Introduction to R for Data Analysis in the Health Sciences: Lecture 5

Amy Willis, Biostatistics, UW

01 November, 2019

Today

- Mid quarter feedback
- t-tests
  - The formula syntax `Y ~ X`
- Simple linear regression
- Multiple linear regression
- Factors
- Optional: Brief demo of R Markdown

Feedback

Many thanks to those of you who submitted feedback in ICE 4! I'll be doing a couple of things differently as a result of your feedback, including recording the lectures, and putting more details on the slides.

Disclaimer: This is not intended to replace attending lectures, but to facilitate your revision of the material.

Feedback

$n = 30$ out of possible 50 – thank you for taking the time!

- 90% said the pace is “about right”; remainder was evenly split between “too fast” and “too slow”
- 83% said the amount of homework is “about right”; remainder was evenly split between “too much” and “too little”
- Comments were evenly split between “please cover more material” and “you cover too much”

Want more help?

Especially if this is new material for you, aspects of learning R will be challenging and frustrating

Resources to support you

- Office hours (three days a week!)
- Sticking around after class
    - I don't leave until everyone's questions are answered; I'm usually here until 4:30 or 5pm

Office hours policy: Folks who attend office hours wanting help on the course material get priority; but if no one is in line for help, I'm happy to discuss *your data* and *material outside the scope of the course*.

Want more work?

No class will completely prepare you for the challenges of data analysis. Hence the emphasis on using Google to help you answer your own questions!

If you feel underworked in this class, and want more practice, download the data from a recent paper that you read and try to replicate the results.

Other resources

- "Advanced R" by Hadley Wickham: `http://adv-r.had.co.nz/`
- This is "Intro R", but I previously taught "advanced R". Materials here: `github.com/adw96/biostat561`

As always. . .

```r
library(tidyverse)
```

```
## -- Attaching packages ----------------------------------------

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------- tidyv
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Example: Sleep Data

The results of a 1908 (!!) paper by William Sealy Gosset on the effect of two sleeping pills is available in the dataset `sleep.csv`:

```
gosset <- read_csv("datasets/sleep.csv")
```

```
## Parsed with column specification:
## cols(
##   extra = col_double(),
##   group = col_double(),
##   ID = col_double()
## )
```

Example: Sleep Data

Variables in this dataset include `extra` (the increase in hours of sleep compared to baseline), `group` (which drug was given), and `ID` (an identifier for each patient)

```
gosset %>% print(n = 12)
```

```
## # A tibble: 20 x 3
##    extra group    ID
##    <dbl> <dbl> <dbl>
##  1   0.7     1     1
##  2  -1.6     1     2
##  3  -0.2     1     3
##  4  -1.2     1     4
##  5  -0.1     1     5
##  6   3.4     1     6
##  7   3.7     1     7
##  8   0.8     1     8
##  9   0       1     9
## 10   2       1    10
## 11   1.9     2     1
## 12   0.8     2     2
## # ... with 8 more rows
```

Example: Sleep Data

group and ID are categorical variables; let's change them from numeric data (<dbl>) to character data

```
gosset_char <- gosset %>%
  mutate(group = as.character(group),
         ID = as.character(ID))
```

Example: Sleep Data

```
gosset_char %>%
  group_by(group) %>%
  summarise(mean = mean(extra), sd = sd(extra))
```

```
## # A tibble: 2 x 3
##   group  mean    sd
##   <chr> <dbl> <dbl>
## 1 1      0.75  1.79
## 2 2      2.33  2.00
```

Is this difference due to random chance?

Sleep Data Example: the *t*-test

To compare mean levels of extra sleep in Group 1 versus 2 with a (unpaired) t-test:

```r
t.test(extra ~ group, data = gosset_char)
```

```
##
##  Welch Two Sample t-test
##
## data:  extra by group
## t = -1.8608, df = 17.776, p-value = 0.07939
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.3654832  0.2054832
## sample estimates:
## mean in group 1 mean in group 2
##            0.75            2.33
```

Sleep Data Example: the *t*-test

```r
t.test(extra ~ group, data = gosset_char)
```

How does the syntax `Y ~ X` work?

- `Y` is the outcome; it is modelled as a function of `X`
- Here we are looking at whether the mean of `extra` varies for each `group`

Sleep Data Example: the *t*-test

