Statistics

Summary: Data visualization

- `ggplot()` specifies what data to use and what variables will be mapped to where
- inside `ggplot()`, `aes(x = , y = , color = )` specify what variables correspond to what aspects of the plot in general
- layers of plots can be combined using the `+` at the **`end`** of lines
- use `geom_line()` and `geom_point()` to add lines and points
- sometimes you need to add a `group` element to `aes()` if your plot looks strange
- make sure you are plotting what you think you are by checking the numbers!
- `facet_grid(~variable)` and `facet_wrap(~variable)` can be helpful to quickly split up your plot

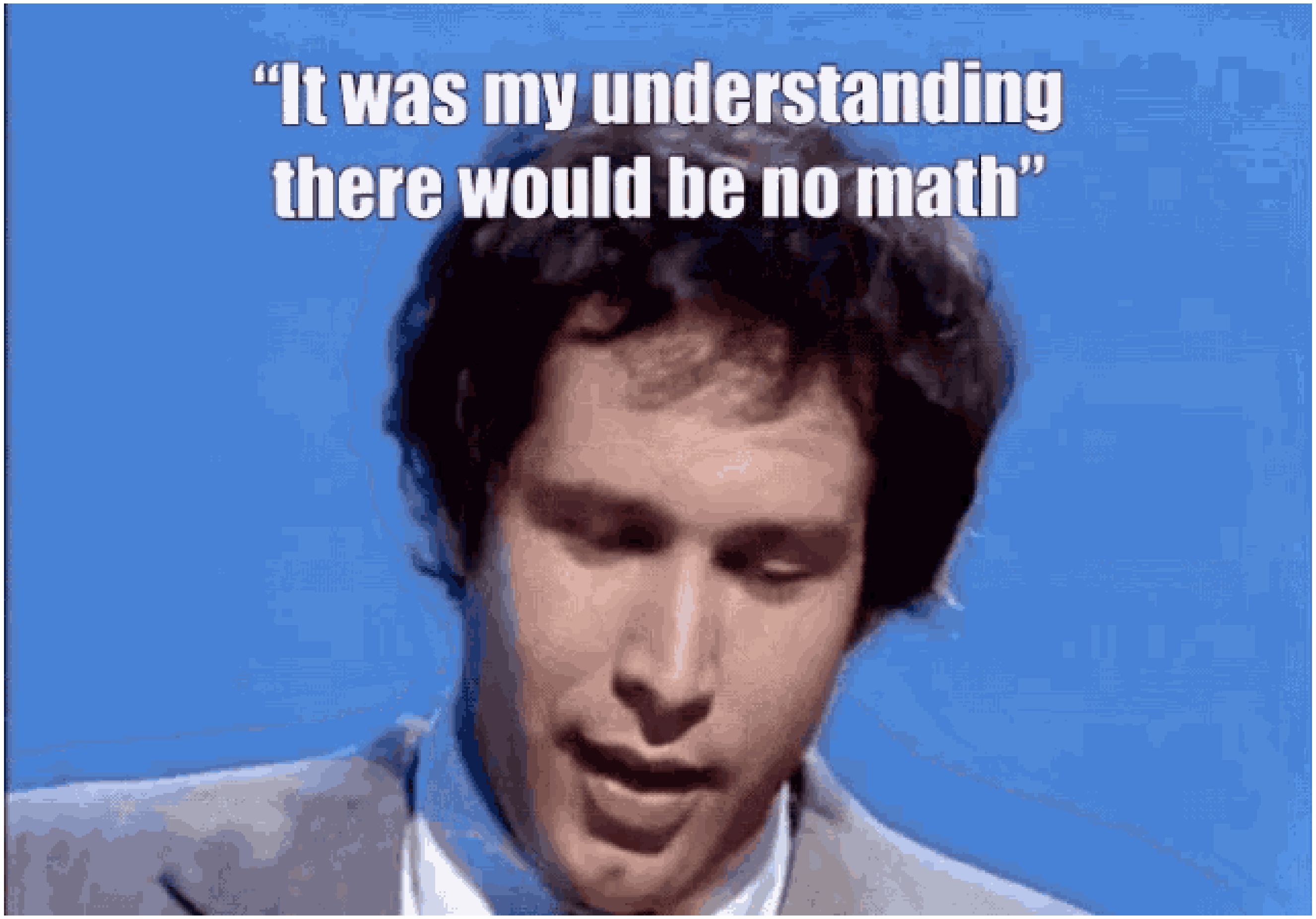
Summary: Factors

- the factor class allows us to have a different order from alphanumeric for categorical data
- we can change data to be a factor variable using `mutate()`, `as_factor()` (in the `forcats` package), or `factor()` functions and specifying the levels with the `levels` argument
- `fct_reorder({variable_to_reorder}, {variable_to_order_by})` helps us reorder a variable by the values of another variable
- arranging, tabulating, and plotting the data will reflect the new order

Overview

We will cover how to use R to compute some of basic statistics and fit some basic statistical models.

- Correlation
- T-test
- Linear Regression / Logistic Regression


“It was my understanding
there would be no math”

Overview

□ We will focus on how to use R software to do these. We will be glossing over the statistical **theory** and “formulas” for these tests. Moreover, we do not claim the data we use for demonstration meet **assumptions** of the methods. □

There are plenty of resources online for learning more about these methods, as well as dedicated Biostatistics series (at different advancement levels) at the JHU School of Public Health.

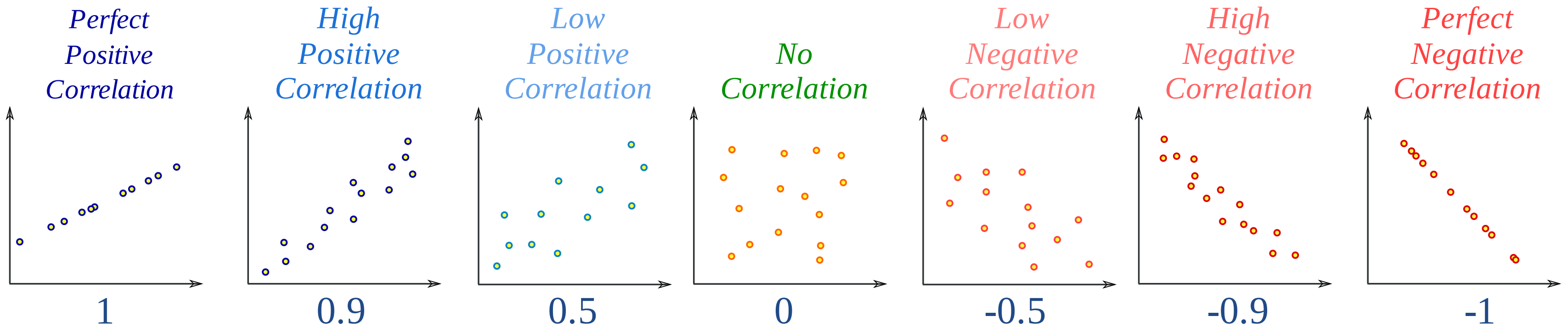
Check out www.opencasestudies.org for deeper dives on some of the concepts covered here and the resource page for more resources.

