Statistical tests

Day 3 - Introduction to Data Analysis with R

Selina Baldauf
Freie Universität Berlin - Theoretical Ecology

March 14, 2025

General approach

Scientific idea

working hypothesis **H1** & null hypothesis **H0**

collect data

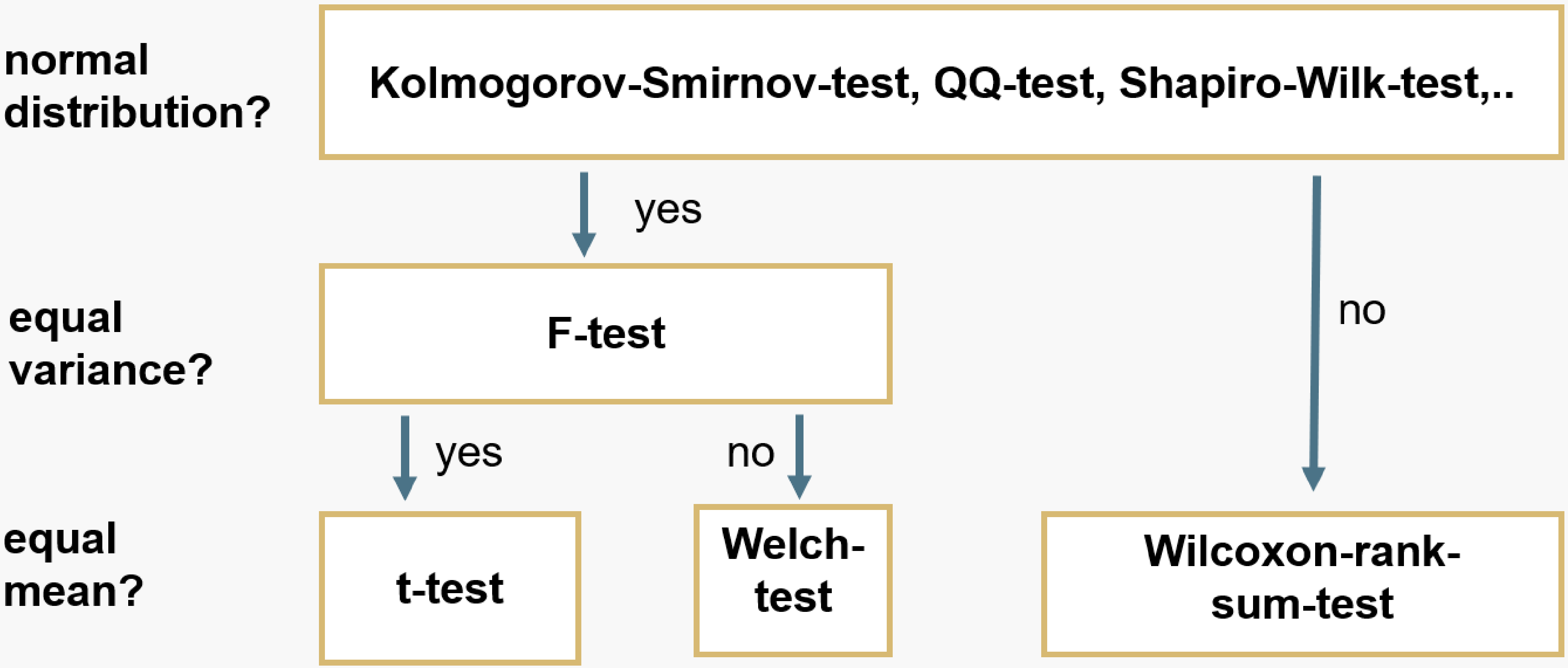
Choose **level of significance α** (often 0.05) to test H0

p: how likely it would be to observe a **deviation from H0** this extreme (or more) **if the H0 were true**?

unlikely (**p ≤ α**):
reject H0
Otherwise (**p>α**):
not reject H0

errors possible

Overview of tests



Tests for normal distribution

Test for normal distribution

There are **various tests** and the outcome might differ!

Shapiro-Wilk-Test

- How much does variance of observed data differ from normal distribution
- Specific test only for normal distribution
- High power, also for few data points