```r
t.test(extra ~ group, data = gosset_char)
```

```
##
##  Welch Two Sample t-test
##
## data:  extra by group
## t = -1.8608, df = 17.776, p-value = 0.07939
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.3654832  0.2054832
## sample estimates:
## mean in group 1 mean in group 2
##            0.75            2.33
```

- `p-value`
    - Null hypothesis is “population mean in group 1 equals the population mean in group 2”
    - Alternative hypothesis is a 2-sided test
- Confidence interval is for the difference in means

Sleep Data Example: the *t*-test

- Default is to treat variances in each group as unequal; to assume equal:

```r
t.test(extra ~ group, data = gosset_char,
       var.equal = TRUE, conf.level = 0.99)
```

```
##
##  Two Sample t-test
##
## data:  extra by group
## t = -1.8608, df = 18, p-value = 0.07919
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##  -4.0240579  0.8640579
## sample estimates:
## mean in group 1 mean in group 2
##            0.75            2.33
```

Sleep Data Example: the *t*-test

For paired testing:

```r
t.test(gosset_char$extra[gosset_char$group == "1"],
       gosset_char$extra[gosset_char$group == "2"],
       paired=TRUE)
```

```
##
##  Paired t-test
##
## data:  gosset_char$extra[gosset_char$group == "1"] and gosset_char$e
## t = -4.0621, df = 9, p-value = 0.002833
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.4598858 -0.7001142
## sample estimates:
## mean of the differences
##                   -1.58
```

Note that the ordering of the group 1 IDs must be the same as the ordering of the group 2 IDs.

Example: linear regression

```r
fev <- read_csv("datasets/fev.csv")
```

```
## Parsed with column specification:
## cols(
##   seqnbr = col_double(),
##   subjid = col_double(),
##   age = col_double(),
##   fev = col_double(),
##   height = col_double(),
##   sex = col_double(),
##   smoke = col_double()
## )
```

Example: linear regression

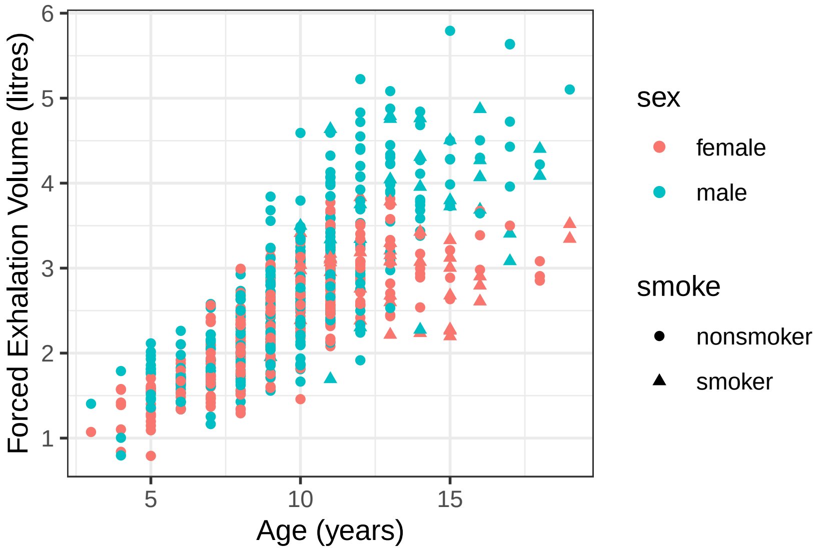
```r
fev_char <- fev %>%
  mutate(sex = ifelse(sex == 1, "male", "female"),
         smoke = ifelse(smoke == 1, "smoker", "nonsmoker"))
```

Example: linear regression

We have previously observed sex and smoking status specific differences:

```r
fev_char %>%
  ggplot(aes(x = age, y = fev, col = sex, pch = smoke)) +
  geom_point() +
  ylab("Forced Exhalation Volume (litres)") +
  xlab("Age (years)") +
  theme_bw()
```

Example: linear regression



Example: linear regression

Let's start by modeling FEV as a function of age:

```r
fev_lm1 <- lm(fev ~ age, data = fev_char)
```

- `lm` stands for *linear model*
- `fev ~ age`: we model `fev` as a function of `age`
    - and that model is *linear* when we use `lm`
- `data = fev_char` indicates that `fev` and `age` are columns in the data frame `fev_char`

Example: linear regression with `lm`

