Assignment 5

Due Thursday October 10 at 11:59pm on Blackboard

As before, the questions without solutions are an assignment: you need to do these questions yourself and hand them in (instructions below).

The assignment is due on the date shown above. An assignment handed in after the deadline is late, and may or may not be accepted (see course outline). My solutions to the assignment questions will be available when everyone has handed in their assignment.

You are reminded that work handed in with your name on it must be *entirely your own work*.

Assignments are to be handed in on Quercus. See `https://www.utsc.utoronto.ca/~butler/c32/quercus1.nb.html` for instructions on handing in assignments in Quercus. Markers' comments and grades will be available there as well.

Start with this. I think it's likely we'll be using something from `smmr` here, so I'm loading that as well. Install it first (see the lecture notes if you need help).

```
library(tidyverse)
library(smmr)
```

1. Work through Chapter 10 of PASIAS (on matched pairs and the matched pairs sign test). This will help you with the catfood question below.
2. Work through problems 9.4 through 9.6 in PASIAS (on Mood's median test). This will help you with the Yukon wildfires problem below.
3. Which cat food do cats prefer? A pet food company was comparing two recipes, A and B. A random sample of 10 cats was taken. Two bowls of food were placed in front of each cat, with each bowl containing food prepared from one of the recipes. Trained observers gave each a score for each food, depending on the cat's behaviour and the amount of food eaten from each bowl. The data are in `http://www.utsc.utoronto.ca/~butler/assgt_data/catfood.txt`. The pet food company is interested in whether there is any evidence that cats prefer one of the recipes over the other.

   (a) (2 marks) Read in and display the data.

   Solution:

   Look at the data file to see that the values are separated by one space:

```r
my_url="http://www.utsc.utoronto.ca/~butler/assgt_data/catfood.txt"
cats <- read_delim(my_url, " ")
## Parsed with column specification:
## cols(
##   cat_id = col_character(),
##   recipe_A = col_double(),
##   recipe_B = col_double()
## )
cats
## # A tibble: 10 x 3
##    cat_id recipe_A recipe_B
##    <chr>     <dbl>    <dbl>
##  1 A           9.4      8.4
##  2 B           4.9      4.7
##  3 C           9.3      7.3
##  4 D           6.8      6
##  5 E           7.5      8.3
##  6 F           6.2      5.4
##  7 G           7.5      8
##  8 H           6.7      5.5
##  9 I           8.6      6.9
## 10 J           7.2      6.1
```

Extra: the first time I did this, I got the URL wrong, and what happened was that I was redirected to U of T's "404 File Not Found" page (try this for yourself, eg. by editing the `~butler` out of the URL, and trying the `read_delim` again): what happens is that `read_delim` will try to read the HTML as a file delimited by spaces! (I eventually realized that this is what was happening, by copying the URL into my web browser). In this kind of case, it would be easier to simply be told that the file didn't exist, and then we'd know to check the URL, rather than having a strange file get read in and have to figure out where on earth it came from.

(b) (2 marks) What kind of experimental design is this: matched pairs, two independent samples, or something else? Explain briefly.

Solution: This is a matched pairs design, because each cat gets to try *both* recipes, and so each cat produces two measurements.

Extra: the usual way of running a matched pairs design is to give each subject one treatment, then take a break, then give each subject the other treatment (with, for example, the order of presentation of treatments being randomized by a coin flip). This one is not quite like that, because you can imagine the recipes "competing" with each other for the cat's attention, and the pet food company wanted to see which one a cat would prefer if you gave them both. The value of the scoring system here is that you can see *how much* each cat preferred one recipe to the other. If they had just recorded which recipe each cat preferred, you would have had less information to base a decision on.

The other usual advantage of a matched pairs is present here as well: you are comparing the recipes on the *same* cats, rather than giving one group of ten cats one recipe and a *different* group of cats the other recipe. The disadvantage of *that* (which would be a two-independent-samples experiment that you might analyze with a Welch or pooled t-test) is that the two groups of cats might have come out different by chance, for example the older cats might have ended up mainly in one group, and, say, older cats tended to prefer recipe B. With a matched pairs experiment you don't have to worry about that.

(c) (3 marks) Run a suitable t-test. What do you conclude, in the context of the data? (If you need to do any data manipulation to do the test, do that first.)

Solution: This looks like "wide format" data, but that is exactly what the matched pairs analysis wants, so no manipulation is needed. Hence, straight to this. This way has no squiggle or `data=`, so you need to specify the data frame using `with` or dollar signs, of which there would be two here, one on each of the recipes. We are looking for *any* difference, so this is a two-sided test, the default:

```
with(cats, t.test(recipe_A, recipe_B, paired=T))
```

```
##
##  Paired t-test
##
## data:  recipe_A and recipe_B
## t = 2.6656, df = 9, p-value = 0.02581
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.1135065 1.3864935
## sample estimates:
## mean of the differences
##                    0.75
```

The P-value is less than 0.05, so we reject the null hypothesis of equal mean scores for the two recipes, and conclude that the recipes do differ from each other (in mean score).

Extra: the confidence interval contains only positive values, and we were comparing recipes A and B in that order (because we input them to `t.test` in that order), which means that cats actually prefer recipe A.

Alternative: what the paired t-test is doing is working out the differences and testing whether those have mean *zero*, so you can physically do it that way as well:

```
cats %>% mutate(diff=recipe_A-recipe_B) -> cats2
with(cats2, t.test(diff, mu=0))
```

```
##
##  One Sample t-test
##
## data:  diff
## t = 2.6656, df = 9, p-value = 0.02581
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.1135065 1.3864935
## sample estimates:
## mean of x
##      0.75
```

You can also do it in a pipeline, though there is a subtlety:

```
cats %>% mutate(diff=recipe_A-recipe_B) %>%
  with(., t.test(diff, mu=0))
```

```
##
##  One Sample t-test
##
## data:  diff
## t = 2.6656, df = 9, p-value = 0.02581
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.1135065 1.3864935
## sample estimates:
## mean of x
##      0.75
```

The first input to **with** has to be a data frame, and the one it needs to be is the one with **diff** in it, which doesn't have a name. So you use the name **.** for it, which means "the data frame that came out of the previous step". Or, of course, you can explicitly give it a name, as I did with **cats2** above, and then use that name.

Can you do it this way?

```
cats %>% mutate(diff=recipe_A-recipe_B) %>%
  with(t.test(diff, mu=0))
```

```
##
##  One Sample t-test
##
## data:  diff
## t = 2.6656, df = 9, p-value = 0.02581
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.1135065 1.3864935
## sample estimates:
## mean of x
##      0.75
```

In fact, you can, but I think this is very confusing, because you have to stop and think about where the data frame for the **with** went. It is implicitly the data frame that came out of the previous step (where **diff** was calculated), but I don't think we've seen one like this before, and the implication of **with** is that you specify a data frame to get things from first. So, from that point of view, I think **with(., t.test(...))** is much clearer.

Since the t-test on the differences is a *one*-sample test, you ought to specify a null mean (for clarity), although it still works if you don't because the default null mean is zero.

If you're a fan of the dollar sign, you might try this, on the same principle as above:

```
cats %>% mutate(diff=recipe_A-recipe_B) %>%
    t.test(.$diff, mu=0)
```

```
## Must use a vector in `[`, not an object of class matrix.
```

which doesn't work, but this does:

```
cats %>% mutate(diff=recipe_A-recipe_B) %>%
    pull(diff) %>% t.test(., mu=0)

##
##  One Sample t-test
##
## data:  .
## t = 2.6656, df = 9, p-value = 0.02581
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.1135065 1.3864935
## sample estimates:
## mean of x
##      0.75
```

In this case, it seems to be necessary to get the differences *as a vector* first, and then feed that vector into `t.test`. I'm not quite sure why the previous way doesn't work.

