# Alternative tests

# When there isn't sufficient normality

- When your samples are not normal enough, cannot use $t$ procedures
- Sometimes transforming the data (eg taking logs of all the values) will help
- Or, use test with no assumptions about normality:
  - for one sample, use *sign test* for median
  - for two samples, use *Mood's median test*
  - for matched pairs, use *sign test* on differences.

# One-sample: the IRS data

- The IRS ("Internal Revenue Service") is the US authority that deals with taxes (like Revenue Canada).
- One of their forms is supposed to take no more than 160 minutes to complete. A citizen's organization claims that it takes people longer than that on average.
- Sample of 30 people; time to complete form recorded.
- Read in data, and do $t$-test of $H_0 : \mu = 160$ vs. $H_a : \mu > 160$.
- Only one column, so pretend it is delimited by something.

# Packages

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

- installation instructions for `smmr` later

# Read in data

```
my_url <- "http://ritsokiguess.site/datafiles/irs.txt"
irs <- read_csv(my_url)
irs
```

```
# A tibble: 30 x 1
    Time
   <dbl>
 1    91
 2    64
 3   243
 4   167
 5   123
 6    65
 7    71
 8   204
 9   110
10   178
# i 20 more rows
```

# Test whether mean is 160 or greater

```
with(irs, t.test(Time, mu = 160,
                 alternative = "greater"))
```

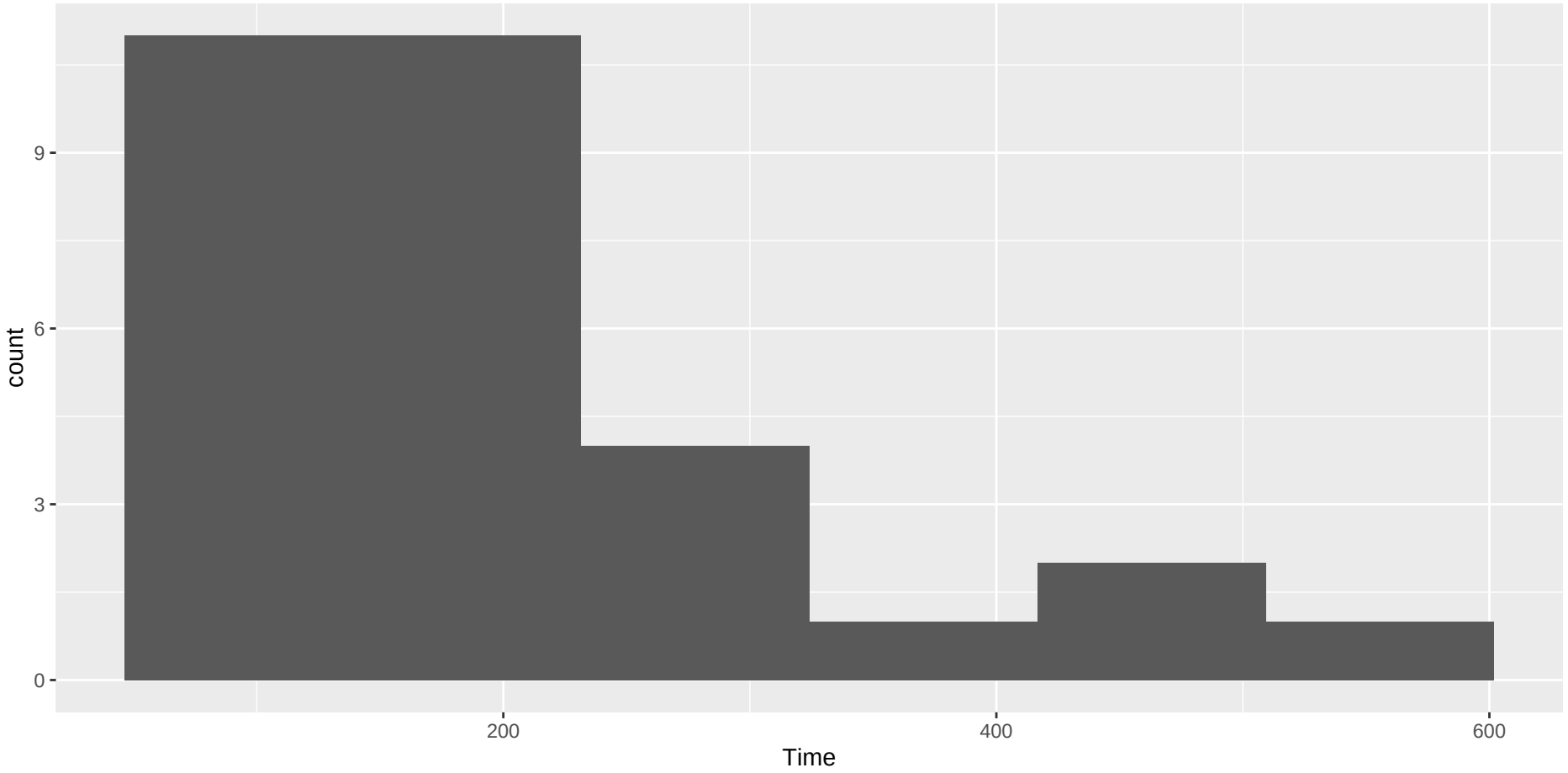
```
    One Sample t-test

data:  Time
t = 1.8244, df = 29, p-value = 0.03921
alternative hypothesis: true mean is greater than 160
95 percent confidence interval:
 162.8305      Inf
sample estimates:
mean of x
 201.2333
```

