
systemic…
4            0         3           32.2        54 a patient with severe
systemic…
5            0         2           28.0        68 a patient with mild systemic
d…
```

Now we have only the five rows that appear in both tables.

`left_join()`: keep rows from x

```
1  lg_sample %>%
2    select(1:4) %>%
3    left_join(y = asa_status, by = join_by(preOp_asa == asa_status_id))
```

```
# A tibble: 6 × 5
  preOp_gender preOp_asa preOp_calcBMI preOp_age asa_status
         <dbl>     <dbl>          <dbl>     <dbl> <chr>
1            0         2           31.3        80 a patient with mild systemic
d…
2            0         1           32.3        58 a normal healthy patient
3            1         3           30.5        65 a patient with severe
systemic…
4            0         3           32.2        54 a patient with severe
systemic…
5            1        NA           36.3        56 <NA>
6            0         2           28.0        68 a patient with mild systemic
d…
```

Now we have increased to 6 rows, because we kept all of the left table, including the patient with no pre-op ASA status reported.

`right_join()`: keep rows from y

```
1  lg_sample %>%
2    select(1:4) %>%
3    right_join(y = asa_status, by = join_by(preOp_asa == asa_status_id))
```

```
# A tibble: 6 × 5
  preOp_gender preOp_asa preOp_calcBMI preOp_age asa_status
         <dbl>     <dbl>          <dbl>     <dbl> <chr>
1            0         2           31.3        80 a patient with mild systemic
d…
2            0         1           32.3        58 a normal healthy patient
3            1         3           30.5        65 a patient with severe
systemic…
4            0         3           32.2        54 a patient with severe
systemic…
5            0         2           28.0        68 a patient with mild systemic
d…
6           NA         4           NA          NA UNUSED CATEGORY
```

6 rows again, but this time we have dropped the nonreporting patient, and we see instead the unused category.

`semi_join()`, `anti_join()`: filter x vs. y

Filtering joins decide which rows to keep from x, based on the presence (or absence) of the value in y

Example: you applied inclusion criteria to get a list of patient ID's to include in analysis, and now you would like to filter just those patients' records into a data frame.

```
inclusion_ids <- tibble(subject_id = c(9134, 663, 5408))

covid_testing %>%
  semi_join(y = inclusion_ids, by = join_by(subject_id))
```

```
# A tibble: 3 × 17
  subject_id fake_…¹ fake_…² gender pan_day test_id clini…³ result demo_…⁴   age
       <dbl> <chr>   <chr>   <chr>    <dbl> <chr>   <chr>   <chr>  <chr>   <dbl>
1       9134 grunt   rivers  male         7 covid   clinic… negat… patient   0.8
2        663 ithoke  targar… male         9 covid   clinic… negat… patient   0.8
3       5408 alia    ryswell female      10 covid   clinic… negat… patient   0.9
# … with 7 more variables: drive_thru_ind <dbl>, ct_result <dbl>,
#   orderset <dbl>, payor_group <chr>, patient_class <chr>, col_rec_tat <dbl>,
#   rec_ver_tat <dbl>, and abbreviated variable names ¹fake_first_name,
#   ²fake_last_name, ³clinic_name, ⁴demo_group
```

Factors

Factors

- Factors are special vectors for categorical variables
- A factor has one or more **levels** of accepted values
- Values are integers with a human-readable label
- Levels may be ordered or unordered

```
1 c("A", "B", "C", "B", "A", "B") %>% summary()
```

```
   Length     Class      Mode 
        6 character character 
```

```
1 as_factor(c("A", "B", "C", "B", "A", "B")) %>% summary()
```

```
A B C 
2 3 1 
```

Jobs involving factors

Typical things you'll need to do with factors:

- Convert character vector into factor: `as_factor()`
- Change the label associated with a level: `fct_recode()`
- Collapse two or more levels into one level (e.g., "Other"): `fct_collapse()`
- Change factor's order of levels: several functions