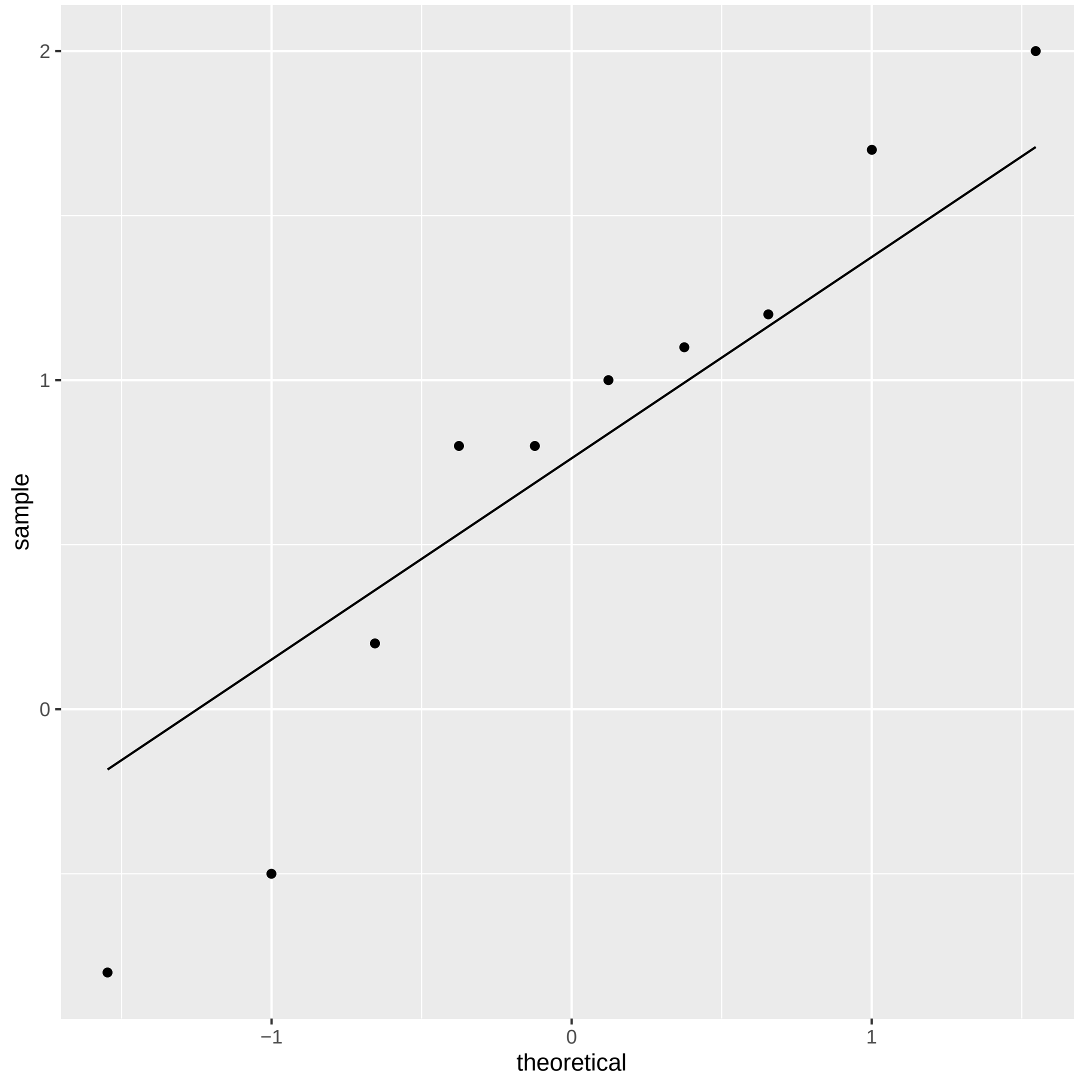
Any one of these alternatives, with a proper conclusion, is also full marks. In these cases, you have to calculate the differences first (which is what I meant by "data manipulation": there is no other tidying required). In particular, if you try some kind of `gather`, you will lose the association between the two recipes for each cat, which is what this question is all about.

The guideline for the grader is that if the right P-value comes out of what looks like a sensible t-test, then it's going to be good. (You could even take the differences the other way around, and because it's a two-sided test, the P-value will be identical: the evidence that the mean difference is not zero is still the same.)

(d) (4 marks) Make a suitable graph to assess the assumptions of your t-test. What do you conclude about the validity of your t-test? Explain briefly.

Solution: The right graph is something to assess the normality of the *differences*, which you will have to calculate first if you have not already done so. This works rather more smoothly in a pipeline, because the data frame with the differences in it can get passed straight into `ggplot`, and then you don't name a data frame at the start of `ggplot`:

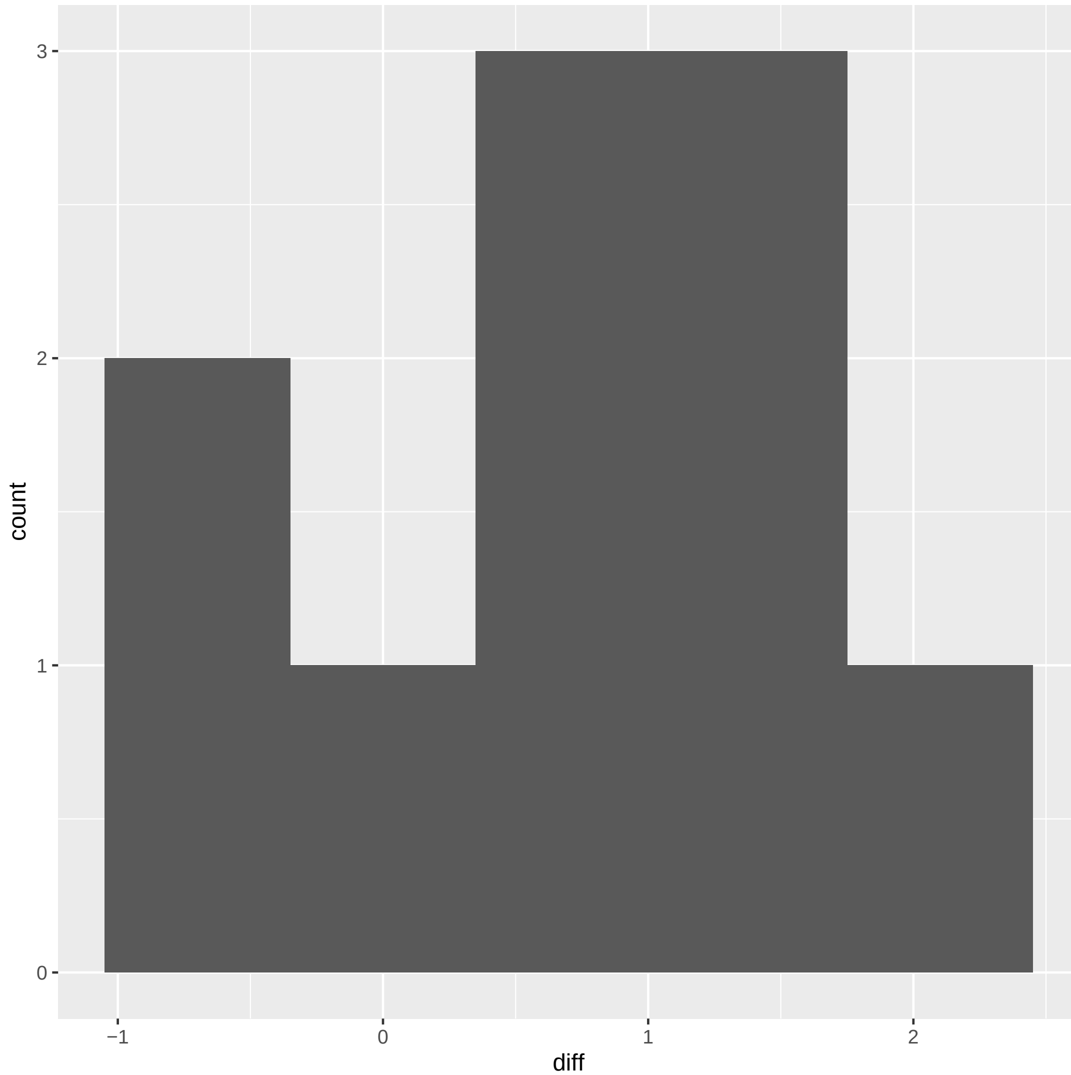
```
cats %>% mutate(diff=recipe_A-recipe_B) %>%
  ggplot(aes(sample=diff)) + stat_qq() + stat_qq_line()
```



Say something about what you see here: for example, you might see two outliers at the bottom, or you might see a long-tailed distribution (the two highest values are a bit too high as well). In either of those cases, you would say that you doubt whether the matched pairs t-test can be trusted. I think you can also reasonably say that those two values at the bottom are not too far off the line, and therefore that normality is at least tolerably good, and the matched-pairs t-test is OK. Say what you see and what that implies. Get the logic straight and I'm good.

