Data Science for Environmental Health

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

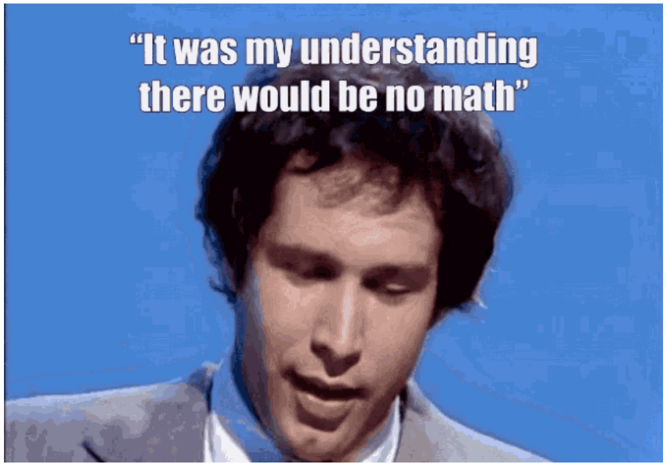
Summary

- 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.

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

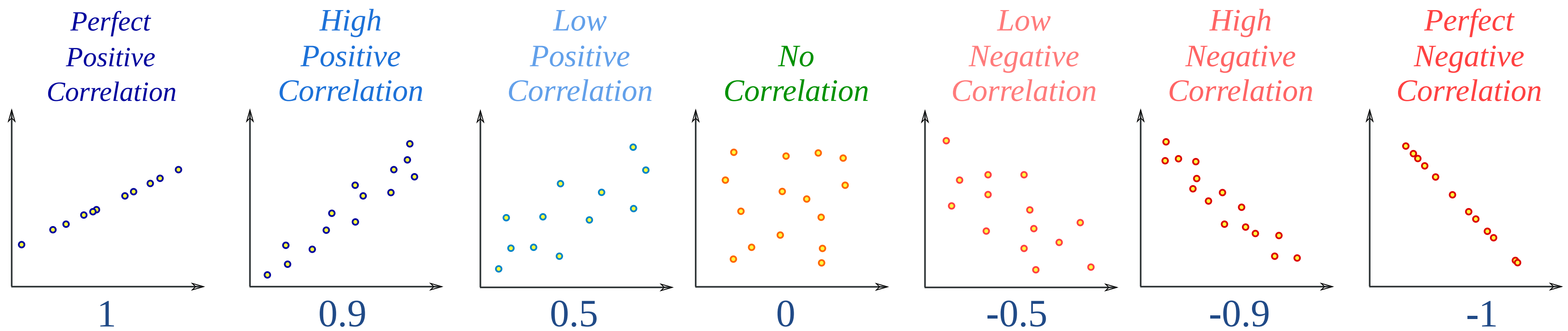
Correlation

Processing math: 100%

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.

Processing math: 100%

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` `176
  <chr>      <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <db
1 Afghani…      NA     NA     NA     NA     NA     NA     NA     NA     NA
2 Albania       NA     NA     NA     NA     NA     NA     NA     NA     NA
3 Algeria       NA     NA     NA     NA     NA     NA     NA     NA     NA
4 Andorra       NA     NA     NA     NA     NA     NA     NA     NA     NA
5 Angola        NA     NA     NA     NA     NA     NA     NA     NA     NA
6 Antigua…      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.

```
# x and y must be numeric vectors
y1980 <- yearly_co2_emissions %>% pull(`1980`)
y1985 <- yearly_co2_emissions %>% pull(`1985`)
```

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.

```
cor(y1980, y1985, use = "complete.obs")
```

```
[1] 0.9936257
```

Correlation coefficient calculation and test

```
cor.test(y1980, y1985)
```

```
    Pearson's product-moment correlation

data:  y1980 and y1985
t = 114.59, df = 169, p-value < 0.00000000000000022
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9913844 0.9952853
sample estimates:
      cor
0.9936257
```

