Lecture 12: Inference for the Mean

Taylor Arnold

One-way Inference

inference

The first model we will look at is modeling the mean value of some random process. In this handout we will start to see how to implement this model directly in R and how to analyze how well it fits the data at hand.

a simple example

We want to take the observations of some numeric variable and provide an estimate of its **true** mean. The techniques we cover today will apply to two similar but conceptually different cases. These are:

- a variable sampled independently from a larger population
- a variable observed from repeated trials of a random process

a simple example

In the first case, the true mean is the mean of the entire population. For the second case, the true mean is the average value we would get from an infinite number of trials. As long as the sample from the population is taken at random and the output collected from each random trial is independent of prior trials, the same exact technique is used for estimating the mean of both situations.

a simple example

Consider a random sample of coins from a cup similar to the one we have in class:

```
coins
```

```
## # A tibble: 8 x 1
##   number
##    <dbl>
## 1      1
## 2      2
## 3      2
## 4      3
## 5      4
## 6      4
## # ... with 2 more rows
```

a simple example

Our best guess for the average value of all of the coins in the cups might be the mean of the sample we took:

```
mean(coins$number)
```

```
## [1] 3.125
```

using lm_basic

Let's do this is a different way that will allow us to extrapolate on this single number:

```
model <- lm_basic(number ~ 1, data = coins)
```

This says to construct a model for the variable `number` from the data set `coins`. The `1` indicates that we are fitting a single mean to the dataset; we will see later how to fit more complex models.

using lm_basic

To see the output of the model, run `reg_table`:

```
reg_table(model)
```

The model calls the mean an intercept, for reasons that will become clear shortly, and it gives the exact same value as with our old technique. The other numbers above and below the table can be useful but are not our primary subject of interest at the moment.

using lm_basic

```
reg_table(model)
```

```
##
## Call:
## lm_basic(formula = number ~ 1, data = coins)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.125 -1.125  0.375  0.875  1.875
##
## Coefficients:
##             Estimate
## (Intercept)    3.125
##
## Residual standard error: 1.356 on 7 degrees of freedom
```

using lm_basic - level

Why bother with this more involved method for finding a mean? For one thing, `reg_table` provides an option called `level` that can be set to a number between 0 and 1. For example:

```
reg_table(model, level = 0.9)
```

using lm_basic - level

```
reg_table(model, level = 0.9)
```

```
##
## Call:
## lm_basic(formula = number ~ 1, data = coins)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.125 -1.125  0.375  0.875  1.875
##
## Coefficients:
##             Estimate   5 %  95 %
## (Intercept)    3.125 2.217 4.033
##
## Residual standard error: 1.356 on 7 degrees of freedom
```

confidence interval

The table now includes two additional numbers of the mean: the 10th and 90th percentiles of a *confidence interval*. A confidence interval provides a guess for where the true mean actually lies.

The construction of a confidence interval involves some surprisingly deep mathematics, including the law of large numbers and the central limit theorem.

Using confidence intervals is, however, incredibly simple! The confidence level, here 90%, gives the probability that the testing procedure will lead to a correct result if a sample or experiment is repeated many times. Common confidence levels include 90%, 95%, and 99%.

a second example

Taking a set of sampled flight times from paper helicopters, we can run the exact same analysis:

```
model <- lm_basic(flight_time ~ 1, data = helicopter)
reg_table(model, level = 0.95)
```

Unless we have a specific reason to use a different level, we will usually use a 95% confidence interval in this course.

a second example

```
model <- lm_basic(flight_time ~ 1, data = helicopter)
reg_table(model, level = 0.95)
```

```
##
## Call:
## lm_basic(formula = flight_time ~ 1, data = helicopter)
##
## Residuals:
##      1      2      3      4      5      6
## -0.155  0.055  0.075 -0.135  0.105  0.055
##
## Coefficients:
##             Estimate  2.5 % 97.5 %
## (Intercept)   1.0550 0.9354  1.175
##
## Residual standard error: 0.114 on 5 degrees of freedom
```

