

# C91AR | Advanced Statistics using R

## Lecture 5: Screening Data & Tests of Normality

Dr Peter E McKenna

2025-02-21

### Contents

<b>1</b>	<b>Packages for today</b>	<b>2</b>
<b>2</b>	<b>Reading for Today</b>	<b>2</b>
<b>3</b>	<b>Aims for this module</b>	<b>3</b>
<b>4</b>	<b>Read in tidy data</b>	<b>3</b>
<b>5</b>	<b>Maze study metadata</b>	<b>3</b>
<b>6</b>	<b>Untidy the dataset</b>	<b>3</b>
<b>7</b>	<b>Examine the data</b>	<b>4</b>
<b>8</b>	<b>Recode our condition levels</b>	<b>4</b>
<b>9</b>	<b>Still not happy with the level labels?</b>	<b>4</b>
<b>10</b>	<b>Tidy up our object list</b>	<b>5</b>
<b>11</b>	<b>Listwise deletion</b>	<b>5</b>
<b>12</b>	<b>Using <code>drop_na</code></b>	<b>5</b>
<b>13</b>	<b>Remove <code>df_messy_na</code></b>	<b>6</b>
<b>14</b>	<b>Replacing missing values with the mean</b>	<b>6</b>

15 Cleaning	7
16 Calculate the mean for log(rt)	8
17 Replace missing values with avg_rt	8
18 Examine the vector	9
19 How big are the group sizes after removing missing cases?	9
20 Plot the data	9
21 TidypLOTS equivalent	10
22 Checking the normality of the data using psych	11
23 Boxplot with comparison analysis from tidypLOTS	11
24 Roundup	12
25 Cleanup	12
26 References	13

## 1 Packages for today

```
# load packages
pacman::p_load(tidyverse,
               snakecase,
               psych,
               summarytools,
               messy,
               tidypLOTS)
```

## 2 Reading for Today

Chapter 12: Screening data

### 3 Aims for this module

- I'll show you how to deliberately untidy data using the **messy** package
- We'll do some more label tidying
- Examined the case of listwise deletion (the reading covers other approaches)
- Test distribution normality
- Show you **tidyplots** as an alternative to **ggplot2**

### 4 Read in tidy data

```
# read in dataset
df <-
  read_csv("data_tidy/c91ar_maze_data.csv",
           col_types = "cccdccd")
```

### 5 Maze study metadata

- **id** = anon participant code
- **condition** = ToM manipulation with three levels (Baseline, No-ToM, and ToM)
- **trial** = experiment trial
- **rt** = response time
- **follow\_robot** = whether participants followed the robot's suggestion
- **accuracy** = whether or not their maze route selection was correct
- **conf** = participants self reported confidence for each route decision.

### 6 Untidy the dataset

```
# make script reproducible
set.seed(1234)

# untidy our maze data
df_messy <-
  df |>
  make_missing(cols = "follow_robot",
               missing = NA,
               messiness = 0.05) |> # add missing values to follow_robot
```

```
make_missing(cols = "rt",
             missing = NA,
             messiness = 0.3) |> # add missing and erroneous values to rt
add_special_chars(cols = "condition") # add special chars to condition levels
```

## 7 Examine the data

```
df_messy |>
  dfSummary()
```

- What do you notice from the output of `dfSummary()`?
- What have we done??

## 8 Recode our condition levels

- Check variable levels

```
unique(df_messy$condition) # what a mess!
```

- What a mess! This was created using the `messy::add_special_chars` function
- Tidying up the mess

```
# using a combination of case_when and str_detect
df_messy_levels <-
  df_messy |>
  mutate(
    condition = str_replace_all(condition,
                                "[^[:alnum:][:space:]]", "")
  )

unique(df_messy_levels$condition)
```

## 9 Still not happy with the level labels?

```
# Probably we do a little more tidying
df_messy_levels <-
  df_messy_levels |>
  mutate(condition = case_match(condition,
                                "NoToM" ~ "No_ToM",
                                "baseline" ~ "Baseline",
                                .default = condition))

unique(df_messy_levels$condition)
```

