Biological Stats II : Lab 5

Gavin Fay

02/20/2025

Lab schedule

1/22: Introduction to R and R Studio, working with data
1/29: Visualizing data
2/05: Probability, linear modeling
2/12: Data wrangling, model summaries
2/20: Simulation, Resampling
2/26: Iteration
3/19: Creating functions, debugging
4/02: working with Spatial data
4/09: Flex: automated reporting & Quarto

Segue: Packages in R

Installing packages:

```
install.packages("packagename")
```

If in doubt, use the source:

```
install.packages("packagename", type="source")
```

Installing packages from github:

```
remotes::install_github("repositoryname")
```

Loading (attaching) packages into the workspace:

```
library(packagename)
```

Some people use `require()` instead of `library()`.
Don't do this!
`require()` is basically `try(library())`

Often description of a package and how to use its functions is in the form of a vignette.
`vignette()` lists available vignettes.
`vignette(packagename)` views the vignette for `packagename`.

Programming practices I

Use projects (RStudio)

Write scripts (or markdown files)

Include whitespace in code
- blank lines, spaces in functions

Use an editor with syntax highlighting

Use a style guide

Indent code

Use meaningful object names

Programming practices II

Test code
- Write smallest possible amount (e.g. 1 line).
- Knit early and often.
- Try simple examples that you know the answer to.
- Always assume that there will be an error somewhere.

View results / objects (e.g. with `print()`).

Plot results - are they what you expect?

Be careful when copying sections of code and changing a variable name (it's super common to forget to change all occurrences).
- hint: use your text editor's "Find: Replace all" functionality.

Commenting

`R` ignores everything on a line that follows a #

Comment at the top of your script.
- What the script does, your name, email, date started.

Comment before each function or section of code - What is the purpose of that section of code, what does it do?
- Comment the 'why' not the 'what'

Comment throughout:
- whenever an unusual function is used
- whenever the code is hard to understand
- whenever an algorithm is particularly useful

Commenting out code

When you make modifications to your code: - Copy the code that works then comment it out by prefixing it with `#`. - Change the new copy of the code.

If you need to revert to the old code, just remove the # before each line ("uncomment").

`ctrl+shift+C` is a shortcut in Rstudio to comment/uncomment large blocks of code.

In `.Rmd` files, you can comment out blocks of the file using

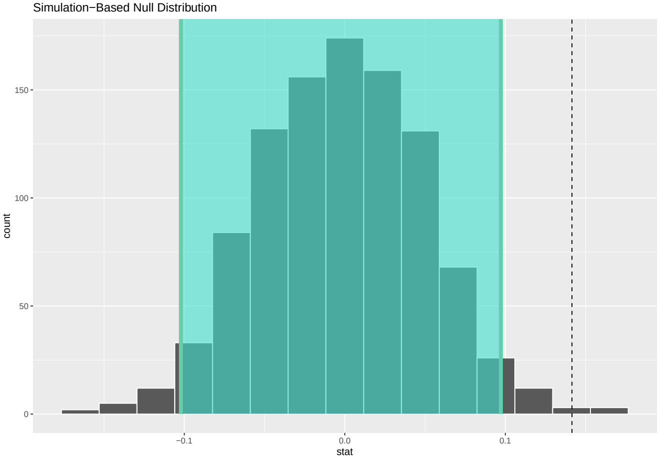
```
<!---
Lines of text and code you want to not be included
--->
```

Permutation tests

```r
library(moderndive)
null_evals <- evals |>
  specify(score ~ gender) |>
  hypothesize(null = "independence") |>
  generate(reps = 1000, type = "permute") |>
  calculate(stat = "diff in means",
            order = c("male", "female"))
null_evals_ci <- null_evals |>
  summarize(
    l = quantile(stat, 0.025),
    u = quantile(stat, 0.975)
    )
score_means <- evals |>  group_by(gender) |>
  summarize(avg_score = mean(score))
dscore <- score_means[2,2]-score_means[1,2]

percentile_ci <- null_evals |>
  get_confidence_interval(type = "percentile",
                          level = 0.95)
```

```
visualize(null_evals) +
  shade_confidence_interval(endpoints = percentile_ci) +
  geom_vline(xintercept = as.numeric(dscore),linetype = "da
```