Correlation

Correlation

The correlation coefficient is a summary statistic that measures the strength of a linear relationship between two numeric variables.

- The strength of the relationship - based on how well the points form a line
- The direction of the relationship - based on if the points progress upward or downward



source

See this case study for more information.

Correlation

Function `cor()` computes correlation in R.

```
cor(x, y = NULL, use = c("everything", "complete.obs"),
    method = c("pearson", "kendall", "spearman"))
```

- provide two numeric vectors of the same length (arguments `x`, `y`), or
- provide a data.frame / tibble with numeric columns only
- by default, Pearson correlation coefficient is computed

Correlation test

Function `cor.test()` also computes correlation and tests for association.

```
cor.test(x, y = NULL, alternative = c("two.sided", "less", "greater"),
    method = c("pearson", "kendall", "spearman"), ...)
```

- provide two numeric vectors of the same length (arguments `x`, `y`), or
- provide a data.frame / tibble with numeric columns only
- by default, Pearson correlation coefficient is computed
- alternative values:
    - two.sided means true correlation coefficient is not equal to zero (default)
    - greater means true correlation coefficient is > 0 (positive relationship)
    - less means true correlation coefficient is < 0 (negative relationship)

GUT CHECK!

What class of data do you need to calculate a correlation?

A. Character data

B. Factor data

C. Numeric data

Correlation

Using the Charm City Circulator data.

```
circ <- read_csv("https://jhudatascience.org/intro_to_r/data/Charm_City_Circulator_Ridership.csv")
head(circ)
```

```
# A tibble: 6 × 15
  day       date  orangeBoardings orangeAlightings orangeAverage purpleBoardings
  <chr>     <chr>           <dbl>            <dbl>         <dbl>           <dbl>
1 Monday    01/1…             877             1027          952               NA
2 Tuesday   01/1…             777              815          796               NA
3 Wednesday 01/1…            1203             1220         1212.              NA
4 Thursday  01/1…            1194             1233         1214.              NA
5 Friday    01/1…            1645             1643         1644               NA
6 Saturday  01/1…            1457             1524         1490.              NA
# ℹ 9 more variables: purpleAlightings <dbl>, purpleAverage <dbl>,
#   greenBoardings <dbl>, greenAlightings <dbl>, greenAverage <dbl>,
#   bannerBoardings <dbl>, bannerAlightings <dbl>, bannerAverage <dbl>,
#   daily <dbl>
```

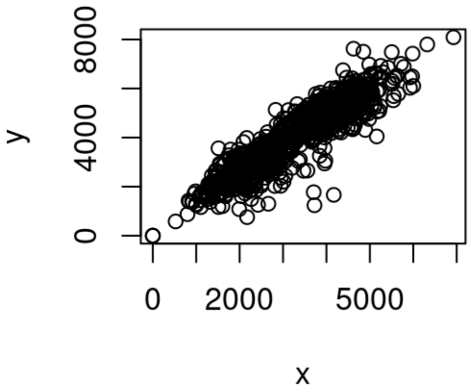
Correlation for two vectors

First, we compute correlation by providing two numeric vectors.

Like other functions, if there are `NA`s, you get `NA` as the result. But if you specify to use only the complete observations, then it will give you correlation using the non-missing data.

```
# x and y must be numeric vectors
x <- circ %>% pull(orangeAverage)
y <- circ %>% pull(purpleAverage)
```

```
# have to specify which data on each axis
# can accomodate missing data
plot(x, y)
```



Correlation coefficient calculation and test

```
library(broom)
cor(x, y)
```

```
[1] NA
```

```
cor(x, y, use = "complete.obs")
```

```
[1] 0.9195356
```

```
cor.test(x, y)
```

```

    Pearson's product-moment correlation

data:  x and y
t = 73.656, df = 991, p-value < 0.00000000000000022
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9093438 0.9286245
sample estimates:
      cor
0.9195356
```

Broom package

The `broom` package helps make stats results look tidy

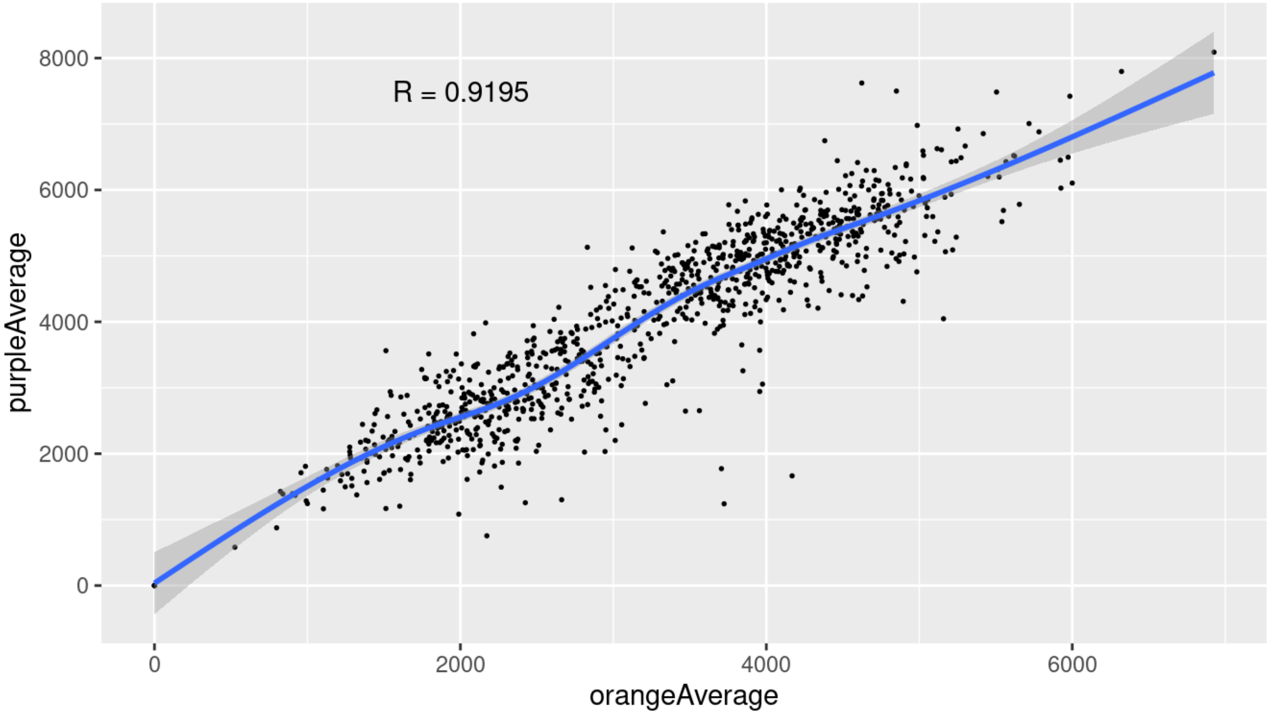
```
cor_result <- tidy(cor.test(x, y))
glimpse(cor_result)
```

```
Rows: 1
Columns: 8
$ estimate    <dbl> 0.9195356
$ statistic   <dbl> 73.65553
$ p.value     <dbl> 0
$ parameter   <int> 991
$ conf.low    <dbl> 0.9093438
$ conf.high   <dbl> 0.9286245
$ method      <chr> "Pearson's product-moment correlation"
$ alternative <chr> "two.sided"
```