Broom package

The `broom` package helps make stats results look tidy

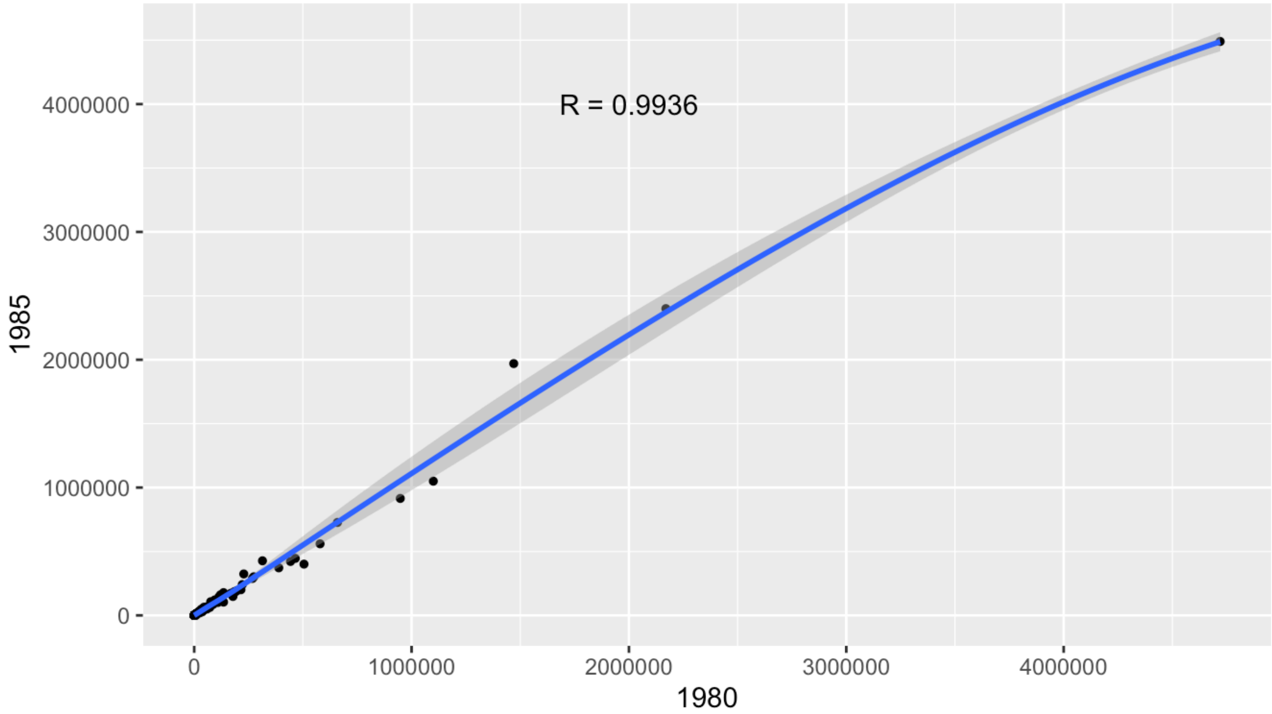
```
library(broom)
cor_result <- tidy(cor.test(y1980, y1985))
glimpse(cor_result)
```

```
Rows: 1
Columns: 8
$ estimate    <dbl> 0.9936257
$ statistic   <dbl> 114.5851
$ p.value     <dbl> 0.00000000000000000000000000000000000000000000000000000000
$ parameter   <int> 169
$ conf.low    <dbl> 0.9913844
$ conf.high   <dbl> 0.9952853
$ 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 = `1980`, y = `1985`)) + geom_point(size = 1) + geom_smooth() +
  annotate("text", x = 2000000, y = 4000000, label = cor_label)
```