```
1 my_fct <- as_factor(c("A", "B", "C", "B", "A", "B"))
2
3 my_fct
```

```
[1] A B C B A B
Levels: A B C
```

```
1 my_fct %>% fct_recode(beta = "B")
```

```
[1] A    beta C    beta A    beta
Levels: A beta C
```

```
1 my_fct %>% fct_collapse(Other = c("A", "B"))
```

```
[1] Other Other C     Other Other Other
Levels: Other C
```

```
1 my_fct %>% fct_infreq()
```

```
[1] A B C B A B
Levels: B A C
```

forcats::fct_infreq()

reorder FACTOR LEVELS by # of observations in each level

(default: largest n = 1st level)

(df)

animal	mass
alligator	281.5
cat	9.4
cat	11.6
whale	52300
alligator	427.9
cat	7.3
cat	10.8

LEVELS REORDERED BASED on COUNTS

see also:

fct_inorder: reorder by order of appearance

fct_reorder: reorder by another variable

fct_relevel: reorder manually

+ more!

forcats.tidyverse.org

Artwork by @allison_horst

ICA 5.1: Joins and factors

Pivoting

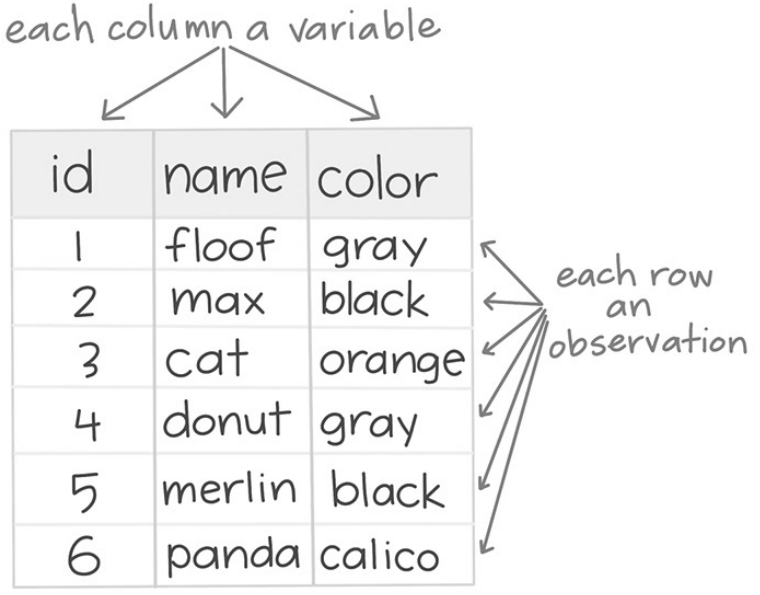
Pivoting: what and why

- To “pivot” tabular data means to restructure part of it in terms of column and row definition
- Example: a variable described by *one column per year* of interest
  - But ggplot expects all values to be in the same column
  - Solution: all values into a new `value` column, and each year into a new `year` column
  - n columns $\rightarrow$ 2 columns