```
fev_lm1
```

```
##
## Call:
## lm(formula = fev ~ age, data = fev_char)
##
## Coefficients:
## (Intercept)          age
##      0.4316       0.2220
```

We see that the fitted model is

$$\hat{FEV} = 0.4316 + 0.2220 \times age$$

summary of an `lm` gives you more details

```
fev_lm1 %>% summary
```

```
##
## Call:
## lm(formula = fev ~ age, data = fev_char)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.57539 -0.34567 -0.04989  0.32124  2.12786
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.431648   0.077895   5.541 4.36e-08 ***
## age         0.222041   0.007518  29.533  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5675 on 652 degrees of freedom
## Multiple R-squared:  0.5722, Adjusted R-squared:  0.5716
## F-statistic: 872.2 on 1 and 652 DF,  p-value: < 2.2e-16
```

`summary` of an `lm` gives you more details

- `Call` restates the formula
- `Residuals` gives a summary of the distribution of the residuals (observations minus fitted values)
- `Coefficient`
  - the fitted line is $\hat{FEV} = 0.4316 + 0.2220 \times age$
  - `Std. Error` gives the estimate of the standard deviation in the `Estimates`
  - `Pr(>|t|)` is a two sided p-value for the null hypothesis that the coefficient is zero
- Other terms give information about residual error (the unexplained variance in the dataset)

Multiple regression

How do we fit a linear model with multiple terms?

Multiple regression

To include additional terms in the linear model, add them to the independent variable:

`Y ~ X1 + X2` fits the model $Y = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \epsilon_i$

For example:

```r
fev_lm2 <- lm(fev ~ age + smoke + sex, data = fev_char)
```

Multiple regression

```r
fev_lm2
```

```
##
## Call:
## lm(formula = fev ~ age + smoke + sex, data = fev_char)
##
## Coefficients:
## (Intercept)          age  smokesmoker      sexmale
##      0.2378       0.2268      -0.1540       0.3153
```

$$\hat{FEV} = 0.2378 + 0.2268 \times age - 0.1540 \times I(smoker) + 0.3153 \times I(male)$$

Multiple regression

$$\hat{FEV} = 0.2378 + 0.2268 \times age - 0.1540 \times I(smoker) + 0.3153 \times I(male)$$

- For a given sex and smoking status, the average increase in FEV for each year of age is 0.23 litres
- For fixed age and sex, smoking decreases FEV by 0.15 litres on average
- On average, males have 0.32 litres more FEV than females for a fixed age and smoking status

Multiple regression

```
fev_lm2 %>% summary
```

```
##
## Call:
## lm(formula = fev ~ age + smoke + sex, data = fev_char)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.46707 -0.35426 -0.03811  0.32199  1.94943
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.237771   0.080228   2.964  0.00315 **
## age          0.226794   0.007884  28.765  < 2e-16 ***
## smokesmoker -0.153974   0.077977  -1.975  0.04873 *
## sexmale      0.315273   0.042710   7.382  4.8e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5432 on 650 degrees of freedom
## Multiple R-squared:  0.6093, Adjusted R-squared:  0.6075
## F-statistic: 337.9 on 3 and 650 DF,  p-value: < 2.2e-16
```

Example: linear regression

To get coefficient estimates for the parameters, use coef

```
fev_lm2 %>% coef
```

```
## (Intercept)         age smokesmoker     sexmale
##   0.2377708   0.2267942  -0.1539741   0.3152733
```

```
fev_lm2 %>% summary %>% coef
```

```
##               Estimate  Std. Error   t value      Pr(>|t|)
## (Intercept)  0.2377708 0.080227888  2.963693  3.150910e-03
## age          0.2267942 0.007884453 28.764737 5.366373e-118
## smokesmoker -0.1539741 0.077976575 -1.974620  4.873410e-02
## sexmale      0.3152733 0.042710389  7.381653  4.800060e-13
```

Example: linear regression

To get confidence intervals for the parameters, use `confint`

```
confint(fev_lm2)
```

```
##                    2.5 %       97.5 %
## (Intercept)  0.08023369  0.395307916
## age          0.21131214  0.242276281
## smokesmoker -0.30709050 -0.000857728
## sexmale      0.23140630  0.399140270
```