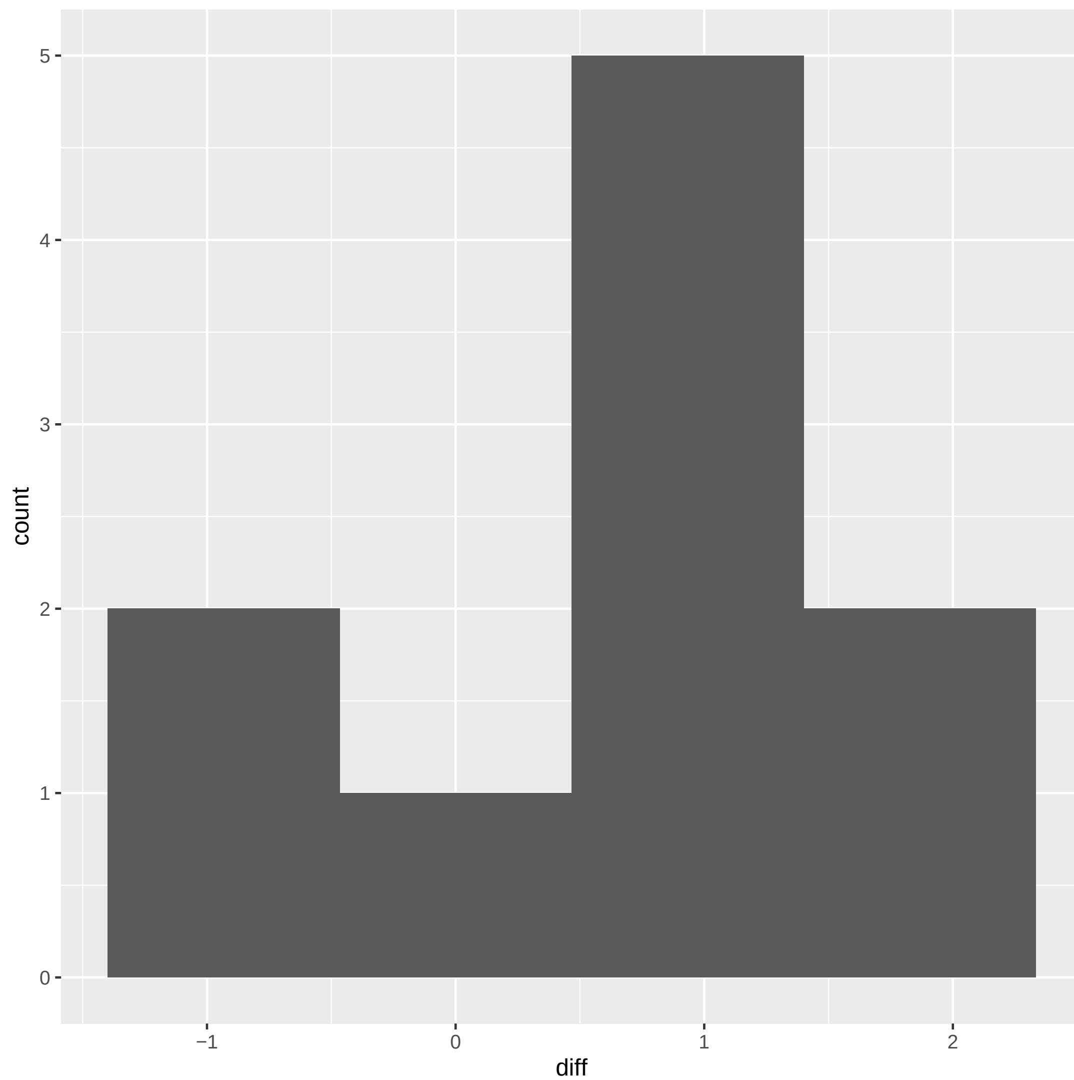
I think this is the best answer, since we are thinking specifically about normality for our assumptions, but I would also take something like a histogram of the differences. Since we don't have very much data, though, it'll be hard to say much about shape:

```
cats %>% mutate(diff=recipe_A-recipe_B) %>%
  ggplot(aes(x=diff)) + geom_histogram(bins=5)
```



You could also say that there are too many low values here (in that bin that includes -1), and come to the same conclusion as I did with the normal quantile plot. But the histogram is not very informative, and I suspected its appearance would depend crucially on the number of bins you chose. It turns out that 4 bins tells a similar story:

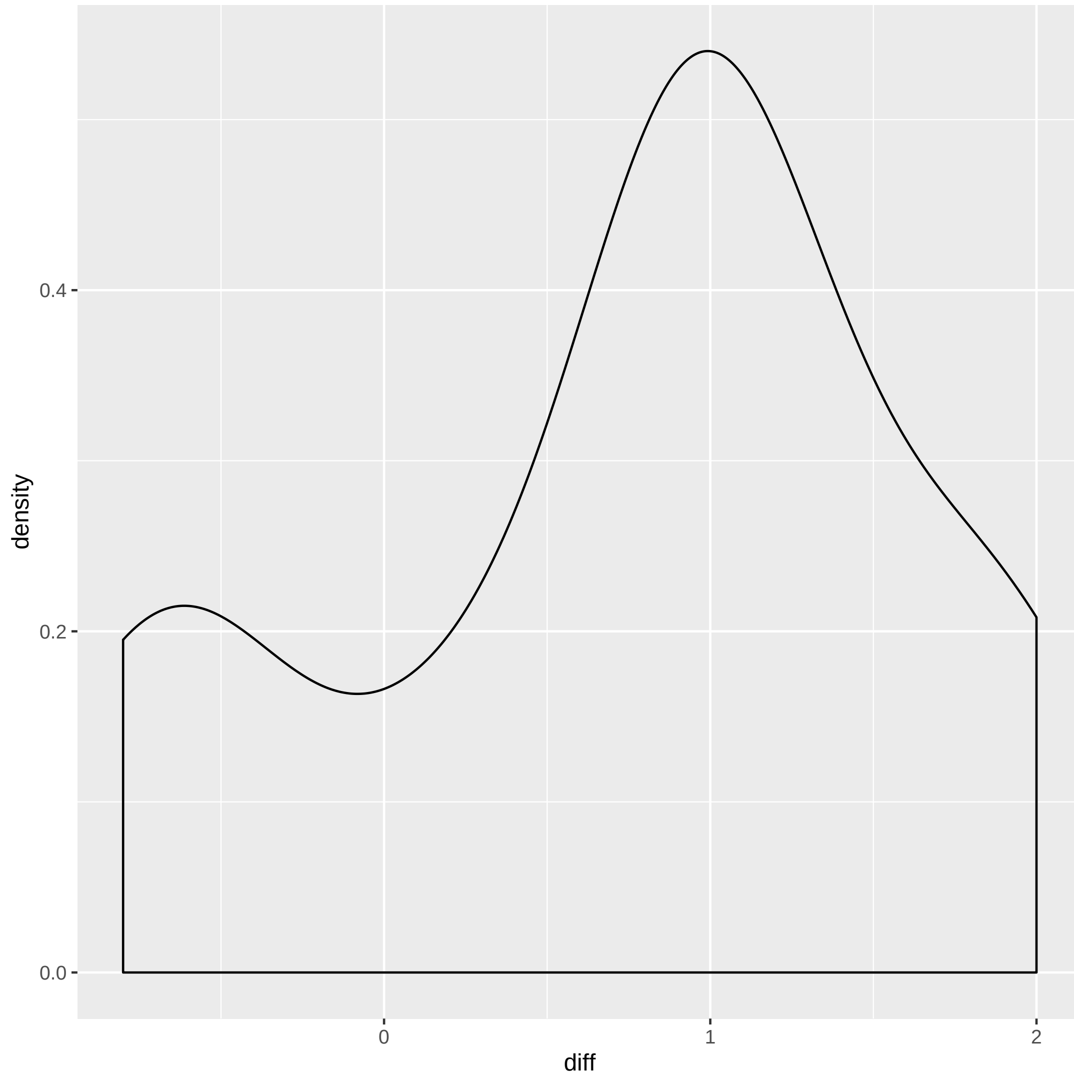
```
cats %>% mutate(diff=recipe_A-recipe_B) %>%
  ggplot(aes(x=diff)) + geom_histogram(bins=4)
```



so maybe I was wrong about that. A valid conclusion from your (reasonable) number of bins is OK. There are only 10 observations, so you really cannot justify more than about 6 bins.