Visual tests: QQ-Plot

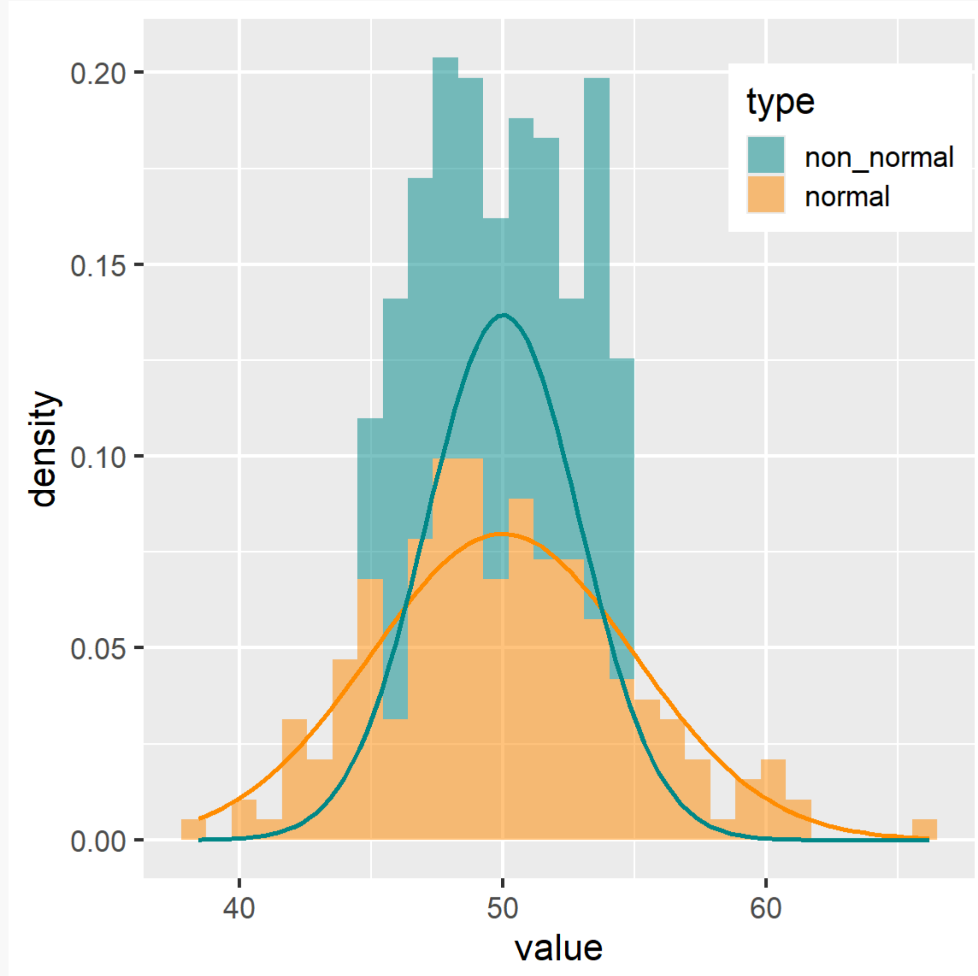
- Quantiles of observed data plotted against quantiles of normal distribution
- Scientist has to decide if normal or not

The data

A tibble with two variables: `normal` and `non_normal`

▶ Expand to reproduce the data

```
mydata
#> # A tibble: 200 × 2
#>    normal non_normal
#>     <dbl>      <dbl>
#>  1   47.2       54.9
#>  2   48.8       46.4
#>  3   57.8       54.1
#>  4   50.4       50.8
#>  5   50.6       49.0
#>  6   58.6       49.5
#>  7   52.3       52.1
#>  8   43.7       45.8
#>  9   46.6       48.4
#> 10   47.8       51.8
#> # ℹ 190 more rows
```



Shapiro-Wilk-Test

H_0: Data does not differ from a normal distribution

```
shapiro.test(mydata$normal)
#>
#>  Shapiro-Wilk normality test
#>
#> data:  mydata$normal
#> W = 0.99076, p-value = 0.2298
```

- W: test statistic
- p: probability to observe data with this level of deviation from normality (or more) if H_0 was true
  - probability is high -> no reason to doubt normality assumption

The data does **not deviate significantly** from a normal distribution (Shapiro-Wilk-Test, W = 0.991, p = 0.23).

```
shapiro.test(mydata$non_normal)
#>
#>  Shapiro-Wilk normality test
#>
#> data:  mydata$non_normal
#> W = 0.95114, p-value = 2.435e-06
```

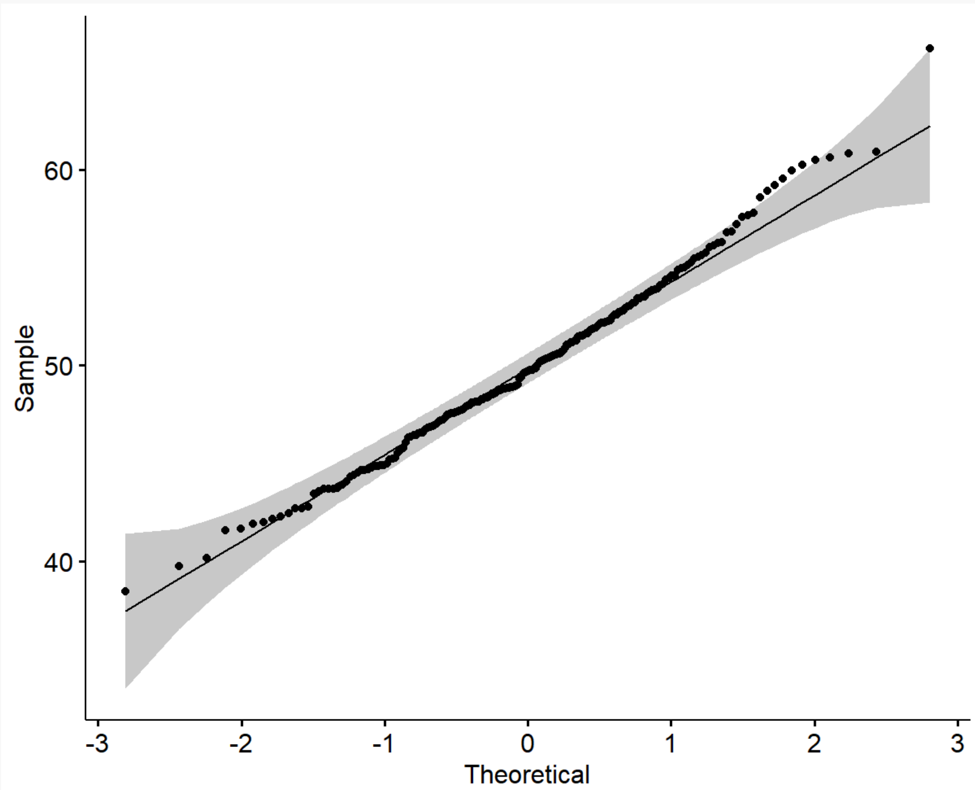
The data **deviates significantly** from a normal distribution (Shapiro-Wilk-Test, W = 0.95, p < 0.001).