Lab exercise 1 - permutation test

Use the Laengelmavesi data in `../data/Laengelmavesi2.csv`

a. Obtain the data for just the lengths of perch and bream.
b. Plot the distribution of lengths for both species, and calculate the mean lengths for both species.
c. Conduct a permutation test to assess whether the difference in mean length between bream and perch is statistically clear.
d. Plot the distribution for the test statistic under the null hypothesis of no difference in length, and indicate the true value for the test statistic relative to the two-tailed 95th percent of the null hypothesis distribution.
e. What are your conclusions?

Auto dataset

in ‘ISLR’ package

`` `geom_smooth()` using formula = 'y ~ x' ``

Bootstrapping

Recall:

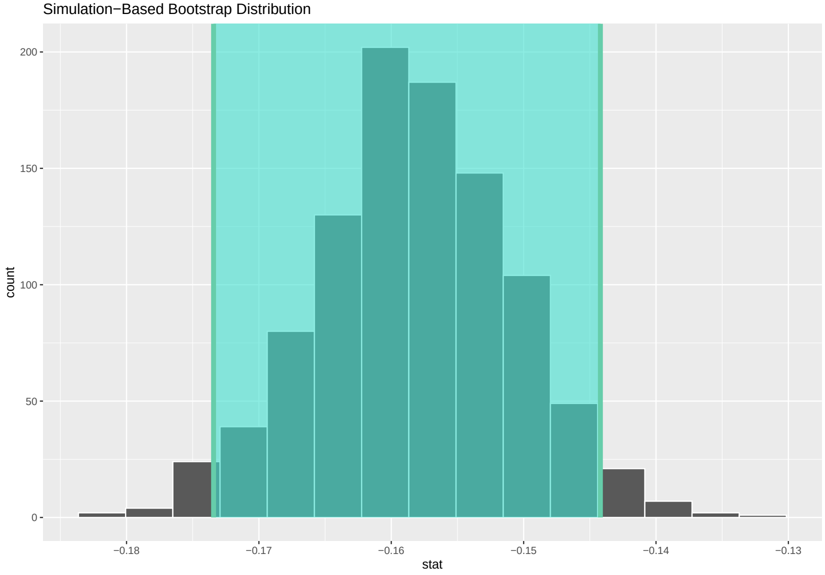
- Have some data x_1, x_2, , . . . , x_n and we are computing a statistic.
- Randomly draw n values from the data *with replacement* (same value can be drawn multiple times).
- Calculate statistic from new random pseudo-data.
- Repeat a large number of times to obtain the distribution: $t_1, t_2, \ldots, t_n$.
- Resulting distribution is the bootstrap sampling distribution.
- Compute standard deviation and 95 percent confidence intervals from this. Finished.

Performing generic case bootstrapping in `R`

Bootstrap estimates of slope from a linear model.

```
slope_bootstrap <- Auto |>
  specify(formula = mpg ~ horsepower) |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate(stat = "slope")
percentile_ci <- slope_bootstrap |>
  get_confidence_interval(type = "percentile",
                          level = 0.95)
percentile_ci
# A tibble: 1 x 2
  lower_ci upper_ci
     <dbl>    <dbl>
1   -0.173   -0.144
```

```
visualize(slope_bootstrap) +
  shade_confidence_interval(endpoints = percentile_ci)
```