Maybe a density plot is better:

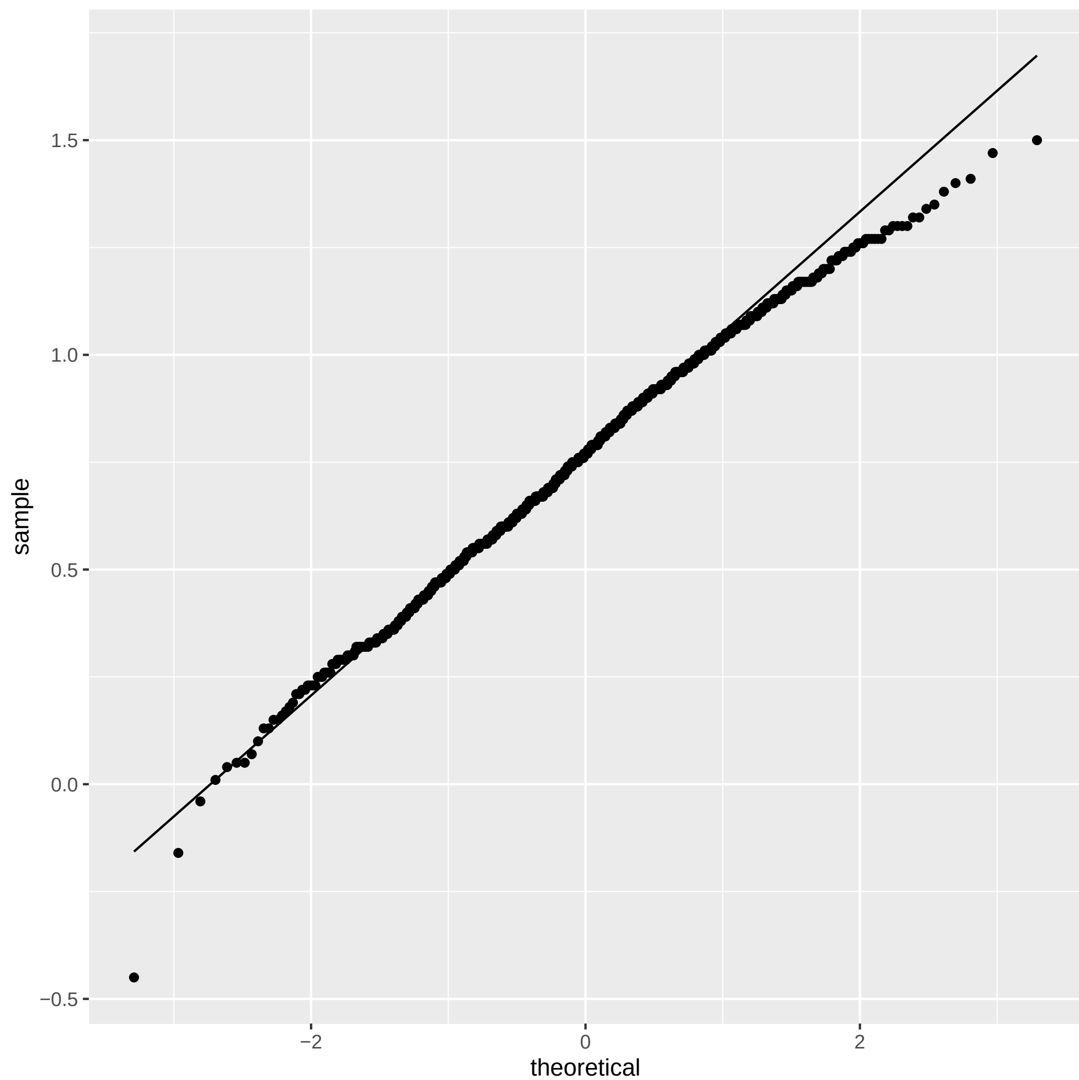
```
cats %>% mutate(diff=recipe_A-recipe_B) %>%
  ggplot(aes(x=diff)) + geom_density()
```



The extra mini-peak on the left is those two negative differences. So the inference here is that these are where any non-normality arises.

Extra: what *actually* matters is the sampling distribution of the mean difference. With 10 cats, we have a tiny bit of help from the Central Limit Theorem (not much, but some). Is this enough to make the sampling distribution of the mean approximately normal? The bootstrap is a way of assessing this. I start from my data frame `cats2` with the differences in it already:

```
rerun(1000, sample_frac(cats2, replace=T)) %>%
  map_df(~summarize(., m=mean(diff))) %>%
  ggplot(aes(sample=m)) + stat_qq() + stat_qq_line()
```



That looks pretty normal. So maybe those two apparent low outliers weren't outlying enough to cause a problem.

The code: the first line samples all 10 (default is all) rows of the data frame `cats2` with replacement (bootstrap). The second line says "for each of those resampled data frames, work out the mean difference and call it `m`". I had to use `map_df` because `summarize` returns a data frame; the output of the `map_df` is a data frame with 1000 rows and one column called `m` (try it and see). Then I make a normal quantile plot of that column `m` of mean differences.

(e) (3 marks) Use `smmr` to run a suitable sign test. How does the result compare to that of your t-test?

Solution: The suitable sign test is on the differences, that they have median zero. So, this means calculating them *again*, or using the data frame you might have saved with the differences in it:

```
cats %>% mutate(diff=recipe_A-recipe_B) %>%
  sign_test(diff, 0)

## $above_below
## below above
##     2     8
##
## $p_values
##   alternative   p_value
## 1       lower 0.9892578
## 2       upper 0.0546875
## 3   two-sided 0.1093750
```

Doing it this way, I don't need the initial data frame on `sign_test`, since I wrote the function to be `tidyverse`-friendly. Or this also works, using my data frame with differences in it:

```
sign_test(cats2, diff, 0)

## $above_below
## below above
##     2     8
##
## $p_values
##   alternative   p_value
## 1       lower 0.9892578
## 2       upper 0.0546875
## 3   two-sided 0.1093750
```

The two-sided P-value, 0.109, is not less than 0.05, so we *do not* reject a median difference of zero. This is different from the matched-pairs t-test, where we *did* reject a mean of zero.

Extra: the reason for the difference in results goes back to my earlier comment about how they chose to record the data. What the sign test is actually doing is to count how many cats preferred recipe A (8 of them) and how many preferred B (2 of them), by looking at the two scores for each cat and seeing which one is higher. The sign test *throws away* how big the difference is, and having an 8–2 split of cats that preferred the two recipes is not *quite* unbalanced enough to reject with. (With a larger sample size, this kind of split certainly *would* be unbalanced enough.)

So the final conclusion you draw will depend on what you thought of your normal quantile plot (or histogram): if you thought the differences were normal enough, you would stick with your t-test and declare a difference between the two recipes; if not, you go with the sign test on the differences and declare that you couldn't find a difference between the recipes.

I'm thinking the final final conclusion ought to be "use more than 10 cats", but the data is what we have.[1]

4. In the Yukon Territory, is more forested area being destroyed by wildfires than in the past? The data are in `http://www.utsc.utoronto.ca/~butler/assgt_data/Yukon_Wildfires.csv` as a `.csv` file. The data file contains five columns: the year, the number of wildfires caused by lightning strikes, the number caused by humans, the total of the last two, and the total number of hectares of forest destroyed by wildfires in that year.

   (a) (2 marks) Read in and display (some of) the data. Note that the variable names have Capital Letters.

Solution: The usual thing:

```
my_url="http://www.utsc.utoronto.ca/~butler/assgt_data/Yukon_Wildfires.csv"
fires=read_csv(my_url)
```

```
## Parsed with column specification:
## cols(
##   Year = col_double(),
##   Lightning = col_double(),
##   Human = col_double(),
##   Total = col_double(),
##   Hectares = col_double()
## )
```

```
fires
```

```
## # A tibble: 55 x 5
##     Year Lightning Human Total Hectares
##    <dbl>     <dbl> <dbl> <dbl>    <dbl>
##  1  1950         9    28    37   177024
##  2  1951        18    53    71   339151
##  3  1952         8    17    25    39805
##  4  1953        16    39    55   175800
##  5  1954        13    44    57    49340
##  6  1955        21    54    75    49926
##  7  1956         4    51    55     1490
##  8  1957        41    47    88    77945
##  9  1958        38    62   100   889032
## 10  1959        27    33    60    39498
## # ... with 45 more rows
```

(b) (2 marks) One way of comparing any of these variables in the past to the same variable more recently is to divide the time period into two parts. For example, we can compare up to (and including) 1980 with 1981 to the present (the data set goes from 1950 to 2004). In your data frame, make a new column called `recent` that is `TRUE` if the year is 1981 or greater and `FALSE` otherwise. Save the resulting data frame, and display at least some of it. (Hint: set your new variable equal to the appropriate logical condition, or, if you must, use `ifelse`.)

Solution: This is a `mutate`. Since we are quite happy to have the values of the new variable be `TRUE` and `FALSE`, we just have to define the new variable equal to a logical condition. There is no need to use `ifelse`:

```
fires %>% mutate(recent=(Year>=1981)) -> fires
fires
```

```
## # A tibble: 55 x 6
##     Year Lightning Human Total Hectares recent
##    <dbl>     <dbl> <dbl> <dbl>    <dbl> <lgl>
##  1  1950         9    28    37   177024 FALSE
##  2  1951        18    53    71   339151 FALSE
##  3  1952         8    17    25    39805 FALSE
##  4  1953        16    39    55   175800 FALSE
##  5  1954        13    44    57    49340 FALSE
##  6  1955        21    54    75    49926 FALSE
##  7  1956         4    51    55     1490 FALSE
##  8  1957        41    47    88    77945 FALSE
##  9  1958        38    62   100   889032 FALSE
## 10  1959        27    33    60    39498 FALSE
## # ... with 45 more rows
```

I assigned back into the original data frame. (You can create a new data frame if you like.)