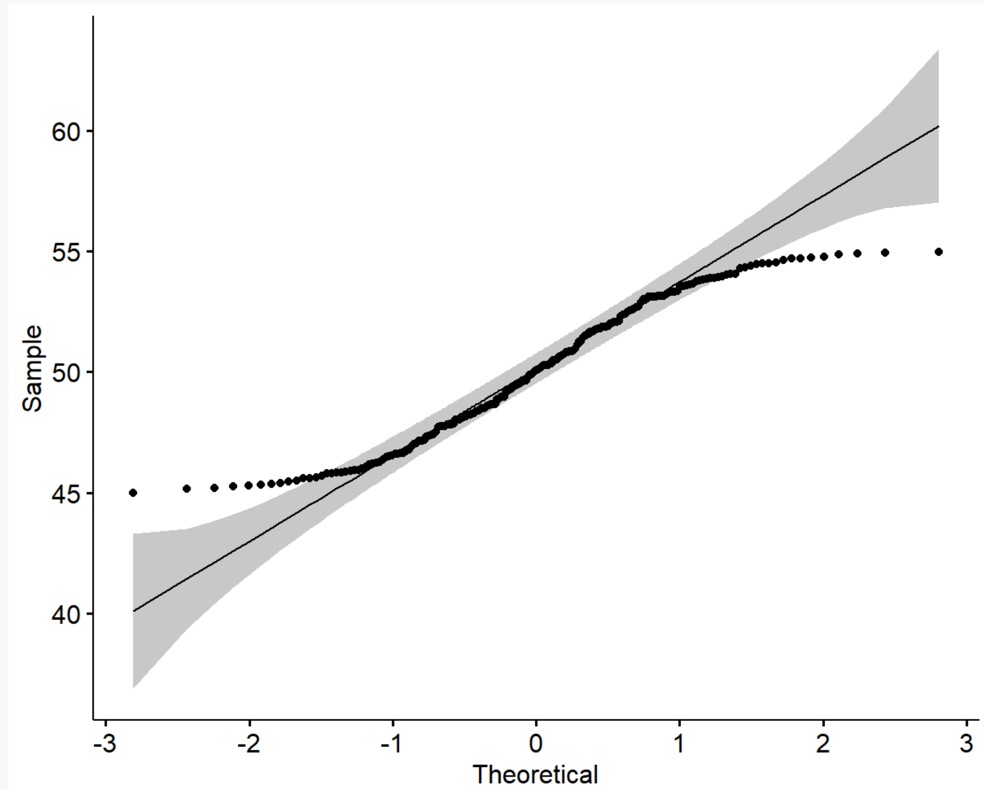
Visual test with QQ-Plot

Points should match the straight line. Small deviations are okay.

```
# ggplot(
#   mydata,
#   aes(sample = normal)
# ) +
#   stat_qq() +
#   stat_qq_line()
ggpubr::ggqqplot(mydata$normal)
```

```
# ggplot(
#   mydata,
#   aes(sample = non_normal)
# ) +
#   stat_qq() +
#   stat_qq_line()
ggpubr::ggqqplot(mydata$non_normal)
```





Tests for equal variance

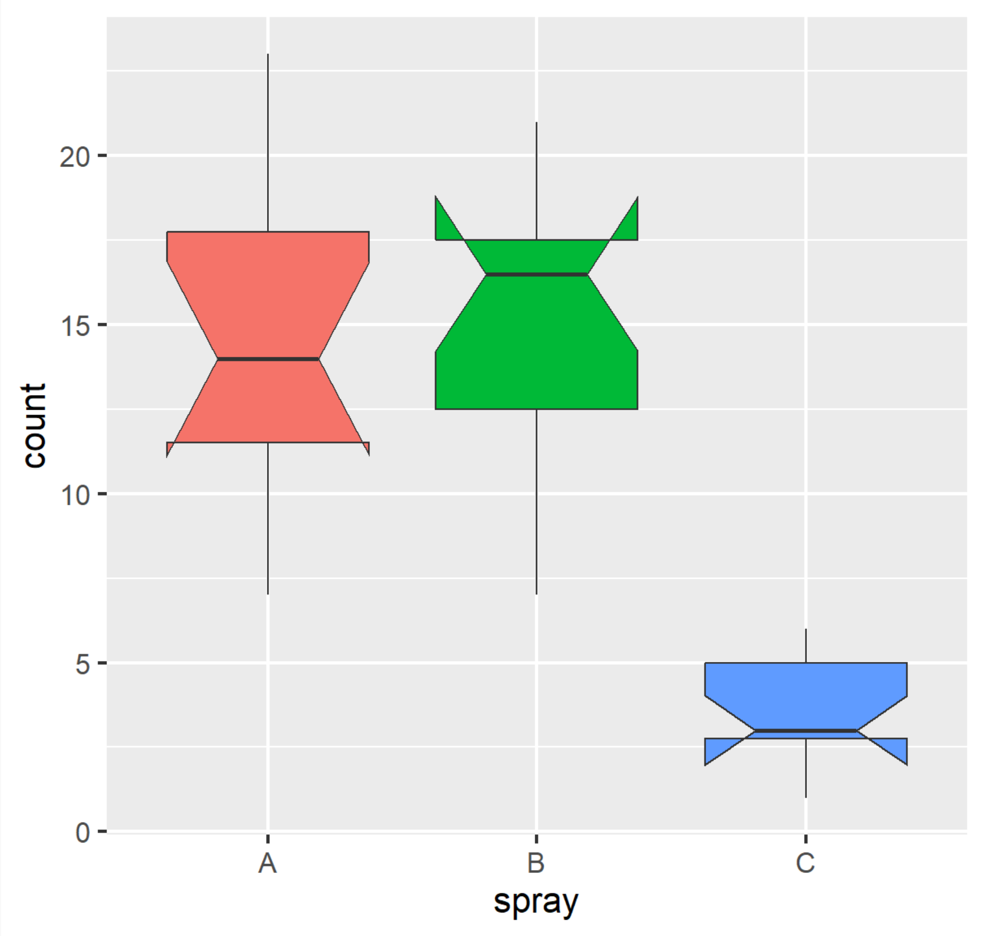
The data

Counts of insects in agricultural units treated with different insecticides.

