ANOVA in R

Avalon C.S. Owens, Eric R. Scott

11/02/2018

Overview

```
library(dplyr)
library(ggplot2)
library(car) # for leveneTest()
```

- Conducting an ANOVA in R
  - Setting up ANOVA models
  - Getting residuals from ANOVA models
  - Testing assumptions of ANOVA on those residuals
- Kruskal-Wallis test

ANOVA example

chickwts

…Sorry if you're sick of this dataset!

```
str(chickwts)
```

```
## 'data.frame':    71 obs. of  2 variables:
##  $ weight: num  179 160 136 227 217 168 108 124 143 140 ...
##  $ feed  : Factor w/ 6 levels "casein","horsebean",..: 2 2 2 2 2 2 2 2 2 2 ...
```

Only two columns: one for weight, one describing the type of feed

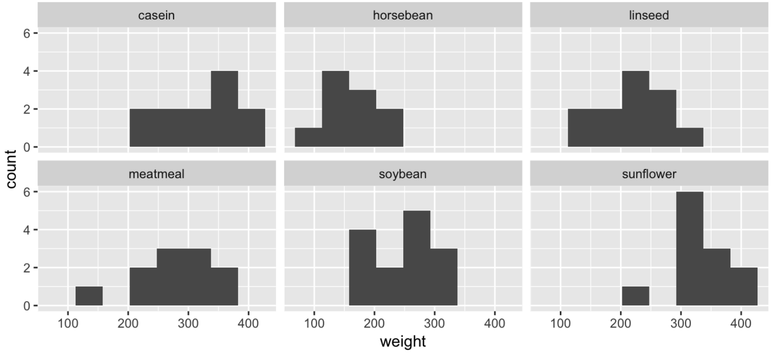
Variable name = `feed`, 6 *levels* to that variable

Faceted histogram

Let's start by making a faceted histogram to check normality

Later I will show you a **better** way to check for normality.

```
ggplot(chickwts, aes(x = weight)) + geom_histogram(bins = 8) + facet_wrap("feed")
```



Doing an ANOVA in R

1. Make the ANOVA model with `aov()`. This sets up the model, calculates sums of squares, but doesn't do the statistical test
2. Get residuals with `fortify()` from the `ggplot2` package and check normality of **residuals**.
3. If your model passes the check (#2), run `anova()` on your model (this calculates the statistics)

ANOVA step 1: set up the model

Make an ANOVA model with **`aov()`**

This calculates sums of squares, but doesn't calculate F or p-value yet

```
chick.aov <- aov(weight ~ feed, data = chickwts)
chick.aov #don't need to do this, just for demo purposes!
```

```
## Call:
##    aov(formula = weight ~ feed, data = chickwts)
##
## Terms:
##                      feed Residuals
## Sum of Squares  231129.2  195556.0
## Deg. of Freedom        5        65
##
## Residual standard error: 54.85029
## Estimated effects may be unbalanced
```

The **aov** model object

If you type `chick.aov$`, you'll get a dropdown menu that shows what it contains (you can also use `str(chick.aov)`)

We're going to use the residuals!

```
str(chick.aov)
```

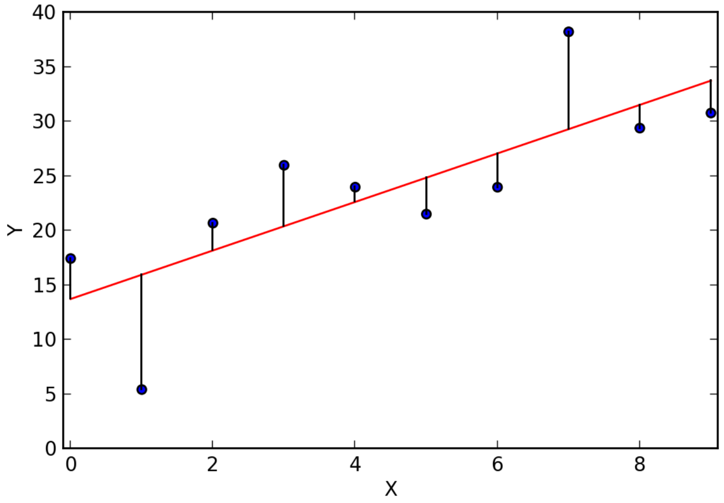
```
## List of 13
##  $ coefficients : Named num [1:6] 323.6 -163.4 -104.8 -46.7 -77.2 ...
##   ..- attr(*, "names")= chr [1:6] "(Intercept)" "feedhorsebean" "feedlinseed" "feedmeatmeal'
##  $ residuals    : Named num [1:71] 18.8 -0.2 -24.2 66.8 56.8 ...
##   ..- attr(*, "names")= chr [1:71] "1" "2" "3" "4" ...
##  $ effects      : Named num [1:71] -2201.8 345 228.6 -58.2 -237.4 ...
##   ..- attr(*, "names")= chr [1:71] "(Intercept)" "feedhorsebean" "feedlinseed" "feedmeatmeal
##  $ rank         : int 6
##  $ fitted.values: Named num [1:71] 160 160 160 160 160 ...
##   ..- attr(*, "names")= chr [1:71] "1" "2" "3" "4" ...
##  $ assign       : int [1:6] 0 1 1 1 1 1
##  $ qr           :List of 5
##   ..$ qr   : num [1:71, 1:6] -8.426 0.119 0.119 0.119 0.119 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:71] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:6] "(Intercept)" "feedhorsebean" "feedlinseed" "feedmeatmeal" ...
```

