Linear Models in R (M1–MIDO)

Lab Session 3 — Solutions

Henri PANJO

Table of contents

Dataset Overview: *data_pokemon.csv*

This dataset is adapted from a popular Kaggle Pokémon dataset.
Even if you are not familiar with Pokémon, the data is straightforward:
it combines numeric statistics with categorical attributes, making it well-suited for applying **Ordinary Least Squares (OLS)** in R.

What it contains

- Unique identifiers and names for each Pokémon
- Battle statistics (health, attack, defense, special attack, special defense, speed)
- Categorical features (primary/secondary type, generation, legendary flag)

Fields (Codebook)

- `id`: Unique Pokémon ID
- `name`: Pokémon name
- `type_1`: Primary type (e.g., Water, Fire)
- `type_2`: Secondary type (optional)
- `hp`: Hit points (overall health)
- `attack`: Physical attack strength (**we will use this as y** in most regressions)
- `defense`: Physical defense strength
- `sp_attack`: Special (non-physical) attack strength
- `sp_defense`: Special defense strength
- `speed`: Speed / turn order
- `generation`: Game generation label
- `legendary`: Indicator for legendary status (TRUE/FALSE)

Note on notation

- We treat `attack` as the outcome variable Y.
- Predictor variables (e.g., `defense`, `speed`) will be denoted as $x_1, x_2, \ldots$.
- Factors like `type_1` or `legendary` will be included as categorical predictors.

Setup

To keep numbers readable and reproducible, we set display options:

```r
options(scipen = 999, digits = 5)
```

We also load the packages used during this session.

> **Warning**
>
> Don't worry if you don't know them all — we'll introduce functions as we need them. Some provide regression tools, others are for data visualization or diagnostics.

```r
library(broom)
library(performance)
library(parameters)
library(datawizard)
library(see)
library(effectsize)
library(insight)
library(correlation)
library(modelbased)
library(glue)
library(scales)
library(GGally)
library(ggpubr)
library(car)
library(lmtest)
library(multcomp)
library(rstatix)
library(matrixTests)
library(ggfortify)
library(qqplotr)
library(patchwork)
library(gtsummary)
library(kableExtra)
library(openxlsx)
library(janitor)
library(marginaleffects)
library(collapse)
library(tidyverse)
```

```r
source("helper_functions3.R")
```

Question 1. Loading dataset

Import the dataset `data_pokemon.csv` with **`read_csv()`** and save it in an object called `pok`.
Using **`select()`**, keep only the variables `id`, `name`, `attack`, `speed`, `defense`, `hp`, `sp_attack`, and `sp_def`.

Display the first 10 rows of `pok` using **`head()`** or **`slice()`**.

Solutions

- Loading `data_pokemon.csv`

```
pok <- read_csv("data_pokemon.csv", show_col_types = FALSE)
```

- **`head()`** on `pok`

```
head(pok, n = 10)
```

```
# A tibble: 10 x 12
      id name      type_1 type_2    hp attack defense sp_attack sp_def speed generation legendary
   <dbl> <chr>     <chr>  <chr>  <dbl>  <dbl>   <dbl>     <dbl>  <dbl> <dbl>      <dbl> <chr>
 1     1 Bulbasaur Grass  Poison    45     49      49        65     65    45          1 No
 2     2 Ivysaur   Grass  Poison    60     62      63        80     80    60          1 No
 3     3 Venusaur  Grass  Poison    80     82      83       100    100    80          1 No
 4     4 Mega Ven~ Grass  Poison    80    100     123       122    120    80          1 No
 5     5 Charmand~ Fire   None      39     52      43        60     50    65          1 No
 6     6 Charmele~ Fire   None      58     64      58        80     65    80          1 No
 7     7 Charizard Fire   Flying    78     84      78       109     85   100          1 No
 8     8 Mega Cha~ Fire   Dragon    78    130     111       130     85   100          1 No
 9     9 Mega Cha~ Fire   Flying    78    104      78       159    115   100          1 No
10    10 Squirtle  Water  None      44     48      65        50     64    43          1 No
```

Question 2: Exploring categorical variables

The Pokémon dataset contains 4 categorical variables

- `type_1` : the primary type (always present)
- `type_2` : the secondary type (may be missing)
- `legendary`: if the pokemon is legendary of not.
- `generation`: the pokemon generation.

1. Create frequency tables for `type_1`, `type_2`, `legendary` and `generation`. Display both the counts and the relative proportions.