Residual bootstrap

```
slope <- coef(lm(mpg~horsepower,data=Auto))[2]
lm1 <- lm(mpg~horsepower,data=Auto)
automod <- augment(lm1, newdata = Auto) |>
  janitor::clean_names()
automod
# A tibble: 392 x 12
   rownames   mpg cylinders displacement horsepower weight accel
   <chr>    <dbl>     <dbl>        <dbl>      <dbl>  <dbl>
 1 1           18         8          307        130   3504
 2 2           15         8          350        165   3693
 3 3           18         8          318        150   3436
 4 4           16         8          304        150   3433
 5 5           17         8          302        140   3449
 6 6           15         8          429        198   4341
 7 7           14         8          454        220   4354
 8 8           14         8          440        215   4312
 9 9           14         8          455        225   4425
10 10          15         8          390        190   3850
# i 382 more rows
# i 4 more variables: origin <dbl>, name <fct>, fitted <dbl>, re
```

```
autoboot <- automod |> specify(response = resid) |>
  generate(reps = 1000, type = "bootstrap") |>
  #ungroup() |>
  mutate(fitted = automod$fitted) |>
  mutate(new_mpg = fitted + resid) |>
  mutate(horsepower = automod$horsepower)
autoboot
# A tibble: 392,000 x 5
# Groups:   replicate [1,000]
   replicate  resid fitted new_mpg horsepower
       <int>  <dbl>  <dbl>   <dbl>      <dbl>
 1         1 -4.62   19.4    14.8         130
 2         1 -2.05   13.9    11.8         165
 3         1  0.113  16.3    16.4         150
 4         1 -2.07   16.3    14.2         150
 5         1 -5.26   17.8    12.6         140
 6         1 -5.85    8.68    2.84        198
 7         1 -2.57    5.21    2.64        220
 8         1  1.07    6.00    7.07        215
 9         1 -5.97    4.42   -1.54        225
10         1 -3.04    9.95    6.90        190
# i 391,990 more rows
```

```
slope_autoboot <- autoboot |>
  group_by(replicate) |>
  summarize(slope = coef(lm(new_mpg~horsepower))[2])
slope_autoboot
# A tibble: 1,000 x 2
   replicate  slope
       <int>  <dbl>
 1         1 -0.170
 2         2 -0.157
 3         3 -0.160
 4         4 -0.165
 5         5 -0.150
 6         6 -0.161
 7         7 -0.169
 8         8 -0.151
 9         9 -0.166
10        10 -0.172
# i 990 more rows
```

```
percentile_ci <- quantile(slope_autoboot$slope, c(0.025, 0.975))
percentile_ci
      2.5%      97.5%
-0.1688020 -0.1446102
```

Lab exercise 2

`hake.csv` contains abundance data for silver hake from tows in the 2015 NMFS spring bottom trawl survey.
a. Produce 5,000 bootstrapped estimates for the mean abundance per tow based on case resampling (5000 samples).
b. Compare the standard deviation of the bootstrapped estimates of the mean to the standard error of the mean from the original sample.
c. Compute an approximate 95 percent confidence interval for the mean based on the bootstrap, assuming normality. Compare this to the interval based on percentiles of the bootstrap sampling distribution.
d. *BONUS* Plot how the bootstrap confidence interval for the mean changes with the number of bootstrap samples. (100, 500, 1000, 2000, 5000, 10000)

Validation Approach

Recall:
Split data into training set and a validation (or hold-out) set.

Fit model to training set, fitted model used to predict the responses for the observations in the validation set.

Resulting validation set error rate (e.g. MSE for quantitative response) is an estimate of the test error rate.