- The goal is often “tidy data”: one observation per row, one variable per column

> TIDY DATA is a standard way of mapping the meaning of a dataset to its structure.
>
> –HADLEY WICKHAM

In tidy data:

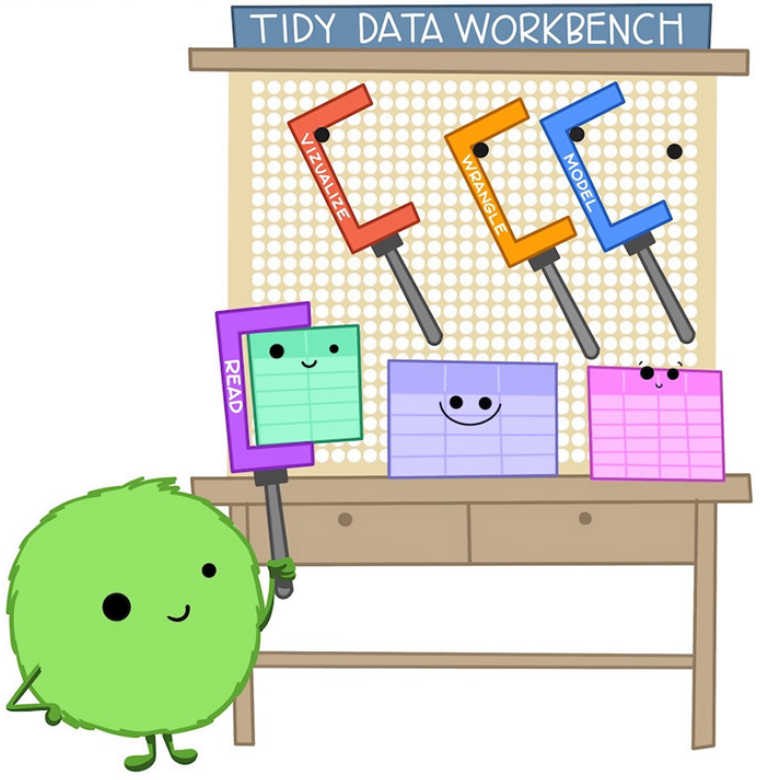
- each variable forms a column
- each observation forms a row
- each cell is a single measurement



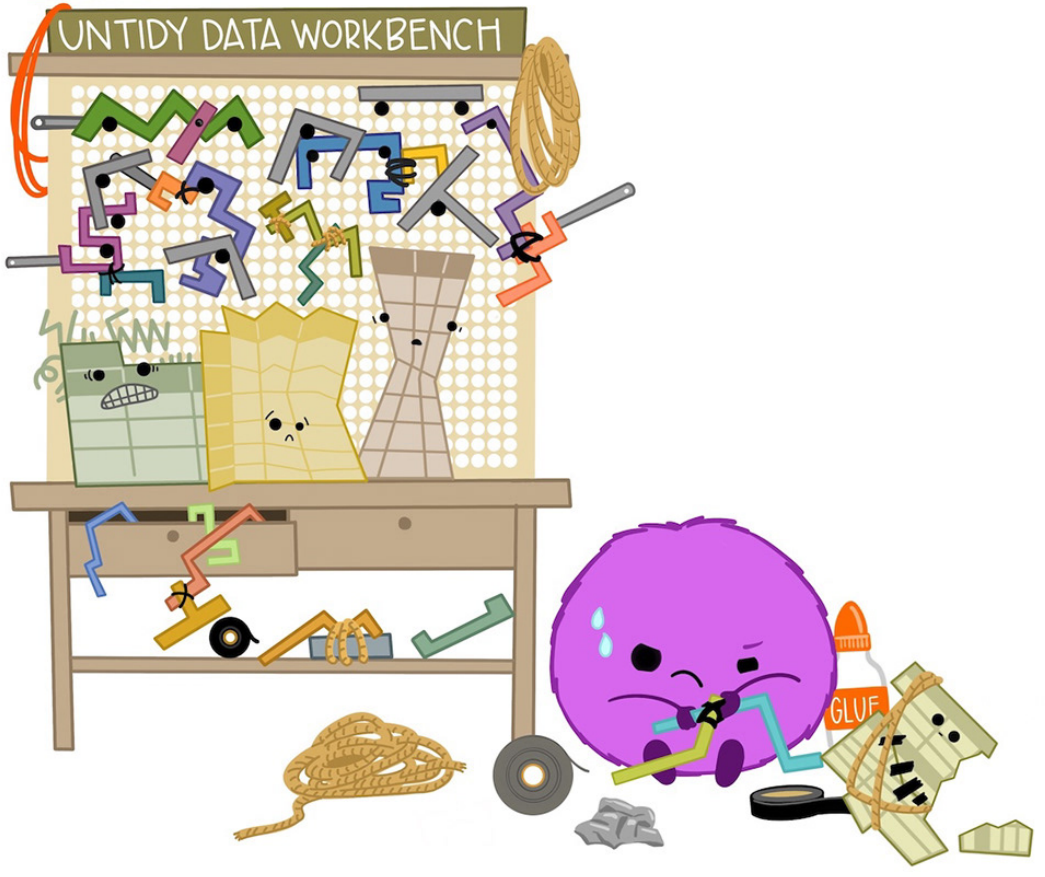
Wickham, H. (2014). Tidy Data. Journal of Statistical Software *59* (10). DOI: 10.18637/jss.v059.i10

Artwork by @allison_horst

When working with tidy data, we can use the same tools in similar ways for different datasets...



...but working with untidy data often means reinventing the wheel with one-time approaches that are hard to iterate or reuse.



Artwork by @allison_horst

`pivot_longer()`

- For when you need reduce the number of columns, making the table *longer* with more rows
- Or: when multiple *observations* are compressed into a single row
- Note required arguments (given in example below)
- Example: annual teen birth counts by state, one column for each year (2013-19); source: CDC
  - `cols =` the year columns (`` `2013`:`2019` ``)
  - `names_to =` new column where you want the years recorded (`year`)
  - `values_to =` new column where you want the values recorded (`births`)

Before pivoting