2. Produce bar plots for the distributions of `type_1` and `type_2`.
   - Make one bar plot for type_1 and one for type_2.
   - Ensure that categories on the x-axis are readable (e.g., rotate labels if necessary).

Solutions

- Frequency table for `type_1` with `count()` from **{dplyr}**

```
table_type1_prop <- mutate(pok, type_1 = fct_infreq(type_1)) |>
  count(type_1) |>
  mutate(pct = 100 * n / sum(n))

table_type1_prop
```

```
# A tibble: 18 x 3
   type_1       n   pct
   <fct>    <int> <dbl>
 1 Water      112 14
 2 Normal      98 12.2
 3 Grass       70  8.75
 4 Bug         69  8.62
 5 Psychic     57  7.12
 6 Fire        52  6.5
 7 Electric    44  5.5
 8 Rock        44  5.5
 9 Dragon      32  4
10 Ghost       32  4
11 Ground      32  4
12 Dark        31  3.88
13 Poison      28  3.5
14 Fighting    27  3.38
15 Steel       27  3.38
16 Ice         24  3
17 Fairy       17  2.12
18 Flying       4  0.5
```

- Frequency table for `type_2` with `tabyl()` from `{janitor}`

```
table_type2_prop <- tabyl(pok, type_2) |>
  adorn_pct_formatting(digits = 2) |>
  arrange(desc(n)) |>
  as_tibble()

table_type2_prop
```

```
# A tibble: 19 x 3
   type_2       n percent
   <chr>    <dbl> <chr>
 1 None       386 48.25%
 2 Flying      97 12.12%
 3 Ground      35 4.38%
 4 Poison      34 4.25%
 5 Psychic     33 4.12%
 6 Fighting    26 3.25%
 7 Grass       25 3.12%
 8 Fairy       23 2.88%
 9 Steel       22 2.75%
10 Dark        20 2.50%
11 Dragon      18 2.25%
12 Ghost       14 1.75%
13 Ice         14 1.75%
14 Rock        14 1.75%
15 Water       14 1.75%
16 Fire        12 1.50%
17 Electric     6 0.75%
18 Normal       4 0.50%
19 Bug          3 0.38%
```

- Frequency table for `legendary`

```
tabyl(pok, legendary) |>
  adorn_pct_formatting(digits = 2) |>
  as_tibble()
```

```
# A tibble: 2 x 3
  legendary     n percent
  <chr>     <dbl> <chr>
1 No          735 91.88%
2 Yes          65 8.12%
```

- Frequency table for `generation`

```
tabyl(pok, generation) |>
  adorn_pct_formatting(digits = 2) |>
  as_tibble()
```

```
# A tibble: 6 x 3
  generation     n percent
       <dbl> <dbl> <chr>
1          1   166 20.75%
2          2   106 13.25%
3          3   160 20.00%
4          4   121 15.12%
5          5   165 20.62%
6          6    82 10.25%
```

- Frequency tables for `type_1` and `type_2` with `tbl_summary()` from `{gtsummary}`