Reject null; mean (for all people to complete form) greater than 160.

# But, look at a graph

```
ggplot(irs, aes(x = Time)) + geom_histogram(bins = 6)
```



# Comments

- Skewed to right.
- Should look at *median*, not mean.

# The sign test

- But how to test whether the median is greater than 160?
- Idea: if the median really is 160 ($H_0$ true), the sampled values from the population are equally likely to be above or below 160.
- If the population median is greater than 160, there will be a lot of sample values greater than 160, not so many less. Idea: test statistic is number of sample values greater than hypothesized median.

# Getting a P-value for sign test 1/3

- How to decide whether "unusually many" sample values are greater than 160? Need a sampling distribution.
- If $H_0$ true, pop. median is 160, then each sample value independently equally likely to be above or below 160.
- So number of observed values above 160 has binomial distribution with $n = 30$ (number of data values) and $p = 0.5$ (160 is hypothesized to be *median*).

# Getting P-value for sign test 2/3

- Count values above/below 160:

```
irs %>% count(Time > 160)
```

```
# A tibble: 2 x 2
  `Time > 160`     n
  <lgl>        <int>
1 FALSE           13
2 TRUE            17
```

- 17 above, 13 below. How unusual is that? Need a *binomial table*.

# Getting P-value for sign test 3/3

- R function `dbinom` gives the probability of eg. exactly 17 successes in a binomial with $n = 30$ and $p = 0.5$:

```
dbinom(17, 30, 0.5)
```

```
[1] 0.1115351
```

- but we want probability of 17 *or more*, so get all of those, find probability of each, and add them up:

```
tibble(x=17:30) %>%
  mutate(prob=dbinom(x, 30, 0.5)) %>%
  summarize(total=sum(prob))
```

```
# A tibble: 1 x 1
  total
  <dbl>
1 0.292
```

## ... or

use cumulative distribution

```
pbinom(17, 30, 0.5) # prob of <= 17
```

```
[1] 0.8192027
```

and hence (note first input):

```
pbinom(16, 30, 0.5, lower.tail = FALSE)
```

```
[1] 0.2923324
```

This last is $P(X \geq 17) = P(X > 16)$.

# Using my package smmr

- I wrote a package smmr to do the sign test (and some other things). Installation is non-standard:

```
install.packages("smmr", repos = "nxskok.r-universe.dev")
```

- Then load it:

```
library(smmr)
```

# smmr for sign test

- smmr's function sign_test needs three inputs: a data frame, a column and a null median:

```
sign_test(irs, Time, 160)
```

```
$above_below
below above
   13    17

$p_values
  alternative   p_value
1       lower 0.8192027
2       upper 0.2923324
3   two-sided 0.5846647
```

# Comments (1/4)

- Testing whether population median *greater than* 160, so want *upper-tail* P-value 0.2923. Same as before.
- Also get table of values above and below; this too as we got.