```
confint(fev_lm2, level=0.99)
```

```
##                    0.5 %     99.5 %
## (Intercept)  0.03050894 0.44503267
## age          0.20642540 0.24716302
## smokesmoker -0.35541990 0.04747167
## sexmale      0.20493466 0.42561191
```

Example: linear regression

To get the fitted values and residuals for each observation, use `fitted` and `residuals`:

```r
fitted(fev_lm2)
residuals(fev_lm2)
```

Example: linear regression

`predict(fev_lm2, new_df)` uses the model to predict the mean `FEV` (i.e. Y) for which you supply age, `sex` and `smoke`

Create a new data frame as follows:

```r
new_df <- tibble("age" = 13:15,
                 "smoke" = "smoker", "sex" = "female")
new_df
```

```
## # A tibble: 3 x 3
##     age smoke  sex
##   <int> <chr>  <chr>
## 1    13 smoker female
## 2    14 smoker female
## 3    15 smoker female
```

Example: linear regression

- `predict(fev_lm2, new_df)` uses the model to predict the mean `FEV` (i.e. Y) for which you supply `age`, `sex` and `smoke`

```r
predict(fev_lm2, new_df)
```

```
##        1        2        3
## 3.032121 3.258916 3.485710
```

Example: linear regression

Confirming that `predict` does what we expect ("plugs in" the new data to the model)

```
fev_lm2 %>% coef
```

```
## (Intercept)         age smokesmoker     sexmale
##   0.2377708   0.2267942  -0.1539741   0.3152733
```

```
0.2378 + 0.2268*(13:15) - 0.1540 + 0
```

```
## [1] 3.0322 3.2590 3.4858
```

Multiple regression

To test hypotheses involving more than one parameter at a time, use `anova()` to compare the fitted models with and without those parameters:

```
lm_age  <- lm(fev ~ age, data = fev_char)
lm_full <- lm(fev ~ age + sex + smoke, data = fev_char)
anova(lm_age, lm_full)
```

```
## Analysis of Variance Table
##
## Model 1: fev ~ age
## Model 2: fev ~ age + sex + smoke
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    652 210.00
## 2    650 191.78  2    18.216 30.869 1.558e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tools for superusers

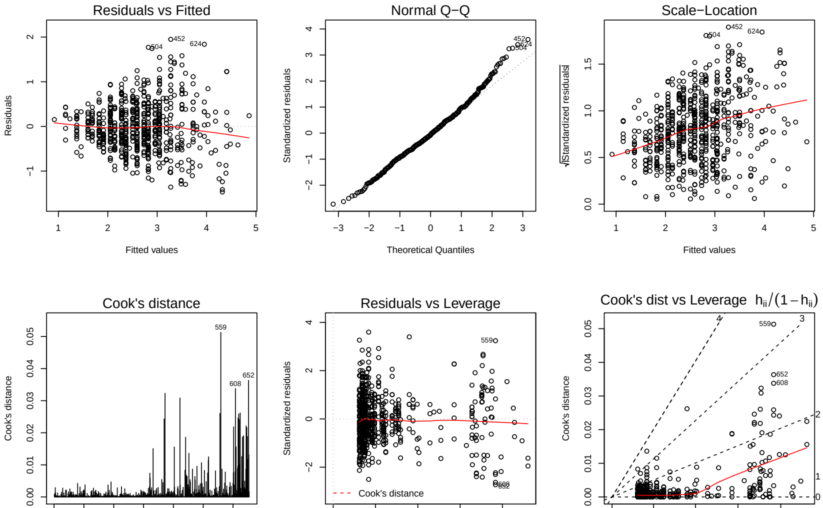
- `geom_abline()` can be used to add a simple linear regression line to a ggplot
- You can obtain the raw numbers using $
  - Use `fev_lm2 %>% names` and `fev_lm2 %>% summary %>% names` to see options
  - e.g. `(fev_lm2 %>% summary)$r.squared` to get R^2
- `vcov(fev_lm2)` gives the variance-covariance matrix for the coefficients
  - `fev_lm2 %>% vcov %>% diag %>% sqrt` is the same as `Std. Error` column in `summary()` output

Looking for something else you don't see here? Run `methods(class="lm")` to see other options