Processing math: 100%

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(`1950`, `1980`, `1985`, `2010`))

head(co2_subset)
```

```
# A tibble: 6 × 4
  `1950` `1980` `1985` `2010`
   <dbl>  <dbl>  <dbl>  <dbl>
1   84.3   1760   3510   8460
2  297     5170   7880   4600
3 3790    66500  72800 119000
4   NA       NA     NA    517
5  187     5350   4700  29100
6   NA      143    249    524
```

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”*.

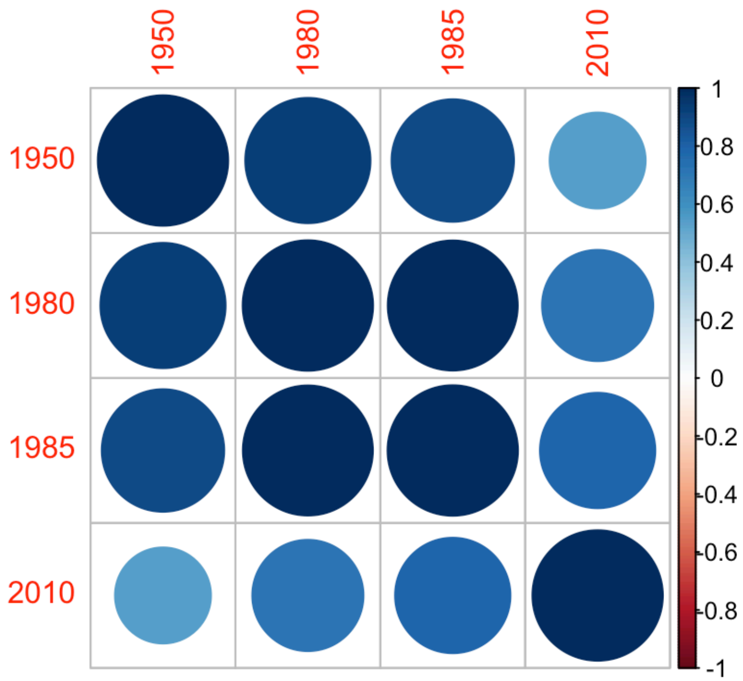
```
cor_mat <- cor(co2_subset, use = "complete.obs")
cor_mat
```

```