# Comments (2/4)

- P-values are:

| Test | P-value |
|---|---|
| $t$ | 0.0392 |
| Sign | 0.2923 |

- These are very different: we reject a mean of 160 (in favour of the mean being bigger), but clearly *fail* to reject a median of 160 in favour of a bigger one.

# Comments (3/4)

- Why is that? Obtain mean and median:

```
irs %>% summarize(mean_time = mean(Time),
                  median_time = median(Time))
```

```
# A tibble: 1 x 2
  mean_time median_time
      <dbl>       <dbl>
1      201.        172.
```

# Comments (4/4)

- The mean is pulled a long way up by the right skew, and is a fair bit bigger than 160.
- The median is quite close to 160.
- We ought to be trusting the sign test and not the t-test here (median and not mean), and therefore there is no evidence that the "typical" time to complete the form is longer than 160 minutes.
- Having said that, there are clearly some people who take a lot longer than 160 minutes to complete the form, and the IRS could focus on simplifying its form for these people.
- In this example, looking at any kind of average is not really helpful; a better question might be "do an unacceptably large fraction of people take longer than (say) 300 minutes to complete the form?": that is, thinking about worst-case rather than average-case.

# CI for median 1/2

- The sign test does not naturally come with a confidence interval for the median.
- So we use the "duality" between test and confidence interval to say: the (95%) confidence interval for the median contains exactly those values of the null median that would not be rejected by the two-sided sign test (at $\alpha = 0.05$).

# CI for median 2/2

- Precisely: Let $C = 100(1 - \alpha)$, so C% gives corresponding CI to level-$\alpha$ test. Then following always true. (Symbol $\iff$ means "if and only if".)

| Test decision | | Confidence interval |
|---|---|---|
| Reject $H_0$ at level $\alpha$ | $\iff$ | $C\%$ CI does not contain $H_0$ value |
| Do not reject $H_0$ at level $\alpha$ | $\iff$ | $C\%$ CI contains $H_0$ value |

- Idea:
  - "Plausible" parameter value inside CI, not rejected;
  - "Implausible" parameter value outside CI, rejected.

# The value of this

- If you have a test procedure but no corresponding CI:
- you make a CI by including all the parameter values that would not be rejected by your test.
- Use:
  - $\alpha = 0.01$ for a 99% CI,
  - $\alpha = 0.05$ for a 95% CI,
  - $\alpha = 0.10$ for a 90% CI, and so on.

# For our data

- The procedure is to try some values for the null median and see which ones are inside and which outside our CI.
- smmr has pval_sign that gets just the 2-sided P-value:

```
pval_sign(160, irs, Time)
```

```
[1] 0.5846647
```

- Try a couple of null medians:

```
pval_sign(200, irs, Time)
```

```
[1] 0.3615946
```

```
pval_sign(300, irs, Time)
```

```
[1] 0.001430906
```

- So 200 inside the 95% CI and 300 outside.

# Doing a whole bunch

```
(d <- tibble(null_median=seq(100,300,20)))
```

```
# A tibble: 11 x 1
   null_median
         <dbl>
 1         100
 2         120
 3         140
 4         160
 5         180
 6         200
 7         220
 8         240
 9         260
10         280
11         300
```

# ... and then

"for each null median, run the function `pval_sign` for that null median and get the P-value":

```
d %>% rowwise() %>%
  mutate(p_value = pval_sign(null_median, irs, Time))
```

# Results

```
# A tibble: 11 x 2
# Rowwise:
   null_median  p_value
         <dbl>    <dbl>
 1         100 0.000325
 2         120 0.0987
 3         140 0.200
 4         160 0.585
 5         180 0.856
 6         200 0.362
 7         220 0.0428
 8         240 0.0161
 9         260 0.00522
10         280 0.00143
11         300 0.00143
```

