Statistical Models/Tests Part1

Welcome to this week's SummeR of R session, our first of two sessions learning about statistical models/tests in R!

Today we'll cover:

- Checking for normality and equal variance
- T-tests and Anovas

Set up

To get started let's load or install the library's we'll be using. If this is your first time using one of the packages uncomment and run the appropriate install.package('package')

```
#install.packages('tidyverse')
library(tidyverse)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

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

```
## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.2     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

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

```
#install.packages('car')
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
#install.packages('broom')
library(broom)
```

We're going to be using some wine_ratings data set from the Tidy Tuesday repository.Let's load that in now.

```
wine_ratings <- readr::read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/da
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   country = col_character(),
##   description = col_character(),
##   designation = col_character(),
##   points = col_double(),
##   price = col_double(),
##   province = col_character(),
##   region_1 = col_character(),
##   region_2 = col_character(),
##   taster_name = col_character(),
##   taster_twitter_handle = col_character(),
##   title = col_character(),
##   variety = col_character(),
##   winery = col_character()
## )
```

Could I be a wine rater? A one sample t-test

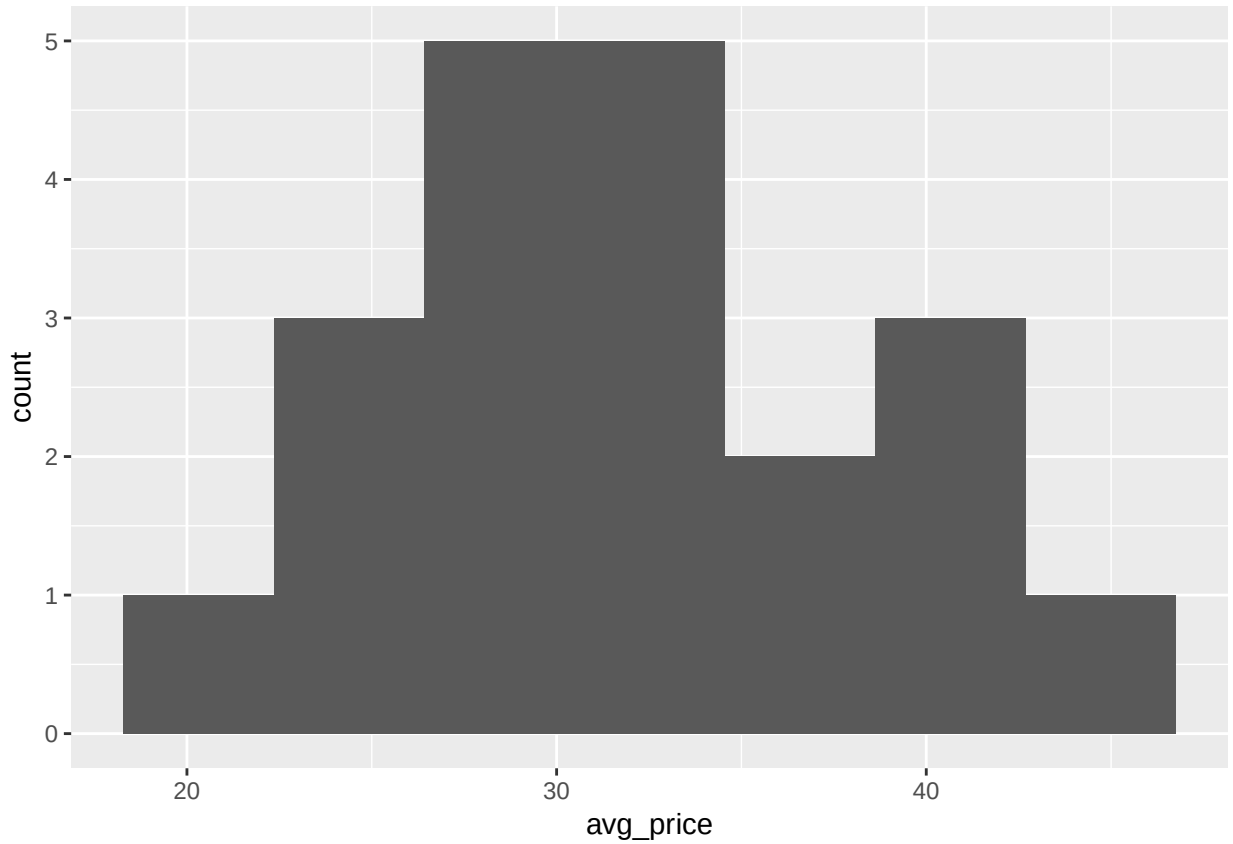
I would like to be a wine rater but I think that most wine raters are probably tasting more expensive wine than any I buy. Let's say that I am willing to spend on average $25 for a bottle of wine (not really). Would I be comparable to other reviewers then? I am going to run a one sample t-test to see if that average cost would be different from the average for the whole group of reviewers.

Below I am going to reorganize my data so that I have the average price for each reviewer.