          1950      1980      1985      2010
1950 1.0000000 0.9228253 0.8818288 0.5415047
1980 0.9228253 1.0000000 0.9935477 0.7270839
1985 0.8818288 0.9935477 1.0000000 0.7827256
2010 0.5415047 0.7270839 0.7827256 1.0000000
```

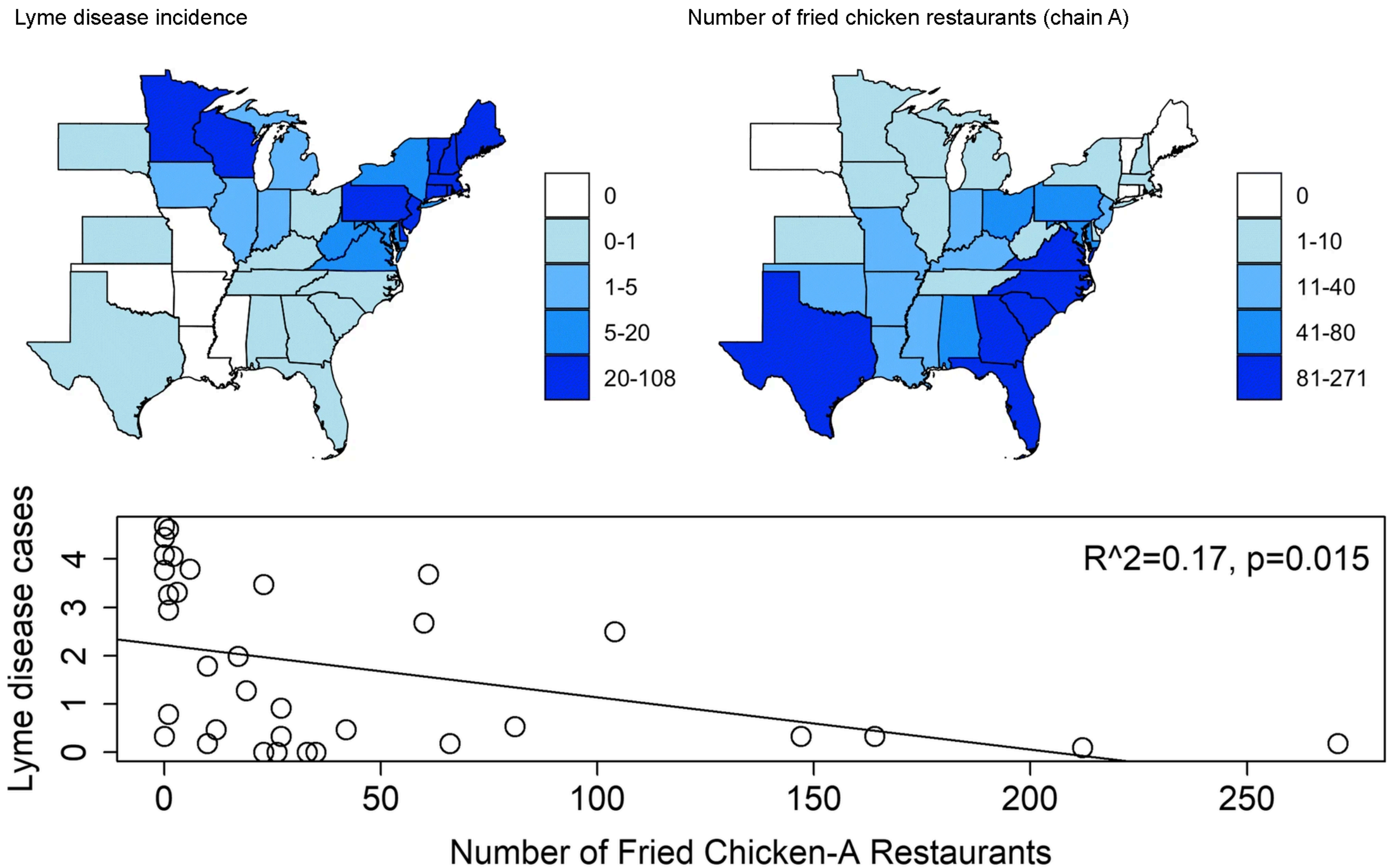
Correlation for data frame columns with plot

`corrplot` package can make correlation matrix plots

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



Correlation does not imply causation



source

Processing math: 100%

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 CO2 emissions data again.