# Make it easier for ourselves

```
d %>% rowwise() %>%
  mutate(p_value = pval_sign(null_median, irs, Time)) %>%
  mutate(in_out = ifelse(p_value > 0.05, "inside", "outside"))
```

```
# A tibble: 11 x 3
# Rowwise:
   null_median  p_value in_out
         <dbl>    <dbl> <chr>
 1         100 0.000325 outside
 2         120 0.0987   inside
 3         140 0.200    inside
 4         160 0.585    inside
 5         180 0.856    inside
 6         200 0.362    inside
 7         220 0.0428   outside
 8         240 0.0161   outside
 9         260 0.00522  outside
10         280 0.00143  outside
11         300 0.00143  outside
```

# Confidence interval for median?

- 95% CI to this accuracy from 120 to 200.
- Can get it more accurately by looking more closely in intervals from 100 to 120, and from 200 to 220.

# A more efficient way: bisection

- Know that top end of CI between 200 and 220:

```
lo <- 200
hi <- 220
```

- Try the value halfway between: is it inside or outside?

```
try <- (lo + hi) / 2
try
```

```
[1] 210
```

```
pval_sign(try,irs,Time)
```

```
[1] 0.09873715
```

- Inside, so upper end is between 210 and 220. Repeat (over):

# ... bisection continued

```
lo <- try
try <- (lo + hi) / 2
try
```

```
[1] 215
```

```
pval_sign(try, irs, Time)
```

```
[1] 0.06142835
```

- 215 is inside too, so upper end between 215 and 220.
- Continue until have as accurate a result as you want.

# Bisection automatically

- A loop, but not a `for` since we don't know how many times we're going around. Keep going `while` a condition is true:

```
lo = 200
hi = 220
while (hi - lo > 1) { # replace 1 by desired accuracy
  try = (hi + lo) / 2
  ptry = pval_sign(try, irs, Time)
  print(c(try, ptry))
  if (ptry <= 0.05)
    hi = try
  else
    lo = try
}
```

# The output from this loop

```
[1] 210.00000000   0.09873715
[1] 215.00000000   0.06142835
[1] 217.50000000   0.04277395
[1] 216.25000000   0.04277395
[1] 215.62500000   0.04277395
```

- 215 inside, 215.625 outside. Upper end of interval to this accuracy is 215.

# Using smmr

- `smmr` has function `ci_median` that does this (by default 95% CI):

```
ci_median(irs, Time)
```

```
[1] 119.0065 214.9955
```

- Uses a more accurate bisection than we did.
- Or get, say, 90% CI for median:

```
ci_median(irs, Time, conf.level=0.90)
```

```
[1] 123.0031 208.9960
```

- 90% CI is shorter, as it should be.