Correlation for two vectors with plot

In plot form, `geom_smooth()` and `annotate()` can help.

```
corr_value <- pull(cor_result, estimate) %>% round(digits = 4)
cor_label <- paste0("R = ", corr_value)
circ %>%
  ggplot(aes(x = orangeAverage, y = purpleAverage)) +
  geom_point(size = 0.3) +
  geom_smooth() +
  annotate("text", x = 2000, y = 7500, label = cor_label)
```



Correlation for data frame columns

We can compute correlation for all pairs of columns of a data frame / matrix. This is often called, *"computing a correlation matrix"*.

Columns must be all numeric!

```
circ_subset_Average <- circ %>% select(ends_with("Average"))
head(circ_subset_Average)
```

```
# A tibble: 6 × 4
  orangeAverage purpleAverage greenAverage bannerAverage
          <dbl>         <dbl>        <dbl>         <dbl>
1          952             NA           NA            NA
2          796             NA           NA            NA
3         1212.            NA           NA            NA
4         1214.            NA           NA            NA
5         1644             NA           NA            NA
6         1490.            NA           NA            NA
```

Correlation for data frame columns

We can compute correlation for all pairs of columns of a data frame / matrix. This is often called, *“computing a correlation matrix”*.

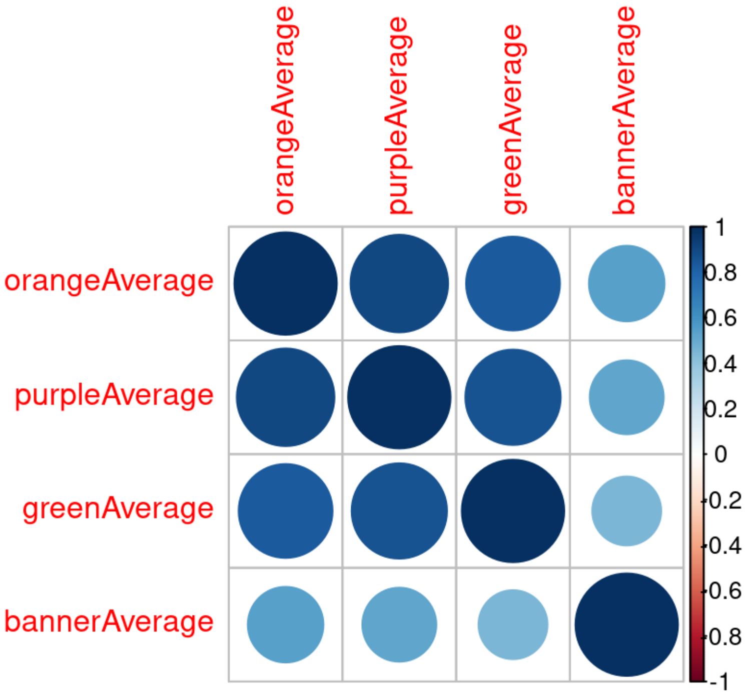
```
cor_mat <- cor(circ_subset_Average, use = "complete.obs")
cor_mat
```

```
              orangeAverage purpleAverage greenAverage bannerAverage
orangeAverage     1.0000000     0.9078826    0.8395806     0.5447031
purpleAverage     0.9078826     1.0000000    0.8665630     0.5213462
greenAverage      0.8395806     0.8665630    1.0000000     0.4533421
bannerAverage     0.5447031     0.5213462    0.4533421     1.0000000
```