## 10 Tidy up our object list

- You can see how our list of objects is growing in the global environment.
- You may decide at a certain point to drop some of the objects to keep the list short
- Let's update `df_messy` and remove `df_messy_levels`

```
# Update df_messy with df_messy_levels values
df_messy <-
  df_messy_levels

# remove df_messy_levels - as this is now contained in df_messy
rm(df_messy_levels)

# check data
unique(df_messy$condition)
```

## 11 Listwise deletion

- The `drop_na` function from **tidyr** removes any row of data that contains an NA (i.e., missing) value
- But this may not be ideal as we might end up removing a lot of useful observations from the dataset
- It may only pertain to a variable that is not integral to our research question, so could be ignored

## 12 Using `drop_na`

```
df_messy_na <-  
  df_messy |>  
  drop_na()
```

- What percentage of the observations is missing?

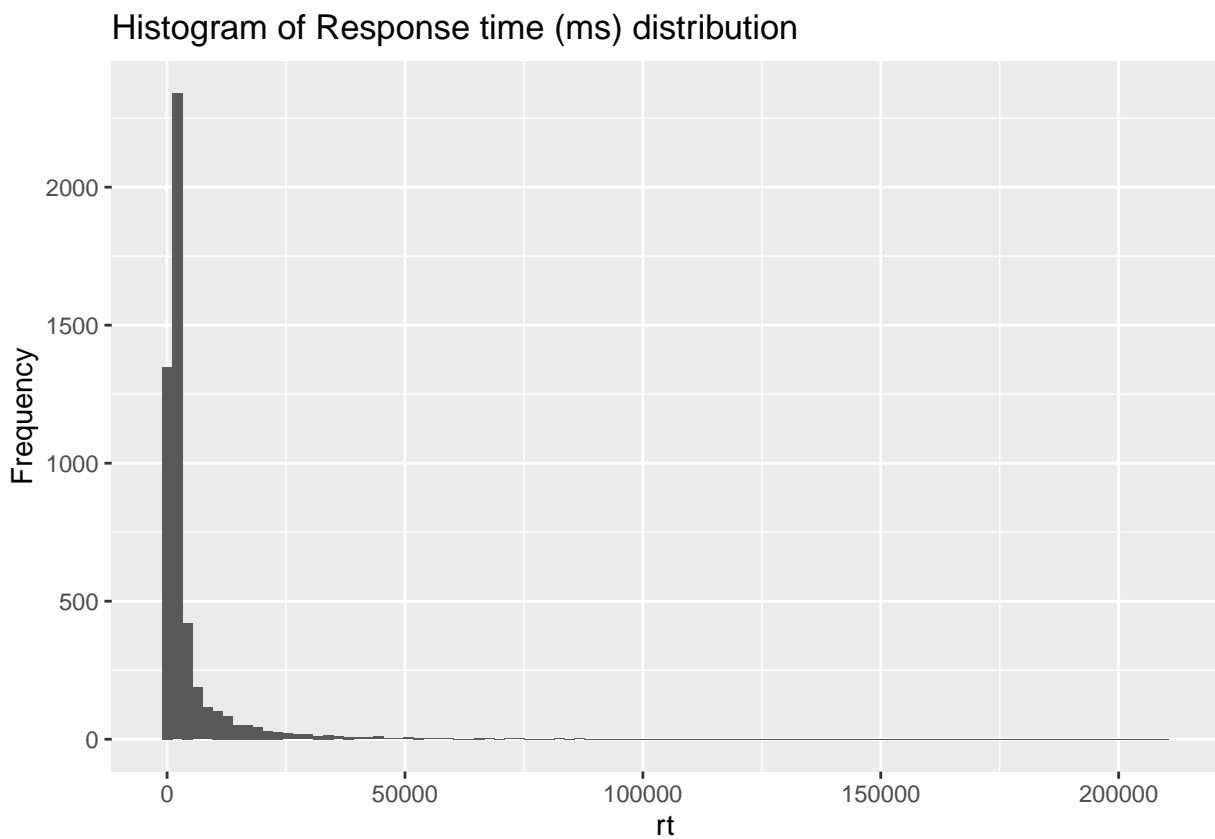
## 13 Remove df\_messy\_na

```
rm(df_messy_na)
```

## 14 Replacing missing values with the mean

- Say we have a continuous variable that **is integral** to our analysis
- An option is to replace missing values in a normal distribution with the mean