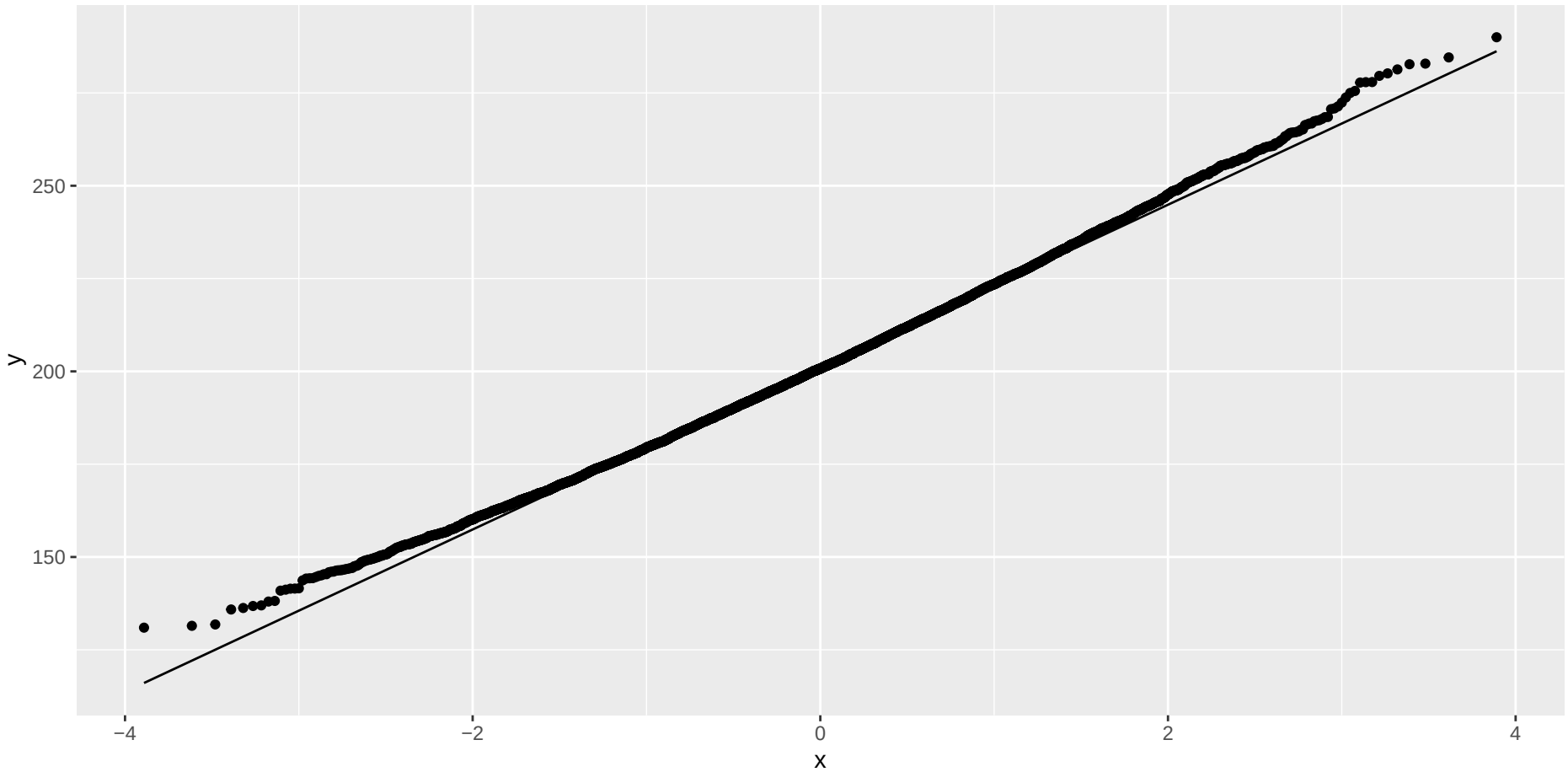
# Bootstrap

- but, was the sample size (30) big enough to overcome the skewness?
- Bootstrap, again:

```
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(irs$Time, replace = TRUE))) %
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(sample = my_mean)) +
    stat_qq() + stat_qq_line() -> g
```

# The sampling distribution

g



# Comments

- A little skewed to right, but not nearly as much as I was expecting.
- The $t$-test for the mean might actually be OK for these data, *if the mean is what you want*.
- In actual data, mean and median very different; we chose to make inference about the median.
- Thus for us it was right to use the sign test.

# Two samples: Mood's median test

- If normality fails (for one or both of the groups), what do we do then?
- Again, can compare medians: use the thought process of the sign test, which does not depend on normality and is not damaged by outliers.
- A suitable test called Mood's median test.
- Before we get to that, a diversion.

# The chi-squared test for independence

Suppose we want to know whether people are in favour of having daylight savings time all year round. We ask 20 males and 20 females whether they each agree with having DST all year round ("yes") or not ("no").

