One-way ANOVA + Assumptions check

2025-11-24

Assignment download and submission

1) Clone this repo to your computer (into your already established GitHub Repo, under a directory named Labs) https://github.com/suyoghc/PSY503_Lab10_AssumptionChecks_OneWayAnova
2) Work on the assignment with regular commits
3) Push the files to your own repo when you are done.

Today's lab

Experiment Design

Let's use a dataset from https://journals.sagepub.com/doi/10.1177/0956797615583071 The study looks at whether playing Tetris 24 hours after trauamatic memory formation reduces intrusive memories.

Participants were randomly allocated to one of four groups and watched a video designed to be traumatic:

1. Control
2. Reactivation + Tetris
3. Tetris
4. Reactivation

They measured the number of intrusive memories prior to the start of the study, then participants kept a diary to record intrusive memories about the film in the 7 days after watching it.

The authors were interested in whether the combination of reactivation and playing Tetris would lead to the largest reduction in intrusive memories. You will recreate their analyses using a one-way ANOVA.

Analysis

```
# Loading packages and reading data
#install.packages('performance')
library(pwr)
library(effectsize)
library(broom)
library(afex) #n-way anova; not being used here
```

```
## Loading required package: lme4
```

```
## Loading required package: Matrix

## ************
## Welcome to afex. For support visit: http://afex.singmann.science/

## - Functions for ANOVAs: aov_car(), aov_ez(), and aov_4()
## - Methods for calculating p-values with mixed(): 'S', 'KR', 'LRT', and 'PB'
## - 'afex_aov' and 'mixed' objects can be passed to emmeans() for follow-up tests
## - Get and set global package options with: afex_options()
## - Set sum-to-zero contrasts globally: set_sum_contrasts()
## - For example analyses see: browseVignettes("afex")
## ************

##
## Attaching package: 'afex'

## The following object is masked from 'package:lme4':
##
##     lmer
```

```
library(emmeans)
```

```
## Welcome to emmeans.
## Caution: You lose important information if you filter this package's results.
## See '? untidy'
```

```
library(performance)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   4.0.0     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x tidyr::pack()   masks Matrix::pack()
## x tidyr::unpack() masks Matrix::unpack()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# Read data
# Create a new variable called PID that equals row_number() to act as a participant ID
# Convert Condition to a factor

james_data <- read_csv("James_2015.csv") %>%
  mutate(PID = row_number(),
         Condition = as.factor(Condition)) %>%
  select(PID,
         Condition,
         intrusions = Days_One_to_Seven_Image_Based_Intrusions_in_Intrusion_Diary)
```

```
## Rows: 72 Columns: 28
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (28): Condition, Time_of_Day, BDI_II, STAI_T, pre_film_VAS_Sad, pre_film...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Q1. Intepret the output of the following chunk in connection to the experiment design outlined above. That is, how many participants, how many conditions? the type of measurements the DV? The values in the dataset can also be viewed via the workspace , via head(), or by just printing the dataset.

```
str(james_data)
```

```
## tibble [72 x 3] (S3: tbl_df/tbl/data.frame)
##  $ PID       : int [1:72] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Condition : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ intrusions: num [1:72] 4 3 6 2 3 4 0 4 2 11 ...
```