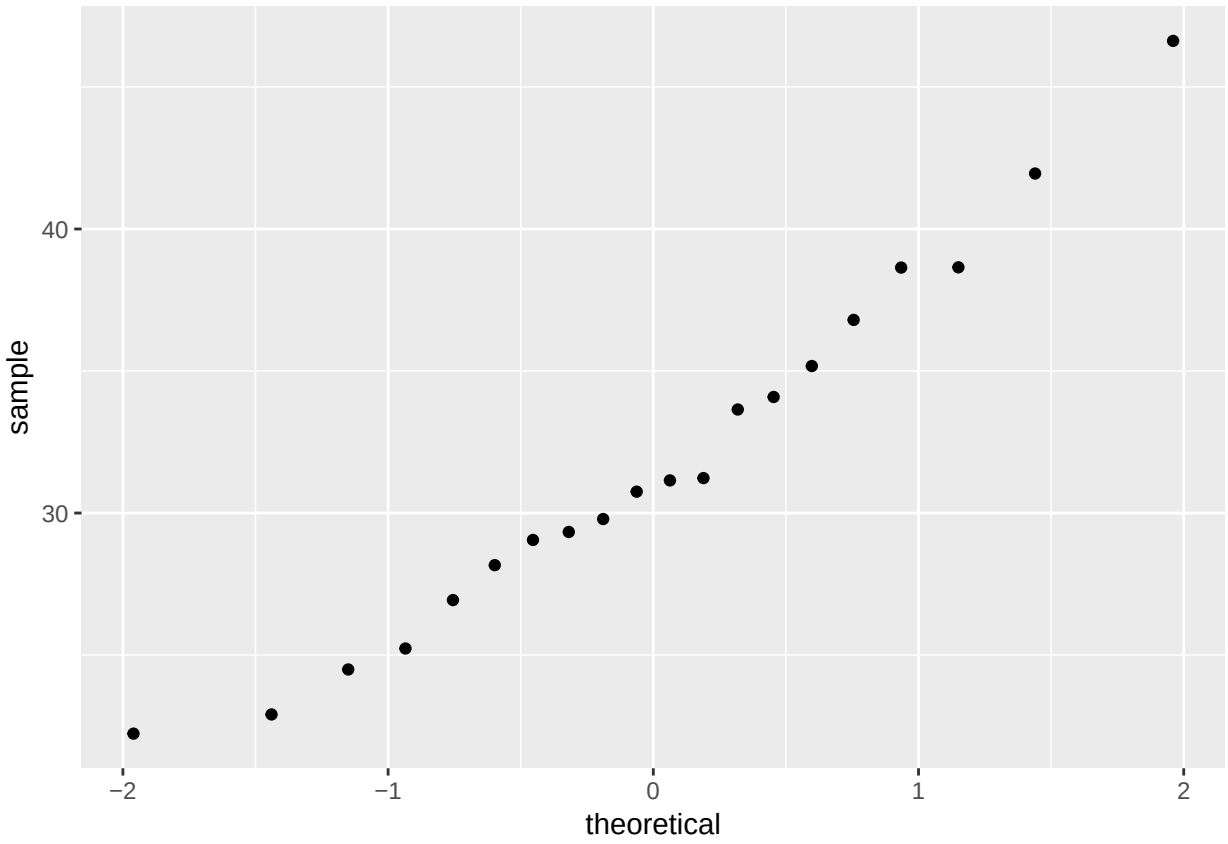
```
reviewers<-wine_ratings %>% group_by(taster_name) %>% summarize(avg_price = mean(price, na.rm=TRUE))
```

I'm going to plot my data now to check to see if it's normally distributed and to make sure there are no outliers.

```
# A histogram
ggplot(reviewers, aes(x=avg_price))+
  geom_histogram(bins=7)
```




```
#A qqplot
ggplot(reviewers, aes(sample=avg_price))+
  geom_qq()
```



The qqplot suggests the data are normally distributed, but let's do a Shapiro-Wilks test to find out. The *shapiro.test()* function is part of the stats package which is included in base R so you don't need to load or install it. If you take a look at the documentation, you'll see its input is a vector of data values. Each column in our data frame is a vector. When we want to call a specific column in R, we call the dataframe name and then follow it with a dollar sign then type the column name (see below reviewer$avg_price) this is how we index specific columns in R.

```
shapiro.test(reviewers$avg_price)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  reviewers$avg_price
## W = 0.96825, p-value = 0.7175
```

Okay, so our p-value suggests we fail to reject null that the data are normally distributed. We'll go on using the parametric t-test.

So is the average for all the reviewers significantly different than my proposed average of $25?

```
t.test(reviewers$avg_price, mu=25)
```

```
##
##  One Sample t-test
##
## data:  reviewers$avg_price
## t = 4.7607, df = 19, p-value = 0.000136
## alternative hypothesis: true mean is not equal to 25
## 95 percent confidence interval:
##  28.83418 34.85075
```

```
## sample estimates:
## mean of x
##  31.84247
```

We see that my proposed average cost per bottle would be significantly different from the reviewers rating. But I actually don't just care about if its different or not, I care about the direction of the difference. I don't want to be rating on average cheaper wine. So I want to test if the mean of the population is significantly greater than my proposed cost.

```
t.test(reviewers$avg_price, mu=25, alternative = "greater")
```

```
##
##  One Sample t-test
##
## data:  reviewers$avg_price
## t = 4.7607, df = 19, p-value = 6.799e-05
## alternative hypothesis: true mean is greater than 25
## 95 percent confidence interval:
##  29.3572     Inf
## sample estimates:
## mean of x
##  31.84247
```

Am I missing out? Comparing two means: expensive and affordable wines

So the average reviewer would spend more on a bottle of wine than I would be willing to. But does that really mean the yare tasting better wines than I would be?