# Some of the data:

```
my_url <- "http://ritsokiguess.site/datafiles/dst.txt"
dst <- read_delim(my_url," ")
dst %>% slice_sample(n = 10)
```

```
# A tibble: 10 x 2
   gender agree
   <chr>  <chr>
 1 male   yes
 2 male   yes
 3 female yes
 4 male   yes
 5 male   yes
 6 male   yes
 7 male   yes
 8 male   no
 9 male   yes
10 male   yes
```

# ... continued

Count up individuals in each category combination, and arrange in contingency table:

```
tab <- with(dst, table(gender, agree))
tab
```

```
        agree
gender   no yes
  female 11   9
  male    3  17
```

- Most of the males say "yes", but the females are about evenly split.
- Looks like males more likely to say "yes", ie. an association between gender and agreement.

# ... continued

- Test an $H_0$ of "no association" ("independence") vs. alternative that there is really some association.
- Done with `chisq.test`.

# ...And finally

```
chisq.test(tab, correct=FALSE)
```

```
    Pearson's Chi-squared test

data:  tab
X-squared = 7.033, df = 1, p-value = 0.008002
```

- Reject null hypothesis of no association (P-value 0.008)
- therefore there is a difference in rates of agreement between (all) males and females (or that gender and agreement are associated).
- Same answers as by hand. (Omitting `correct = FALSE` uses "Yates correction".

# Mood's median test

- Earlier: compare medians of two groups.
- Sign test: count number of values above and below something (there, hypothesized median).
- Mood's median test:
  - Find "grand median" of all the data, regardless of group
  - Count data values in each group above/below grand median.
  - Make contingency table of group vs. above/below.
  - Test for association.

# Why it works

- If group medians equal, each group should have about half its observations above/below grand median. If not, one group will be mostly above grand median and other below.

# Mood's median test for reading data

- Find overall median score:

```
kids %>% summarize(med=median(score)) %>% pull(med) -> m
m
```

```
[1] 47
```

- Make table of above/below vs. group:

```
tab <- with(kids, table(group, score > m))
tab
```

```
group FALSE TRUE
    c    15    8
    t     7   14
```

- Treatment group scores mostly above median, control group scores mostly below, as expected.

# The test

- Do chi-squared test:

```
chisq.test(tab, correct=FALSE)
```

```
    Pearson's Chi-squared test

data:  tab
X-squared = 4.4638, df = 1, p-value = 0.03462
```

- Two-sided (tests for any association).
- Here, is reading method *better*? (one-sided).
- Most of treatment children above overall median, so halve P-value to get 0.017.
- Again, children learn to read better using new method.

# Or by smmr

- `median_test` does the whole thing:

```
median_test(kids,score,group)
```

```
$grand_median
[1] 47

$table
     above
group above below
    c     8    15
    t    14     7

$test
       what      value
1 statistic 4.46376812
2        df 1.00000000
3   P-value 0.03462105
```

- P-value again two-sided.

# Comments 1/2

- P-value 0.013 for (1-sided) t-test, 0.017 for (1-sided) Mood median test.
- Like the sign test, Mood's median test doesn't use the data very efficiently (only, is each value above or below grand median).
- Thus, if we can justify doing *t*-test, we should do it. This is the case here.

# Comments 2/2

- The *t*-test will usually give smaller P-value because it uses the data more efficiently.
- The time to use Mood's median test is if we are definitely unhappy with the normality assumption (and thus the t-test P-value is not to be trusted).
- There is no obvious way to get a confidence interval for the difference between the two medians.

# Matched pairs: the pain relief data

Values aligned in columns:

```
my_url <-
  "http://ritsokiguess.site/datafiles/analgesic.txt"
pain <- read_table(my_url)
pain %>% mutate(diff = druga - drugb) -> pain
glimpse(pain)
```