```
head(james_data)
```

```
## # A tibble: 6 x 3
##     PID Condition intrusions
##   <int> <fct>          <dbl>
## 1     1 1                  4
## 2     2 1                  3
## 3     3 1                  6
## 4     4 1                  2
## 5     5 1                  3
## 6     6 1                  4
```

A1

There are 72 participants and 4 conditions. The memory intrusion (DV) has numerical values in a ratio scale?

Next, we want to calculate some descriptive statistics to see some overall trends in the data. We are really interested in the scores from each experimental group rather than overall.

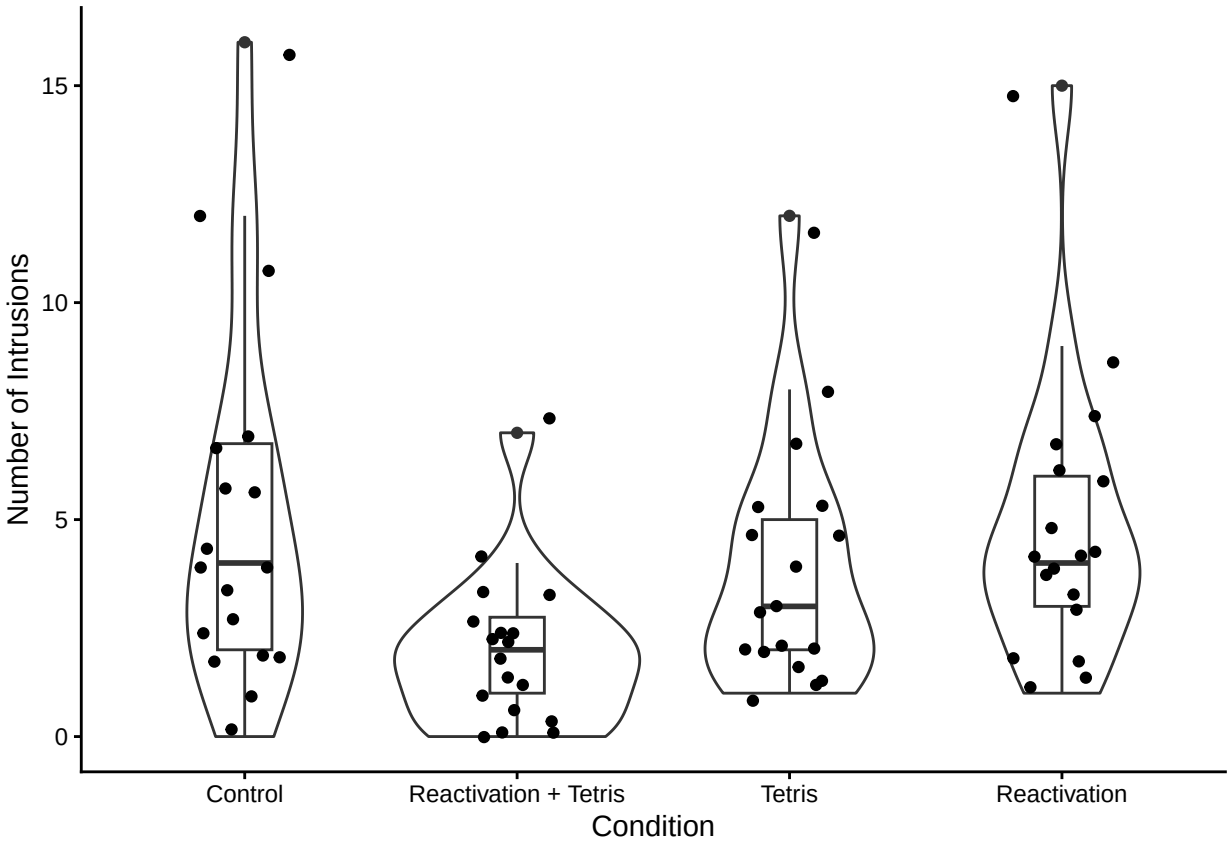
```
#Create a violin-boxplot with the number of intrusive memories on the y-axis and condition on the x-axi
#Change the labels on the x-axis to something more informative for the condition names.
james_data %>%
  group_by(Condition) %>%
  summarise(mean = round(mean(intrusions), 2),
            sd = round(sd(intrusions), 2),
            se = round(sd/sqrt(length(intrusions)), 2))
```

```
## # A tibble: 4 x 4
##   Condition  mean    sd    se
```

```
##   <fct>     <dbl> <dbl> <dbl>
## 1 1          5.11  4.23  1
## 2 2          1.89  1.75  0.41
## 3 3          3.89  2.89  0.68
## 4 4          4.83  3.33  0.78
```

Now we can visualise the data. In the original paper they use a bar plot, but let's use a better plot that gives us more information about the data.

```
james_data %>%
  ggplot(aes(x = Condition, y = intrusions))+
  geom_violin()+
  geom_boxplot(width = .2) +
  geom_jitter(width = .2) +
  scale_y_continuous(name = "Number of Intrusions") +
  scale_x_discrete(labels = c("Control", "Reactivation + Tetris", "Tetris", "Reactivation")) +
  theme_classic()
```



While violin plots are more informative than bar plots, notice that they don't give you a good idea about the sample sizes.

Q2a Use geom_jitter and edit the above graph into a dot+violin plot

Q2b: By just looking at the updated graph, what is your guess (along with reasoning) for the results of carrying out an anova? First, with respect to the omnibus test, and then with respect to the pairwise contrasts. ###### A2b

Overall, it seems like there are some group differences in the number of intrusive memories. The range and overall shape of the violin plots vary across groups. For pairwise contrast, I think only <Control-Reactivation+Tetris> and <Reactivation-Reactivation+Tetris> pairs would show significant differences. Their median values look very much apart from each other.

One-way ANOVA

We can carry out the one-way ANOVA using the aov() function.

Q3: Fit a one-way ANOVA model to the data using the aov function. (don't change the variable name 'mod', as it gets referred to later in the assignment) ###### A3:

```
?aov
```

```
## starting httpd help server ... done
```

```
mod <- aov(intrusions~Condition, data=james_data)
summary(mod)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## Condition    3  114.8   38.27   3.795 0.0141 *
## Residuals   68  685.8   10.09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Before running the main F-test, the key assumptions of ANOVA must be checked. Since ANOVA is a parametric test based on the normal distribution, the assumptions include the DV being interval or ratio, independence, homogeneity of variance, and that the distributions within groups (or the residuals) are normally distributed

From the study design, we know that assumptions 1 and 2 are satisfied.

For other checks, it turns out that adding the argument which=i, to the plot function produces different types of model diagnostic outputs when the fitted model,mode is passed as an object.

plot(mod, which = 1)

Q4: Iterate through different values (numbers from 1-6) of the which parameter to check modeling assumptions.