Compare treatments A, B and C:

Create subsets before: count variable for each treatment as a vector

```
TreatA <- filter(
  InsectSprays,
  spray == "A"
)$count
TreatB <- filter(
  InsectSprays,
  spray == "B"
)$count
TreatC <- filter(
  InsectSprays,
  spray == "C"
)$count
```



Test for equal variance

First, test for normal distribution!

F-Test

- **Normal distribution** of groups
- Calculates ratio of variances (if equal, ratio = 1)
- p: How likely is this (or a more extreme) ratio of variances if the variances were truly equal?

Levene test

- **Non-normal distribution** of groups
- Compare difference between data sets with difference within data sets

Test for equal variance

First, test for normal distribution

```
shapiro.test(TreatA)
#> 
#>  Shapiro-Wilk normality test
#> 
#> data:  TreatA
#> W = 0.95757, p-value = 0.7487
shapiro.test(TreatB)
#> 
#>  Shapiro-Wilk normality test
#> 
#> data:  TreatB
#> W = 0.95031, p-value = 0.6415
shapiro.test(TreatC)
#> 
#>  Shapiro-Wilk normality test
#> 
#> data:  TreatC
#> W = 0.92128, p-value = 0.2967
```

Result: All 3 treatments are normally distributed.

F-Test

H_0: Variances do not differ between groups

```
var.test(TreatA, TreatB)
#>
#>  F test to compare two variances
#>
#> data:  TreatA and TreatB
#> F = 1.2209, num df = 11, denom df = 11, p-value = 0.7464
#> alternative hypothesis: true ratio of variances is not equal to 1
#> 95 percent confidence interval:
#>  0.3514784 4.2411442
#> sample estimates:
#> ratio of variances
#>            1.22093
```

- F: test statistics, ratio of variances (if F = 1, variances are equal)
- df: degrees of freedom of both groups
- p-value: probability to observe this (or more extreme) F if H_0 was true

Variances of sprays A & B **don't differ significantly** (F-Test, $F_{11,11}$ = 1.22, p = 0.75)

F-Test

H_0: Variances do not differ between groups

```
var.test(TreatA, TreatC)
#> 
#>  F test to compare two variances
#> 
#> data:  TreatA and TreatC
#> F = 7.4242, num df = 11, denom df = 11, p-value = 0.002435
#> alternative hypothesis: true ratio of variances is not equal to 1
#> 95 percent confidence interval:
#>   2.137273 25.789584
#> sample estimates:
#> ratio of variances 
#>           7.424242
```

Variances of sprays A & C **differ significantly** (F-Test, $F_{11,11}$ = 7.42, p = 0.002)

Test for equal means

Test for equal means

t-test

- **Normal distribution** AND **equal variance**
- Compares if mean values are within range of standard error of each other
- p: How likely is this (or more extreme) difference between means if the population means were truly equal?

Welch-Test (corrected t-test)

- **Normal distribution** but **unequal variance**

Wilcoxon rank sum test

- **Non-normal distribution** and **unequal variance**
- Compares rank sums of the data
- Non-parametric

t-test

H_0: The samples do not differ in their mean

Treatment A and B: **normally distributed** and **equal variance**

```
t.test(TreatA, TreatB, var.equal = TRUE)
#>
#>  Two Sample t-test
#>
#> data:  TreatA and TreatB
#> t = -0.45352, df = 22, p-value = 0.6546
#> alternative hypothesis: true difference in means is not equal to 0
#> 95 percent confidence interval:
#>  -4.643994  2.977327
#> sample estimates:
#> mean of x mean of y
#>  14.50000  15.33333
```

- t: test statistics (t = 0 means equal means)
- df: degrees of freedom of t-statistics
- p-value: how likely is this extreme of a difference if H_0 was true?

The means of spray A and B **don't differ significantly** (t = -0.45, df = 22, p = 0.66)

Welch-Test

H_0: The samples do not differ in their mean

Treatment A and C: **normally distributed** and **non-equal variance**

```
t.test(TreatA, TreatC, var.equal = FALSE)
#> 
#>  Welch Two Sample t-test
#> 
#> data:  TreatA and TreatC
#> t = 7.5798, df = 13.91, p-value = 2.655e-06
#> alternative hypothesis: true difference in means is not equal to 0
#> 95 percent confidence interval:
#>   7.885546 14.114454
#> sample estimates:
#> mean of x mean of y 
#>      14.5       3.5
```

The means of spray A and C do **differ significantly** ($t = 7.58$, $df = 13.9$, $p < 0.001$)

Wilcoxon-rank-sum Test

H_0: The samples do not differ in their mean

We don't need the Wilcoxon test to compare treatment A and B, but for the sake of an example:

```
wilcox.test(TreatA, TreatB)
#>
#>  Wilcoxon rank sum test with continuity correction
#>
#> data:  TreatA and TreatB
#> W = 62, p-value = 0.5812
#> alternative hypothesis: true location shift is not equal to 0
```

The means of spray A and B do **not differ significantly** (W = 62, p = 0.58)

Paired values

Are there pairs of data points?