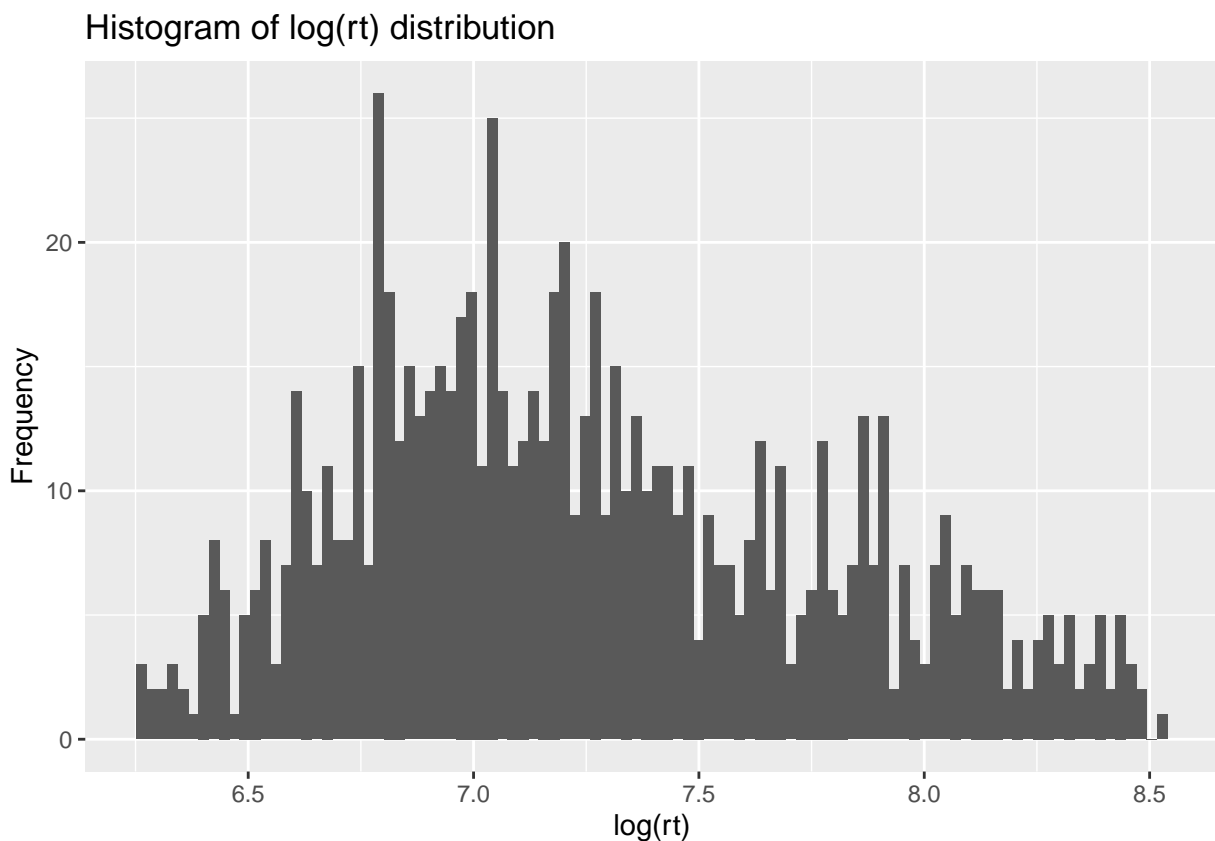
```
# examine the histogram  
df_messy |>  
  ggplot(mapping = aes(rt)) +  
  geom_histogram(bins = 100) +  
  labs(y = "Frequency",  
       title = "Histogram of Response time (ms) distribution")
```



## 15 Cleaning

- Let's set the bounds that we had previously, between 500:5000 ms and take the log

```
df_messy |>
  filter(rt %in% c(500:5000)) |>
  ggplot(mapping = aes(log(rt))) +
  geom_histogram(bins = 100) +
  labs(y = "Frequency",
       title = "Histogram of log(rt) distribution")
```



## 16 Calculate the mean for log(rt)

```
df_messy |>
  filter(rt %in% c(500:5000)) |>
  summarise(avg_rt = mean(rt))
```