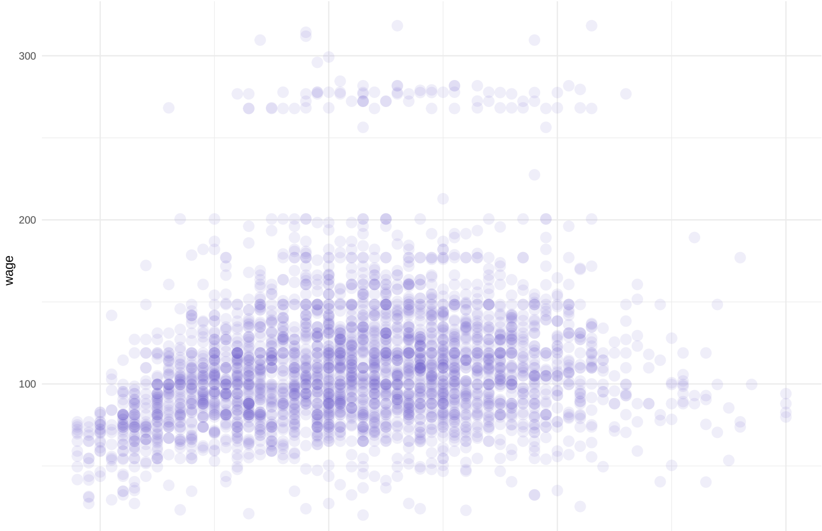
```
set.seed(1)
train <- sample(392,196)
lm.fit <- lm(mpg ~ horsepower, data=Auto, subset=train)
mean((Auto$mpg - predict(lm.fit, Auto))[-train]^2)
[1] 23.26601
```

```
lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data=Auto, subset=train)
mean((Auto$mpg - predict(lm.fit2, Auto))[-train]^2)
[1] 18.71646
```

```
lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data=Auto, subset=train)
mean((Auto$mpg - predict(lm.fit3, Auto))[-train]^2)
[1] 18.79401
```

Wages (from ISLR)

```r
ggplot(Wage, aes(y = wage, x = age)) +
  geom_point(col = "slateblue", alpha = 0.1, size = 4) +
  theme_minimal() #+
```



k-Fold Cross-Validation

Use `cv.glm()` with argument `K=k` to perform k-fold cross-validation.

```
set.seed(17)
cv.error.10 <- matrix(0,nrow=10,ncol=10)
for (isim in 1:10) {
  cv.error <- map(1:10,
            ~cv.glm(Auto,
            glm(mpg~poly(horsepower,.x),data = Auto),
            K=10)$delta[1])
  cv.error.10[isim,] <- as.numeric(cv.error)
}
cv.error.10
          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
 [1,] 24.27207 19.26909 19.34805 19.29496 19.03198 18.89781 19.12061 19
 [2,] 24.32221 19.20485 19.20345 19.19061 18.81104 18.93199 18.64288 18
 [3,] 24.40176 19.28566 19.47768 19.45928 19.11895 18.96636 19.14254 19
 [4,] 24.34410 19.21850 19.23349 19.29482 19.20885 18.89513 18.83750 19
 [5,] 24.12350 19.15993 19.30270 19.43933 19.14562 18.77230 18.95511 18
 [6,] 24.26450 19.20361 19.21035 19.50279 19.03784 19.00893 18.80578 20
 [7,] 24.21276 19.32266 19.25201 19.57620 18.94538 18.98961 19.36863 18
 [8,] 24.23926 19.19939 19.36453 19.18138 18.98821 19.20072 18.82207 19
 [9,] 24.25723 19.19239 19.43090 19.64706 19.03607 18.83338 18.74457 18
[10,] 24.06964 19.23239 19.41729 19.54851 19.02514 18.90808 18.94559 19
          [,9]    [,10]
```

Lab exercise 3, k-fold cross validation

Using the Wage data set, evaluate the predictive ability of models for wages.
a. Define a unique random number seed. Use 10-fold cross validation to estimate the test error rate for models fitting a polynomial of age of order 2, 3, 4, 5, and 6.
b. Conduct the validation 20 times for each polynomial. Plot the distribution (use boxplots) for the validation test error rate as a function of the degree of polynomial. Based on the results, what order polynomial would you use?
c. Use 5-fold cross-validation to compare the performance of models that include combinations of:

- a polynomial of age
- education level
- race
- industry

e. What model would you choose based on the test error rates?
f. *BONUS* How does the model chosen by 5-fold CV compare to that from using AIC as a model selection tool?

HINTS
Write down the steps you need to take to perform the calculations.
Make use of existing code from earlier in the lab to help.

Lab schedule

1/22: Introduction to R and R Studio, working with data
1/29: Visualizing data
2/05: Probability, linear modeling
2/12: Data wrangling, model summaries
2/20: Simulation, Resampling
2/26: Iteration
3/19: Creating functions, debugging
4/02: working with Spatial data
4/09: Flex: automated reporting & Quarto