Example: samples of invertebrates across various rivers before and after sewage plants.

- For each plant, there is a pair of data points (before and after the plant)
- Question: Is the change (before-after) significant

Use `paired = TRUE` in the test.

```
t.test(TreatA, TreatB, var.equal = TRUE, paired = TRUE)
t.test(TreatA, TreatB, var.equal = FALSE, paired = TRUE)
wilcox.test(TreatA, TreatB, paired = TRUE)
```

Careful: your treatment vector both have to have the same order

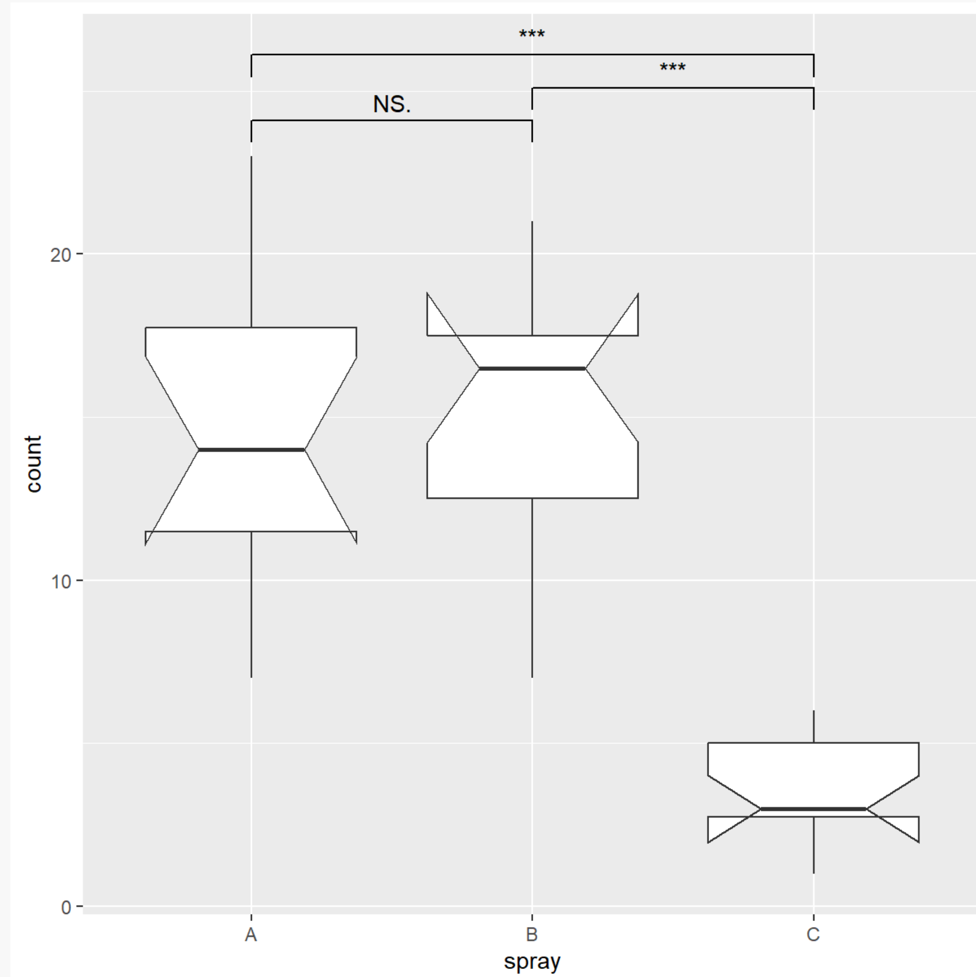
Plot test results with `ggsignif`

The `ggsignif` package offers a `geom_signif()` layer that can be added to a ggplot to annotate significance levels

```
# install.packages("ggsignif")
library(ggsignif)
```

