Statistics

Summary

- `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

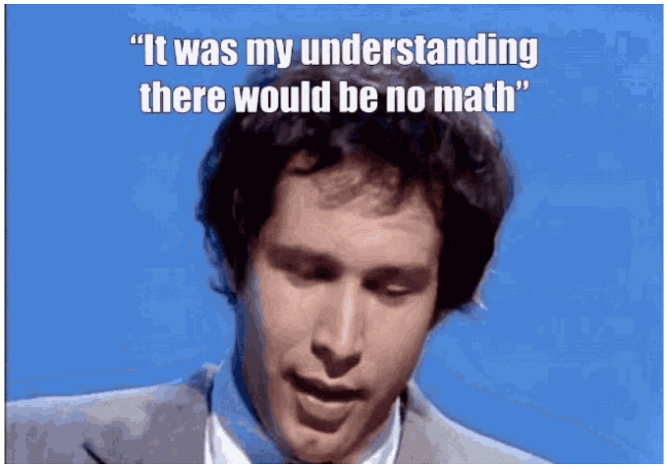
Summary

- 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.

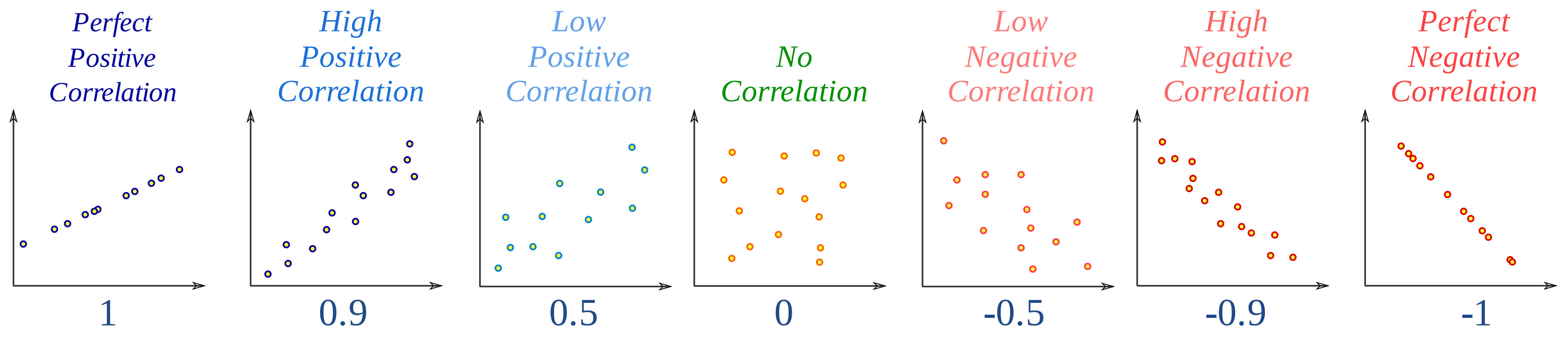
Check out www.opencasestudies.org for deeper dives on some of the concepts covered here and the resource page for more resources.

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)

Correlation

https://daseh.org/data/Yearly_CO2_Emissions_1000_tonnes.csv

```
library(dasehr)

head(yearly_co2_emissions)
```

```
# A tibble: 6 × 265
  country   `1751` `1752` `1753` `1754` `1755` `1756` `1757` `1758` `1759` `1760`
  <chr>      <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 Afghani…      NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
2 Albania       NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
3 Algeria       NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
4 Andorra       NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
5 Angola        NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
6 Antigua…      NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
# ℹ 254 more variables: `1761` <dbl>, `1762` <dbl>, `1763` <dbl>, `1764` <dbl>,
#   `1765` <dbl>, `1766` <dbl>, `1767` <dbl>, `1768` <dbl>, `1769` <dbl>,
#   `1770` <dbl>, `1771` <dbl>, `1772` <dbl>, `1773` <dbl>, `1774` <dbl>,
#   `1775` <dbl>, `1776` <dbl>, `1777` <dbl>, `1778` <dbl>, `1779` <dbl>,
#   `1780` <dbl>, `1781` <dbl>, `1782` <dbl>, `1783` <dbl>, `1784` <dbl>,
#   `1785` <dbl>, `1786` <dbl>, `1787` <dbl>, `1788` <dbl>, `1789` <dbl>,
#   `1790` <dbl>, `1791` <dbl>, `1792` <dbl>, `1793` <dbl>, `1794` <dbl>, …
```