If you want to use `ifelse`, it goes like this:

```
fires %>% mutate(recent2=ifelse(Year>=1981, TRUE, FALSE))
```

```
## # A tibble: 55 x 7
##     Year Lightning Human Total Hectares recent recent2
##    <dbl>     <dbl> <dbl> <dbl>    <dbl> <lgl>  <lgl>
##  1  1950         9    28    37   177024 FALSE  FALSE
##  2  1951        18    53    71   339151 FALSE  FALSE
##  3  1952         8    17    25    39805 FALSE  FALSE
##  4  1953        16    39    55   175800 FALSE  FALSE
##  5  1954        13    44    57    49340 FALSE  FALSE
##  6  1955        21    54    75    49926 FALSE  FALSE
##  7  1956         4    51    55     1490 FALSE  FALSE
##  8  1957        41    47    88    77945 FALSE  FALSE
##  9  1958        38    62   100   889032 FALSE  FALSE
## 10  1959        27    33    60    39498 FALSE  FALSE
## # ... with 45 more rows
```

The warning sign is this: if you find yourself using `TRUE` and `FALSE` inside an `ifelse`, that's a sign that you don't really need the `ifelse` at all. R is perfectly happy setting a new variable equal to something that could be true or false.

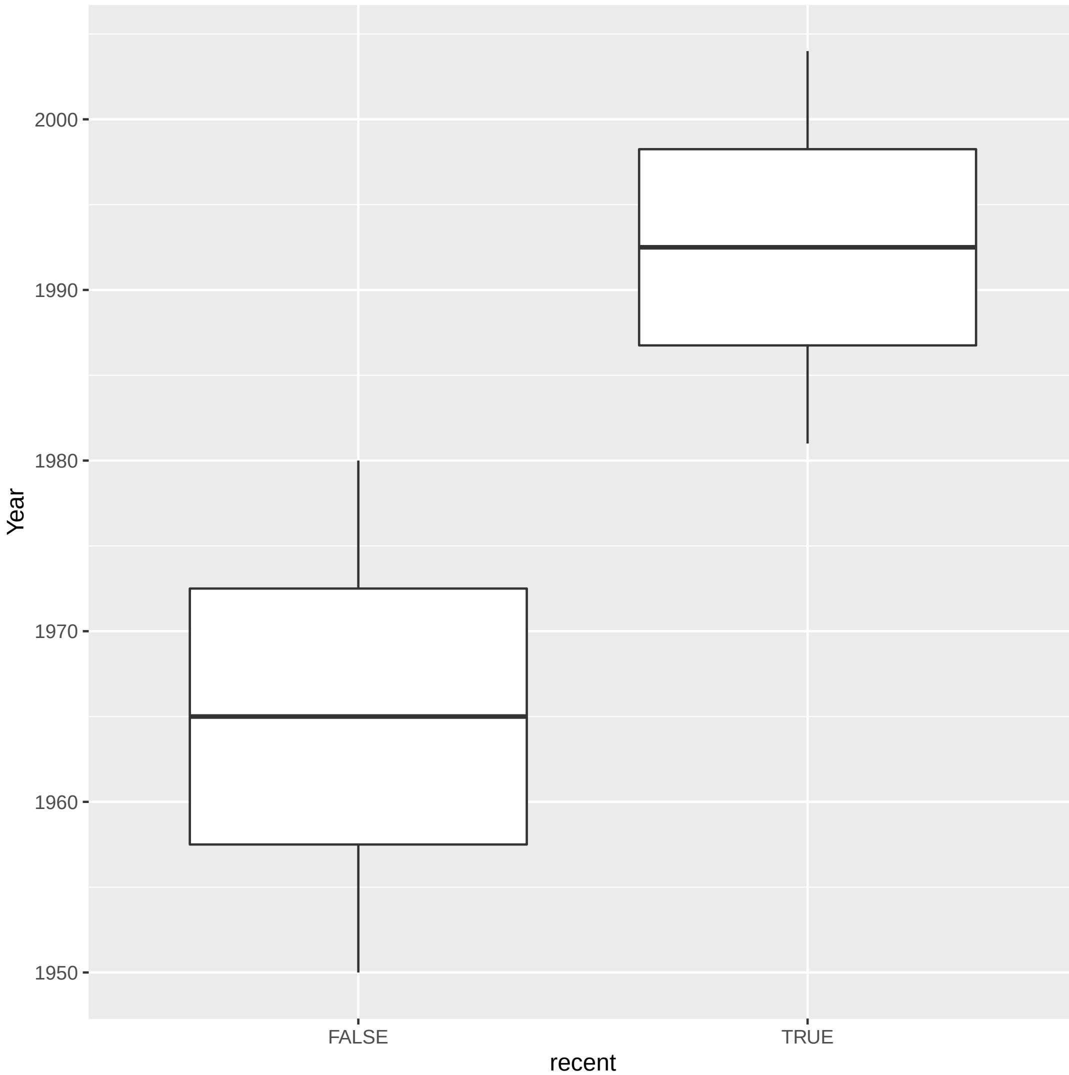
Extra: the above is a not very convincing demonstration of the association between `Year` and `recent`. Other ways to do it include passing the data frame into `View`, which will create a new tab with a display of the whole data frame (which you can then check). Or you can use `sample_n`, which takes a random sample of rows, here 15 of them:

```
fires %>% sample_n(15)
```

```
## # A tibble: 15 x 6
##     Year Lightning Human Total Hectares recent
##    <dbl>     <dbl> <dbl> <dbl>    <dbl> <lgl>
##  1  1958        38    62   100   889032 FALSE
##  2  1961         9    40    49    44037 FALSE
##  3  2004       249    33   282  1714875 TRUE
##  4  1974        47    46    93     3465 FALSE
##  5  1984        88    80   168    23229 TRUE
##  6  1980        57    93   150   154205 FALSE
##  7  1967        49    48    97   123975 FALSE
##  8  1956         4    51    55     1490 FALSE
##  9  1998        87   110   197   343672 TRUE
## 10  1975       104    62   166    31555 FALSE
## 11  1994       185    70   255   421710 TRUE
## 12  1990        73    81   154     7389 TRUE
## 13  1992        68    48   116    37815 TRUE
## 14  1963        16    27    43    17506 FALSE
## 15  1971        72    67   139   301730 FALSE
```

The early years all have `recent=FALSE` and the later years all have `recent=TRUE`. This is a bit more convincing. Or you could even make a graph: year is quantitative and `recent` is categorical:

```
fires %>% ggplot(aes(x=recent, y=Year)) + geom_boxplot()
```