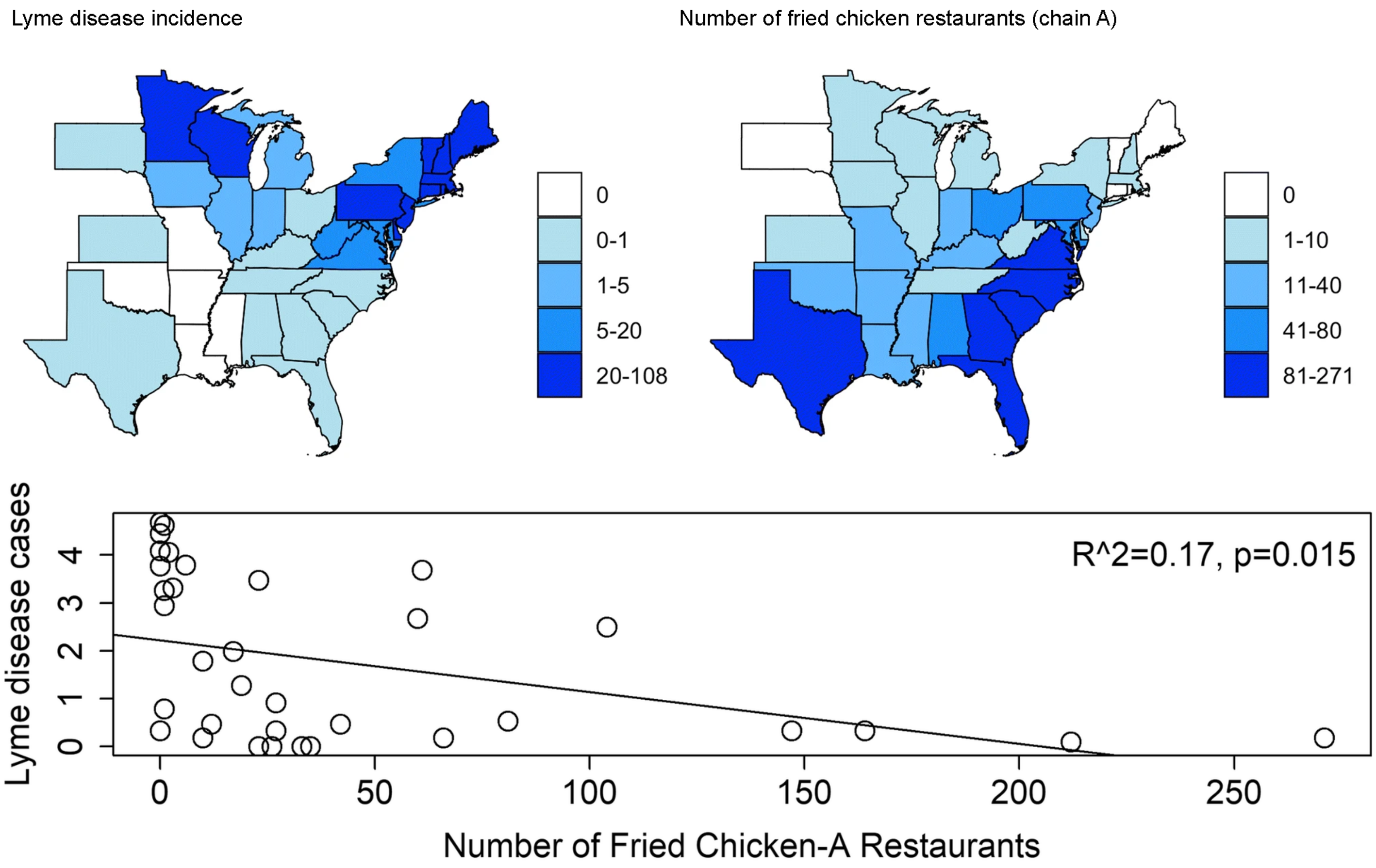
Correlation for data frame columns with plot

`corrplot` package can make correlation matrix plots

```
library(corrplot)
corrplot(cor_mat)
```



Correlation does not imply causation



source

T-test

T-test

The commonly used are:

- **one-sample t-test** – used to test mean of a variable in one group
- **two-sample t-test** – used to test difference in means of a variable between two groups (if the “two groups” are data of the *same* individuals collected at 2 time points, we say it is two-sample paired t-test)

The `t.test()` function in R is one to address the above.

```
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, ...)
```

Running one-sample t-test

It tests the mean of a variable in one group. By default (i.e., without us explicitly specifying values of other arguments):

- tests whether a mean of a variable is equal to 0 (`mu = 0`)
- uses “two sided” alternative (`alternative = "two.sided"`)
- returns result assuming confidence level 0.95 (`conf.level = 0.95`)
- omits `NA` values in data

```
sum(is.na(x)) # count NAs in x
```

```
[1] 10
```

```
t.test(x)
```

```
    One Sample t-test

data:  x
t = 83.279, df = 1135, p-value < 0.00000000000000022
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 2961.700 3104.622
sample estimates:
mean of x 
 3033.161 
```

Running two-sample t-test

It tests the difference in means of a variable between two groups. By default:

- tests whether difference in means of a variable is equal to 0 (`mu = 0`)
- uses “two sided” alternative (`alternative = "two.sided"`)
- returns result assuming confidence level 0.95 (`conf.level = 0.95`)
- assumes data are not paired (`paired = FALSE`)
- assumes true variance in the two groups is not equal (`var.equal = FALSE`)
- omits `NA` values in data

Check out this this case study and this case study for more information.

Running two-sample t-test in R

```
sum(is.na(x))
```

```
[1] 10
```

```
sum(is.na(y)) # count NAs in x and y
```

```
[1] 153
```

```
t.test(x, y)
```

```

    Welch Two Sample t-test

data:  x and y
t = -17.076, df = 1984, p-value < 0.00000000000000022
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1096.7602  -870.7867
sample estimates:
mean of x mean of y 
 3033.161  4016.935 
```

T-test: retrieving information from the result with `broom` package

The `broom` package has a `tidy()` function that can organize results into a data frame so that they are easily manipulated (or nicely printed).

```
result <- t.test(x, y)
result_tidy <- tidy(result)
glimpse(result_tidy)
```

```
Rows: 1
Columns: 10
$ estimate    <dbl> -983.7735
$ estimate1   <dbl> 3033.161
$ estimate2   <dbl> 4016.935
$ statistic   <dbl> -17.07579
$ p.value     <dbl> 0.00000000000000000000000000000000000000000000000000000000…
$ parameter   <dbl> 1983.954
$ conf.low    <dbl> -1096.76
$ conf.high   <dbl> -870.7867
$ method      <chr> "Welch Two Sample t-test"
$ alternative <chr> "two.sided"
```

P-value adjustment

□ You run an increased risk of Type I errors (a “false positive”) when multiple hypotheses are tested simultaneously. □

Use the `p.adjust()` function on a vector of p-values. Use `method =` to specify the adjustment method:

```
my_pvalues <- c(0.049, 0.001, 0.31, 0.00001)
p.adjust(my_pvalues, method = "BH") # Benjamini Hochberg
```

```
[1] 0.06533333 0.00200000 0.31000000 0.00004000
```

```
p.adjust(my_pvalues, method = "bonferroni") # multiply by number of tests
```

```
[1] 0.19600 0.00400 1.00000 0.00004
```

```
my_pvalues * 4
```

```
[1] 0.19600 0.00400 1.24000 0.00004
```

See here for more about multiple testing correction. Bonferroni also often done as p-value threshold divided by number of tests (0.05/test number).

Some other statistical tests

- `wilcox.test()` – Wilcoxon signed rank test, Wilcoxon rank sum test
- `shapiro.test()` – Shapiro-Wilk test of normality
- `ks.test()` – Kolmogorov-Smirnov test
- `var.test()`– Fisher’s F-Test
- `chisq.test()` – Chi-squared test
- `aov()` – Analysis of Variance (ANOVA)

Summary

- use `cor()` to calculate correlation between two vectors, `cor.test()` can give more information
- `corrplot()` is nice for a quick visualization!
- `t.test()` one sample test to test the difference in mean of a single vector from zero (one input)
- `t.test()` two sample test to test the difference in means between two vectors (two inputs)
- `tidy()` in the `broom` package is useful for organizing and saving statistical test output
- remember to adjust p-values with `p.adjust()` when doing multiple tests on data

Lab Part 1

- Class Website
- Lab

Regression

Linear regression

Linear regression is a method to model the relationship between a response and one or more explanatory variables.