1) for normality using a Q-Q plot
2) constant variance across groups

NOTE 1 that there are many options because some visualizations are better for different contexts (e.g. continuous predictors vs categorical predictors)

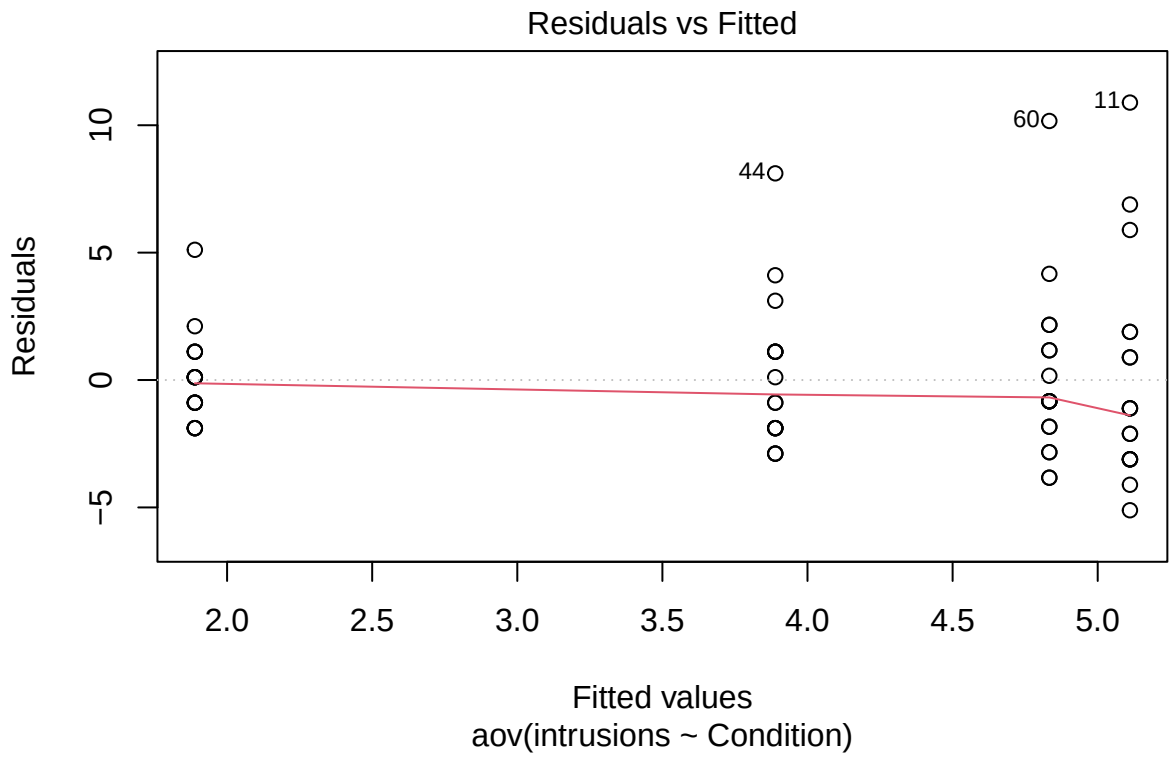
NOTE 2: Assessing independence requires considering study design (based on theory)! However, as the number of predictors / information about them increases, one can check clustering of residuals along predictor

variables. (E.g. is there a random effect due to some variable) but here we do not have other information other than the grouping and the potential fixed effects due to them)

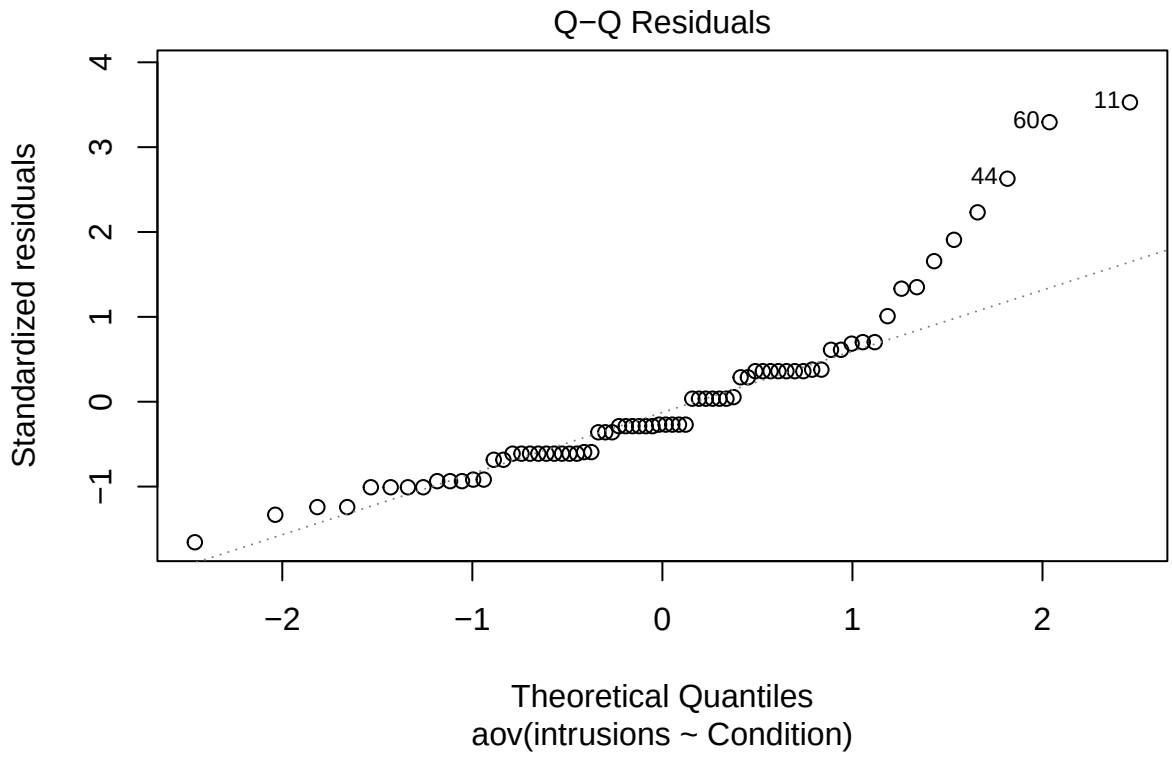
A4:

Plots of relevant visual model checks

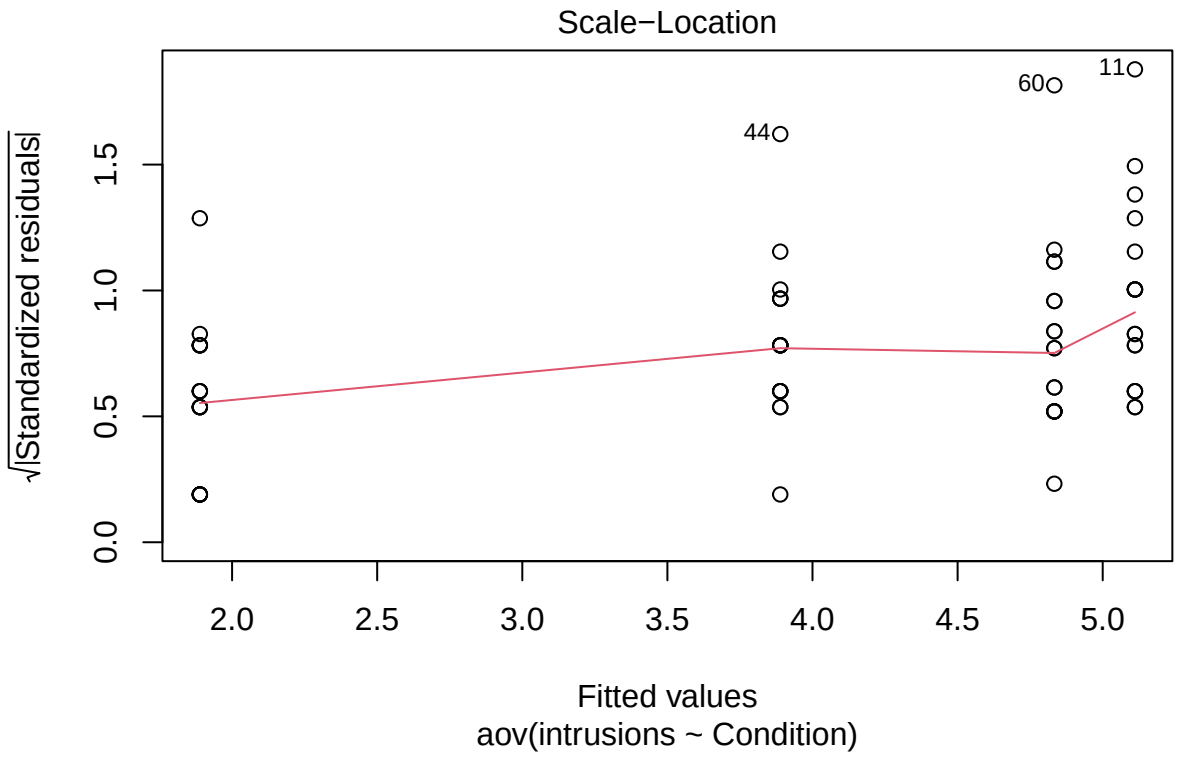
```
#Your answer here
plot(mod, which=1)
```