Correlation for two vectors

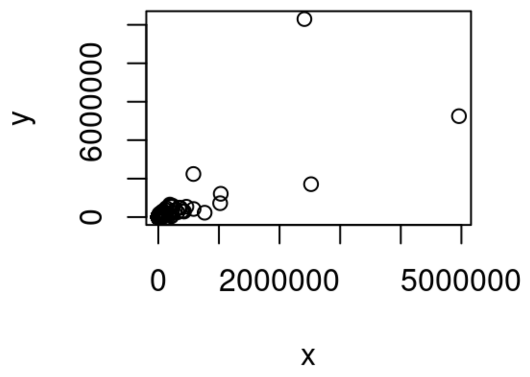
First, we compute correlation by providing two vectors.

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

```
# x and y must be numeric vectors
x <- pull(yearly_co2_emissions, `1989`)

y <- pull(yearly_co2_emissions, `2014`)

# 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] 0.7644918
```

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

```

	Pearson's product-moment correlation

data:  x and y
t = 15.463, df = 170, p-value < 0.00000000000000022
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.6942789 0.8202897
sample estimates:
      cor 
0.7644918 
```

Broom package

The `broom` package helps make stats results look tidy

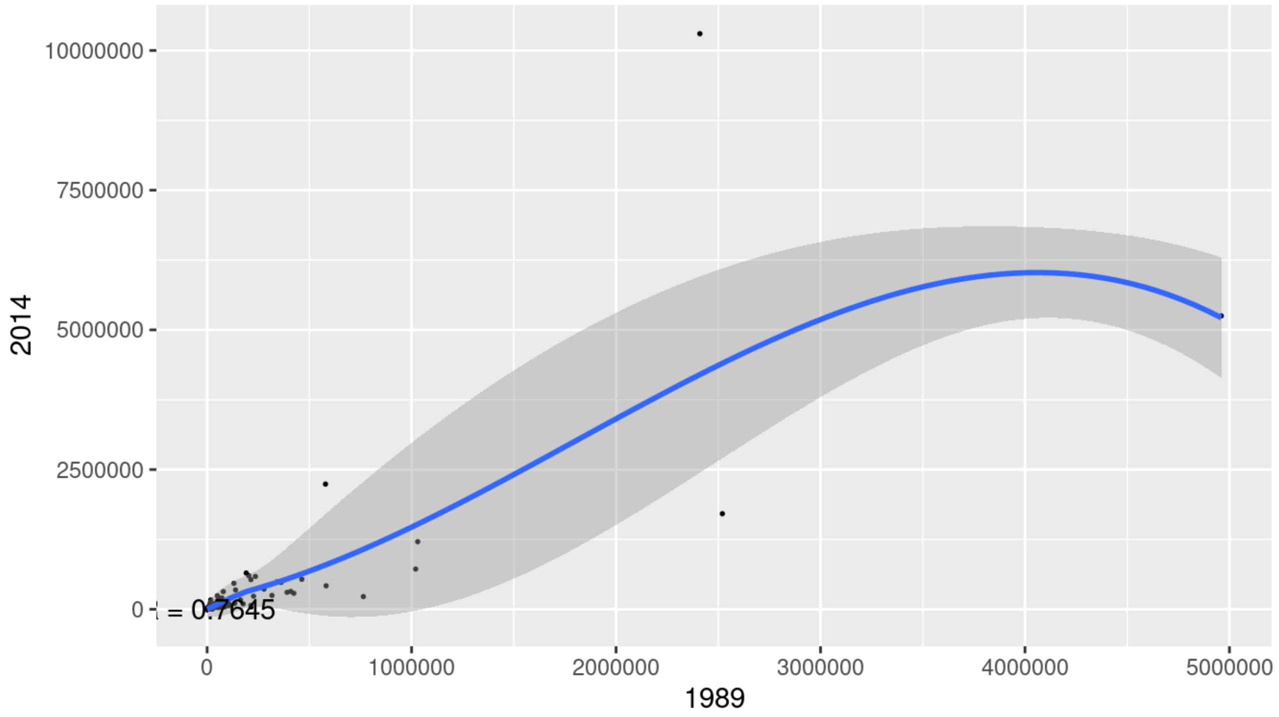
```
cor_result <- tidy(cor.test(x, y))
glimpse(cor_result)
```

```
Rows: 1
Columns: 8
$ estimate    <dbl> 0.7644918
$ statistic   <dbl> 15.46267
$ p.value     <dbl> 0.00000000000000000000000000000000030487
$ parameter   <int> 170
$ conf.low    <dbl> 0.6942789
$ conf.high   <dbl> 0.8202897
$ 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)
yearly_co2_emissions %>%
  ggplot(aes(x = `1989`, y = `2014`)) +
  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!