## 17 Replace missing values with avg\_rt

```
# Create mean variable
avg_rt <- 1646.523

# create new object that replaces missing rt values with the mean
df_messy1 <-
  df_messy |>
  filter(rt %in% c(500:5000)) |>                                # remember to set the new bounds!
  mutate(rt = if_else(is.na(rt),
                      avg_rt, # if the cell is empty, enter the mean of the vector
                      rt))                                         # otherwise, put what was there already
```



## 18 Examine the vector

```
# Check for old
df_messy |>
  summarise(count_nas = is.na(rt) |>
    sum()) |>
  pluck("count_nas")

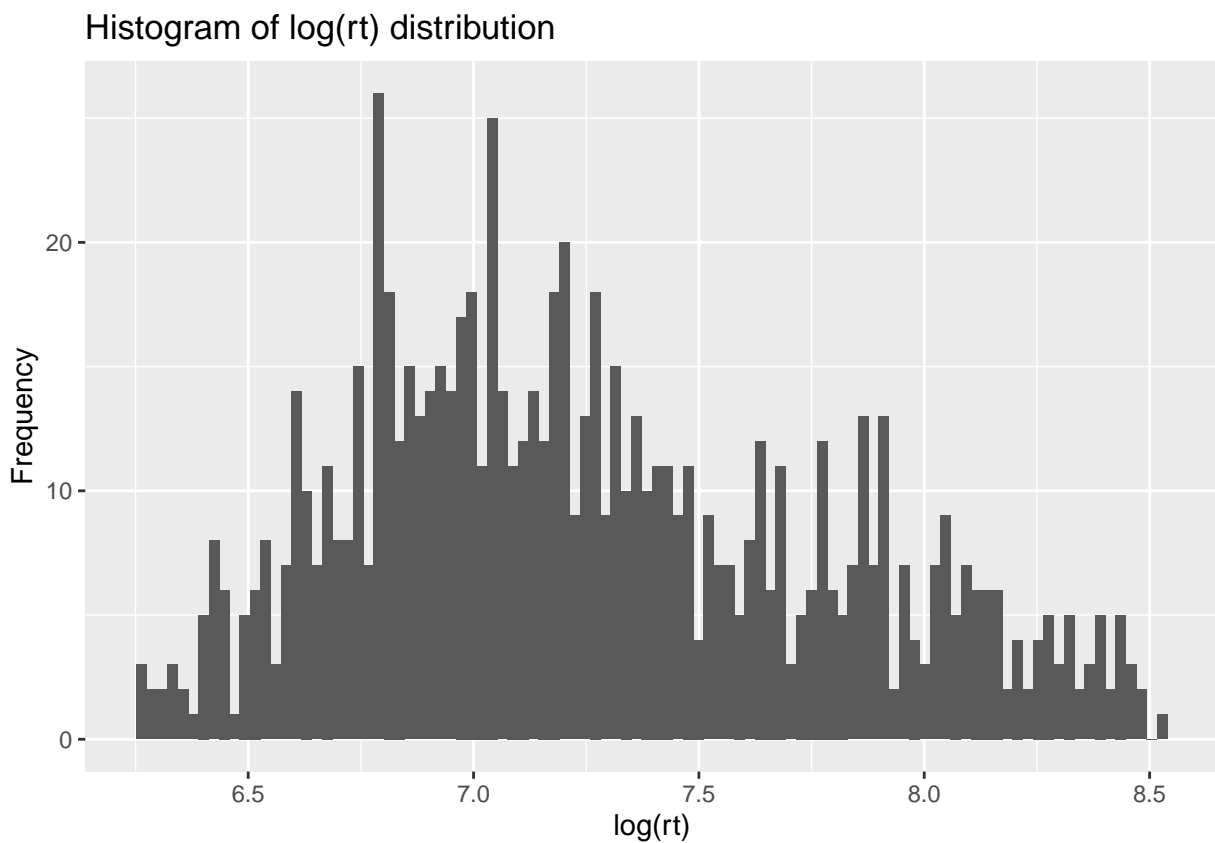
# Check for new
df_messy1 |>
  summarise(count_nas = is.na(rt) |>
    sum()) |>
  pluck("count_nas")
```

## 19 How big are the group sizes after removing missing cases?

```
# What would I write here to check the group sizes?
```