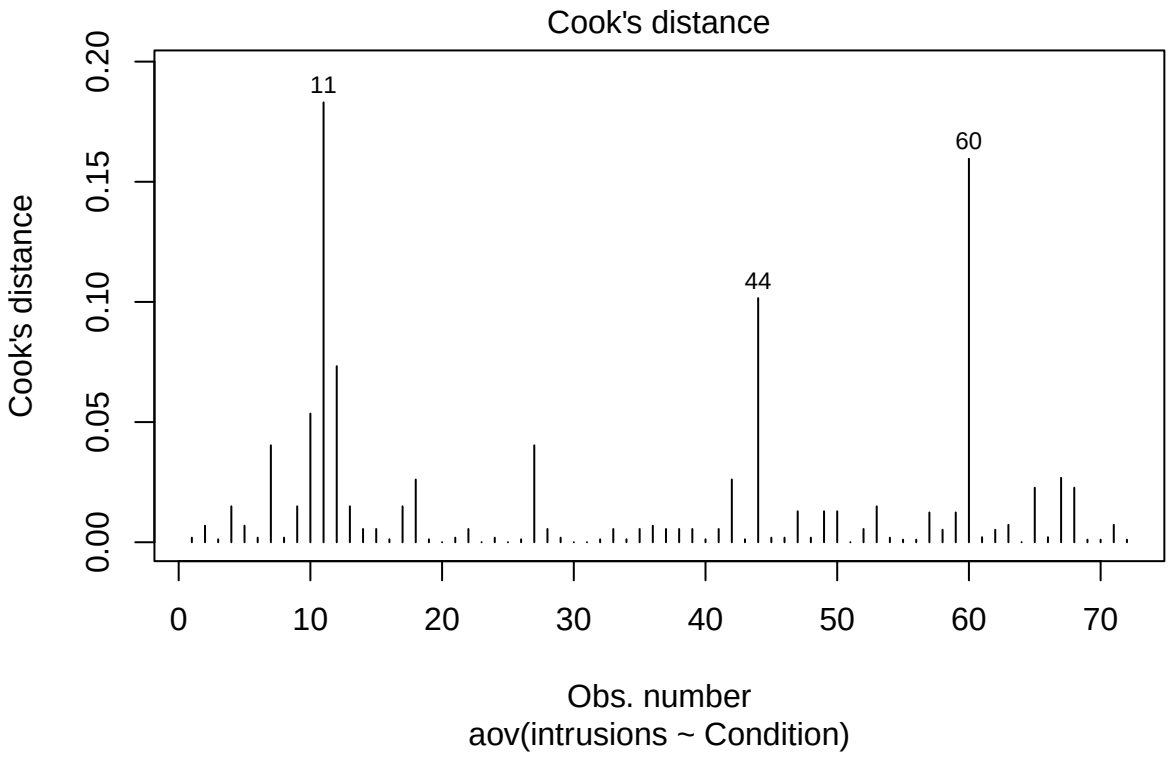

```
plot(mod, which=2) #Q-Q plot
```



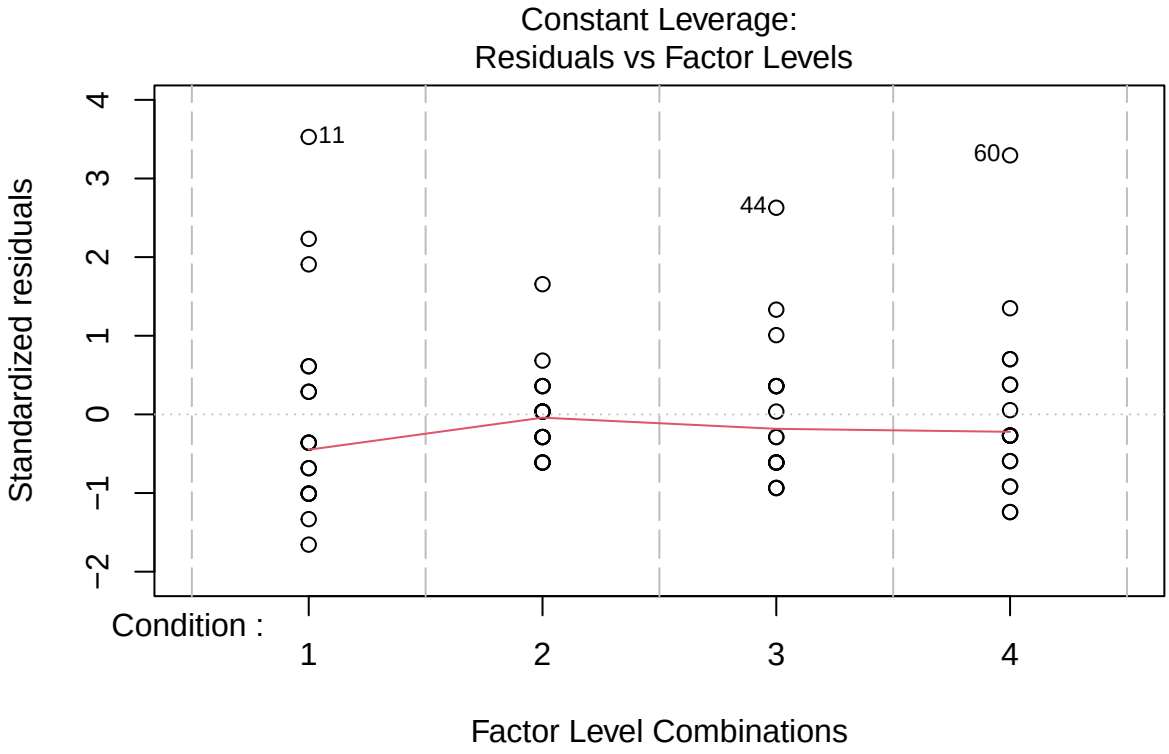

```
plot(mod, which=3)