Most commonly used statistical tests are actually specialized regressions, including the two sample t-test, see here for more.

Linear regression notation

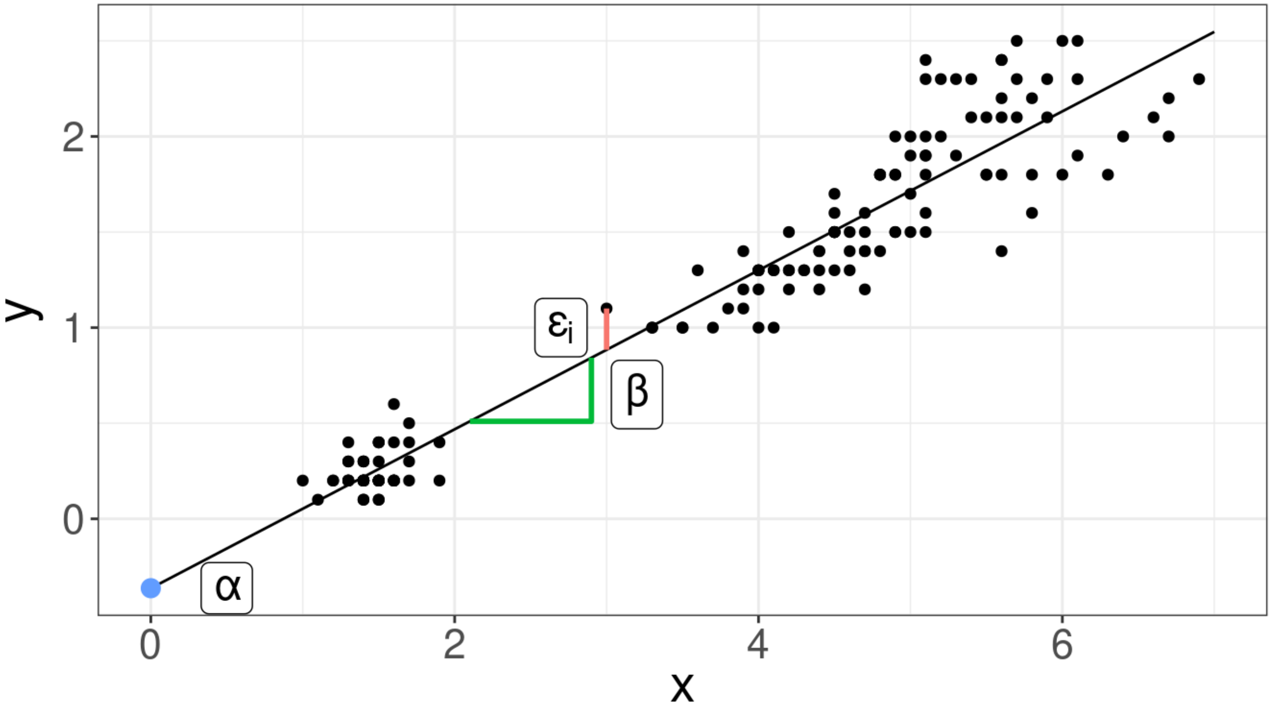
Here is some of the notation, so it is easier to understand the commands/results.

$$y_i = \alpha + \beta x_i + \varepsilon_i$$

where:

- y_i is the outcome for person i
- α is the intercept
- β is the slope (also called a coefficient) - the mean change in y that we would expect for one unit change in x (“rise over run”)
- x_i is the predictor for person i
- ε_i is the residual variation for person i

Linear regression



Linear regression

Linear regression is a method to model the relationship between a response and one or more explanatory variables.

We provide a little notation here so some of the commands are easier to put in the proper context.

$$y_i = \alpha + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \varepsilon_i$$

where:

- y_i is the outcome for person i
- α is the intercept
- β_1, β_2, β_2 are the slopes/coefficients for variables x_{i1}, x_{i2}, x_{i3} - average difference in y for a unit change (or each value) in x while accounting for other variables
- x_{i1}, x_{i2}, x_{i3} are the predictors for person i
- ε_i is the residual variation for person i

See this case study for more details.

Linear regression fit in R

To fit regression models in R, we use the function `glm()` (Generalized Linear Model).

You may also see `lm()` which is a more limited function that only allows for normally/Gaussian distributed error terms (aka typical linear regressions).

We typically provide two arguments:

- `formula` – model formula written using names of columns in our data
- `data` – our data frame

Linear regression fit in R: model formula

Model formula

$$y_i = \alpha + \beta x_i + \varepsilon_i$$

In R translates to

```
y ~ x
```

Linear regression fit in R: model formula

Model formula

$$y_i = \alpha + \beta x_i + \varepsilon_i$$

In R translates to

```
y ~ x
```

In practice, `y` and `x` are replaced with the **names of columns from our data set**.

For example, if we want to fit a regression model where outcome is `income` and predictor is `years_of_education`, our formula would be:

```
income ~ years_of_education
```

Linear regression fit in R: model formula

Model formula

$$y_i = \alpha + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \varepsilon_i$$

In R translates to

```
y ~ x1 + x2 + x3
```

In practice, `y` and `x1`, `x2`, `x3` are replaced with the **names of columns from our data set**.

For example, if we want to fit a regression model where outcome is `income` and predictors are `years_of_education`, `age`, and `location` then our formula would be:

```
income ~ years_of_education + age + location
```

Linear regression

We will use data about emergency room doctor complaints.