## 20 Plot the data

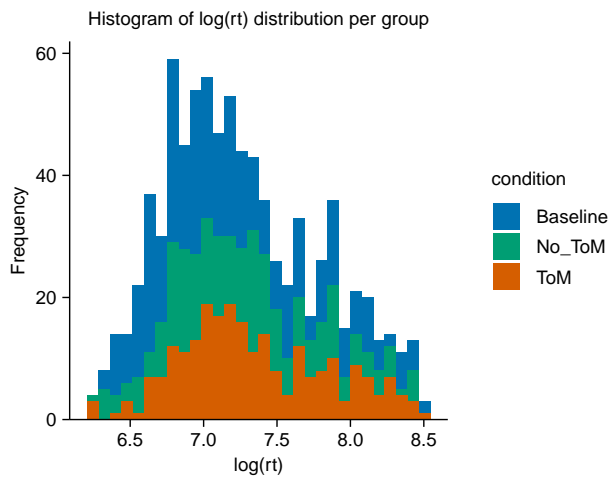
```
df_messy1 |>
  ggplot(mapping = aes(log(rt))) +
  geom_histogram(bins = 100) +
  labs(y = "Frequency",
    title = "Histogram of log(rt) distribution")
```



## 21 TidypLOTS equivalent

```
# using tidypLOTS
df_messy1 <-
  df_messy1 |>
  mutate(log_rt = log(rt)) # tidypLOTS needs to be passed existing vectors

df_messy1 |>
  tidypLOT(x = log_rt, color = condition) |>
  add_histogram(bins = 30) |>
  add_title("Histogram of log(rt) distribution per group") |>
  adjust_x_axis_title("log(rt)") |>
  adjust_y_axis_title("Frequency")
```



## 22 Checking the normality of the data using psych

```
# Overall
describe(df_messy1$log_rt) |>
  select(11,12)

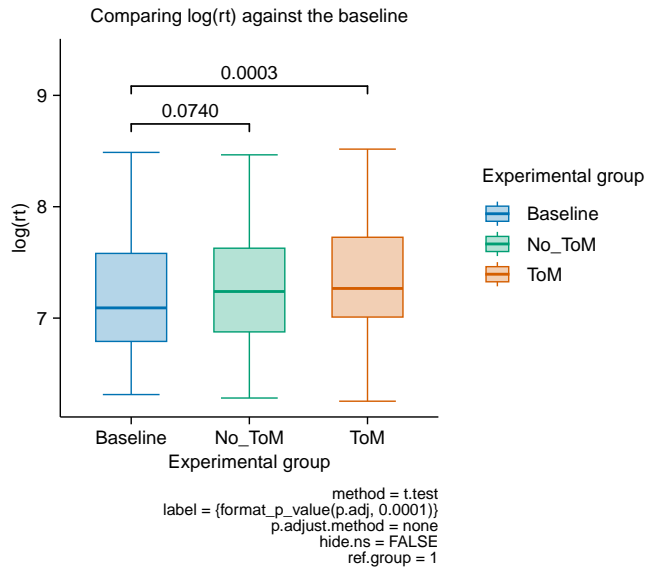
# By group
describeBy(log_rt ~ condition,
            mat = TRUE,
            data = df_messy1) |>
  select(2, 13:14)
```

- What is `select` doing here?
- What are the rules for violations to Skew and Kurtosis?

## 23 Boxplot with comparison analysis from tidyplots

```
df_messy1 |>
  tidyplot(x = condition,
            y = log_rt,
            color = condition) |>
  add_boxplot() |>
  add_test_pvalue(ref.group = 1) |>
  add_title("Comparing log(rt) against the baseline") |>
  adjust_x_axis_title("Experimental group") |>
```

```
adjust_y_axis_title("log(rt)") |>
adjust_legend_title("Experimental group")
```



## 24 Roundup

- I showed you how to deliberately untidy data using the **messy** package
- We did some label tidying
- We examined the case of listwise deletion, with your reading covering other approaches to missing data
- We examined test statistics for distribution normality
- Gave you a first look at **tidyplots** as an alternative to **ggplot2**

## 25 Cleanup

```
# Clear data
rm(list = ls()) # Removes all objects from environment

# Clear packages
p_unload(all) # Remove all contributed packages

# Clear plots
graphics.off() # Clears plots, closes all graphics devices
```

```
# Clear console  
cat("\014") # Mimics ctrl+L
```

## 26 References