Inference Across Groups

coins2

The `lm_basic` function allows for much more complex models than describing a simple mean. Consider a second set of data where coins have been taken from two different cups:

```
coins2
```

```
## # A tibble: 8 x 2
##   number   cup
##    <dbl> <chr>
## 1      1     A
## 2      1     A
## 3      4     A
## 4      5     A
## 5      1     B
## 6      3     B
## # ... with 2 more rows
```

lm_basic again

In this case, we may want to model the mean of both cups. To do this with `lm_basic`, we just add the new variable to the formula:

```
model <- lm_basic(number ~ 1 + cup, data = coins2)
reg_table(model)
```

lm_basic again

```
reg_table(lm_basic(number ~ 1 + cup, data = coins2))
```

```
##
## Call:
## lm_basic(formula = number ~ 1 + cup, data = coins2)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.000 -1.750  0.500  1.062  2.250
##
## Coefficients:
##             Estimate
## (Intercept)     2.75
## cupB            0.25
##
## Residual standard error: 1.768 on 6 degrees of freedom
## Multiple R-squared:  0.006623,   Adjusted R-squared:  -0.1589
## F-statistic:  0.04 on 1 and 6 DF,  p-value: 0.8481
```

lm_basic again

How do we read this new table? The intercept gives the mean value for the **A** cup and the second term, called a slope, gives the additional amount needed to get the mean of the coins from cup **B**. So, the best guess of cup B's mean is equal to 3.

lm_basic - confidence intervals

What does the table look like when we add confidence intervals:

```
reg_table(model, level = 0.95)
```

lm_basic - confidence intervals

```
reg_table(model, level = 0.95)
```

```
##
## Call:
## lm_basic(formula = flight_time ~ 1, data = helicopter)
##
## Residuals:
##      1      2      3      4      5      6
## -0.155  0.055  0.075 -0.135  0.105  0.055
##
## Coefficients:
##             Estimate  2.5 % 97.5 %
## (Intercept)   1.0550 0.9354  1.175
##
## Residual standard error: 0.114 on 5 degrees of freedom
```

lm_basic - confidence intervals

The mean of cup A is predicted to be between 0.5872 and 4.913. The difference between cup B's mean and cup A's mean is somewhere between -2.8086 and 3.309. Because this difference includes zero, we say that there is no statistical evidence (at the 95% level) that the mean of the two cups is different.

Think about this statement for a bit. Why would a value of zero be important in this model?

another example

Let's apply this to a more complex situation using the mammals sleep dataset. We can model the average time spent awake as a function of the diet type of a given mammal:

```
model <- lm_basic(awake ~ 1 + vore, data = msleep)
reg_table(model, level = 0.95)
```

another example

```
reg_table(lm_basic(awake ~ 1 + vore, data = msleep), level = 0.95)
```

another example

Now each of these values gives the difference between the base level, carnivores, and all of the others. So the predicted mean for hours spent awake for insectivores is 13.6263 + (-4.5662), or about 9 hours.

The confidence intervals tell us whether there is evidence that a given diet type is different from carnivores. We see, for example, that there is statistical evidence that insectivores differ from carnivores, but no evidence for distinctions between carinvores and any other groups.

the base level

The careful observer will notice that there is a problem with this approach: what if we want to compare two values when one is not the base level? To do so, use the `fct_relevel` command with the new baseline used as the second parameter:

```
msleep <- mutate(msleep, vore_new = fct_relevel(vore, "inse
model <- lm_basic(awake ~ 1 + vore_new, data = msleep)
```

Now everything is compared to the insecti category.

the base level

```
reg_table(model, level = 0.95)
```

```
##
## Call:
## lm_basic(formula = awake ~ 1 + vore_new, data = msleep)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.0263 -4.0128  0.4237  3.4255  7.7237
##
## Coefficients:
##                Estimate   2.5 % 97.5 %
## (Intercept)      9.0600  5.0773 13.043
## vore_newcarni    4.5663  0.0901  9.043
## vore_newherbi    5.4306  1.1480  9.713
## vore_newomni     4.0150 -0.4378  8.468
##
## Residual standard error: 4.467 on 72 degrees of freedom
```