Lets sort the wines into two categories based on their affordability. Then I can compare the wines in each group to see if they actually score better.

```
wine_ratings<-wine_ratings %>% mutate(affordability = factor(case_when(price > 25 ~'expensive', TRUE ~
```

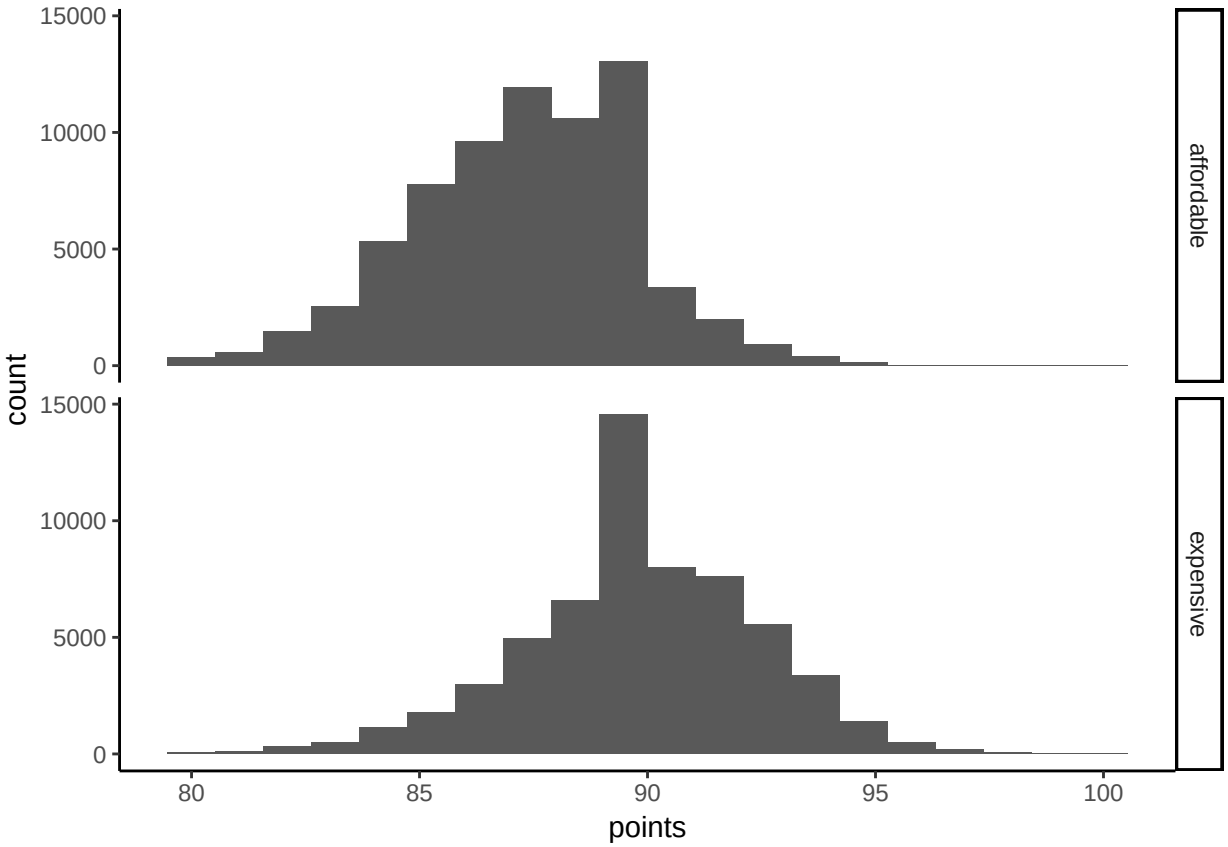
You might see something different in that bit of code. I put the *case_when()* function inside the *factor()* function. Factors can be handy when you have categorical variables. They are a computationally more efficient way to work with your variables. Also now when you get a summary of your dataframe you get the count of each factor in your dataframe.

```
summary(wine_ratings$affordability)
```

```
## affordable  expensive
##      70272      59699
```

We have so many observations, I'm not going to worry about assumptions much. But for good measure let's plot the two groups to make sure there's nothing weird going on.

```
ggplot(wine_ratings)+
  geom_histogram(aes(x=points), bins=20)+ #default is 30 bins, you can play around with this a bit
  facet_grid(affordability~.)+
  theme_classic()
```



We can check to see if the two samples have equal variance by performing a leveneTest, inside the functions we put a formula of the structure 'continuous_variable ~ grouping' our continuous variable is the points value and the samples are grouped by their category in the affordability column.

```
leveneTest(points~affordability, data=wine_ratings)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##           Df F value    Pr(>F)
## group      1   620.5 < 2.2e-16 ***
##       129969
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Levene's test has a null hypothesis that the variances are equal,based on the results of our test we would reject the null and assume unequal variance.

We need the data to be in vectors (i.e. a list of elements all of the same data type) in order to perform the t-test. To do the two sample t-test we actually need one vector with the points data for the affordable wines and then another vector of the points data for the expensive wines. We can use filter to grab just the rows that are affordable and then we use pull to just grab the points values for those rows. Pull will return the points values as a vector (unlike *select()* which would give them in the form of a dataframe). We do the same thing for the expensive wines.

```
affordable<-filter(wine_ratings, affordability == 'affordable') %>% pull(points)
expensive <- filter(wine_ratings, affordability == 'expensive') %>% pull(points)
```

Now that we have the data in the form of two vectors we can do another variance test using *var.test()* which uses and F test, where the null hypothesis is that the ratio of the two variances is 1.