```
t.test(y1980)
```

```
    One Sample t-test

data:  y1980
t = 3.3324, df = 170, p-value = 0.001056
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
  44745.81 174792.25
sample estimates:
mean of x
   109769
```

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

```
t.test(y1980, y1985)
```

```
    Welch Two Sample t-test

data:  y1980 and y1985
t = -0.090533, df = 341, p-value = 0.9279
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -95902.79  87462.97
sample estimates:
mean of x mean of y 
 109769.0  113988.9 
```

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(y1980, y1985)
result_tidy <- tidy(result)
glimpse(result_tidy)
```

```
Rows: 1
Columns: 10
$ estimate    <dbl> -4219.909
$ estimate1   <dbl> 109769
$ estimate2   <dbl> 113988.9
$ statistic   <dbl> -0.09053303
$ p.value     <dbl> 0.9279168
$ parameter   <dbl> 340.999
$ conf.low    <dbl> -95902.79
$ conf.high   <dbl> 87462.97
$ 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 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.

Processing math: 100%

Linear regression notation

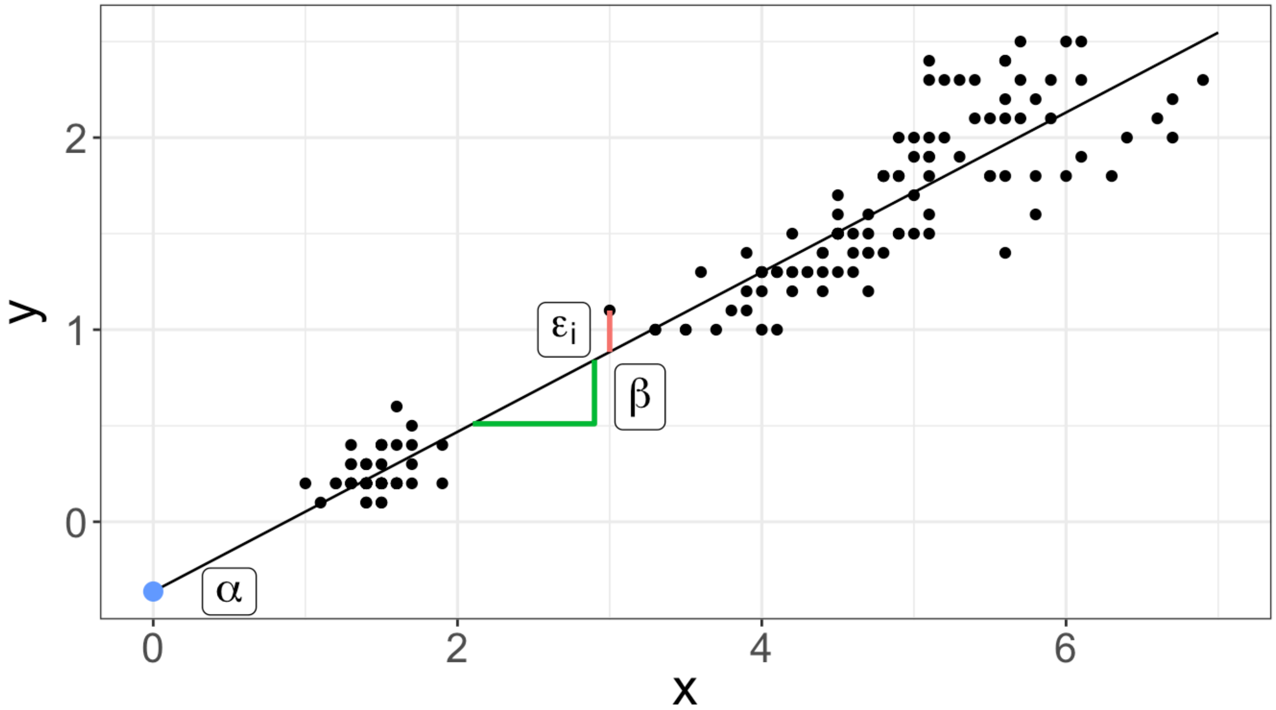
Here is some of the notation, so it is easier to understand the commands/results.

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

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
- $\beta_1, \beta_2, \beta_2$ 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 the calenviroscreen dataset from the `dasehr` package to examine how traffic estimates predict diesel particulate emissions.

Processing math: 100%

Linear regression: model fitting

For this model, we will use two variables:

- **DieselPM** - estimated diesel particulate emissions from on-road and non-road sources
- **TrafficPctl** - percentile ranking of traffic density

```
fit <- glm(DieselPM ~ TrafficPctl, data = calenviroscreen)
fit
```

```
Call:  glm(formula = DieselPM ~ TrafficPctl, data = calenviroscreen)

Coefficients:
(Intercept)  TrafficPctl
   0.042452     0.003637

Degrees of Freedom: 7999 Total (i.e. Null);  7998 Residual
  (35 observations deleted due to missingness)
Null Deviance:      537.2
Residual Deviance: 449.1    AIC: -330.9
```

Linear regression: model summary

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

```
summary(fit)
```

```
Call:
glm(formula = DieselPM ~ TrafficPctl, data = calenviroscreen)

Coefficients:
              Estimate Std. Error t value             Pr(>|t|)
(Intercept) 0.04245151 0.00529915   8.011   0.0000000000000013 ***
TrafficPctl 0.00363651 0.00009177  39.625 < 0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.05614936)

    Null deviance: 537.24  on 7999  degrees of freedom
Residual deviance: 449.08  on 7998  degrees of freedom
  (35 observations deleted due to missingness)
AIC: -330.9

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 the traffic percentile, we see 0.003637 more Diesel particulate emissions. The error for this estimate is relatively small at 0.00009. This relationship appears to be significant with a small p value < 2e-16.