```



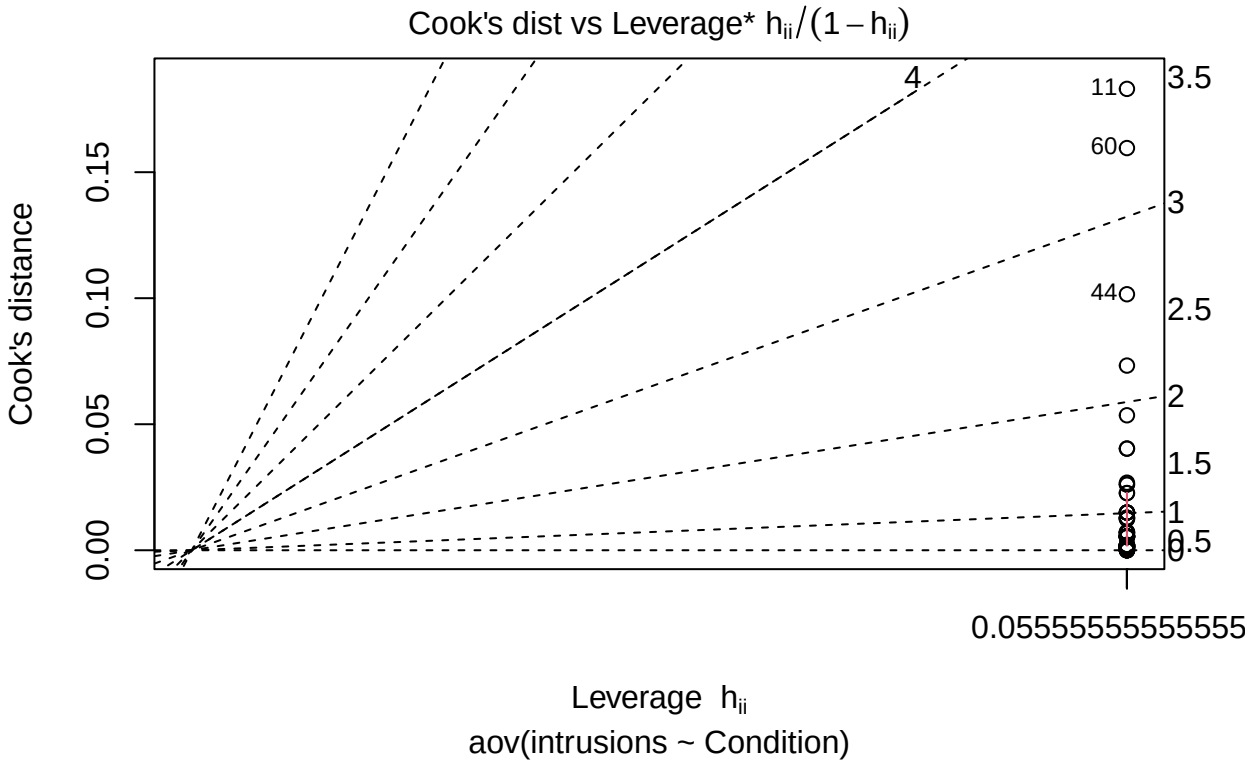

```
plot(mod, which=4)
```




```
plot(mod, which=5) #constant variance across groups?
```