Tools for superusers

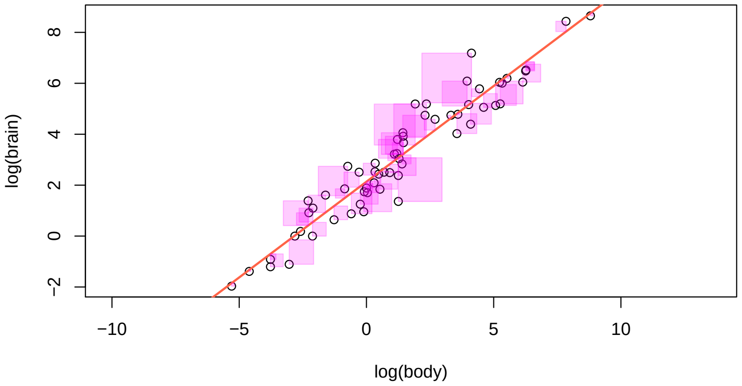
`plot()` has a method for `lm` objects that show some useful diagnostics

```
par(mfrow=c(2,3)); plot(fev_lm2, which=1:6); par(mfrow=c(1,1))
```



Brief motivation: linear regression

`lm` finds the least-squares fit:



Any other choice of line would use more purple ink.

Disclaimer & warning

The linear model, and R's `lm` is an extremely useful tool for data analysis, modeling, and estimation.

However, it is important to understand what you are modeling, and what assumptions go into your model.

Knowing how to use `lm` and correctly interpreting the output are *different skills.*

Please take care when interpreting the results of any model and any hypothesis test!

Factors: Categorical data in R

There are multiple ways to treat categorical data in R:

- Factors: `R`'s formal way of dealing with categories
- Characters: text

Most analysis methods (`e.g.`, `lm`) are invariant to whether a variable is a character or a factor

Factors in regression

Whichever is alphabetically first will be the “baseline” for both character and factor.

```
# sex is a character:
lm(fev ~ sex, data = fev_char) %>% coef
```

```
## (Intercept)     sexmale
##   2.4511698   0.3612766
```

```
# sex is a factor
lm(fev ~ sex,
   data = fev_char %>% mutate(sex = as.factor(sex))) %>% coef
```

```
## (Intercept)     sexmale
##   2.4511698   0.3612766
```

Characters: Categorical data in R

If you want to change the baseline, this is easy with categories: just set the baseline to be alphabetically earlier:

```
fev_male_baseline <- fev_char %>%
  mutate(sex = ifelse(sex == "male", "amale", "female"))
lm(fev ~ sex, data = fev_male_baseline) %>% coef
```

```
## (Intercept)   sexfemale
##   2.8124464  -0.3612766
```

Factors: Categorical data in R

Changing the baseline is more tedious with factors: use ‘`relevel`

```
fev_factor <- fev_char %>%
  mutate(sex = as.factor(sex))
fev_factor_male_baseline <- fev_factor %>%
  mutate(sex = relevel(sex, ref = "male"))
lm(fev ~ sex, data = fev_factor_male_baseline) %>% coef
```

```
## (Intercept)   sexfemale
##   2.8124464  -0.3612766
```

Factors vs categories

Factors can be useful when

- You have a set number of categories (e.g., treatment group vs not treatment group)
- The set of categories is large (e.g., more than 1000)
- Computation time is a bottleneck

Factors vs categories

Characters can be useful when

- You have a unknown set of categories (e.g., ID numbers, names. . .)
- You have missing or unknown values
  - see `https://www.r-bloggers.com/factors-are-not-first-class-citizens-in-r/` for some dangerous/pathological behaviour

Categorical data in R

My personal preference is to never use factors. I personally find that the risks and tedium never outweigh the benefits. *Not all of my colleagues would agree.*

Summary

- `t.test`
- The default `lm()` function can be used for simple linear regression as well as multiple regression
- Both `t.test` and `lm` use the syntax `response ~ covariate`
- Take care when running and interpreting models and tests!

Next week: More regression methods, including logistic regression

The plan

- 5 minute break
- In-class exercise available via Canvas
  - Designed to be completed by 3:20 p.m.
  - Due today 6:30 p.m.
  - *Yellow sticky note* = urgent; *blue sticky note* = non-urgent
- At 3:00pm, I'm going to give a brief demonstration of R Markdown
  - Learning R Markdown is *OPTIONAL*
- Homework due next week by 1 p.m. Friday
- Office hours as always: Tuesday, Wednesday and Thursday