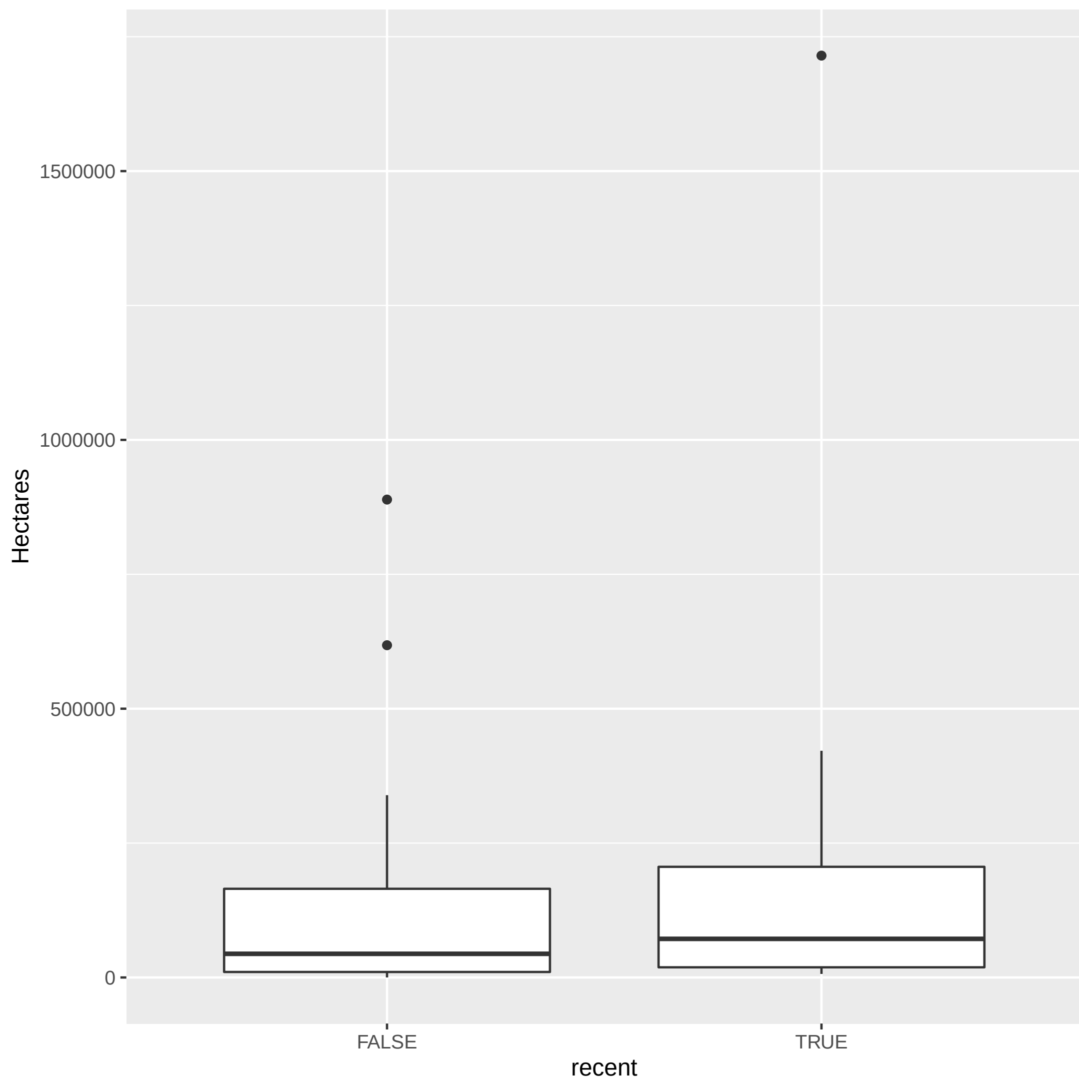


All the years that really are recent are classified as recent. This is more convincing still.

(c) (2 marks) Use your data frame with `recent` in it to make a suitable plot of the `Hectares` values, one that could be used to assess the assumptions for a two-sample t-test.

Solution: This means the plot has to include both `Hectares` and `recent`. The obvious thing is a boxplot:

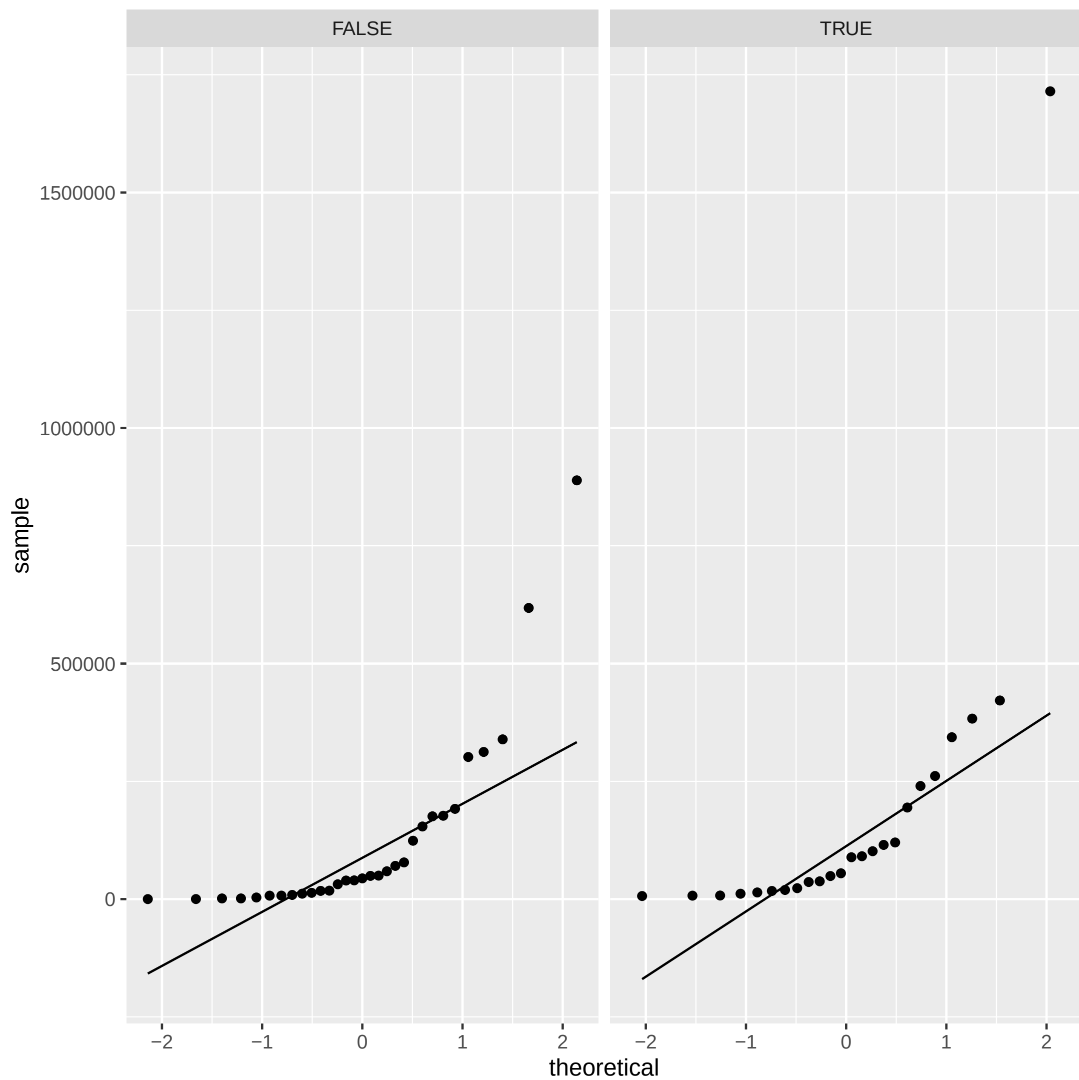
```
ggplot(fires, aes(x=recent, y=Hectares)) + geom_boxplot()
```



I didn't ask for comment, but you might say that there are too many big outliers to trust normality, and also that the medians don't look very different. Or you could note that the outliers are at the same (top) end as the long tails, and thus both distributions are right-skewed. Either is a good conclusion from the boxplots.

If you want to take the point of view that we are really assessing normality within the two groups of `Hectares` values defined by `recent` (which is also a sensible choice), facetted normal quantile plots are called for, thus:

```
ggplot(fires, aes(sample=Hectares)) + stat_qq() + stat_qq_line() +
  facet_wrap(~recent)
```



This takes a bit of care to get everything in the right place. Again, no comment required, but it is clear from the curves that both distributions are skewed to the right. You might say that the right-hand plot has an outlier in addition; maybe in the left-hand plot the two highest values are consistent with the right-skewness, that is, part of the curve rather than too high compared to where the curve goes, as opposed to being outliers. Up to you.[2]

(d) (3 marks) Use something from `smmr` to compare the median hectares destroyed in the two time periods. What precisely do you conclude, in the context of the data? Explain briefly.