```r
var.test(affordable, expensive)
```

```
##
##  F test to compare two variances
##
## data:  affordable and expensive
## F = 0.83579, num df = 70271, denom df = 59698, p-value < 2.2e-16
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.8229916 0.8487822
## sample estimates:
## ratio of variances
##          0.8357915
```

We also reject the null hypothesis here and we'll assume unequal variances for the t-test.

Using this dataset I found an interesting limit to the *shapiro.test()*:

```r
#shapiro.test(affordable)
#shapiro.test(expensive)
```

It has a 5000 sample maximum, at that point you may want to look at a qqplot to see if the sample deviates from normal in a way that is meaningful for your analysis.

We'll use the *t.test()* to compare the two means, the default for the function is to assume variances are not equal (which works here), and it performs a Welch Two Sample t-test. If you had equal variance you would set the argument var.equal=TRUE which would run a Two-sample t-test.

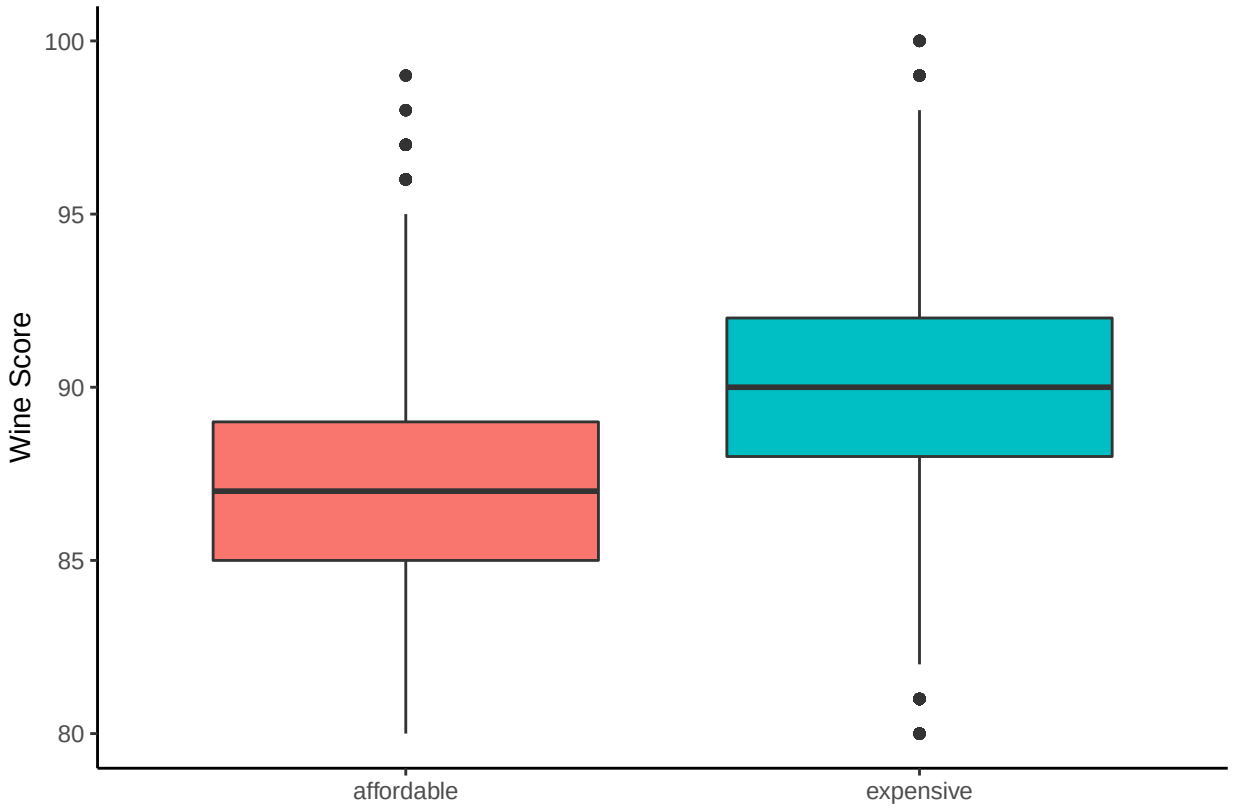
```r
t.test(affordable, expensive) #if variances were equal we would have added the argument var.equal=TRUE
```

```
##
##  Welch Two Sample t-test
##
## data:  affordable and expensive
## t = -183.17, df = 122223, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.808689 -2.749218
## sample estimates:
## mean of x mean of y
##  87.17069  89.94965
```

The two samples are significantly different from each other, the affordable wines have just slightly lower scores than the expensive wines.

We can see the two samples below:

```r
ggplot(wine_ratings)+
  geom_boxplot(aes(x=affordability, y=points, fill=affordability))+
  theme_classic()+
  ylab('Wine Score')+
  xlab('')+
  theme(legend.position = 'none')
```



Note on paired t-tests
If you wanted to run a paired t-test you would have the two vectors and specify the argumend *paired=TRUE* inside the t.test() function

Note on wilcoxon test If the data had not been normally distributed and you could not transform it to be normal, then you could run the wilcoxon / mann-whitney test with the function *wilcox.test()* inside you have one vector of data and either another vector or a mean value you want to compare your sample to, you can also specify an argument alternative = 'greater' or 'less'

Comparing means from multiple groups

Can I somehow select wines that give me the best bang for my buck (most points/price)? Would it make sense to choose wines only from certain countries? Let's if country of origin signficantly affects the ratio of points/price.

First we'll narrow down the dataset a bit, to include only the 5 countries with most wine reviews and clean the data a little bit.

```
#grab the 5 most reviewed countries
country_df<-wine_ratings %>% group_by(country) %>% summarise(count=n()) %>% top_n(5)
```

```
## Selecting by count
```

```
#use that to filter-join the wine_ratings list and select only columns I'm going to be interested in.
country_df<-semi_join(wine_ratings, country_df)%>% select(country, price, points, variety)%>%drop_na()
```