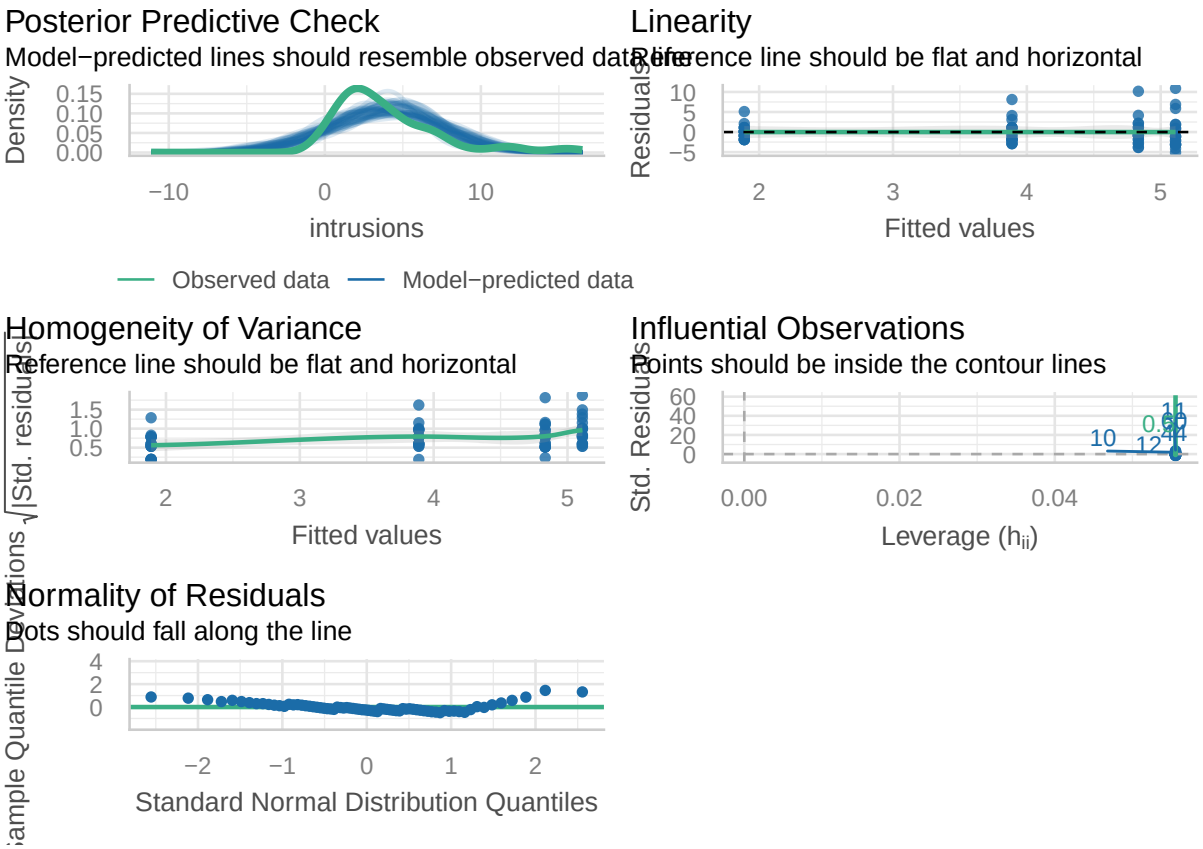



```
plot(mod, which=6)
```



Q5: The performance package has a neat function 'check_model()' which allows one to carry out a wide array of visual tests with a single command. Run this function and report your observations ###### A5:

```
check_model(mod, check = "all")
```



The model-predicted lines are not perfectly aligned with the observed data line, which implies some systemic discrepancies between observed and simulated data. Linearity plot looks ok, while the reference line in homogeneity of variance plot isn't quite flat. It means the variance might vary across condition groups. Normality of residuals also look quite off the grid.

Now, we can check how many participants are in each condition using count():

```
james_data %>%
  count(Condition)
```

```
## # A tibble: 4 x 2
##   Condition     n
##   <fct>     <int>
## 1 1            18
## 2 2            18
## 3 3            18
## 4 4            18
```

Generally, equal sample sizes result in more robust ANOVA results (particularly when assumption of normality and homegeniety of variance are not met). Thankfully, the sample sizes are equal, and we should be OK to proceed with the ANOVA.

Q6a: Display the results of ANOVA stored in 'mod'. Interpret what it means. And express this result the way they are generally reported in papers. ###### A6a:

```r
summary(mod)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## Condition    3  114.8   38.27   3.795 0.0141 *
## Residuals   68  685.8   10.09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
emmeans(mod, ~Condition)
```

```
##  Condition emmean    SE df lower.CL upper.CL
##  1           5.11 0.749 68    3.617     6.60
##  2           1.89 0.749 68    0.395     3.38
##  3           3.89 0.749 68    2.395     5.38
##  4           4.83 0.749 68    3.340     6.33
##
## Confidence level used: 0.95
```

A one-way ANOVA showed a significant effect of condition on the number of intrusive memories, $F(3, 68) = 3.80$, $p = .014$.

Q6b: As per usual we can make model fit outputs easier to work with by using tidy(). Use this output to directly extract and report the result of the omnibus test in 6a

```r
mod_output <- mod %>% tidy()
mod_output
```

```
## # A tibble: 2 x 6
##   term         df sumsq meansq statistic p.value
##   <chr>     <dbl> <dbl>  <dbl>     <dbl>   <dbl>
## 1 Condition     3  115.   38.3      3.79  0.0141
## 2 Residuals    68  686.   10.1     NA    NA
```

A6b:

A one-way ANOVA showed a significant effect of condition on the number of intrusive memories, $F(3, 68) = 3.79$, $p = 0.014$.

Post-hoc tests For post-hoc comparisons, the paper appears to have computed Welch t-tests but there is no mention of any multiple comparison correction! We could reproduce these results by using t.test() for each of the contrasts.

For example, to compare condition 1 (the control group) with condition 2 (the reactivation plus tetris group) we could run:

```r
james_data %>%
  filter(Condition %in% c("1", "2")) %>%
  droplevels() %>% # ignore unused factor levels
  t.test(intrusions ~ Condition,
         data = .)
```

```
##
##  Welch Two Sample t-test
##
## data:  intrusions by Condition
## t = 2.9893, df = 22.632, p-value = 0.006627
## alternative hypothesis: true difference in means between group 1 and group 2 is not equal to 0
## 95 percent confidence interval:
##  0.990331 5.454113
## sample estimates:
## mean in group 1 mean in group 2
##        5.111111        1.888889
```

Because Condition has four levels, we cannot just specify intrustion ~ Condition because a t-test compares two groups and it would not know which of the four to compare so first we have to filter the data and use a new function droplevels(). It's important to remember that when it comes to R there are two things to consider, the data you can see and the underlying structure of that data. In the above code we use filter() to select only conditions 1 and 2 so that we can compare them. However, that does not change the fact that R "knows" that Condition has four levels - it does not matter if two of those levels do not have any observations any more, the underlying structure still says there are four groups. droplevels() tells R to remove any unused levels from a factor. You could try running the above code but without droplevels() and see what happens.