```
1  teen_births %>% print(width=58)
```

```
# A tibble: 52 × 8
# Groups:   State [52]
   State  `2013` `2014` `2015` `2016` `2017` `2018`
`2019`
   <chr>   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
<dbl>
 1 Alaba…  10784  10018   9478   8960   8482   7848
7910
 2 Alaska   1416   1290   1324   1166    972    846
786
 3 Arizo…  14464  13244  11820  10714  10050   9300
8636
 4 Arkan…   8310   7564   7354   6744   6356   5856
5764
 5 Calif…  61010  54050  48350  42824  37870  33858
30712
 6 Color…   7668   6754   6540   6136   5580   5044
4942
 7 Conne…   3212   2840   2482   2272   2106   1976
1804
 8 Delaw…   1456   1232   1080   1166   1104    994
888
```

After pivoting longer

```
1  teen_births %>%
2    pivot_longer(cols = `2013`:`2019`,
3                 names_to = "year",
4                 values_to = "births")
```

```
# A tibble: 364 × 3
# Groups:   State [52]
   State    year  births
   <chr>    <chr>  <dbl>
 1 Alabama  2013   10784
 2 Alabama  2014   10018
 3 Alabama  2015    9478
 4 Alabama  2016    8960
 5 Alabama  2017    8482
 6 Alabama  2018    7848
 7 Alabama  2019    7910
 8 Alaska   2013    1416
 9 Alaska   2014    1290
10 Alaska   2015    1324
# … with 354 more rows
```

Data source: CDC

`pivot_wider()`

- For when you need to increase the number of columns, making the table *wider* with fewer rows
- Rare that you will need to use this for tidying; usually for generating a summary/comparison table, or for transforming to the format expected by another tool
- Note required arguments `names_from`, `values_from`

ICA 5.2: Pivoting

Inference and modeling

Checking for normality: histogram + theoretical

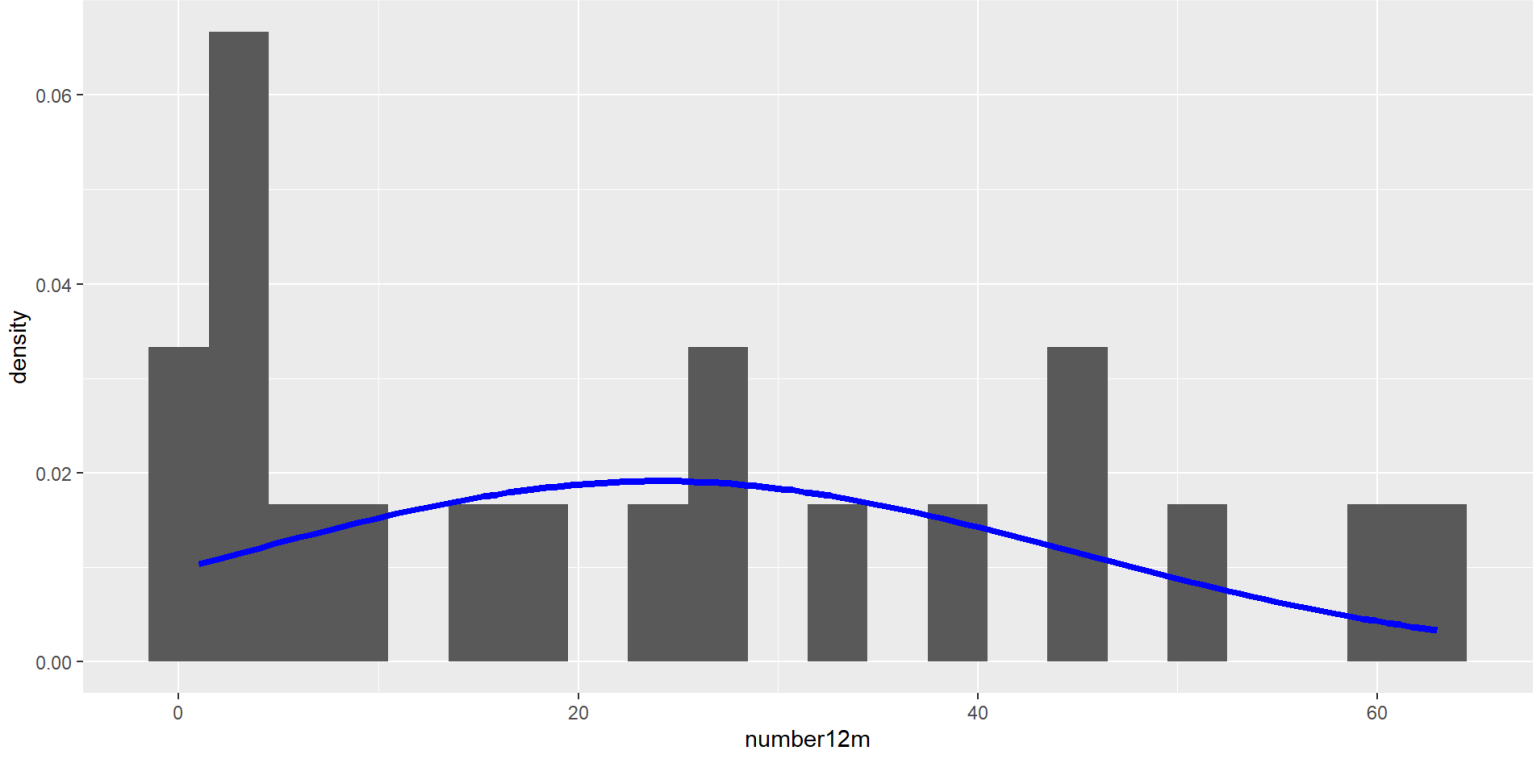
The ggplot2 `stat_function()` layer allows you to plot various statistical functions (such as a theoretical distribution). Note that for the two layers to be compatible, we transform the histogram's `y` to density. `dnorm()` is the base R function for the Normal distribution.