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

```
Rows: 2
Columns: 5
$ term      <chr> "(Intercept)", "TrafficPctl"
$ estimate  <dbl> 0.042451513, 0.003636512
$ std.error <dbl> 0.00529915058, 0.00009177366
$ statistic <dbl> 8.011003, 39.624789
$ p.value   <dbl> 0.00000000000000129833968572387431567327971127744949621403422
```

Processing math: 100%

Linear regression: multiple predictors

Let's try adding another explanatory variable to our model, amount of daily Ozone concentration (`Ozone`). Ozone is usually inversely related to particulate measures.

```
fit2 <- glm(DieselPM ~ TrafficPctl + Ozone, data = calenviroscreen)
summary(fit2)
```

```
Call:
glm(formula = DieselPM ~ TrafficPctl + Ozone, data = calenviroscreen)

Coefficients:
               Estimate  Std. Error t value            Pr(>|t|)    
(Intercept)  0.23025068  0.01347754   17.08 <0.0000000000000002 ***
TrafficPctl  0.00355094  0.00009067   39.16 <0.0000000000000002 ***
Ozone       -3.77418894  0.24967138  -15.12 <0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.0545963)

    Null deviance: 537.24  on 7999  degrees of freedom
Residual deviance: 436.61  on 7997  degrees of freedom
  (35 observations deleted due to missingness)
AIC: -554.3

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)", "TrafficPctl", "Ozone"
$ estimate  <dbl> 0.23025068, 0.00355094, -3.77418894
$ std.error <dbl> 0.01347753532, 0.00009067243, 0.24967137612
$ statistic <dbl> 17.08403, 39.16229, -15.11663
$ p.value   <dbl> 0.0000000000000000000000000000000000000000000000000000000000000
```


Processing math: 100%

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.

Let's create a variable that tells us whether a census tract has a high, middle, or low percentage of the population below the poverty line.

```
calenviroscreen <- calenviroscreen %>% mutate(
  PovertyPctl_level = case_when(
    PovertyPctl > 0.75 ~ "high",
    PovertyPctl > 0.25 & PovertyPctl <= 0.75 ~ "middle",
    PovertyPctl <= 0.25 ~ "low",
    TRUE ~ NA
  )
)
```

Linear regression: factors

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

```
fit3 <- glm(DieselPM ~ TrafficPctl + Ozone + factor(PovertyPctl_level), data = calenviroscreen)
summary(fit3)
```

```
Call:
glm(formula = DieselPM ~ TrafficPctl + Ozone + factor(PovertyPctl_level),
    data = calenviroscreen)

Coefficients:
                                   Estimate  Std. Error t value
(Intercept)                       0.22893847  0.01343002  17.047
TrafficPctl                       0.00353551  0.00009034  39.137
Ozone                            -3.73294307  0.24881820 -15.003
factor(PovertyPctl_level)low     -0.09685048  0.05463573  -1.773
factor(PovertyPctl_level)middle -0.11329720  0.03672457  -3.085
                                            Pr(>|t|)
(Intercept)                      < 0.0000000000000002 ***
TrafficPctl                      < 0.0000000000000002 ***
Ozone                            < 0.0000000000000002 ***
factor(PovertyPctl_level)low                 0.07632 .
factor(PovertyPctl_level)middle              0.00204 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.05357268)

    Null deviance: 523.61  on 7933  degrees of freedom
Residual deviance: 424.78  on 7929  degrees of freedom
  (101 observations deleted due to missingness)
AIC: -697.85

Number of Fisher Scoring iterations: 2
```