```
co2_subset <- yearly_co2_emissions %>% select(c(`1909`, `1929`, `1949`, `1969`,
head(co2_subset)
```

```
# A tibble: 6 × 6
  `1909` `1929` `1949` `1969` `1989` `2009`
   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1     NA   NA     14.7    942   2780   6770
2     NA   NA   1020     3250   8980   4380
3     NA   80.7  909    11300  80000 121000
4     NA   NA     NA       NA     NA    517
5     NA   NA     NA     2790   5010  27800
6     NA   NA     NA     1260    286    510
```

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"*.

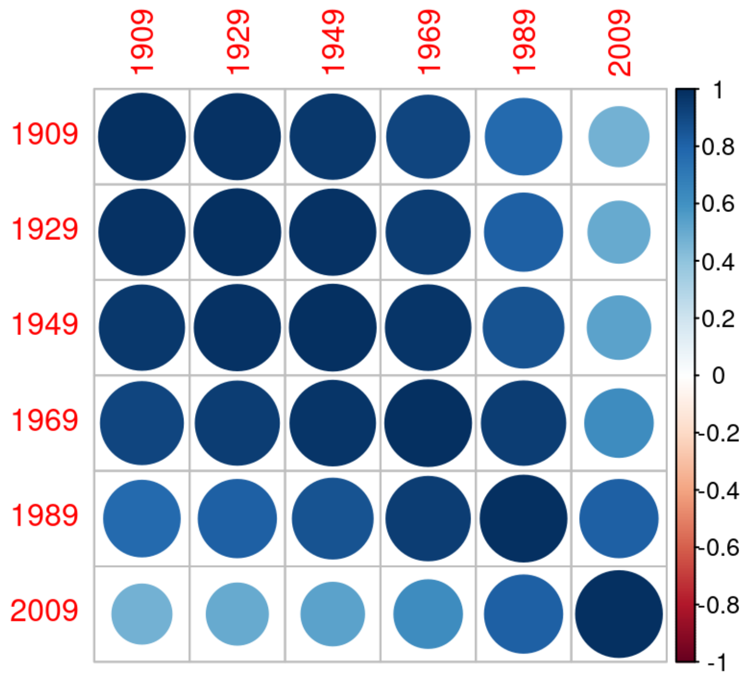
```
cor_mat <- cor(co2_subset, use = "complete.obs")
cor_mat
```