“Data was recorded on 44 doctors working in an emergency service at a hospital to study the factors affecting the number of complaints received.”

```
# install.packages("faraway")
library(faraway)
data(esdcomp)
esdcomp
```

```
   visits complaints residency gender revenue   hours
1    2014          2         Y      F  263.03 1287.25
2    3091          3         N      M  334.94 1588.00
3     879          1         Y      M  206.42  705.25
4    1780          1         N      M  226.32 1005.50
5    3646         11         N      M  288.91 1667.25
6    2690          1         N      M  275.94 1517.75
7    1864          2         Y      M  295.71  967.00
8    2782          6         N      M  224.91 1609.25
9    3071          9         N      F  249.32 1747.75
10   1502          3         Y      M  269.00  906.25
11   2438          2         N      F  225.61 1787.75
12   2278          2         N      M  212.43 1480.50
13   2458          5         N      M  211.05 1733.50
14   2269          2         N      F  213.23 1847.25
15   2431          7         N      M  257.30 1433.00
16   3010          2         Y      M  326.49 1520.00
17   2234          5         Y      M  290.53 1404.75
18   2906          4         N      M  268.73 1608.50
```

Linear regression: model fitting

We fit linear regression model with the number of patient visits (`visits`) as an outcome and total number of hours worked (`hours`) as a predictor. In other words, we are evaluation if the number of hours worked is predictive of the number of visits a doctor had.

```
fit <- glm(visits ~ hours, data = esdcomp)
fit

Call:  glm(formula = visits ~ hours, data = esdcomp)

Coefficients:
(Intercept)        hours
    140.288        1.584

Degrees of Freedom: 43 Total (i.e. Null);  42 Residual
Null Deviance:      16920000
Residual Deviance: 5383000  AIC: 646.3
```

Linear regression: model summary

The `summary()` function returns a list that shows us some more detail.

```
summary(fit)
```

```
Call:
glm(formula = visits ~ hours, data = esdcomp)

Coefficients:
            Estimate Std. Error t value         Pr(>|t|)
(Intercept)  140.288    242.723   0.578            0.566
hours          1.584      0.167   9.488 0.00000000000526 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 128155.3)

    Null deviance: 16919101  on 43  degrees of freedom
Residual deviance:  5382524  on 42  degrees of freedom
AIC: 646.3

Number of Fisher Scoring iterations: 2
```

tidy results

The broom package can help us here too!

The estimate is the coefficient or slope – for one change in hours worked (1 hour increase), we see 1.58 more visits. The error for this estimate is relatively small at 0.167. This relationship appears to be significant with a small p-value <0.001.

```
tidy(fit) %>% glimpse()
```

```
Rows: 2
Columns: 5
$ term      <chr> "(Intercept)", "hours"
$ estimate  <dbl> 140.28841, 1.58408
$ std.error <dbl> 242.7225866, 0.1669579
$ statistic <dbl> 0.5779784, 9.4879004
$ p.value   <dbl> 0.566364879085634154, 0.000000000005262224
```

Linear regression: multiple predictors

Let’s try adding another explanatory variable to our model, dollars per hour earned by the doctor (`revenue`). The meaning of coefficients is more complicated here.

```
fit2 <- glm(visits ~ hours + revenue, data = esdcomp)
summary(fit2)
```

```
Call:
glm(formula = visits ~ hours + revenue, data = esdcomp)

Coefficients:
              Estimate Std. Error t value             Pr(>|t|)    
(Intercept) -2078.1369   327.9157  -6.337        0.00000014326 ***
hours           1.6179     0.1081  14.968 < 0.0000000000000002 ***
revenue         8.3437     1.0828   7.706        0.00000000169 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 53620.97)

    Null deviance: 16919101  on 43  degrees of freedom
Residual deviance:  2198460  on 41  degrees of freedom
AIC: 608.91

Number of Fisher Scoring iterations: 2
```

Linear regression: multiple predictors

Can also use `tidy` and `glimpse` to see the output nicely.

```
fit2 %>%
  tidy() %>%
  glimpse()
```

```
Rows: 3
Columns: 5
$ term      <chr> "(Intercept)", "hours", "revenue"
$ estimate  <dbl> -2078.136879, 1.617854, 8.343689
$ std.error <dbl> 327.9156731, 0.1080845, 1.0827657
$ statistic <dbl> -6.337412, 14.968422, 7.705904
$ p.value   <dbl> 0.00000014326245193176067, 0.00000000000000000324554, 0.0000…
```

Linear regression: factors

Factors get special treatment in regression models – lowest level of the factor is the comparison group, and all other factors are **relative** to its values.

`residency` takes values Y or N to indicate whether the doctor is a resident.

```
esdcomp %>% count(residency)
```

```
  residency  n
1         N 24
2         Y 20
```

Linear regression: factors

Yes relative to No – baseline is No

```
fit_3 <- glm(visits ~ residency, data = esdcomp)
summary(fit_3)
```

```
Call:
glm(formula = visits ~ residency, data = esdcomp)

Coefficients:
            Estimate Std. Error t value            Pr(>|t|)    
(Intercept)   2510.8      126.3   19.87 <0.0000000000000002 ***
residencyY    -275.5      187.4   -1.47               0.149    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 383122.6)

    Null deviance: 16919101  on 43  degrees of freedom
Residual deviance: 16091148  on 42  degrees of freedom
AIC: 694.49

Number of Fisher Scoring iterations: 2
```

Linear regression: factors

Comparison group is not listed – treated as intercept. All other estimates are relative to the intercept.

```r
circ <- read_csv("https://jhudatascience.org/intro_to_r/data/Charm_City_Circulator_Ridership.csv")
fit_4 <- glm(orangeBoardings ~ factor(day), data = circ)
summary(fit_4)
```

```

Call:
glm(formula = orangeBoardings ~ factor(day), data = circ)

Coefficients:
                     Estimate Std. Error t value             Pr(>|t|)    
(Intercept)           3744.04      89.09  42.027 < 0.0000000000000002 ***
factor(day)Monday     -667.67     125.99  -5.300     0.00000014090070 ***
factor(day)Saturday   -883.37     126.60  -6.978     0.00000000000525 ***
factor(day)Sunday    -1865.57     127.02 -14.687 < 0.0000000000000002 ***
factor(day)Thursday   -528.83     126.39  -4.184     0.00003099042385 ***
factor(day)Tuesday    -591.25     126.19  -4.685     0.00000315254564 ***
factor(day)Wednesday  -487.93     126.39  -3.860              0.00012 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1238057)

    Null deviance: 1627179072  on 1078  degrees of freedom
Residual deviance: 1327197363  on 1072  degrees of freedom
  (67 observations deleted due to missingness)
AIC: 18208

Number of Fisher Scoring iterations: 2
```