```
mu <- mean(polyps$number12m, na.rm=TRUE)
sigma <- sd(polyps$number12m, na.rm=TRUE)

polyps %>%
  drop_na(number12m) %>%
  ggplot() +
  geom_histogram(aes(x=number12m, y=after_stat(density)), binwidth = 3) +
  stat_function(fun=dnorm, geom="line", color="blue", linewidth=1.5,
                args=list(mean=mu, sd=sigma))
```

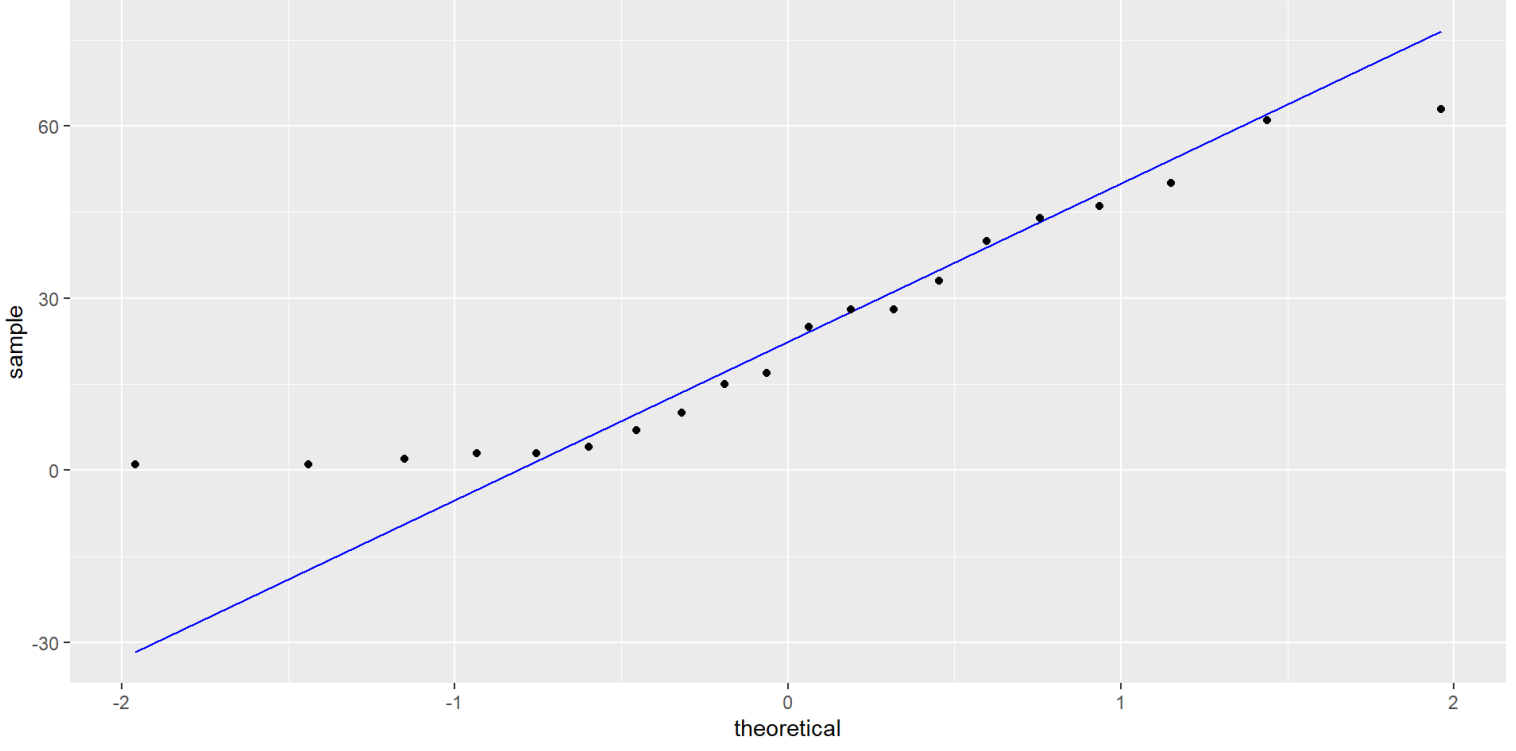
```r
mu <- mean(polyps$number12m, na.rm=TRUE)
sigma <- sd(polyps$number12m, na.rm=TRUE)