```
          1909      1929      1949      1969      1989      2009
1909 1.0000000 0.9843659 0.9654205 0.9134400 0.7770616 0.4752203
1929 0.9843659 1.0000000 0.9870474 0.9401504 0.8115660 0.5085648
1949 0.9654205 0.9870474 1.0000000 0.9720538 0.8659073 0.5327061
1969 0.9134400 0.9401504 0.9720538 1.0000000 0.9451710 0.6215263
1989 0.7770616 0.8115660 0.8659073 0.9451710 1.0000000 0.8110514
2009 0.4752203 0.5085648 0.5327061 0.6215263 0.8110514 1.0000000
```

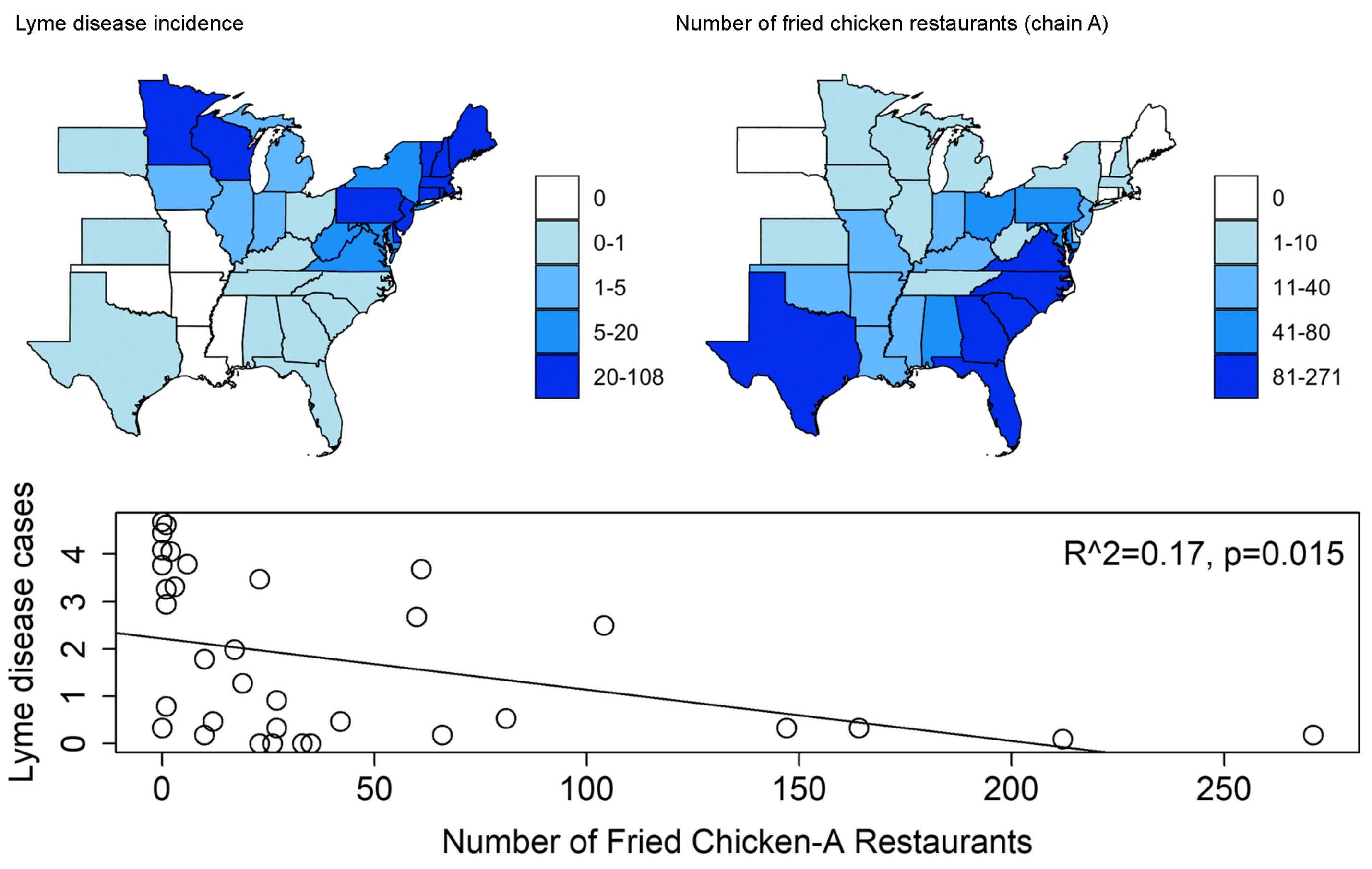
Correlation for data frame columns with plot

`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

Let’s look at the dataset of haloacetic acid levels in public water sources in Washington, saved as `haa5` in the `dasehr` package.

```
x <- haa5 %>% pull(perc_pop_exposed_to_exceedances)
sum(is.na(x)) # count NAs in x
```

```
[1] 11
```

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

```

	One Sample t-test

data:  x
t = 3.7753, df = 21, p-value = 0.00111
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 0.02654098 0.09164084
sample estimates:
 mean of x 
0.05909091 
```

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

```
x <- haa5 %>% pull(pop_on_sampled_PWS)
y <- haa5 %>% pull(pop_exposed_to_exceedances)

sum(is.na(x))

[1] 11

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

[1] 11

t.test(x, y)

    Welch Two Sample t-test

data:  x and y
t = 12.23, df = 21, p-value = 0.00000000005122
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 3501529 4936249
sample estimates:
  mean of x   mean of y
4221499.045    2610.364
```

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> 4218889
$ estimate1   <dbl> 4221499
$ estimate2   <dbl> 2610.364
$ statistic   <dbl> 12.23048
$ p.value     <dbl> 0.00000000005122359
$ parameter   <dbl> 21.00014
$ conf.low    <dbl> 3501529
$ conf.high   <dbl> 4936249
$ 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:

```r
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
```

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

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

```r
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 test
- `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