Linear regression: factors

Relative to the level is not listed.

```
calenviroscreen <-
  calenviroscreen %>%
  mutate(PovertyPctl_level = factor(
    PovertyPctl_level,
    levels = c("low", "middle", "high")
  ))
fit4 <- glm(DieselPM ~ TrafficPctl + Ozone + PovertyPctl_level, data = calenviroscreen)
summary(fit4)
```

```

Call:
glm(formula = DieselPM ~ TrafficPctl + Ozone + PovertyPctl_level,
    data = calenviroscreen)

Coefficients:
                           Estimate  Std. Error t value            Pr(>|t|)
(Intercept)              0.13208799  0.05585244   2.365              0.0181 *
TrafficPctl              0.00353551  0.00009034  39.137 <0.0000000000000002 ***
Ozone                   -3.73294307  0.24881820 -15.003 <0.0000000000000002 ***
PovertyPctl_levelmiddle -0.01644672  0.06569819  -0.250              0.8023
PovertyPctl_levelhigh    0.09685048  0.05463573   1.773              0.0763 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.05357268)

    Null deviance: 523.61  on 7933  degrees of freedom
Residual deviance: 424.78  on 7929  degrees of freedom
  (101 observations deleted due to missingness)
AIC: -697.85

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(DieselPM ~ TrafficPctl + Ozone + PovertyPctl_level - 1, data = calenviroscreen)
summary(fit5)
```

```
Call:
glm(formula = DieselPM ~ TrafficPctl + Ozone + PovertyPctl_level - 
    1, data = calenviroscreen)

Coefficients:
                           Estimate  Std. Error t value            Pr(>|t|)    
TrafficPctl              0.00353551  0.00009034  39.137 <0.0000000000000002 ***
Ozone                   -3.73294307  0.24881820 -15.003 <0.0000000000000002 ***
PovertyPctl_levellow     0.13208799  0.05585244   2.365              0.0181 *  
PovertyPctl_levelmiddle  0.11564127  0.03838198   3.013              0.0026 ** 
PovertyPctl_levelhigh    0.22893847  0.01343002  17.047 <0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.05357268)

    Null deviance: 919.65  on 7934  degrees of freedom
Residual deviance: 424.78  on 7929  degrees of freedom
  (101 observations deleted due to missingness)
AIC: -697.85

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.

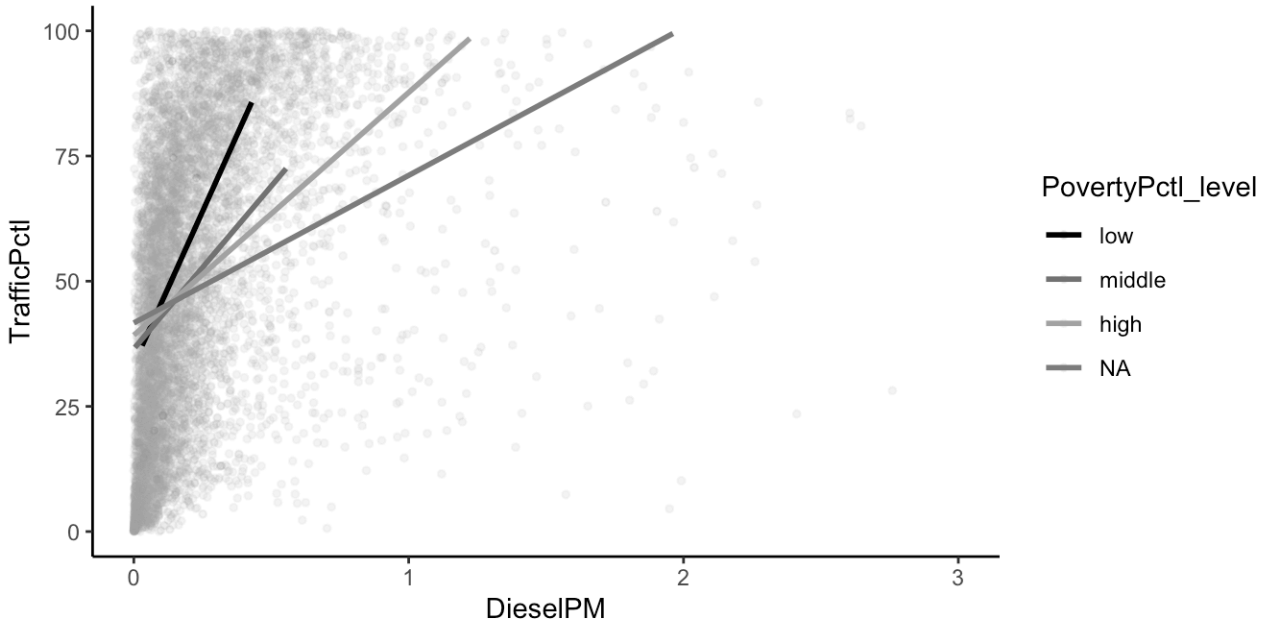
```
fit6 <- glm(
  DieselPM ~ TrafficPctl + Ozone + PovertyPctl_level + TrafficPctl * PovertyPctl_level,
  data = calenviroscreen
)
tidy(fit6)
```