Linear regression: factors

Relative to the level is not listed.

```
circc <- circ %>% mutate(day = factor(day,
  levels =
    c(
      "Monday", "Tuesday", "Wednesday",
      "Thursday", "Friday", "Saturday", "Sunday"
    )
))
fit_5 <- glm(orangeBoardings ~ day, data = circ)
summary(fit_5)
```

```

Call:
glm(formula = orangeBoardings ~ day, data = circ)

Coefficients:
             Estimate Std. Error t value             Pr(>|t|)    
(Intercept)   3076.37      89.09  34.533 < 0.0000000000000002 ***
dayTuesday      76.42     126.19   0.606               0.5449    
dayWednesday   179.73     126.39   1.422               0.1553    
dayThursday    138.84     126.39   1.098               0.2723    
dayFriday      667.67     125.99   5.300          0.000000141 ***
daySaturday   -215.71     126.60  -1.704               0.0887 .  
daySunday    -1197.91     127.02  -9.431 < 0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1238057)

    Null deviance: 1627179072  on 1078  degrees of freedom
Residual deviance: 1327197363  on 1072  degrees of freedom
  (67 observations deleted due to missingness)
AIC: 18208

Number of Fisher Scoring iterations: 2
```

Linear regression: factors

You can view estimates for the comparison group by removing the intercept in the GLM formula

```
y ~ x - 1
```

Caveat is that the p-values change, and interpretation is often confusing.

```
fit_6 <- glm(orangeBoardings ~ factor(day) - 1, data = circ)
summary(fit_6)
```

```

Call:
glm(formula = orangeBoardings ~ factor(day) - 1, data = circ)

Coefficients:
                     Estimate Std. Error t value            Pr(>|t|)    
factor(day)Monday     3076.37      89.09   34.53 <0.0000000000000002 ***
factor(day)Tuesday    3152.79      89.37   35.28 <0.0000000000000002 ***
factor(day)Wednesday  3256.10      89.66   36.31 <0.0000000000000002 ***
factor(day)Thursday   3215.21      89.66   35.86 <0.0000000000000002 ***
factor(day)Friday     3744.04      89.09   42.03 <0.0000000000000002 ***
factor(day)Saturday   2860.67      89.95   31.80 <0.0000000000000002 ***
factor(day)Sunday     1878.46      90.55   20.75 <0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1238057)

    Null deviance: 11540692004  on 1079  degrees of freedom
Residual deviance:  1327197363  on 1072  degrees of freedom
  (67 observations deleted due to missingness)
AIC: 18208

Number of Fisher Scoring iterations: 2
```

Linear regression: interactions

You can also specify interactions between variables in a formula `y ~ x1 + x2 + x1 * x2`. This allows for not only the intercepts between factors to differ, but also the slopes with regard to the interacting variable.

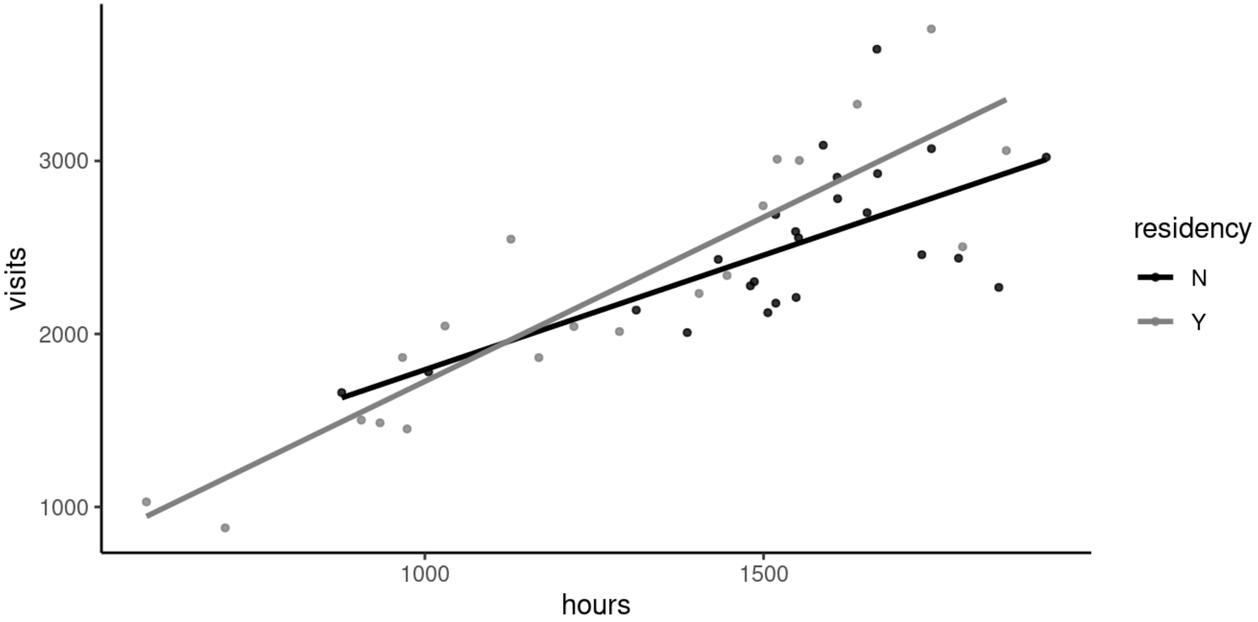
```
fit_7 <- glm(visits ~ hours + residency + hours * residency, data = esdcomp)
tidy(fit_7)
```

```
# A tibble: 4 × 5
  term             estimate std.error statistic  p.value
  <chr>               <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)       469.      481.        0.976 0.335
2 hours               1.32      0.308     4.30  0.000108
3 residencyY       -642.      559.       -1.15  0.258
4 hours:residencyY    0.574     0.377     1.52  0.136
```

Linear regression: interactions

By default, `ggplot` with a factor added as a color will look include the interaction term. Notice the different intercept and slope of the lines.

```
ggplot(esdcomp, aes(x = hours, y = visits, color = residency)) +
  geom_point(size = 1, alpha = 0.8) +
  geom_smooth(method = "glm", se = FALSE) +
  scale_color_manual(values = c("black", "grey50")) +
  theme_classic()
```



Generalized linear models (GLMs)

Generalized linear models (GLMs) allow for fitting regressions for non-continuous/normal outcomes. Examples include: logistic regression, Poisson regression.

Add the `family` argument – a description of the error distribution and link function to be used in the model. These include:

- `binomial(link = "logit")` - outcome is binary
- `poisson(link = "log")` - outcome is count or rate
- others

Very important to use the right test!

See this case study for more information.

See `?family` documentation for details of family functions.

Logistic regression

We will use data about breast cancer tumors.