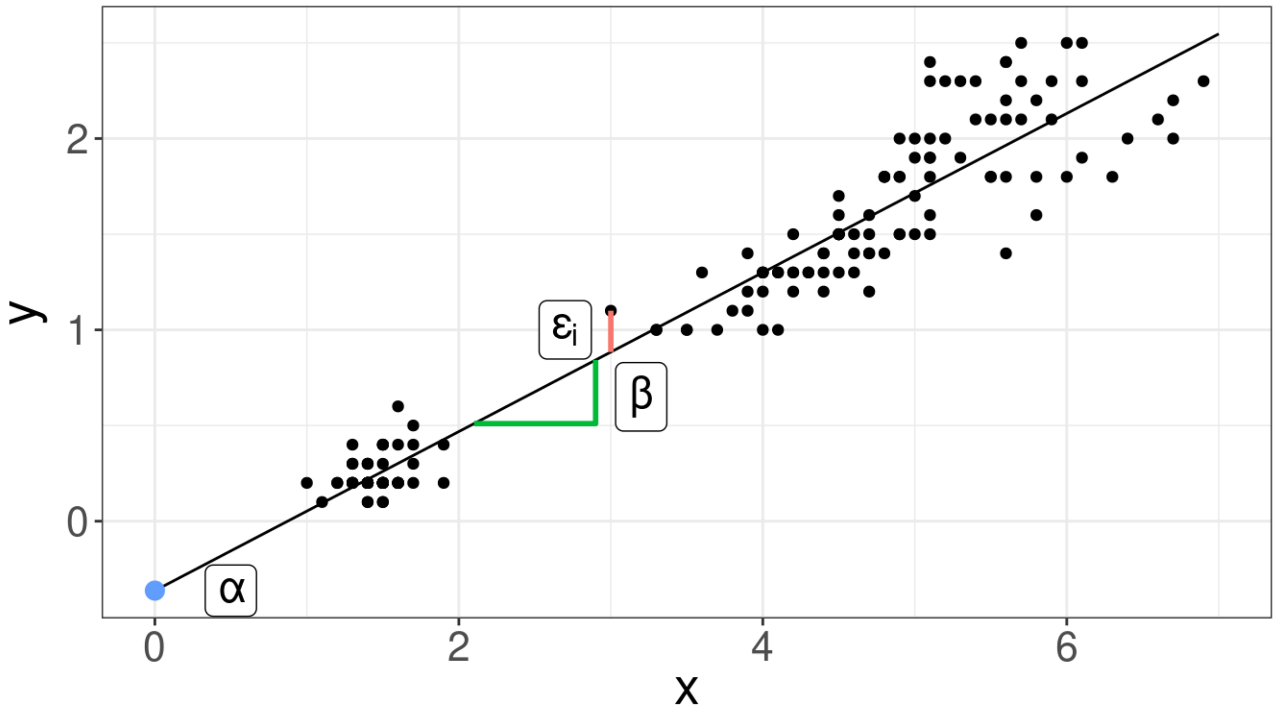
Here is some of the notation, so it is easier to understand the commands/results.

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

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 our dataset about nitrate levels by quarter in public water sources in Washington. We'll load a slightly different version of this dataset, which can be found at "https://daseh.org/data/Nitrate_Exposure_for_WA_Public_Water_Systems_byquarter_v2

```
nitrate <- read_csv(file = "https://daseh.org/data/Nitrate_Exposure_for_WA_Publi

head(nitrate)
```

```
# A tibble: 6 × 12
   year quarter half_of_year pop_on_sampled_PWS `pop_0-3ug/L` `pop_>3-5ug/L`
  <dbl> <chr>   <chr>                     <dbl>         <dbl>          <dbl>
1  1999 Q1      first                    106720         67775              0
2  2000 Q1      first                     34793          5904              0
3  2001 Q1      first                     90054         49552            150
4  2002 Q1      first                    293486        223488           1474
5  2003 Q1      first                   1473586        743676         323767
6  2004 Q1      first                   1272283        486491         529354
# ℹ 6 more variables: `pop_>5-10ug/L` <dbl>, `pop_>10-20ug/L` <dbl>,
#   `pop_>20ug/L` <dbl>, `pop_on_PWS_with_non-detect` <dbl>,
#   pop_exposed_to_exceedances <dbl>, perc_pop_exposed_to_exceedances <dbl>
```