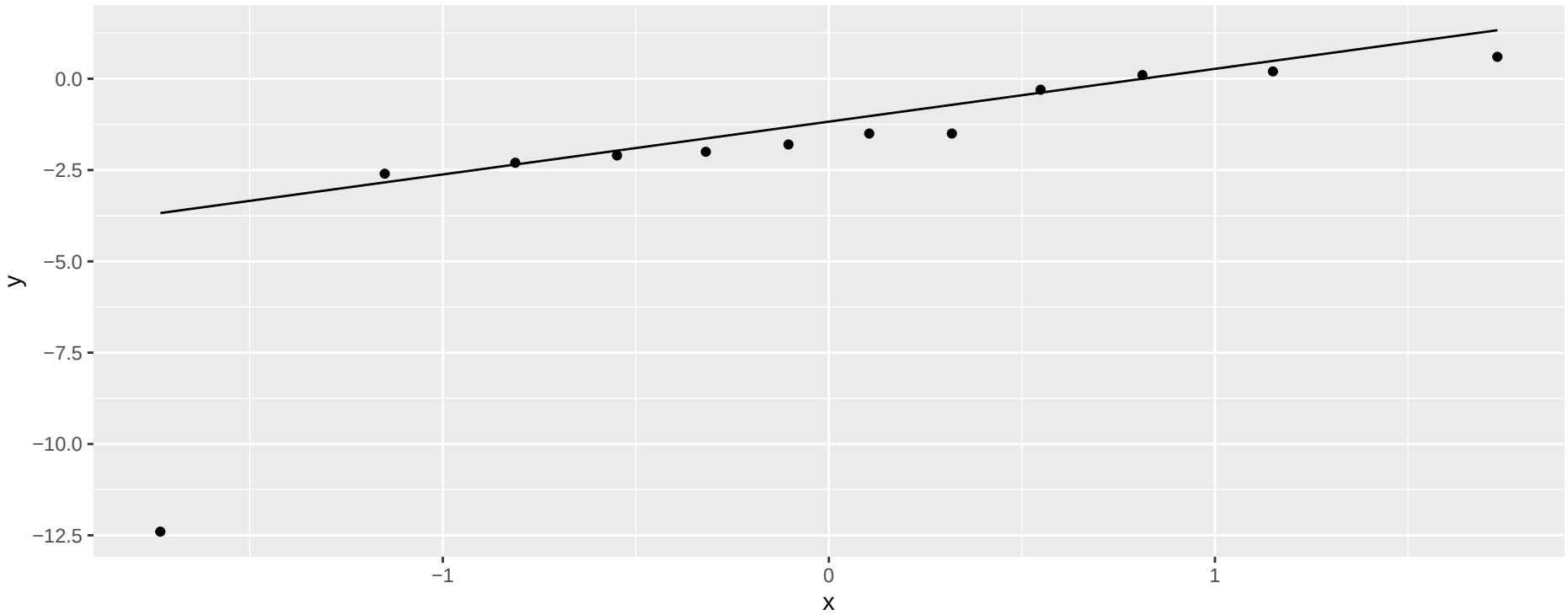
```
Rows: 12
Columns: 4
$ subject <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
$ druga   <dbl> 2.0, 3.6, 2.6, 2.6, 7.3, 3.4, 14.9, 6.6, 2.3,
$ drugb   <dbl> 3.5, 5.7, 2.9, 2.4, 9.9, 3.3, 16.7, 6.0, 3.8,
$ diff    <dbl> -1.5, -2.1, -0.3, 0.2, -2.6, 0.1, -1.8, 0.6, -
```

# Assessing normality

- Matched pairs analyses assume (theoretically) that differences normally distributed.
- How to assess normality? A normal quantile plot.

# The normal quantile plot (of differences)

```
ggplot(pain, aes(sample = diff)) + stat_qq() + stat_qq_line()
```




- Points should follow the straight line. Bottom left one way off, so normality questionable here: outlier.

# What to do instead?

- Matched pairs $t$-test based on one sample of differences
- the differences not normal (enough)
- so do *sign test* on differences, null median 0:

# ... continued

```
sign_test(pain, diff, 0)
```

```
$above_below
below above
    9     3

$p_values
  alternative    p_value
1       lower 0.07299805
2       upper 0.98071289
3   two-sided 0.14599609
```

- No evidence of any difference between the drugs (that the median difference is not zero).