Doing an ANOVA in R

1. Make the ANOVA model with `aov()`
2. **Check assumptions on residuals from this model**
3. If your model passes the check (#2), run `anova()` on your model

ANOVA step 2a: extract the residuals

What are residuals?



What are residuals?

Residuals are individual values minus group means

$$Y_{ij} - \bar{Y}_i$$

What are residuals?

We *could* calculate residuals manually with `mutate()`

```
chickwts %>%
  group_by(feed) %>%
  mutate(group_mean = mean(weight),
         residuals = weight - group_mean)
```

Get residuals with fortify()

You can get residuals into a data frame format with `fortify()`

`fortify(<<some model>>)` returns a data frame with your original data and some extra columns extracted from the model:

- `.fitted` = "fitted" values (*i.e.* group means)
- `.resid` = residual values
- Don't worry about the rest!

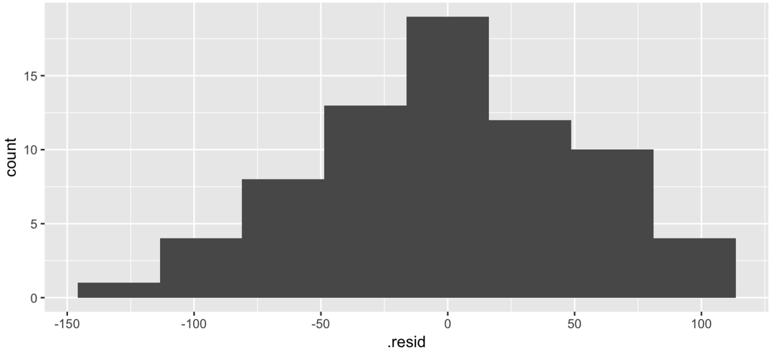
```
fortify(chick.aov)
```

ANOVA step 2b: check normality of residuals

Histogram of residuals

No need to separate by `feed`!

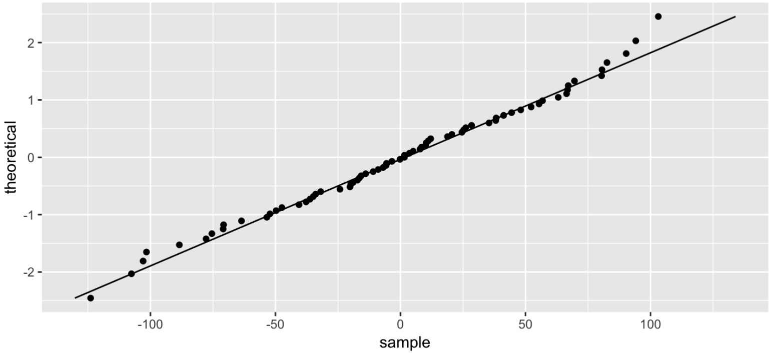
```
ggplot(fortify(chick.aov), aes(x = .resid)) + geom_histogram(bins = 8)
```



Normal probability plot of residuals

No need to separate by `feed`!

```
ggplot(fortify(chick.aov), aes(sample = .resid)) +
  geom_qq() + geom_qq_line() + coord_flip()
```



Shapiro test on residuals

No need to separate by `feed`!