```
## Joining, by = "country"
```

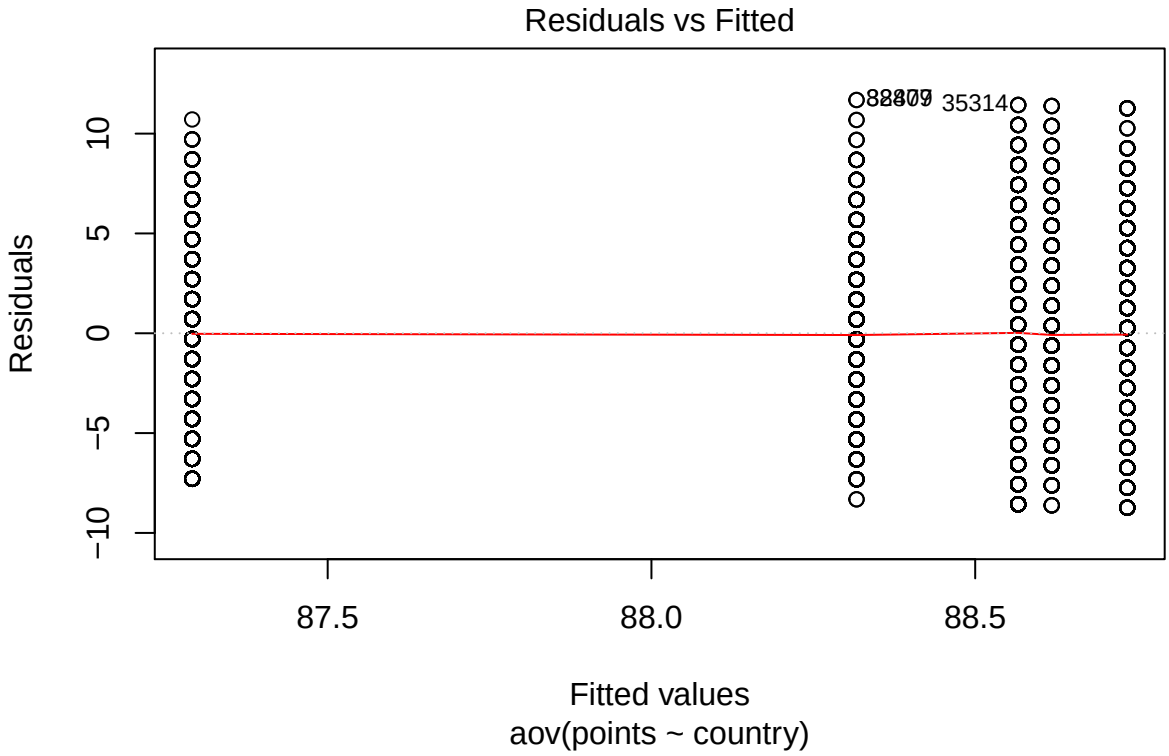
```
#we'll turn country into a factor
country_df<-mutate(country_df, country=factor(country))
```

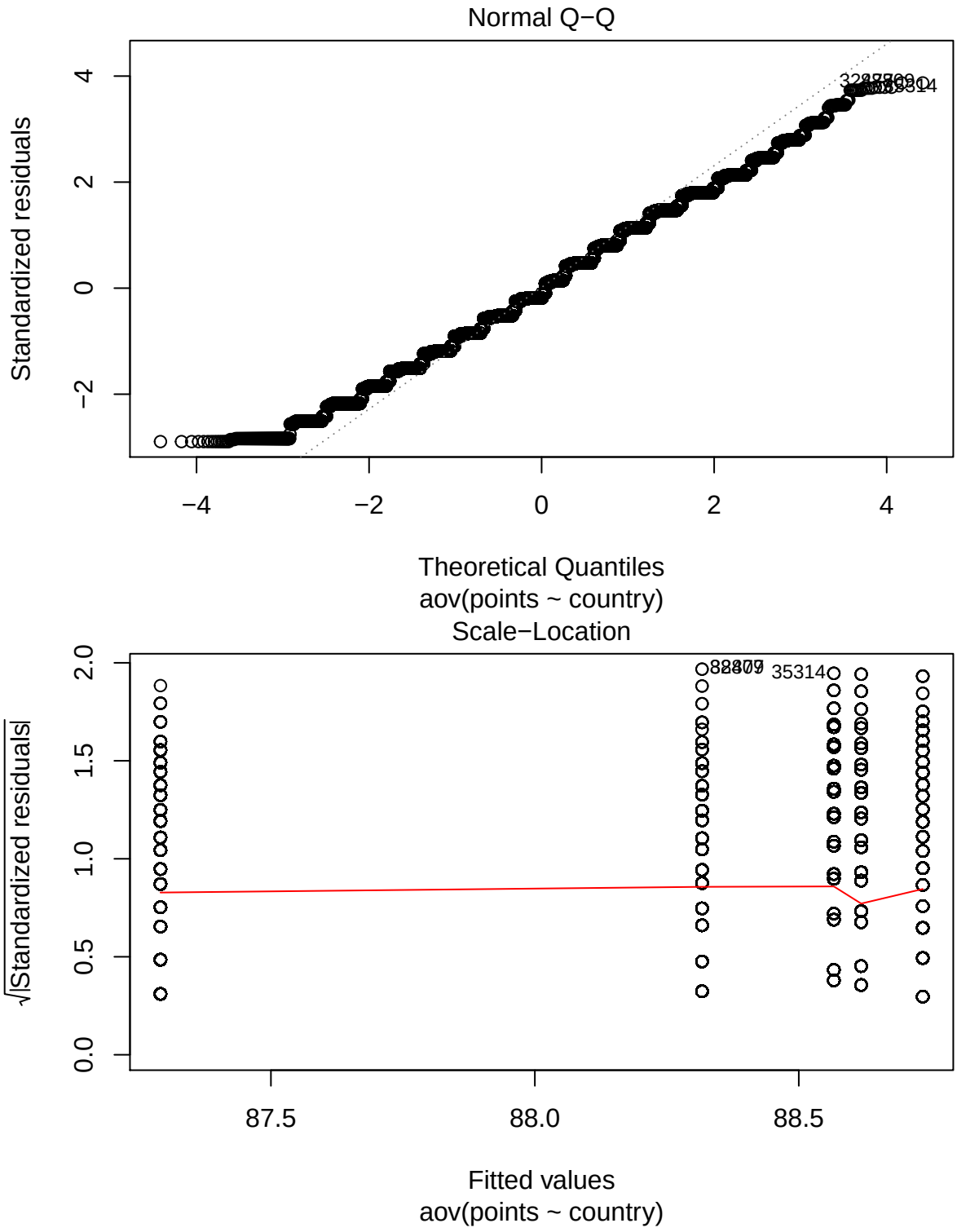
We can use the *aov()* function to run our anova. We'll check our model assumptions after we fit the model. You want to assign the model fit to a variable name because we'll use it to get the statistics and check assumptions.