However, a quicker and better way of doing this that allows you apply a correction for multiple comparisons easily is to use emmeans() which computes all possible pairwise comparison t-tests and applies a correction to the p-value.

First, we use emmeans() to run the comparisons and then we can pull out the contrasts and use tidy() to make it easier to work with. (NOTE: contrasts just refers to the different pairwise comparisons between means that are possible)

Q7: Run the code below. Which conditions are significantly different from each other? Are any of the comparisons different from the ones reported in the paper now that a correction for multiple comparisons has been applied?

```
mod_pairwise <-emmeans(mod,
                       pairwise ~ Condition,
                       adjust = "bonferroni")

mod_contrasts <- mod_pairwise$contrasts %>%
  tidy()

mod_contrasts
```

```
## # A tibble: 6 x 8
##   term      contrast   null.value estimate std.error    df statistic adj.p.value
##   <chr>     <chr>           <dbl>    <dbl>     <dbl> <dbl>     <dbl>       <dbl>
## 1 Condition Condition~          0    3.22       1.06    68     3.04       0.0199
## 2 Condition Condition~          0    1.22       1.06    68     1.15       1
## 3 Condition Condition~          0    0.278      1.06    68     0.262      1
## 4 Condition Condition~          0   -2.00       1.06    68    -1.89       0.379
## 5 Condition Condition~          0   -2.94       1.06    68    -2.78       0.0420
## 6 Condition Condition~          0   -0.944      1.06    68    -0.892      1
```

A7

In the paper, the authors reported that participants showed significantly lower intrusive memories in condition 2(reactivation+tetris) than in condition 3(tetris only). But this wasn't the case after applying bonferroni correction.

Power and effect sizes Finally, we can replicate their power analysis using pwr.anova.test from the pwr package.

In the paper they say, *"On the basis of the effect size of d = 1.14 from Experiment 1, we assumed a large effect size of f = 0.4. A sample size of 18 per condition was required in order to ensure an 80% power to detect this difference at the 5% significance level."*

Q8 Fill in the parameter values below to obtain the right power analysis ###### A8

```
pwr.anova.test(k = 4,
               f = 0.4,
               sig.level = 0.05,
               power = 0.8)
```

```
##
##      Balanced one-way analysis of variance power calculation
##
##               k = 4
##               n = 18.04262
##               f = 0.4
##       sig.level = 0.05
##           power = 0.8
##
## NOTE: n is number in each group
```

We have already got the effect size for the overall ANOVA, however, we should also really calculate Cohen's d using cohens_d() from effectsize for each of the pairwise comparisons. This code is a little long because you need to do it separately for each comparison, bind them all together and then add them to mod_contrasts - just make sure your understand which bits of the code you would need to change to run this on different data. As we are binding rows and columns rather than joining, it is critical the comparisons are already in the correct order.

```
# Calculate Cohen's d for all comparisons
d_1_2 <- cohens_d(intrusions ~ Condition,
                  data = filter(james_data,
                                Condition %in% c(1, 2)) %>%
                    droplevels())

d_1_3 <- cohens_d(intrusions ~ Condition,
                  data = filter(james_data,
                                Condition %in% c(1, 3)) %>%
                    droplevels())

d_1_4 <- cohens_d(intrusions ~ Condition,
                  data = filter(james_data,
                                Condition %in% c(1, 4)) %>%
                    droplevels())
```

```
d_2_3 <- cohens_d(intrusions ~ Condition,
                  data = filter(james_data,
                                Condition %in% c(2, 3)) %>%
                    droplevels())

d_2_4 <- cohens_d(intrusions ~ Condition,
                  data = filter(james_data,
                                Condition %in% c(2, 4)) %>%
                    droplevels())

d_3_4 <- cohens_d(intrusions ~ Condition,
                  data = filter(james_data,
                                Condition %in% c(3, 4)) %>%
                    droplevels())

# Bind all the comparisons in the order of mod_contrasts
pairwise_ds <- bind_rows(d_1_2,
                         d_1_3,
                         d_1_4,
                         d_2_3,
                         d_2_4,
                         d_3_4)

# Bind this object to the mod_contrasts object
mod_contrasts <- mod_contrasts %>%
  bind_cols(pairwise_ds)
```

What are your options if the data do not meet the assumptions and it's really not appropriate to continue with a regular one-way ANOVA? As always, there are multiple options and it is a judgement call.

You could run a non-parametric test, the Kruskal-Wallis for between-subject designs and the Friedman test for within-subject designs. If normality is the problem, you could try transforming the data. You could use bootstrapping too.