Linear regression: model fitting

The upper limit of acceptable nitrates in water is 10 ug/L. We fit linear regression model with the number of people on public water systems with more than the limit of nitrates (`pop_exposed_to_exceedances`) as an outcome and the number of people exposed to 3-5 ug/L of nitrates (`pop_>3-5ug/L`) as a predictor. In other words, we are evaluating if the number of people exposed to low levels of nitrates in their water is predictive of the number of people exposed to excess nitrates in their water.

```
fit <- glm(pop_exposed_to_exceedances ~ `pop_>3-5ug/L`, data = nitrate)
fit
```

```
Call:  glm(formula = pop_exposed_to_exceedances ~ `pop_>3-5ug/L`, data = nitrate

Coefficients:
   (Intercept)  `pop_>3-5ug/L`  
    837.633357        0.002052  

Degrees of Freedom: 87 Total (i.e. Null);  86 Residual
Null Deviance:      166900000 
Residual Deviance: 149100000    AIC: 1518
```

Linear regression: model summary

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

```
summary(fit)
```

```

Call:
glm(formula = pop_exposed_to_exceedances ~ `pop_>3-5ug/L`, data = nitrate)

Coefficients:
                  Estimate  Std. Error t value Pr(>|t|)
(Intercept)    837.6333569 268.5458637   3.119  0.00247 **
`pop_>3-5ug/L`   0.0020516   0.0006391   3.210  0.00186 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1733185)

    Null deviance: 166917415  on 87  degrees of freedom
Residual deviance: 149053912  on 86  degrees of freedom
AIC: 1517.9

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)", "`pop_>3-5ug/L`"
$ estimate  <dbl> 837.633356922, 0.002051615
$ std.error <dbl> 268.5458637101, 0.0006390501
$ statistic <dbl> 3.119145, 3.210413
$ p.value   <dbl> 0.002468135, 0.001864329
```

Linear regression: multiple predictors

Let’s try adding another explanatory variable to our model, year (`year`). The tidy function will not work with this unfortunately. The meaning of coefficients is more complicated here.

```
fit2 <- glm(pop_exposed_to_exceedances ~ `pop_>3-5ug/L` + year, data = nitrate)
summary(fit2)
```

```

Call:
glm(formula = pop_exposed_to_exceedances ~ `pop_>3-5ug/L` + year,
    data = nitrate)

Coefficients:
                     Estimate     Std. Error t value   Pr(>|t|)    
(Intercept)    193320.5254972  46164.1477089   4.188 0.00006852 ***
`pop_>3-5ug/L`      0.0033674      0.0006653   5.062 0.00000238 ***
year              -96.0210515     23.0288882  -4.170 0.00007319 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1455811)

    Null deviance: 166917415  on 87  degrees of freedom
Residual deviance: 123743924  on 85  degrees of freedom
AIC: 1503.5

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)", "`pop_>3-5ug/L`", "year"
$ estimate  <dbl> 193320.525497167, 0.003367383, -96.021051499
$ std.error <dbl> 46164.147708904, 0.000665288, 23.028888212
$ statistic <dbl> 4.187677, 5.061541, -4.169591
$ p.value   <dbl> 0.000068518905, 0.000002375999, 0.000073186127
```

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.

`quarter` takes values Q1, Q2, Q3, and Q4 to indicate the quarter of the year for each observation.

```
nitrate %>% count(quarter)
```

```
# A tibble: 4 × 2
  quarter     n
  <chr>   <int>
1 Q1         22
2 Q2         22
3 Q3         22
4 Q4         22
```

Linear regression: factors

The comparison group that is not listed is treated as intercept. All other estimates are relative to the intercept.

```
fit3 <- glm(pop_exposed_to_exceedances ~ `pop_>3-5ug/L` + year + factor(quarter), data = nitrate)
summary(fit3)
```

```

Call:
glm(formula = pop_exposed_to_exceedances ~ `pop_>3-5ug/L` + year + 
    factor(quarter), data = nitrate)

Coefficients:
                        Estimate     Std. Error t value   Pr(>|t|)    
(Intercept)       208033.8242583  47597.8780270   4.371 0.00003603 ***
`pop_>3-5ug/L`         0.0038251      0.0007477   5.115 0.00000202 ***
year                -103.5354110     23.7644678  -4.357 0.00003794 ***
factor(quarter)Q2     12.4362229    366.7866716   0.034      0.973    
factor(quarter)Q3    473.5397889    390.3813390   1.213      0.229    
factor(quarter)Q4    405.3578933    368.5826835   1.100      0.275    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1466474)

    Null deviance: 166917415  on 87  degrees of freedom
Residual deviance: 120250886  on 82  degrees of freedom
AIC: 1507

Number of Fisher Scoring iterations: 2
```