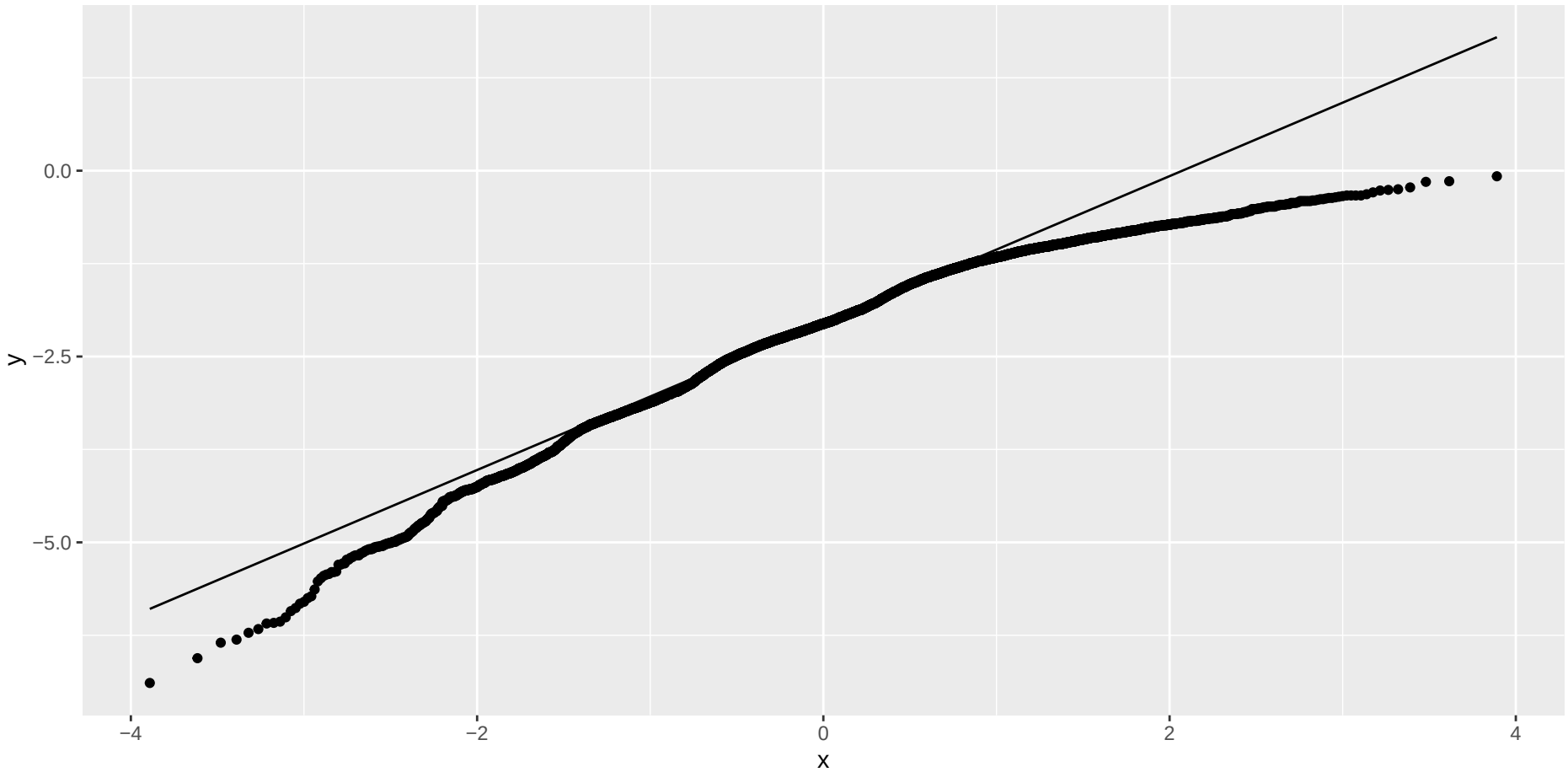
# Did we really need to worry about that outlier?

Bootstrap sampling distribution of sample mean differences:

```
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(pain$diff, replace = TRUE))) %>%
  mutate(my_mean = mean(my_sample)) %>%
  ggplot(aes(sample = my_mean)) + stat_qq() + stat_qq_line() +
```

# The normal quantile plot

g




Yes we did need to worry; this is clearly skewed left and not normal.