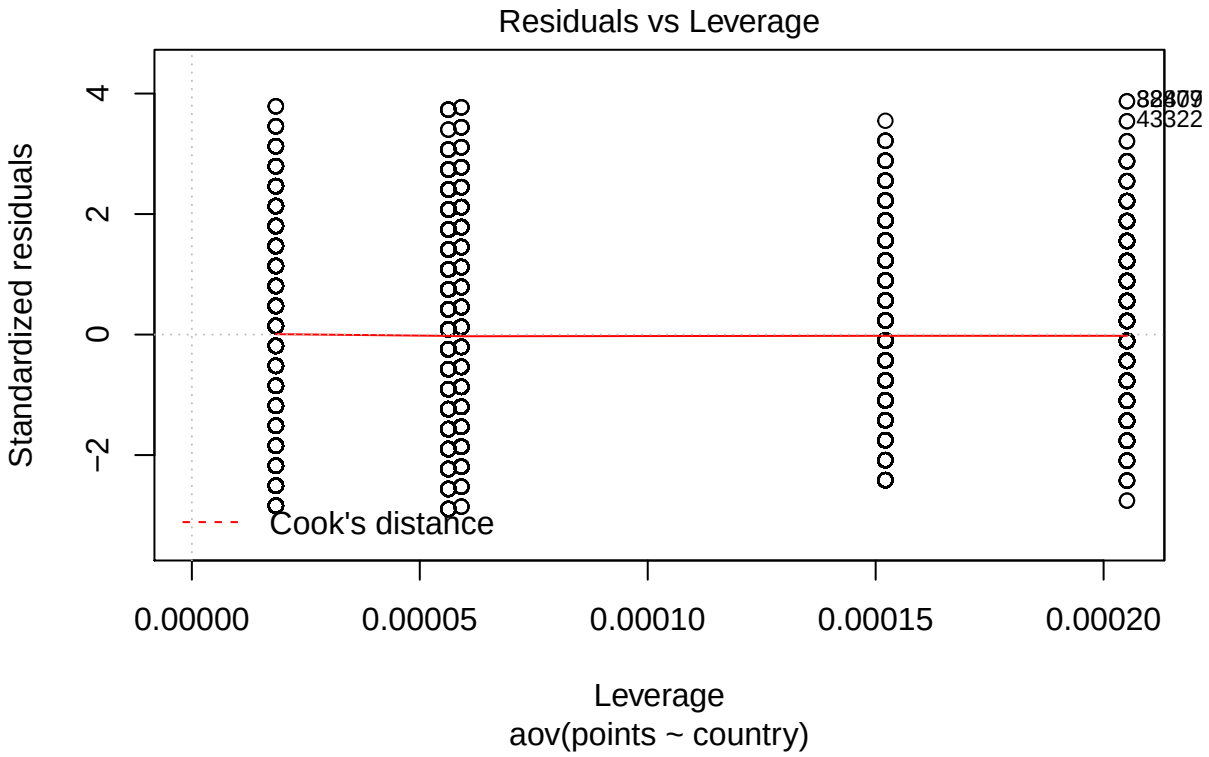
```
anova<-aov(points~country, data=country_df)
```

You can use the functions we used earlier to test model assumptions (e.g. shapiro.test(), leveneTest()). You can also check them out by using the base R *plot()* and putting variable name you assigned the model fit to inside of it. This will give you four plots that can let you interpet if residuals meet assumptions for heteroscedacity and normality.