```
# A tibble: 7 × 5
  term                                  estimate std.error statistic  p.value
  <chr>                                    <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)                           0.200      0.113      1.77   7.62e- 2
2 TrafficPctl                           0.00224    0.00186    1.21   2.28e- 1
3 Ozone                                -3.72       0.249    -14.9    7.90e-50
4 PovertyPctl_levelmiddle               0.00335    0.131      0.0256 9.80e- 1
5 PovertyPctl_levelhigh                 0.0280     0.112      0.249  8.03e- 1
6 TrafficPctl:PovertyPctl_levelmiddle -0.000721   0.00227   -0.317  7.51e- 1
7 TrafficPctl:PovertyPctl_levelhigh     0.00131    0.00186    0.702  4.83e- 1
```

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(calenviroscreen, aes(x = DieselPM, y = TrafficPctl, color = PovertyPctl_level)) +
  geom_point(size = 1, alpha = 0.1) +
  geom_smooth(method = "glm", se = FALSE) +
  scale_color_manual(values = c("black", "grey45", "grey65", "grey85")) +
  theme_classic() +
  ylim(0,100) +
  xlim(0, 3)
```



Processing math: 100%

/

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.

Processing math: 100%

Logistic regression

Let's look at a logistic regression example. We'll use the `calenviroscreen` dataset again. We will create a new binary variable based on the DieselPM percentile variable, so we can tell whether a census tract has high or low DieselPM emissions compared to the others.

```
calenviroscreen <-
  calenviroscreen %>%
    mutate(
      DieselPM_level = case_when
      (DieselPMPctl > 0.75 ~ 1,
        DieselPMPctl <= 0.75 ~ 0))
```

Logistic regression

Now that we've created the `DieselPM_level` variable (where a `1` indicates the census tract is one of the top 75% when it comes to dieselPM emissions), we can run a logistic regression.

Let's explore how `PovertyPctl_level` might predict `DieselPM_level`.

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

```
binom_fit <- glm(DieselPM_level ~ PovertyPctl_level, data = calenviroscreen, family = binomial(link = "logit"))
summary(binom_fit)
```

```
Call:
glm(formula = DieselPM_level ~ PovertyPctl_level, family = binomial(link = "logit"),
    data = calenviroscreen)

Coefficients:
                                Estimate       Std. Error z value Pr(>|z|)
(Intercept)                17.56606843873  932.48063065847   0.019    0.985
PovertyPctl_levelmiddle     0.00000004734 1118.59764091255   0.000    1.000
PovertyPctl_levelhigh     -12.62378846430  932.48064030187  -0.014    0.989

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 666.77  on 7959  degrees of freedom
Residual deviance: 665.93  on 7957  degrees of freedom
  (75 observations deleted due to missingness)
AIC: 671.93

Number of Fisher Scoring iterations: 16
```

Logistic Regression

See this case study for more information.

Processing math: 100%

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 living in a high traffic area predicts high levels of DieselPM.

```
library(epitools)

calenviroscreen <-
  calenviroscreen %>%
    mutate(
      Traffic_level = case_when
      (TrafficPctl > 0.75 ~ 1,
        TrafficPctl <= 0.75 ~ 0))

response <- calenviroscreen %>% pull(DieselPM_level)
predictor <- calenviroscreen %>% pull(Traffic_level)
```

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 living in a high traffic area predicts high levels of DieselPM.

```
oddsratio(predictor, response)
```

```
$data
         Outcome
Predictor  0    1 Total
    0     23   37    60
    1     35 7905  7940
    Total 58 7942  8000

$measure
         odds ratio with 95% C.I.
Predictor estimate    lower    upper
        0   1.0000       NA       NA
        1 139.3968 74.58837 260.5596

$p.value
         two-sided
Predictor midp.exact                                fisher.exact
        0         NA                                          NA
        1          0 0.00000000000000000000000000000000000007956334
         two-sided
Predictor
        0
        1 0.000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

$correction
[1] FALSE

attr(,"method")
Processing math: 100% iased 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

Processing math: 100%