“Data come from a study of breast cancer in Wisconsin. There are 681 cases of potentially cancerous tumors of which 238 are actually malignant. Determining whether a tumor is really malignant is traditionally determined by an invasive surgical procedure. The purpose of this study was to determine whether a new procedure called fine needle aspiration which draws only a small sample of tissue could be effective in determining tumor status.”

```
data(wbca)
head(wbca)
```

```
  Class Adhes BNucl Chrom Epith Mitos NNucl Thick UShap USize
1     1     1     1     3     2     1     1     5     1     1
2     1     5    10     3     7     1     2     5     4     4
3     1     1     2     3     2     1     1     3     1     1
4     1     1     4     3     3     1     7     6     8     8
5     1     3     1     3     2     1     1     4     1     1
6     0     8    10     9     7     1     7     8    10    10
```

Logistic regression

`Class` is a 0/1-valued variable indicating if the tumor was malignant (0 if malignant, 1 if benign).

```
# General format
glm(y ~ x, data = DATASET_NAME, family = binomial(link = "logit"))
```

```
binom_fit <- glm(Class ~ UShap + USize, data = wbca, family = binomial(link = "logit"))
summary(binom_fit)
```

```
Call:
glm(formula = Class ~ UShap + USize, family = binomial(link = "logit"), 
    data = wbca)

Coefficients:
            Estimate Std. Error z value             Pr(>|z|)    
(Intercept)   5.6868     0.4359  13.047 < 0.0000000000000002 ***
UShap        -0.8431     0.1593  -5.292          0.000000121 ***
USize        -0.8686     0.1690  -5.139          0.000000277 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 881.39  on 680  degrees of freedom
Residual deviance: 218.28  on 678  degrees of freedom
AIC: 224.28

Number of Fisher Scoring iterations: 7
```

Odds ratios

An odds ratio (OR) is a measure of association between an exposure and an outcome. The OR represents the odds that an outcome will occur given a particular exposure, compared to the odds of the outcome occurring in the absence of that exposure.

Check out this paper.

Use `oddsratio(x, y)` from the `epitools()` package to calculate odds ratios.

Odds ratios

This data shows whether people became ill after eating ice cream in the 1940s.

```r
# install.packages(epitools)
library(epitools)
data(oswego)
ice_cream <-
  oswego %>%
  select(ill, vanilla.ice.cream) %>%
  mutate(
    ill = recode(ill, "Y" = 1, "N" = 0),
    vanilla.ice.cream = recode(vanilla.ice.cream, "Y" = 1, "N" = 0)
  )
```

Odds ratios

```
head(ice_cream)
```

```
  ill vanilla.ice.cream
1   1                 1
2   1                 1
3   1                 1
4   1                 1
5   1                 1
6   1                 1
```

```
ice_cream %>% count(ill, vanilla.ice.cream)
```

```
  ill vanilla.ice.cream  n
1   0                 0 18
2   0                 1 11
3   1                 0  3
4   1                 1 43
```

Odds ratios

```
response <- ice_cream %>% pull(ill)
predictor <- ice_cream %>% pull(vanilla.ice.cream)
oddsratio(predictor, response)
```

```
$data
         Outcome
Predictor  0  1 Total
    0     18  3    21
    1     11 43    54
    Total 29 46    75

$measure
         odds ratio with 95% C.I.
Predictor estimate    lower    upper
        0  1.00000       NA       NA
        1 21.40719 5.927963 109.4384

$p.value
         two-sided
Predictor      midp.exact    fisher.exact      chi.square
        0              NA              NA              NA
        1 0.0000002698215 0.0000002597451 0.0000001813314

$correction
[1] FALSE

attr(,"method")
[1] "median-unbiased estimate & mid-p exact CI"
```

See this case study for more information.

Odds ratios

The odds ratio is 21.4. When the predictor is TRUE (aka the individual ate vanilla ice cream), the odds of the response (having acute GI illness) were 21 times great than when it is FALSE (did not eat ice cream).

Functions you might also see

- the `stat_cor()` function in the `ggpubr` can add correlation coefficients and p-values as a layer to `ggplot` objects
- the `pairs()` (`graphics` package) or `ggpairs()` (`GGally` package) functions are also useful for visualizing correlations across variables in a data frame
- `acf()` in the `stats` package can compute autocorrelation and cross-correlation with lags
- calculate confidence intervals for intercept and slopes from `glm`/`lm` objects using `confint()`
- principal components analysis – use `prcomp()`

There's an R package for everything

Final note

Some final notes:

- Researcher's responsibility to **understand the statistical method** they use – underlying assumptions, correct interpretation of method results
- Researcher's responsibility to **understand the R software** they use – meaning of function's arguments and meaning of function's output elements

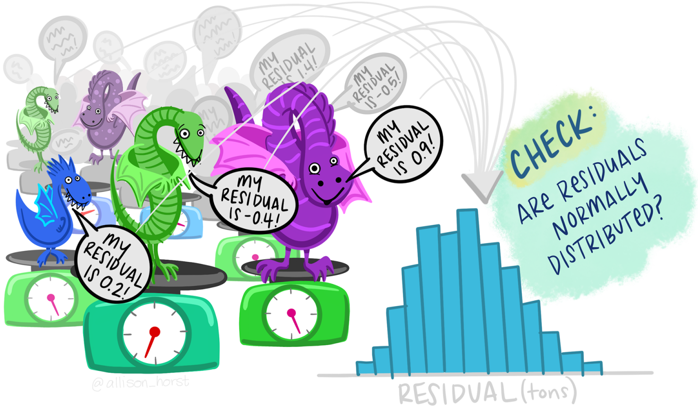


Image by Allison Horst.

Summary

- `glm()` fits regression models:
  - Use the `formula =` argument to specify the model (e.g., `y ~ x` or `y ~ x1 + x2` using column names)
  - Use `data =` to indicate the dataset
  - Use `family =` to do a other regressions like logistic, Poisson and more
  - `summary()` gives useful statistics
- `oddsratio()` from the `epitools` package can calculate odds ratios (outside of logistic regression - which allows more than one explanatory variable)
- this is just the tip of the iceberg!

Resources (also on the website!)

For more check out:

- this chapter on modeling in this tidyverse book
- this chart on when to do what test
- opencasestudies.org

For classes at JHU School of Public Health:

- PH.140.621, PH.140.622, PH.140.623, PH.140.62 - Statistical Methods in Public Health I, II, III, and IV - The class is mostly taught in STATA, but you can also join a group of students working in R. The class covers many topics in statistical analysis for public health data.
- PH.140.778 - Statistical Computing, Algorithm, and Software Development - A more advanced course for working with data in R. Content for similar topics as this course can also be found on Leanpub.

Lab Part 2

- Class Website
- Lab
- Day 8 Cheatsheet

Image by Gerd Altmann from Pixabay