Solution:

This is Mood's median test — data frame, quantitative column, categorical column (that makes the two groups):

```
library(smmr)
median_test(fires, Hectares, recent)
```

```
## $table
##        above
## group   above below
##   FALSE    14    16
##   TRUE     13    11
##
## $test
##        what     value
## 1 statistic 0.3000000
## 2        df 1.0000000
## 3   P-value 0.5838824
```

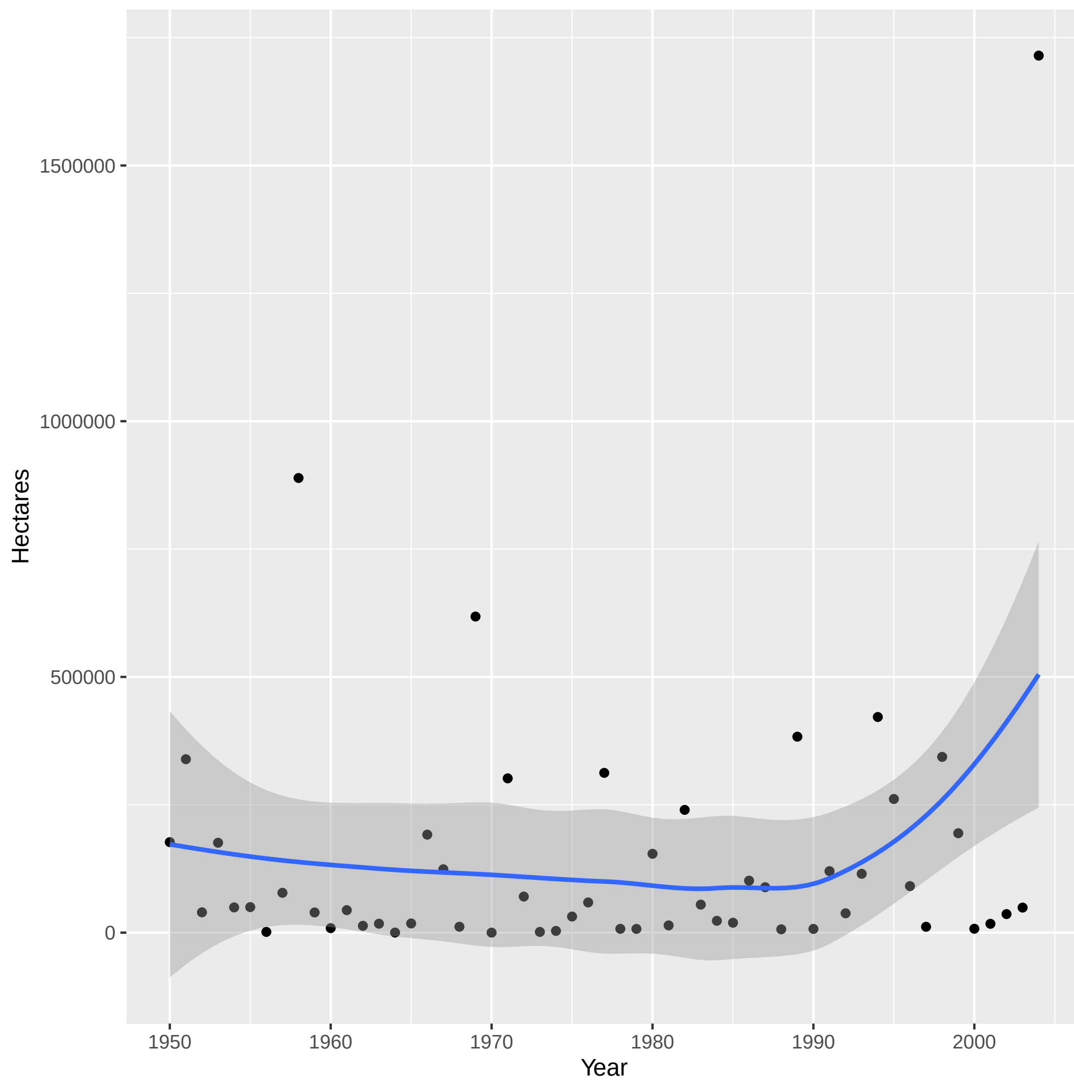
The P-value is 0.584, much greater than 0.05, so there is no evidence at all of a difference in median hectares destroyed between the recent years and the earlier ones. (This is close enough to a 50–50 split above and below in each group, in other words; it would have to be a lot more unbalanced to be significant.)

Extra (1): `median_test` always gives you a two-sided test. Notice how I phrased my conclusion as "no evidence of a difference", a two-sided conclusion. If you want to make it one-sided, with an increase in median, you have some more work to do. We saw above that the direction of change in the median is, if anything, an increase towards the more recent years; this says that we are "on the right side", so we can justifiably halve the P-value, to 0.292, to get a one-sided test. But this is still not significant, so there is no evidence of an *increase* in median either.

Extra (2): there is something arbitrary about splitting between 1980 and 1981. It would be better to use the actual years and test for a time trend. To do that, we should first look at a time plot:

```
ggplot(fires, aes(x=Year, y=Hectares)) +  geom_point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The smooth trend actually shows a more or less level trend up to about 1990, with a sharp increase afterwards. There is also a lot of variability. How much of the apparent increase is due to the very high value in 2004 is another question. In fact, there are a lot of years with a low amount of damage, even recently, and occasional years with much higher damage.

The approved test among environmental science people is called Mann-Kendall. This is based on a non-parametric correlation called the Kendall correlation; the other variable is time. Being non-parametric, it is not affected by outliers, of which natural data tends to have a lot. I wrote a package `mkac` to run it:

```
library(mkac)
kendall_Z_adjusted(fires$Hectares)
```

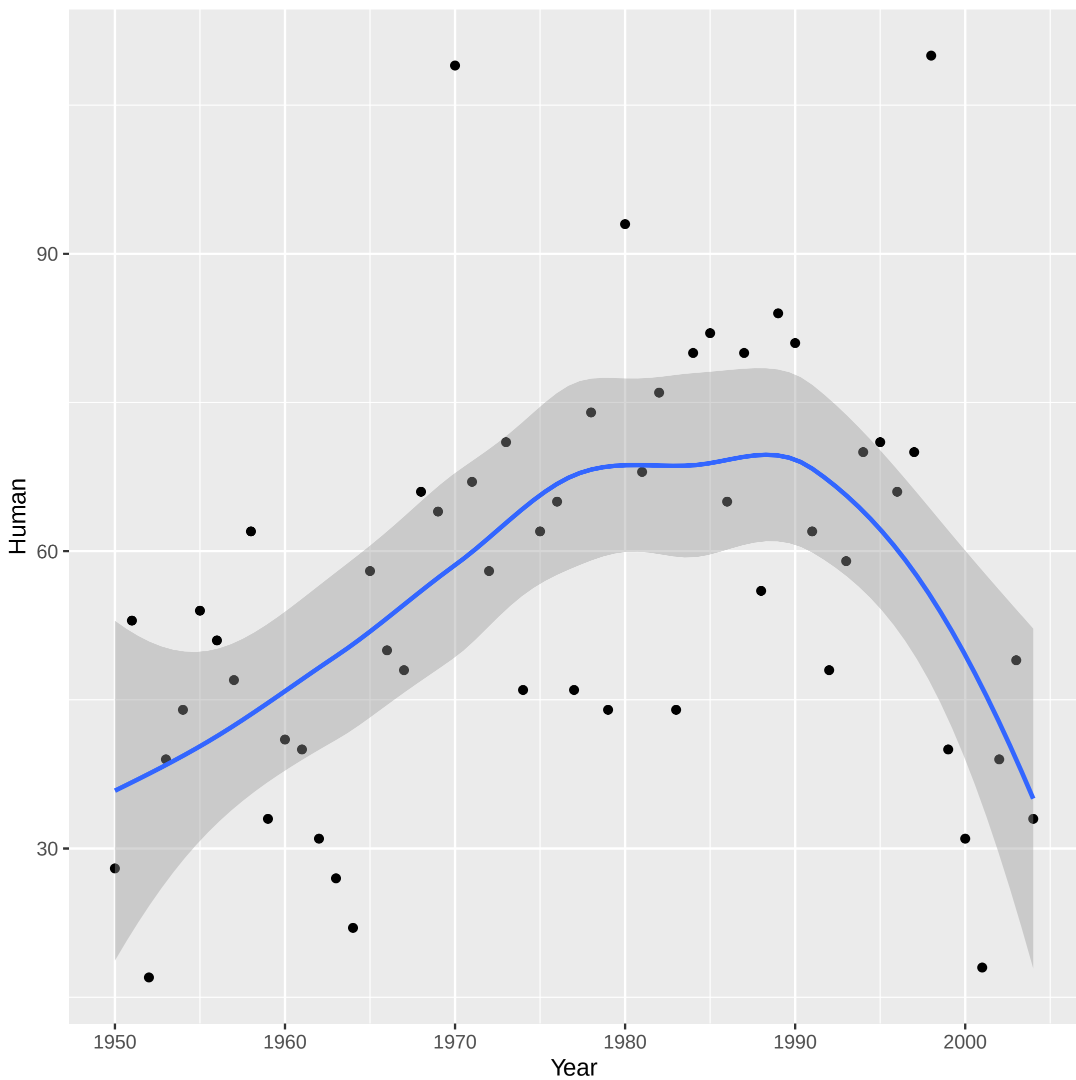
```
## $z
## [1] 0.5952823
##
## $z_star
## [1] 0.5952823
##
## $ratio
## [1] 1
##
## $P_value
## [1] 0.5516548
##
## $P_value_adj
## [1] 0.5516548
```

The P-value to look at is the last one, 0.552. This is once again not significant; there is no time trend. I would expect this method to find a time trend if there really was one, so I think we can be comfortable that any apparent time trend is no more than chance.

Extra (4) (yes, I know): one of the other variables is number of wildfires caused by humans per year. What kind of time trend does that have?