Linear regression: factors

Relative to the level is not listed.

```
nitrate <- nitrate %>% mutate(quarter = factor(quarter,
  levels =
    c(
      "Q1", "Q2", "Q3", "Q4"
    )
))
fit4 <- glm(pop_exposed_to_exceedances ~ `pop_>3-5ug/L` + year + quarter, data = nitrate)
summary(fit4)
```

```

Call:
glm(formula = pop_exposed_to_exceedances ~ `pop_>3-5ug/L` + year +
    quarter, data = nitrate)

Coefficients:
                     Estimate     Std. Error t value   Pr(>|t|)
(Intercept)    208033.8242583  47597.8780270   4.371 0.00003603 ***
`pop_>3-5ug/L`      0.0038251      0.0007477   5.115 0.00000202 ***
year             -103.5354110     23.7644678  -4.357 0.00003794 ***
quarterQ2          12.4362229    366.7866716   0.034      0.973
quarterQ3         473.5397889    390.3813390   1.213      0.229
quarterQ4         405.3578933    368.5826835   1.100      0.275
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1466474)

    Null deviance: 166917415  on 87  degrees of freedom
Residual deviance: 120250886  on 82  degrees of freedom
AIC: 1507

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.

```
fit5 <- glm(pop_exposed_to_exceedances ~ `pop_>3-5ug/L` + year + quarter - 1, data = nitrate)
summary(fit5)
```

```
Call:
glm(formula = pop_exposed_to_exceedances ~ `pop_>3-5ug/L` + year +
    quarter - 1, data = nitrate)

Coefficients:
                     Estimate     Std. Error t value   Pr(>|t|)
`pop_>3-5ug/L`      0.0038251      0.0007477   5.115 0.00000202 ***
year             -103.5354110     23.7644678  -4.357 0.00003794 ***
quarterQ1      208033.8242583  47597.8780270   4.371 0.00003603 ***
quarterQ2      208046.2604813  47580.0330230   4.373 0.00003578 ***
quarterQ3      208507.3640472  47668.7351298   4.374 0.00003558 ***
quarterQ4      208439.1821516  47623.6821872   4.377 0.00003522 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1466474)

    Null deviance: 384570288  on 88  degrees of freedom
Residual deviance: 120250886  on 82  degrees of freedom
AIC: 1507

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.

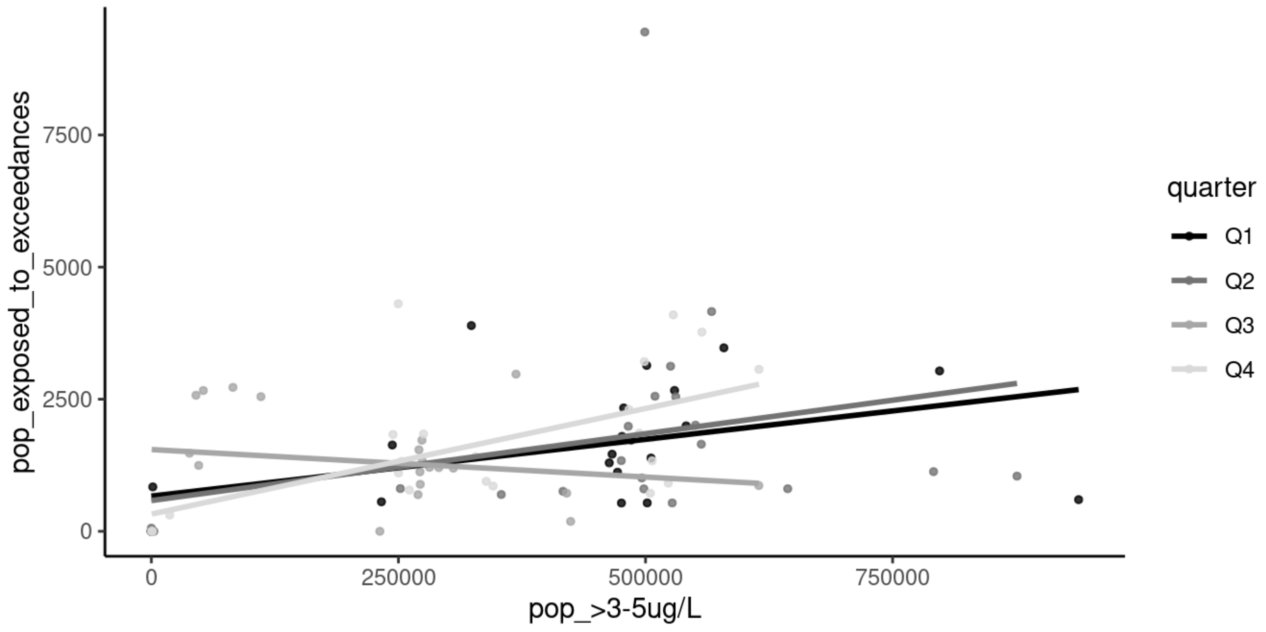
```
fit6 <- glm(pop_exposed_to_exceedances ~ `pop_>3-5ug/L` + year + quarter + year * quarter, data = nitrate)
tidy(fit6)
```

```
# A tibble: 9 × 5
  term                estimate     std.error statistic    p.value
  <chr>                  <dbl>         <dbl>     <dbl>      <dbl>
1 (Intercept)    173920.       89077.            1.95  0.0544
2 `pop_>3-5ug/L`      0.00387      0.000761      5.09  0.00000237
3 year              -86.6         44.4          -1.95  0.0547
4 quarterQ2      163851.      115853.            1.41  0.161
5 quarterQ3       16037.      115668.            0.139 0.890
6 quarterQ4      -36066.      117691.           -0.306 0.760
7 year:quarterQ2    -81.5         57.7          -1.41  0.161
8 year:quarterQ3     -7.74        57.6          -0.134 0.893
9 year:quarterQ4     18.2         58.6           0.310 0.757
```