Plot test results with `geom_signif()`

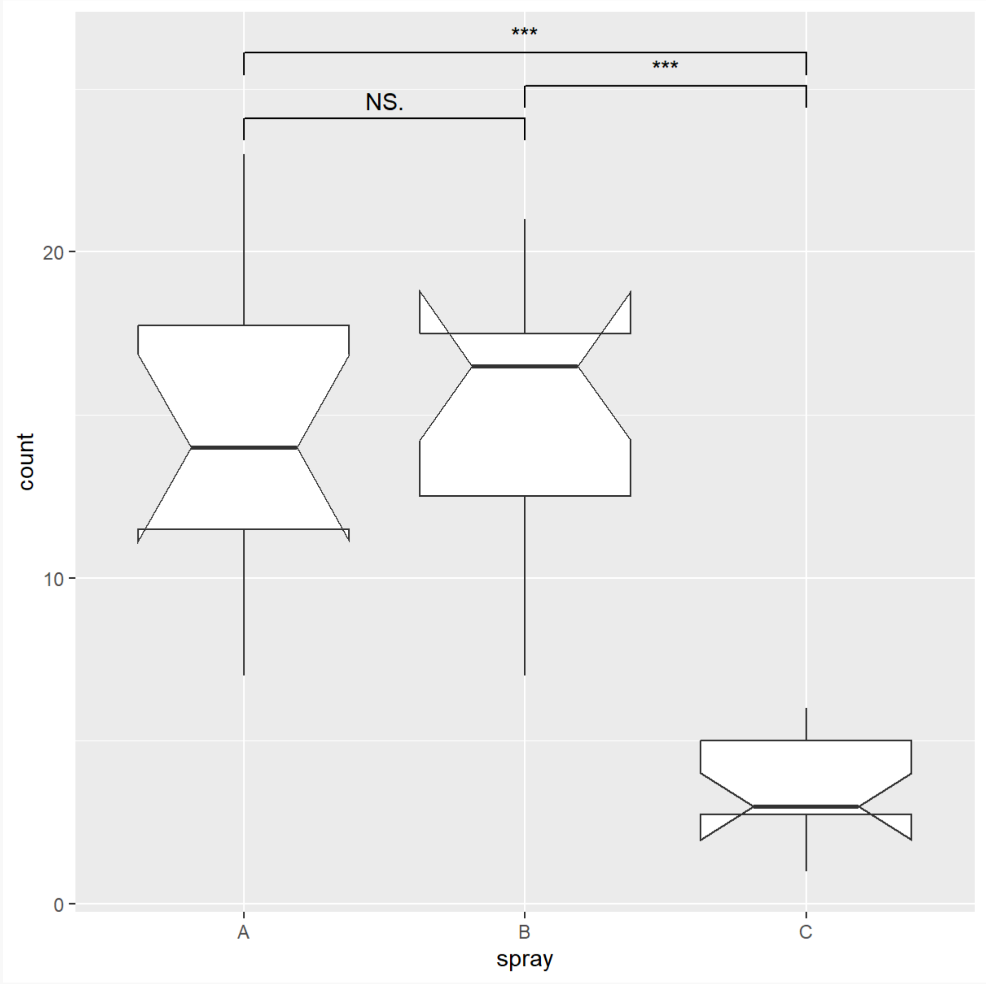
```r
ggplot(
  InsectSprays,
  aes(x = spray, y = count)
) +
  geom_boxplot(notch = TRUE) +
  geom_signif(
    comparisons = list(
      c("A", "B"),
      c("B", "C"),
      c("A", "C")
    ),
    map_signif_level = TRUE,
    y_position = c(23, 24, 25)
  )
```



- By default, a Wilcoxon test is performed

Plot test results with `geom_signif()`

```
ggplot(
  InsectSprays,
  aes(x = spray, y = count)
) +
  geom_boxplot(notch = TRUE) +
  geom_signif(
    comparisons = list(
      c("A", "B"),
      c("B", "C"),
      c("A", "C")
    ),
    test = "t.test",
    test.args = list(
      var.equal = TRUE
    ),
    map_signif_level = TRUE,
    y_position = c(23, 24, 25)
  )
```



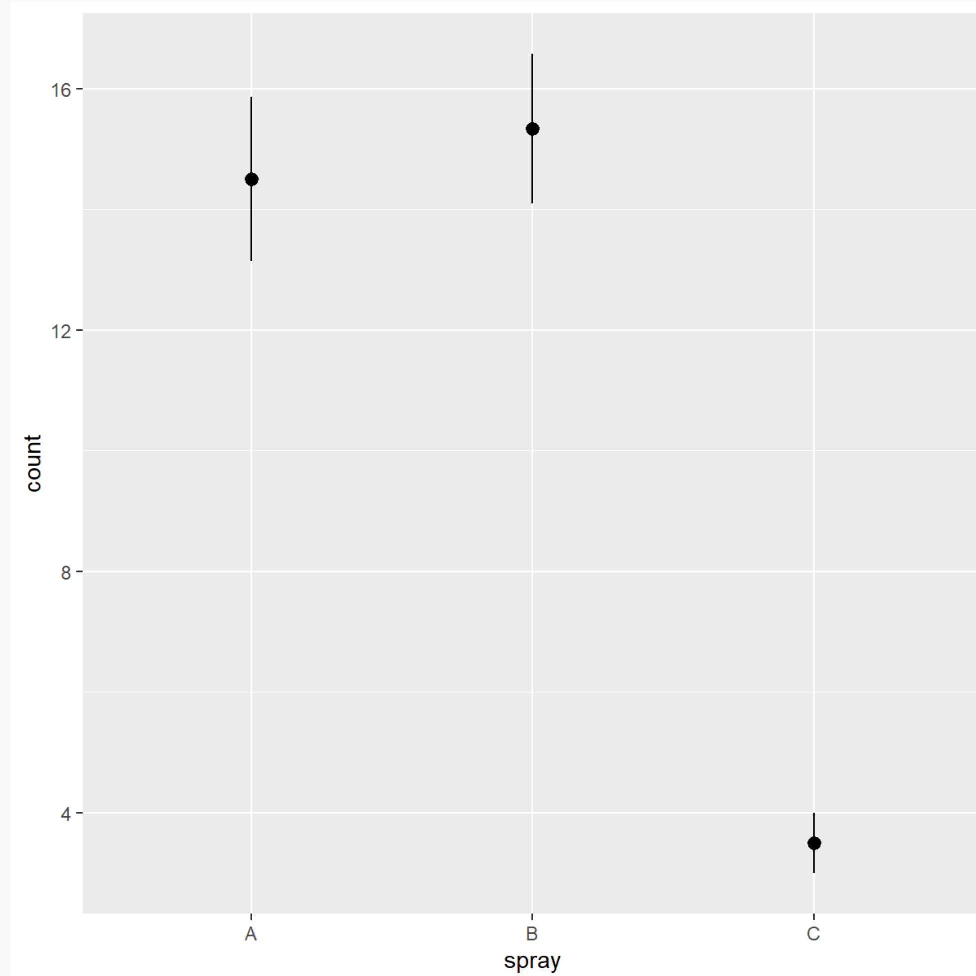
- `test`: run specific test
- `test.args`: pass additional arguments in a list
- `?geom_signif` for more options