polyps %>%
  drop_na(number12m) %>%
  ggplot() +
  geom_histogram(aes(x=number12m, y=after_stat(density)), binwidth = 3) +
  stat_function(fun=dnorm, geom="line", color="blue", linewidth=1.5,
                args=list(mean=mu, sd=sigma))
```



Checking for normality: Q-Q plot

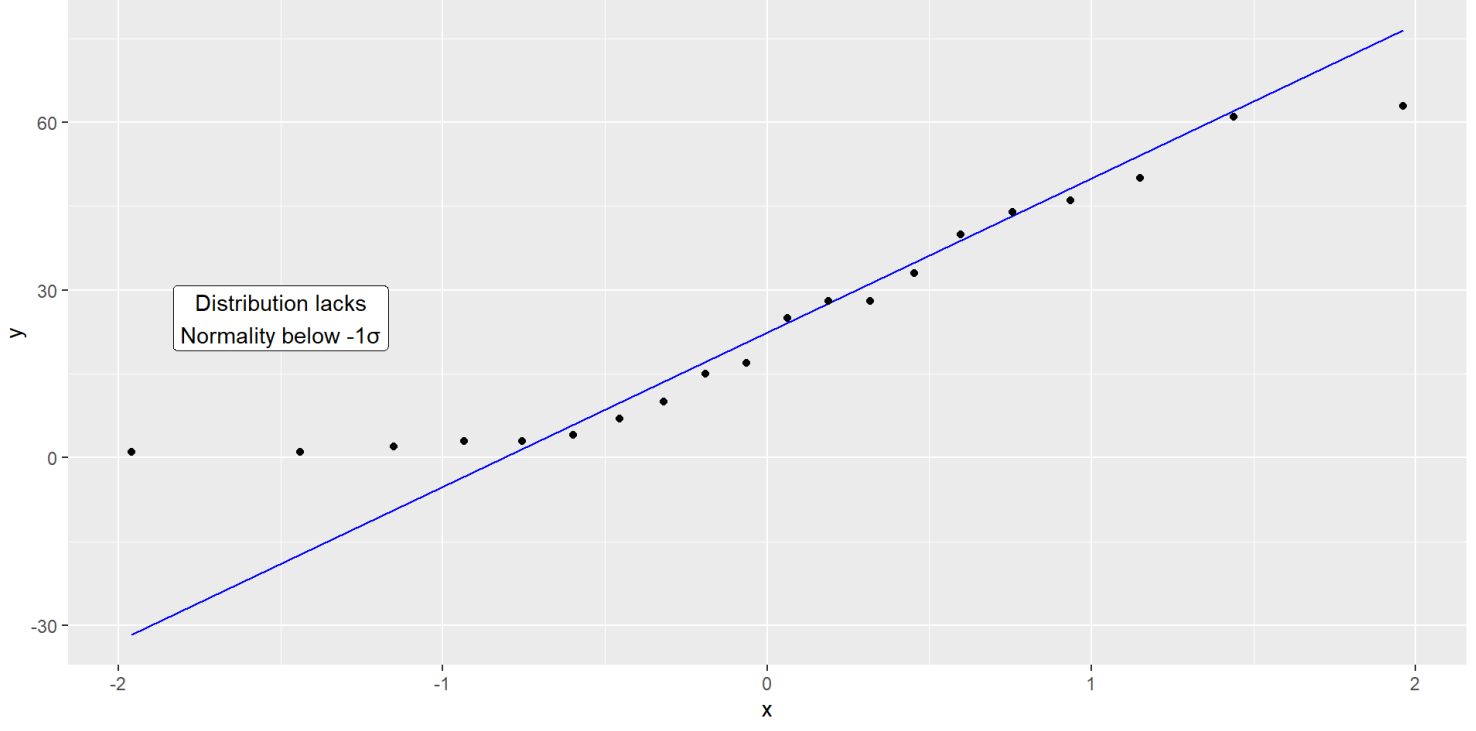
```
num12m_qqp <- polyps %>%
  drop_na(number12m) %>%
  ggplot(aes(sample=number12m)) +
  geom_qq_line(color="blue") +
  geom_qq()
num12m_qqp
```



Sidebar: plot annotations

`annotate()` is an additional ggplot layer you can add to plots:

```
num12m_qqp <- num12m_qqp +
  annotate(geom="label", x=-1.5, y=25,
           label="Distribution lacks\nNormality below -1σ")
num12m_qqp
```



Sidebar: saving plots

`ggplot2::ggsave()` allows us to save plot objects in a variety of formats and parameters. Here are example calls:

```
# scalable vector graphics (SVG) format:
ggsave(filename="num12m-qq.svg", plot=num12m_qqp)

# 500 x 400 px PNG file, 100dpi:
ggsave(filename="num12m-qq.png", plot=num12m_qqp,
       width=500, height=400, units="px", dpi=100)

# 4 in. x 6 in. PDF, 300dpi:
ggsave(filename="num12m-qq.pdf", plot=num12m_qqp,
       width=4, height=6, units="in")
# note that this plot will be in a portrait format
```

Inference and modeling - the base R way

- A variety of functions are available depending on the test / probability distribution / regression you need
- These functions usually expect vectors as input; usually produce some printable object (`summary()` also usable)
- Student's t-test: `t.test()`
- Chi-squared test: `chisq.test()` (note: expects a matrix for 2-way)
- Linear model: `lm()` with `predict()`

These functions can be compatible with tidyverse work (but not `%>%`)

Are mean 12-month counts significantly different between sulindac (treatment) and placebo groups?