```
plot(anova)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
aov(points ~ country)
Scale−Location
√|Standardized residuals|
Fitted values
aov(points ~ country)



Okay now we can check out the fit of our anova model. We use summary to print the model statistics.

```
summary(anova)
```

```
##                 Df Sum Sq Mean Sq F value Pr(>F)
## country          4  11223  2805.7     308 <2e-16 ***
## Residuals   100398 914593     9.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The null hypothesis for the anova is the average points for wine is the same for wines from all five countries. That is:

$H_0 : \mu_{Spain} = \mu_{Portugal} = \mu_{Italy} = \mu_{US} = \mu_{France}$

Our anova results rejects this null, so we know **at least one** of the means is significantly different. Let's do run Tukey's post hoc to see how the groups vary. We use the *TukeyHSD()* and put the variable we named our anova inside.

```
TukeyHSD(anova)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = points ~ country, data = country_df)
##
## $country
##                         diff        lwr         upr     p adj
## Italy-France    -0.11668112 -0.2051156 -0.02824659 0.0029492
## Portugal-France -0.41814929 -0.5512558 -0.28504279 0.0000000
## Spain-France    -1.44413241 -1.5629831 -1.32528175 0.0000000
## US-France       -0.16848006 -0.2396297 -0.09733040 0.0000000
## Portugal-Italy  -0.30146817 -0.4353027 -0.16763367 0.0000000
```

```
## Spain-Italy     -1.32745129 -1.4471167 -1.20778587 0.0000000
## US-Italy        -0.05179894 -0.1243014  0.02070352 0.2914992
## Spain-Portugal  -1.02598312 -1.1815996 -0.87036665 0.0000000
## US-Portugal      0.24966923  0.1265706  0.37276789 0.0000003
## US-Spain         1.27565235  1.1681283  1.38317640 0.0000000
```

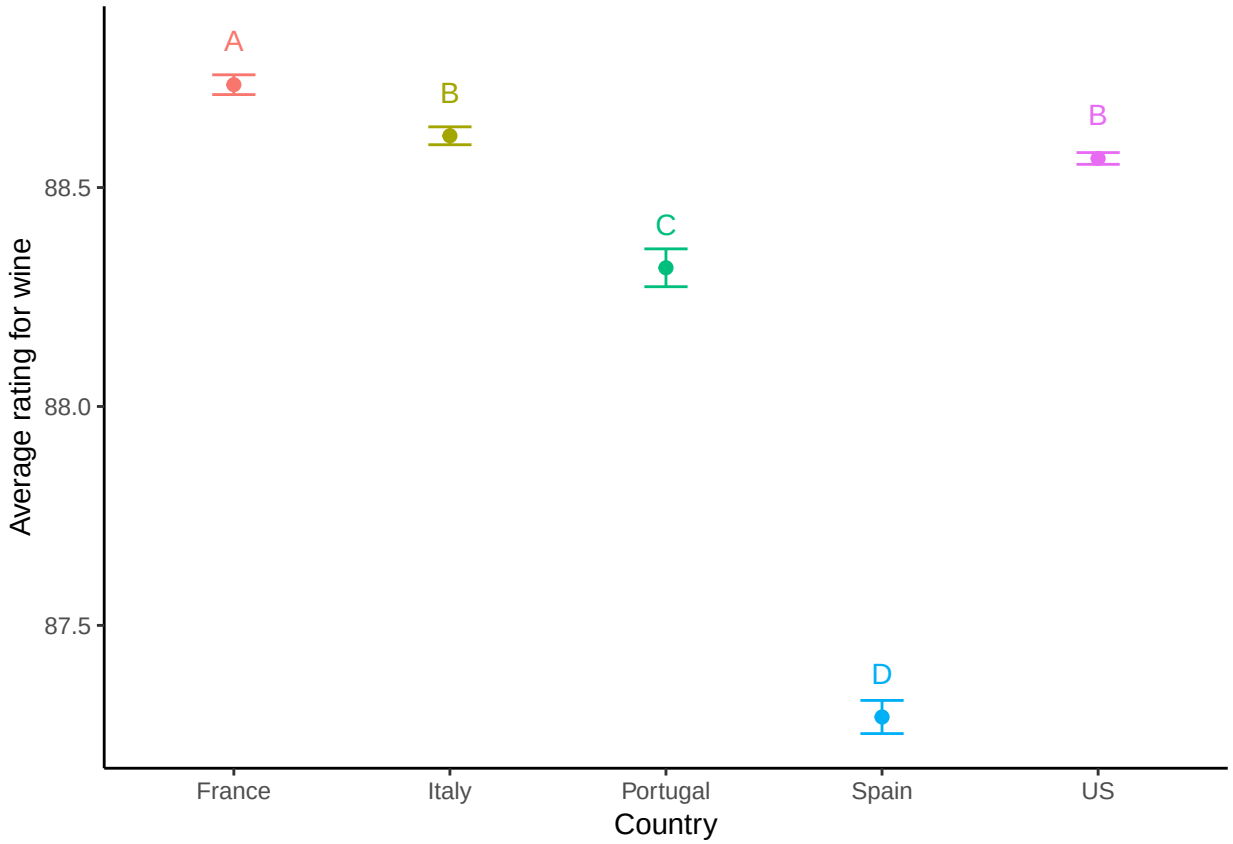
This is kind of hard to work. Luckily the tidyverse has a package called broom (we loaded this in earlier) and broom has a function called *tidy()*

```
Tukeys<-TukeyHSD(anova)%>%tidy()

#Let's highlight significant differences
Tukeys<-Tukeys %>% mutate(sig = case_when(adj.p.value < .05~ '*', TRUE ~''))
Tukeys
```