```
ggplot(fires, aes(x=Year, y=Human)) +  geom_point() + geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



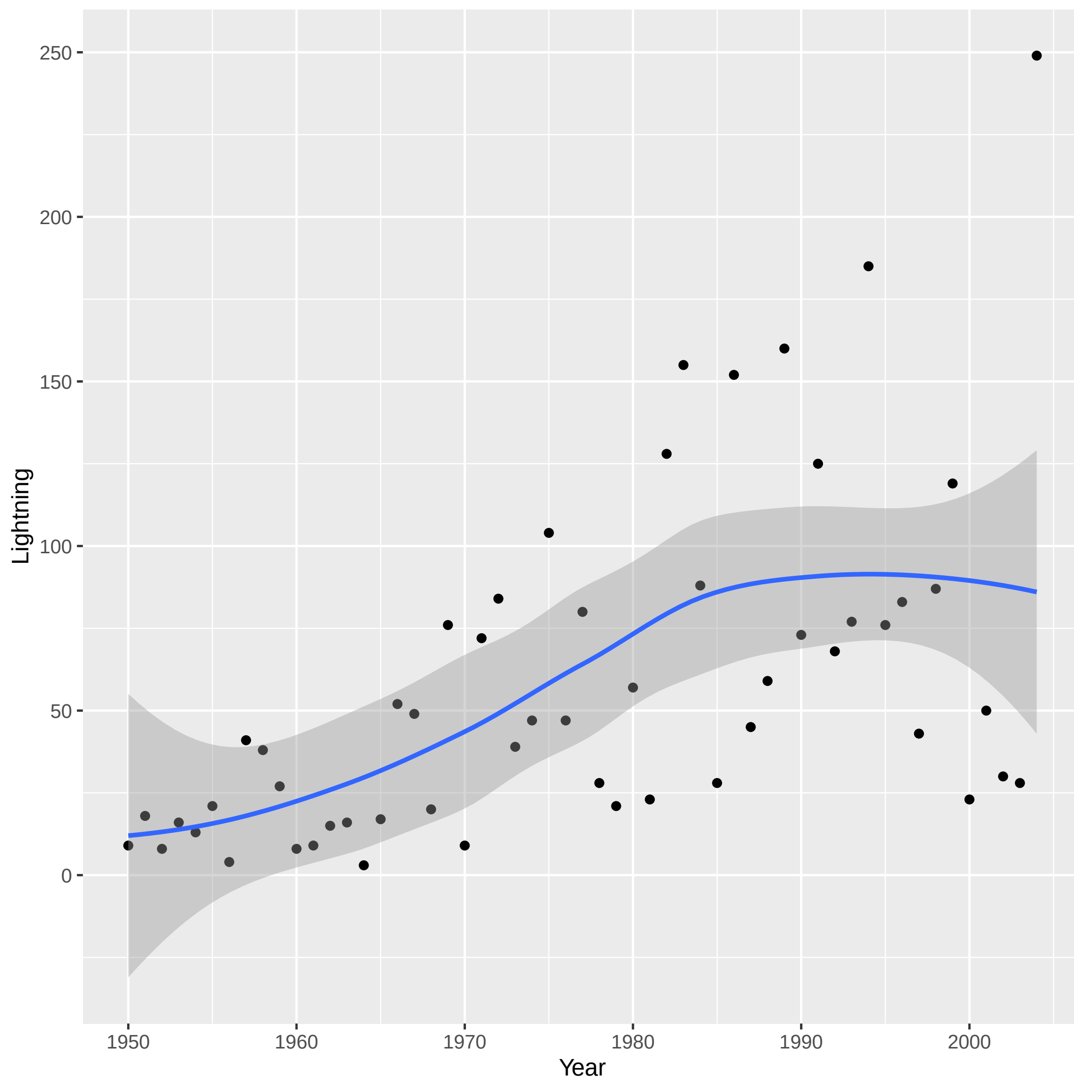
Up, and then down again. Mann-Kendall looks for a "monotone trend": one that keeps going up or keeps going down, so it may not find anything here either:

```
kendall_Z_adjusted(fires$Human)

## $z
## [1] 2.301274
##
## $z_star
## [1] 1.814531
##
## $ratio
## [1] 1.608451
##
## $P_value
## [1] 0.02137614
##
## $P_value_adj
## [1] 0.06959596
```

This is a very nearly significant upward trend (because the z is positive). If you looked only at this, you might conclude that the trend was still going up, whereas if you look at the graph, you see that something else is happening in the most recent years. (This shows the value of looking at graphs as well as inferential statistics.)

So there was an increasing trend of hectares damaged since about 1990 (apparently). Is that something to do with lightning, then?

```
ggplot(fires, aes(x=Year, y=Lightning)) +  geom_point() + geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Number of wildfires caused by lightning shows an increasing trend, but not really since 1990. So it's not that. The conclusion I draw is that the number of wildfires is not really going up (since 1990), but the hectares destroyed is, so the *size* of the fires must be increasing.

This is the end of what you need to hand in, but I originally had a couple of other things I was going to ask you to do (before realizing that the assignment was too long with them in). I've added those back below. You do **not** need to hand these in, but working through them might give you some more insight:

(e) Find the median of all the `Hectares` values.

Solution:

Just `summarize`:

```
fires %>% summarize(med=median(Hectares))

## # A tibble: 1 x 1
##     med
##   <dbl>
## 1 49340
```

(f) Count (using `count`) the number of `Hectares` values above (and below) the overall median for each of your two time periods (`recent` being `TRUE` or `FALSE`). Hints: (i) in `count` you can have either a column or a logical condition based on a column being greater than some value, or both; (ii) you can just type the number you obtained in the previous part.

Solution: There are two things to count: whether a year is recent or not, and whether the value of `Hectares` is greater than that median you calculated in the previous part. Hence:

```
fires %>% count(recent, Hectares>49340)

## # A tibble: 4 x 3
##   recent `Hectares > 49340`     n
##   <lgl>  <lgl>              <int>
## 1 FALSE  FALSE                 17
## 2 FALSE  TRUE                  14
## 3 TRUE   FALSE                 11
## 4 TRUE   TRUE                  13
```

Extra: I was trying to direct you towards this rather than the apparently-equivalent:

```
fires %>% count(recent, Hectares<49340)

## # A tibble: 4 x 3
##   recent `Hectares < 49340`     n
##   <lgl>  <lgl>              <int>
## 1 FALSE  FALSE                 15
## 2 FALSE  TRUE                  16
## 3 TRUE   FALSE                 13
## 4 TRUE   TRUE                  11
```

which comes out slightly differently. The reason for the difference is that with 55 years, the median will be one of the data values, so the value equal to 49340 will flip from being not-above 49340 in the first case to being not-below in the second.

`smmr` does something slightly different. The astute amongst you will note that `smmr` above produced yet another table of aboves and belows, different again from the ones I got with `count`. What `smmr` does, when it gets a value exactly equal to the overall median, is to count it as neither above nor below (that is, to just throw it away). You could guess that it did something like this by looking at the table of counts above and below in the `smmr` output: the four frequencies add up to 54, but there are 55 observations.

I didn't want you to have to get into all that, so I mention it as an extra now.

(g) Does it look as if more recent years have a greater number of hectares of forest destroyed, on average? Explain briefly. (Of course, you know the answer now, but pretend for the moment you don't. I originally had this part in before you did the test and got the P-value.)

Solution:

Looking at my first `count` in (d), before 1981 (where `recent` is FALSE), there was a slight majority of years in which the number of hectares destroyed was less than the overall median. After 1980, there was a slight majority of years in which the number of hectares destroyed was *greater* than the overall median. So there is an increase, but a very small one. You ought to read this to say that both time periods had close to a 50–50 split of years above and below the median, so there's no real difference. (By now, of course, you know how the test came out, so you know that the "actual" answer is that the counts were basically balanced both before and after 1980, and differed from 50–50 only by chance, not by anything meaningful.)

Notes

[1] I am now singing "if 8 out of 10 cats all prefer Whiskas, do the other two prefer Lesley Judd?" This is a lyric from the British band Half Man Half Biscuit, and as is typical for them, there are several cultural references that you won't get unless you grew up in Britain in the 1980s, as I did. Explanation: Whiskas is a cat food that had a commercial stating that 8 out of 10 cats preferred it; Lesley Judd was one of the hosts of a children's TV show called Blue Peter (still running, though with many changes of host since then), where the presenters had pets that appeared on the show, and Lesley Judd did indeed have a cat, whose name I forget.

[2] My take is that skewness is a property of the whole distribution, while outliers are unusual compared to that distribution, whatever it is.