`shapiro.test()` still needs a *vector* rather than a data frame

```
shapiro.test(chick.aov$residuals)
#OR
shapiro.test(fortify(chick.aov)$.resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  chick.aov$residuals
## W = 0.98616, p-value = 0.6272
```

Doing an ANOVA in R

1. Make the ANOVA model with `aov()`
2. Check assumptions on **residuals** from this model
3. **If your model passes the check (#2), run `anova()` on your model**

ANOVA step 3: run the test!

Use **`anova()`** to run the test.

Now we get to see our p-value!

```
anova(chick.aov)
```

```
## Analysis of Variance Table
##
## Response: weight
##           Df Sum Sq Mean Sq F value    Pr(>F)
## feed       5 231129   46226  15.365 5.936e-10 ***
## Residuals 65 195556    3009
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What if data aren't normal?

- `InsectSprays` is data on the effectiveness of insecticides
  - Researchers applied insecticides A through F
  - Then they counted insects in the fields
- Unlikely to be normal since it is **count** data

```
InsectSprays
```

Summarize

One thing we can do is calculate some summary statistics to get an idea if it meets assumptions of ANOVA

```
insectsummary <- InsectSprays %>%
  group_by(spray) %>% summarize(n = n(), var = var(count))
insectsummary
```

Sample size is not exactly *huge*, and variances differ by > 10×

Coding Hack: `kable()`

Making pretty tables

- Data frames look pretty and interactive in your .Rmd file, but print out like boring R output in Word
- Format them as actual tables with `kable()` from `knitr`!
- You already have `knitr` installed, but you have to load it or specify with `::`

```
insectsummary %>% knitr::kable() %>% print()
```

```
##
##
## spray     n         var
## ------  ---  ----------
## A        12   22.272727
## B        12   18.242424
## C        12    3.901515
## D        12    6.265151
## E        12    3.000000
## F        12   38.606061
```

Check for homogeneity of variances

Let's formally test our suspicions about variance

```
leveneTest(count ~ spray, data = InsectSprays)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value   Pr(>F)
## group  5  3.8214 0.004223 **
##       66
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Set up the model

Let's set up the model with `aov()` and `fortify()` it so we can use **residuals** to check the normality

```
insect.m <- aov(count  ~ spray, data = InsectSprays)
insect.fort <- fortify(insect.m)
head(insect.fort) #don't need to inspect fortified data for homeworks.  Just for demonstration.
```

Check assumption of normality

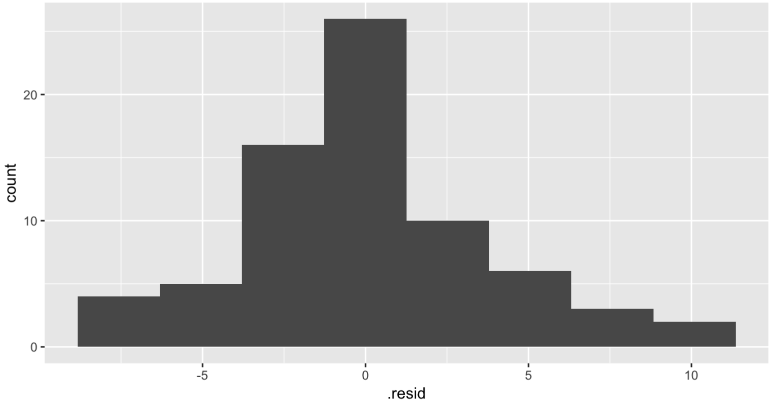
We can do this a few ways:

- With a histogram
- With a normal probability plot
- With `shapiro.test()`

With a histogram

What do you think?

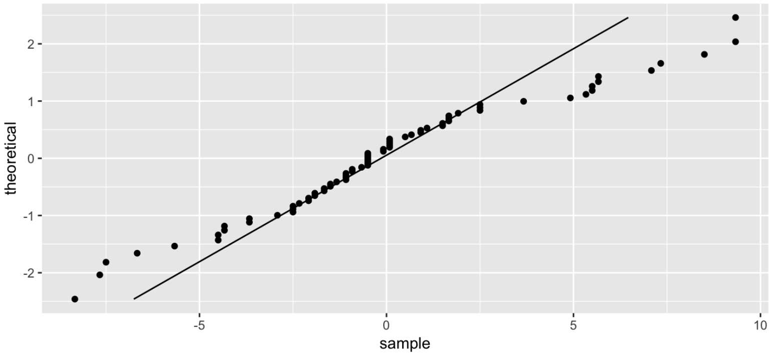
```
ggplot(insect.fort, aes(x = .resid)) + geom_histogram(bins = 8)
```