```
## # A tibble: 10 x 7
##    term    comparison      estimate conf.low conf.high adj.p.value sig
##    <chr>   <chr>              <dbl>    <dbl>     <dbl>       <dbl> <chr>
##  1 country Italy-France     -0.117    -0.205   -0.0282    2.95e- 3 *
##  2 country Portugal-France  -0.418    -0.551   -0.285     4.53e-14 *
##  3 country Spain-France     -1.44     -1.56    -1.33      0.       *
##  4 country US-France        -0.168    -0.240   -0.0973    1.05e- 9 *
##  5 country Portugal-Italy   -0.301    -0.435   -0.168     8.02e- 9 *
##  6 country Spain-Italy      -1.33     -1.45    -1.21      0.       *
##  7 country US-Italy         -0.0518   -0.124    0.0207    2.91e- 1 ""
##  8 country Spain-Portugal   -1.03     -1.18    -0.870     0.       *
##  9 country US-Portugal       0.250     0.127    0.373     3.15e- 7 *
## 10 country US-Spain          1.28      1.17     1.38      0.       *
```

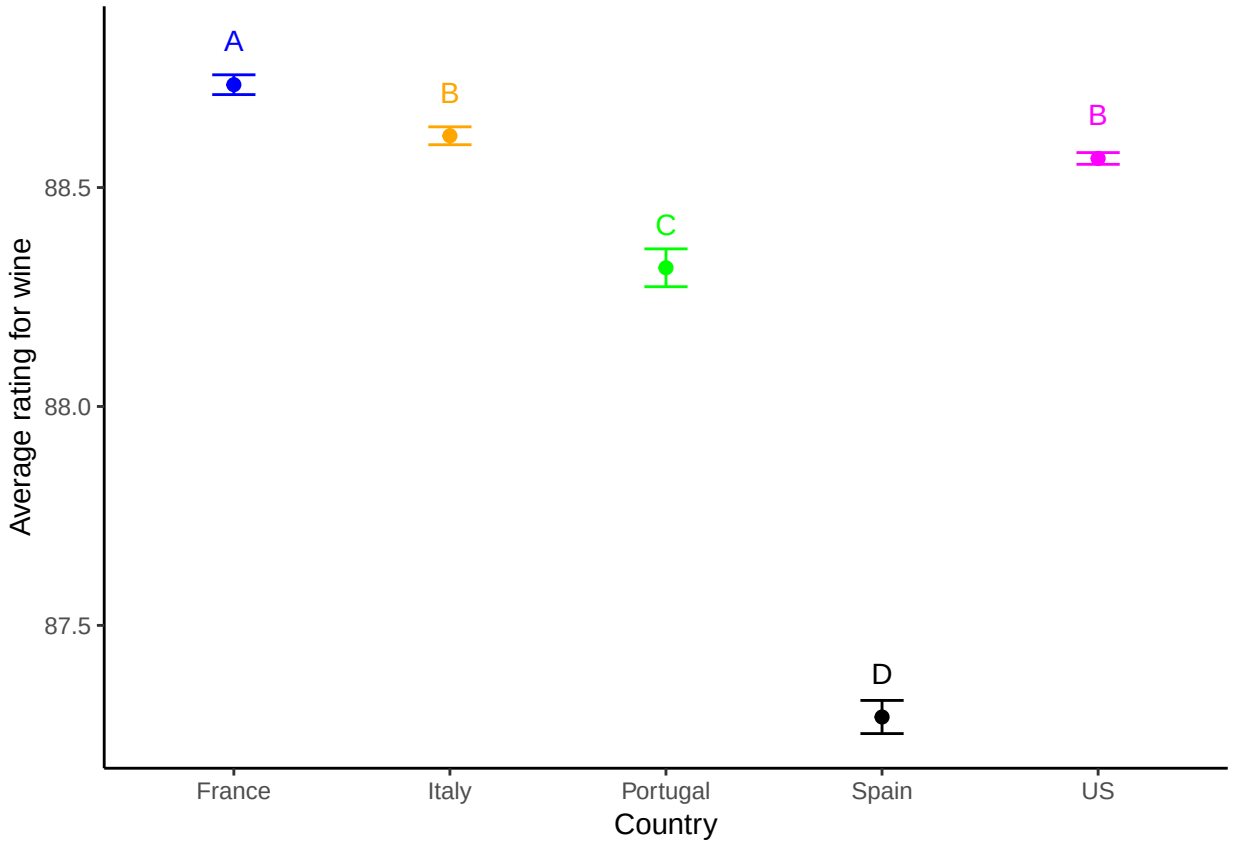
Let's plot this now. We can use summarize to get our means, se, and then add a column with our Tukey groupings

```
country_summary <-country_df %>%
  group_by(country) %>%
  summarise(avg_rating=mean(points), se = sd(points)/ (sqrt(n())))%>%
  mutate(TukeyGroup = c('A', 'B', 'C', 'D', 'B'))

#Then we make the plot
plot<-ggplot(country_summary, aes(x=country, y=avg_rating, color=country))+
        geom_point(size=2)+
        geom_text(aes(label=TukeyGroup), nudge_y=.1)+
        xlab('Country')+
        ylab('Average rating for wine')+
        geom_errorbar(aes(ymin=avg_rating-se, ymax=avg_rating+se), width=.2)+
        theme_classic()+
        theme(legend.position = 'NA')
#Here it is!
plot
```



If you want to change the colors of your graph you can set the colors:

```
plot+scale_color_manual(values=c('blue','orange', 'green', 'black', 'magenta')) # can also use hex#
```



The package ggsci has some color palettes you may want to try! Uncomment the code

```
#install.packages('ggsci')
#library(ggsci)

#plot+scale_color_lancet()
```

Check out more ggsci options

Like your graph and want to save it?

```
#ggsave('CountryWines.pdf') #it will save the last plot you made by default under the name & with the e
```

Real quick other kinds of ANOVA, to test for an interaction

```
anova_interaction<-aov(points~country*variety, data=country_df)
summary(anova_price)
TukeyHSD(anova_price)
```

Note on Kruskal Wallis If your data does not meet the assumptions for normality to run an anova you may want ot run a kruskal-wallis test you can do that in R with the function *kruskal.test()* the arguments inside are similar to the anova where you would have a formula 'y~x'

Good resources to check out for more variations/details:

- R cookbook by Paul Teetor
- Learning Statistics with R by Danielle Navarro

What to do now?

Practice wrangling and running stats on the titanic data set. Some questions you could answer: - Is there a significant difference in age of the people that survived vs. not? - Is there a significant difference in age across passenger classes?

```
#Load this package
#install.packages('titanic')
#library(titanic)
#The package has a dataframe called Titanic that is now available for you to access
#View(Titanic)
```

This week's TidyTuesday

```
ufo_sightings <- readr::read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/
```

```
## Parsed with column specification:
## cols(
##   date_time = col_character(),
##   city_area = col_character(),
##   state = col_character(),
##   country = col_character(),
##   ufo_shape = col_character(),
##   encounter_length = col_double(),
##   described_encounter_length = col_character(),
##   description = col_character(),
##   date_documented = col_character(),
##   latitude = col_double(),
##   longitude = col_double()
## )
```