```r
sulindac12m <- polyps %>%
  filter(treatment == "sulindac") %>%
  pull(number12m)
sulindac12m
```

```
[1] NA  2 17  1 25  3 33 NA  3  1  4
```

```r
placebo12m <- polyps %>%
  filter(treatment == "placebo") %>%
  pull(number12m)
placebo12m
```

```
[1] 63 28 61  7 15 44 28 10 40 46 50
```

```r
t.test(sulindac12m, placebo12m)
```

```
    Welch Two Sample t-test

data:  sulindac12m and placebo12m
t = -3.6114, df = 16.901, p-value = 0.002172
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -40.79597 -10.69898
sample estimates:
mean of x mean of y
 9.888889 35.636364
```

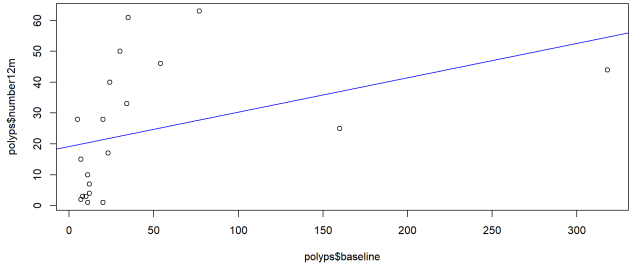
Is there a relationship between baseline and 12-month counts?

```r
polyps_model <- lm(number12m ~ baseline, data=polyps)
polyps_model
```

```
Call:
lm(formula = number12m ~ baseline, data = polyps)

Coefficients:
(Intercept)     baseline
    19.1647       0.1113
```

```r
plot(polyps$baseline, polyps$number12m)
abline(polyps_model, col="blue")
```



What 12-month count does this model predict for a baseline of 50? of 100?

```r
new_baseline <- tibble(
  baseline = c(50, 100)
)
predict(polyps_model, newdata=new_baseline)
```

```
       1        2
24.72882 30.29290
```

Inference and modeling - the tidymodels way

- Specifically, the {infer} and {parsnip} packages
- New verbs for the tidyverse pipeline: (or you can use wrappers like `t_test()`)
  - `specify()` variable/relationship of interest
  - `hypothesize()` the null hypothesis
  - `generate()` (or simulate if theoretical) the null distribution
  - `calculate()` a test statistic like `Chisq, F, t,` or `z`
- Good vignettes for getting started: Getting to Know infer, Introduction to parsnip

Are mean 12-month counts significantly different between sulindac (treatment) and placebo groups?

```r
# infer::t_test()

polyps %>%
  t_test(response = number12m,
         explanatory = treatment)
```

```
# A tibble: 1 × 7
  statistic  t_df p_value alternative estimate lower_ci upper_ci
      <dbl> <dbl>   <dbl> <chr>          <dbl>    <dbl>    <dbl>
1      3.61  16.9 0.00217 two.sided       25.7     10.7     40.8
```

Is there a relationship between baseline and 12-month counts?

```r
# parsnip::linear_reg()

polyps_model <- linear_reg() %>%
  set_engine("lm") %>%
  fit(number12m ~ baseline, data = polyps)
polyps_model
```

```
parsnip model object


Call:
stats::lm(formula = number12m ~ baseline, data = data)

Coefficients:
(Intercept)     baseline
    19.1647       0.1113
```

What 12-month count does this model predict for a baseline of 50? of 100?

```r
new_baseline <- tibble(
  baseline = c(50, 100)
)
predict(polyps_model, new_baseline)
```