Plot mean +- se using `stat_summary`

Another way to plot the results is to plot mean and standard error of the mean:

```r
ggplot(
  InsectSprays,
  aes(x = spray, y = count)
) +
  stat_summary()
```

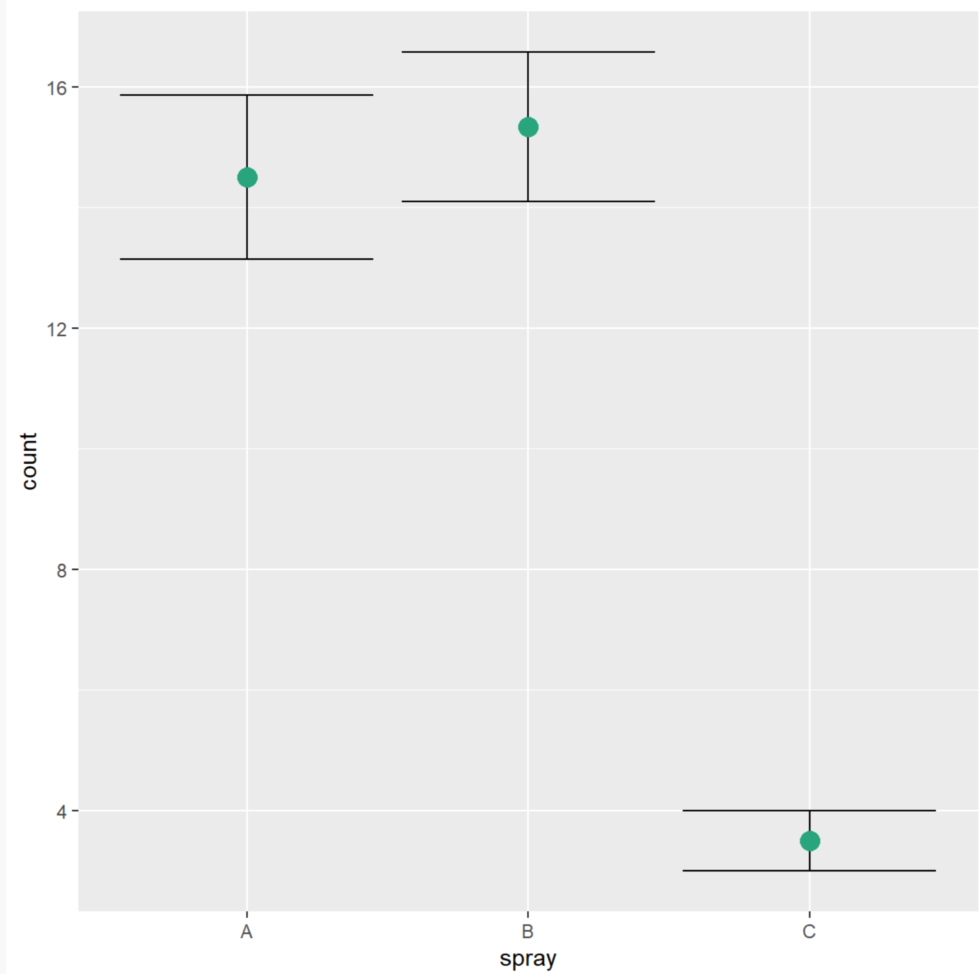
- By default `stat_summary` adds mean and standard error of the mean as pointrange



Plot mean +- se using `stat_summary`

Another way to plot the results is to plot mean and standard error of the mean:

```
ggplot(
  InsectSprays,
  aes(x = spray, y = count)
) +
  stat_summary(
    fun.data = mean_se,
    geom = "errorbar"
  ) +
  stat_summary(
    fun.y = mean,
    geom = "point",
    color = "#28a87d",
    size = 4
  )
```