With a normal probability plot

What do you think? (Feel free to refer to your handout)

```
ggplot(insect.fort, aes(sample = .resid)) +
  geom_qq() + geom_qq_line() + coord_flip()
```



With `shapiro.test()`

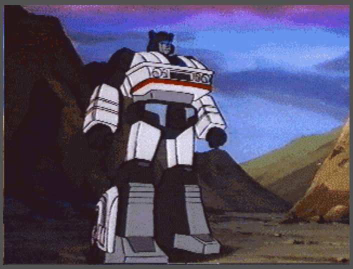
What do you think?

```
shapiro.test(insect.m$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  insect.m$residuals
## W = 0.96006, p-value = 0.02226
```

Do the data meet our assumptions?

- Histogram: a little leptokurtic
- Normal probability plot: even more leptokurtic
- Shapiro-Wilk test: $p > 0.01$, so not terrible
- Levene's test: doesn't pass, unequal variance
- Sample size: 12 each (not great)



Transform!

Transform!

Count data are often "fixed" by a log transformation

But I should check to see if there are zeroes in the data first!

```
InsectSprays %>% filter(count == 0)
```

There are, so I'll try `log(count + 1)`

Transform!

```
insects <- InsectSprays %>% mutate(log_count = log(count + 1))
head(insects, 4)
```

NOTE: You *could* overwrite `InsectSprays`, but it's generally a pretty bad idea to overwrite built-in datasets

If you do, it's not permanent, but you will have to run `data(InsectSprays)` to get the original back!

Re-check the transformed data

I'll start with the normal probability plot, skip the histogram, and then double check with `shapiro.test()`

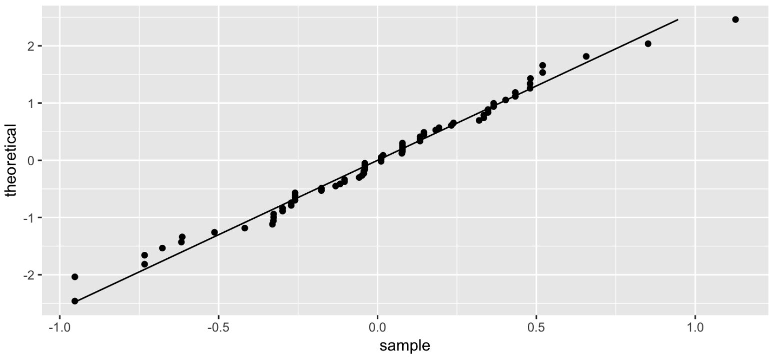
- First, re-fit the `aov()` model and extract residuals again

```
insects.m2 <- aov(log_count ~ spray, data = insects)
insects.fort2 <- fortify(insects.m2)
```

Re-check with a normal probability

Wow! Looks like we got lucky!

```
ggplot(insects.fort2, aes(sample = .resid)) +
  geom_qq() + geom_qq_line() + coord_flip()
```



Re-check with `shapiro.test()`

Nice!

```
shapiro.test(insects.fort2$.resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  insects.fort2$.resid
## W = 0.98475, p-value = 0.5348
```

What about the variances?

Transforming data not only affects normality, but can also mess with homogeneity of variances

Let's check to see if our variance problem is fixed…

```
leveneTest(log_count ~ spray, data = insects)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  5  1.8821 0.1093
##       66
```

Excellent!

ANOVA

Do an ANOVA on the log transformed data that we used to set up `insects.m2`

```
anova(insects.m2)
```

```
## Analysis of Variance Table
##
## Response: log_count
##           Df Sum Sq Mean Sq F value    Pr(>F)
## spray      5 38.518  7.7035  46.007 < 2.2e-16 ***
## Residuals 66 11.051  0.1674
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The mean log of insect count plus 1 significantly differs among spray types (ANOVA, F = 46.007, df = 5, p < 0.0001).

Non-parametric ANOVA alternative

Kruskal-Wallis test

Works like `aov()` and `anova()` combined

Uses the formula interface like `aov()`, but there's no need to save the model and run `anova()`

```
kruskal.test(count ~ spray, data = InsectSprays)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  count by spray
## Kruskal-Wallis chi-squared = 54.691, df = 5, p-value = 1.511e-10
```

Insect count differs significantly by spray type (Kruskal-Wallis test, X^2 = 54.691, df = 5, p < 0.0001)

Homework time!