```
# A tibble: 2 × 1
  .pred
  <dbl>
1  24.7
2  30.3
```

Plotting a model (without `geom_smooth`)

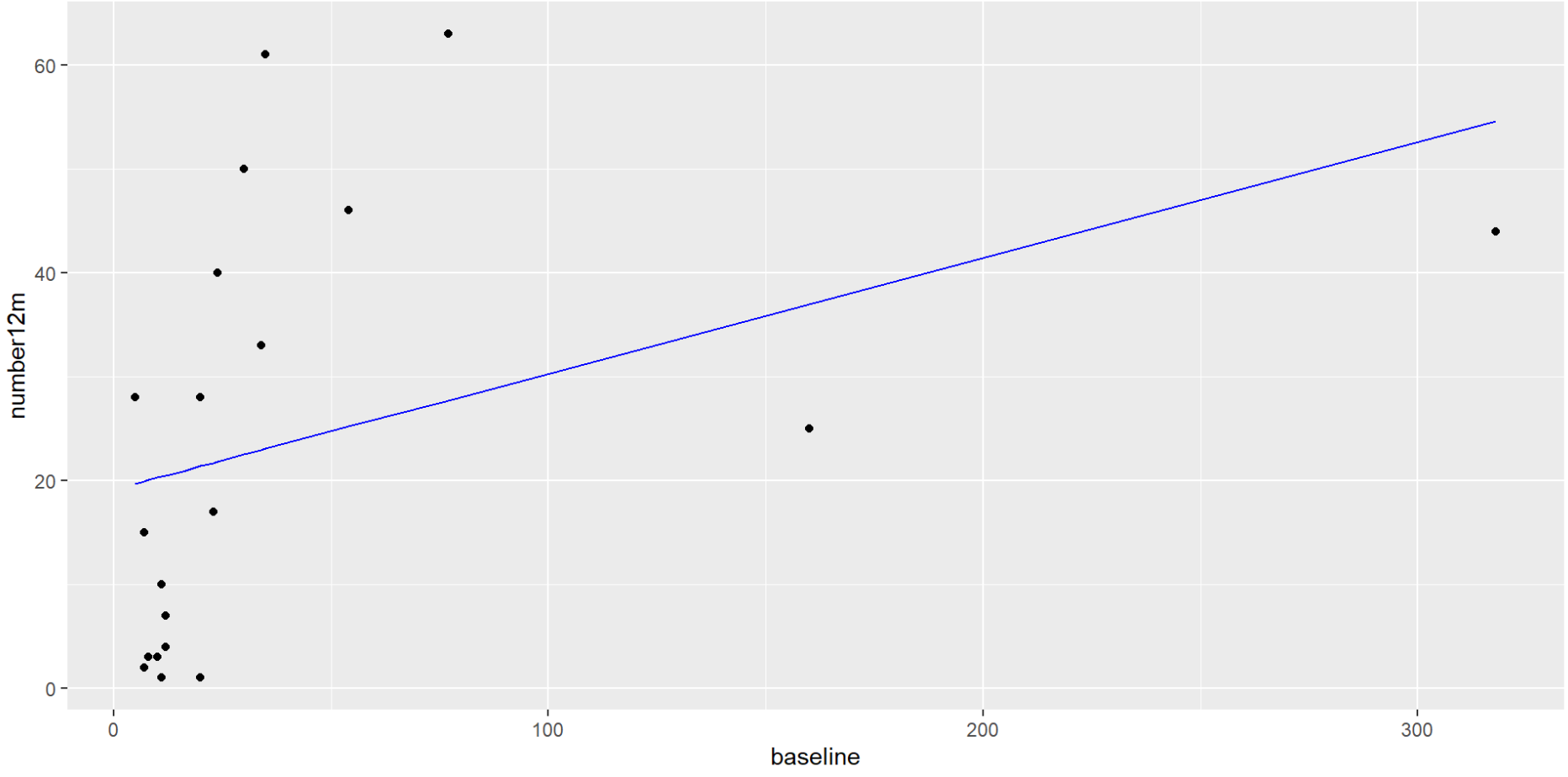
Run `predict()` on a data frame of your explanatory values. Then bind the predicted column back with the original data frame and plot the predictions with their own `geom_line()`.

```
baseline_df <- polyps %>%
  select(baseline)

predicted_df <- predict(polyps_model, new_data = baseline_df) %>%
  rename(predicted_12m = ".pred") %>%
  mutate(predicted_12m = round(predicted_12m, 1))
predicted_df
```

```
# A tibble: 22 × 1
   predicted_12m
           <dbl>
 1          19.9
 2          27.7
 3          19.9
 4          19.7
 5          21.7
 6          23.1
 7          20.4
 8          20.5
 9          19.9
10          54.6
# … with 12 more rows
```

```r
polyps %>%
  bind_cols(predicted_df) %>%
  ggplot() +
  geom_point(aes(x=baseline, y=number12m)) +
  geom_line(aes(x=baseline, y=predicted_12m),
            color="blue")
```



ICA 5.3: Inference and modeling

Wrap up

Conclusion

We learned about:

- joining tables to combine data
- working with factors for categorical variables
- pivoting a data frame to restructure it
- what inference and modeling look like in R

Potential next directions

- Exploring more of tidymodels
- Computational Genomics with R (free ebook)
- …or one of many other topics in the Big Book of R directory

Always feel free to reach out! dbordelon@pitt.edu