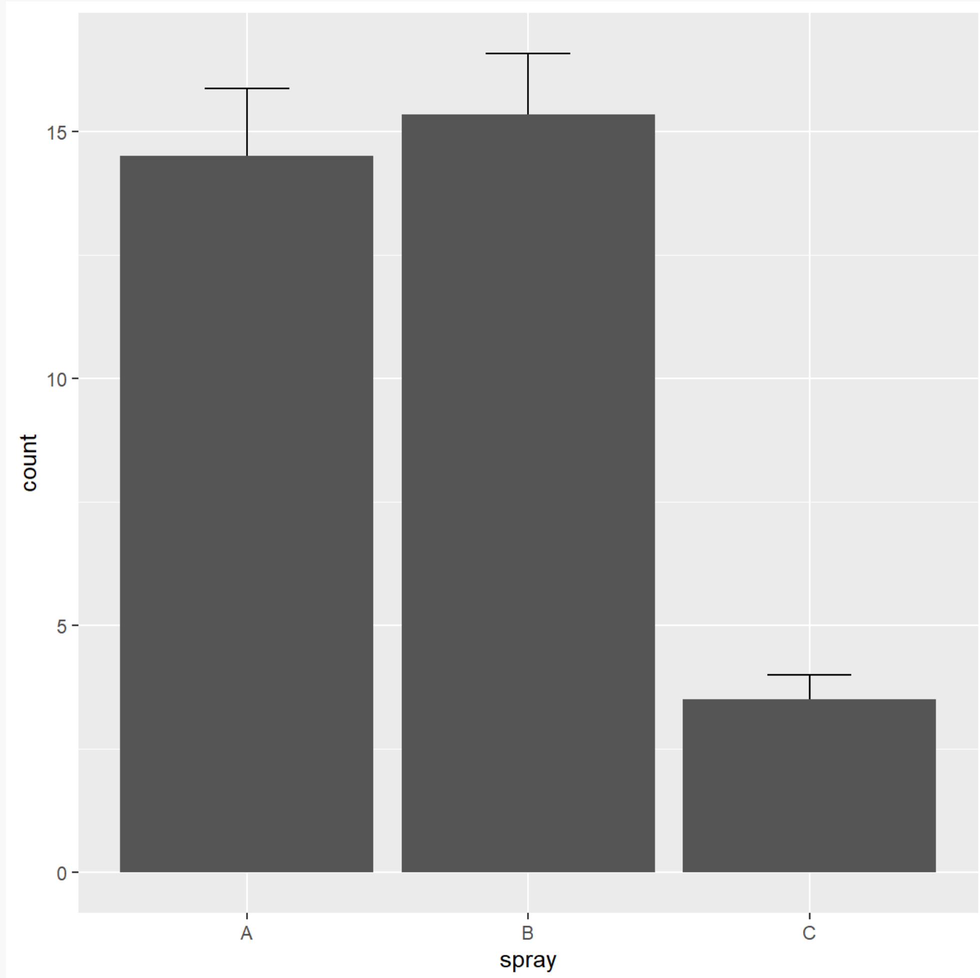


- Inside `stat_summary`, define summary function
  - `fun.data` for errorbars, `fun.y` for points (e.g. mean)

Plot mean +- se using `stat_summary`

Another way to plot the results is to plot mean and standard error of the mean:

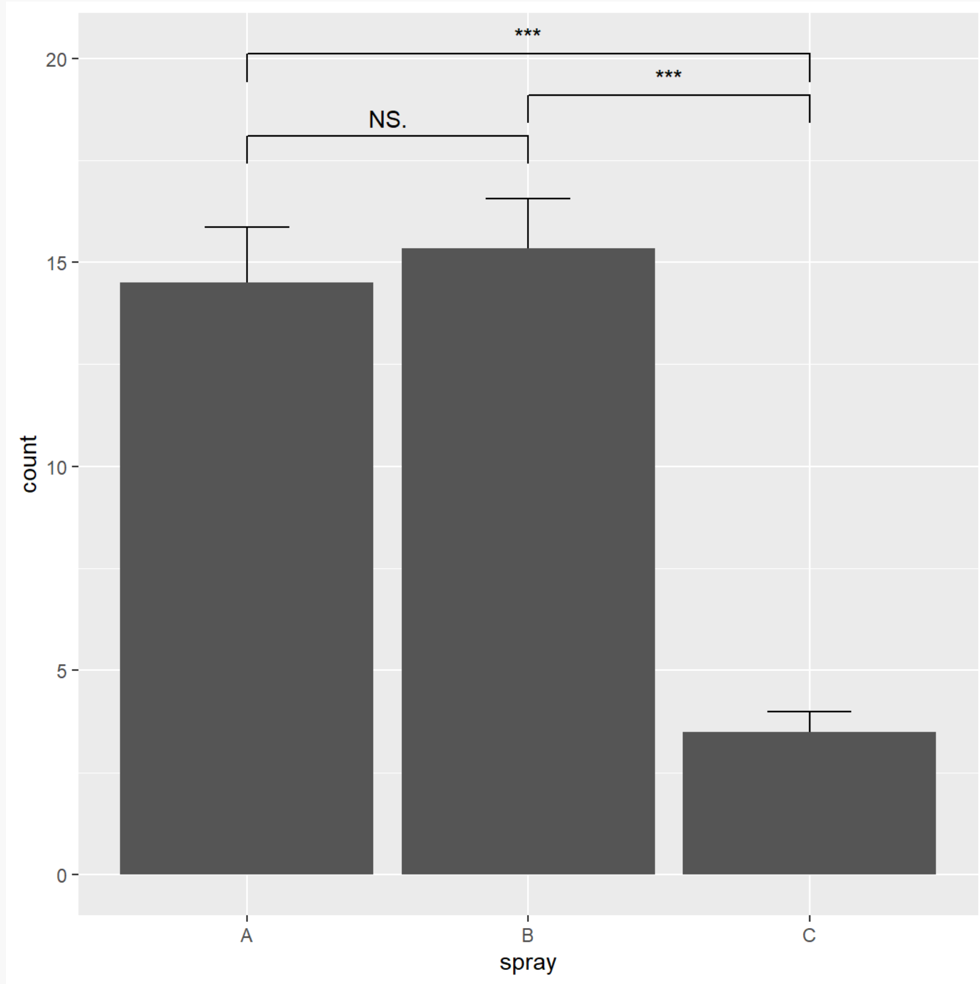
```r
ggplot(
  InsectSprays,
  aes(x = spray, y = count)
) +
  stat_summary(
    fun.data = mean_se,
    geom = "errorbar",
    width = 0.3
  ) +
  stat_summary(
    fun.y = mean,
    geom = "bar",
    size = 4
  )
```



Plot mean +- se using `stat_summary`

Just like before, you can also add a `geom_signif` to a barplot:

```
ggplot(
  InsectSprays,
  aes(x = spray, y = count)
) +
  stat_summary(
    fun.data = mean_se,
    geom = "errorbar",
    width = 0.3
  ) +
  stat_summary(
    fun.y = mean,
    geom = "bar"
  ) +
  ggsignif::geom_signif(
    comparisons = list(
      c("A", "B"),
      c("B", "C"),
      c("A", "C")
    ),
    test = "t.test",
    map_signif_level = TRUE,
    y_position = c(17, 18, 19)
  )
```



Now you

Task 1 (45) min)

Statistical tests

Find the task description here