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(nitrate, aes(x = `pop_>3-5ug/L`, y = pop_exposed_to_exceedances, color = quarter)) +
  geom_point(size = 1, alpha = 0.8) +
  geom_smooth(method = "glm", se = FALSE) +
  scale_color_manual(values = c("black", "grey45", "grey65", "grey85")) +
  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

Let’s look at a logistic regression example. We’ll use the `nitrate` dataset again, but first we’ll need to create a binary variable that tells us whether the percentage of the population exposed to exceedances `perc_pop_exposed_to_exceedances` is greater than 0.1%.

```
nitrate <-
  nitrate %>%
    mutate(
      PercExpMore0.1 = case_when
      (perc_pop_exposed_to_exceedances > 0.1 ~ 1,
        perc_pop_exposed_to_exceedances <= 0.1 ~ 0))
```

Logistic regression

Now that we've created the `PercExpMore0.1` variable (where a `1` indicates the percentage of the population exposed to excessive nitrates is greater than 0.1%), we can run a logistic regression.

Let's explore how `quarter` and `year` might predict `PercExpMore0.1`.

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

```
binom_fit <- glm(PercExpMore0.1 ~ year + quarter, data = nitrate, family = binomial(link = "logit"))
summary(binom_fit)
```

```
Call:
glm(formula = PercExpMore0.1 ~ year + quarter, family = binomial(link = "logit"),
    data = nitrate)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  166.29994  107.61737   1.545   0.1223
year          -0.08285    0.05356  -1.547   0.1219
quarterQ2     -2.21841    0.87930  -2.523   0.0116 *
quarterQ3    -19.46232 2243.38013  -0.009   0.9931
quarterQ4     -1.74955    0.77826  -2.248   0.0246 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 80.363  on 87  degrees of freedom
Residual deviance: 58.693  on 83  degrees of freedom
AIC: 68.693

Number of Fisher Scoring iterations: 18
```

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.

Odds ratios

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

In this case, we're calculating the odds ratio for whether the the first 6 months or last 6 months predicts whether the percentage of population exposed to excess nitrates is more than 0.1%

```
library(epitools)
response <- nitrate %>% pull(PercExpMore0.1)
predictor <- nitrate %>% pull(half_of_year)
oddsratio(predictor, response)
```

```
$data
         Outcome
Predictor  0  1 Total
   first  32 12    44
   second 41  3    44
   Total  73 15    88

$measure
         odds ratio with 95% C.I.
Predictor  estimate      lower     upper
   first  1.0000000         NA        NA
   second 0.2053372 0.04182295 0.7234182

$p.value
         two-sided
Predictor midp.exact fisher.exact chi.square
   first          NA           NA         NA
   second  0.0122317   0.02103458 0.01072943

$correction
[1] FALSE

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

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

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

Content for similar topics as this course can also be found on Leanpub.

Lab Part 2

- Class Website
- Lab

Image by Gerd Altmann from Pixabay