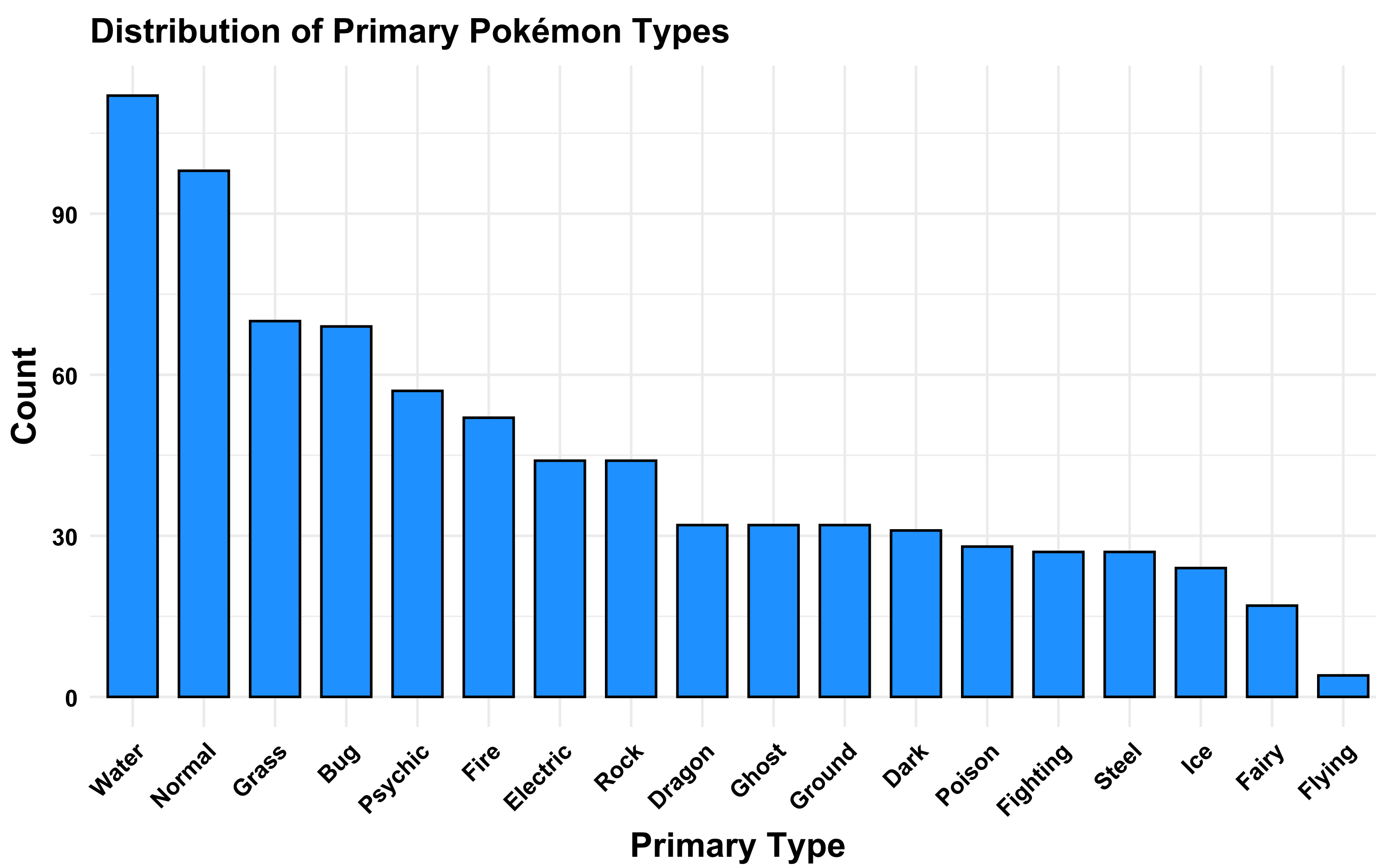
```
select(pok, type_1) |>
  mutate(type_1 = fct_infreq(type_1)) |>
  tbl_summary(label = list(type_1 = "Pokemon primary type")) |>
  modify_header(all_stat_cols() ~ "**{level} (n={n})**") |>
  bold_labels() |>
  remove_footnote_header()

select(pok, type_2) |>
  mutate(type_2 = fct_infreq(type_2)) |>
  tbl_summary(label = list(type_2 = "Pokemon secondary type")) |>
  modify_header(all_stat_cols() ~ "**{level} (n={n})**") |>
  bold_labels() |>
  remove_footnote_header()
```

Characteristic	**Overall (n=800)**
Pokemon primary type	
Water	112 (14%)
Normal	98 (12%)
Grass	70 (8.8%)
Bug	69 (8.6%)
Psychic	57 (7.1%)
Fire	52 (6.5%)
Electric	44 (5.5%)
Rock	44 (5.5%)
Dragon	32 (4.0%)
Ghost	32 (4.0%)
Ground	32 (4.0%)
Dark	31 (3.9%)
Poison	28 (3.5%)
Fighting	27 (3.4%)
Steel	27 (3.4%)
Ice	24 (3.0%)
Fairy	17 (2.1%)
Flying	4 (0.5%)

Characteristic	**Overall (n=800)**
Pokemon secondary type	
None	386 (48%)
Flying	97 (12%)
Ground	35 (4.4%)
Poison	34 (4.3%)
Psychic	33 (4.1%)
Fighting	26 (3.3%)
Grass	25 (3.1%)
Fairy	23 (2.9%)
Steel	22 (2.8%)
Dark	20 (2.5%)
Dragon	18 (2.3%)
Ghost	14 (1.8%)
Ice	14 (1.8%)
Rock	14 (1.8%)
Water	14 (1.8%)
Fire	12 (1.5%)
Electric	6 (0.8%)
Normal	4 (0.5%)
Bug	3 (0.4%)

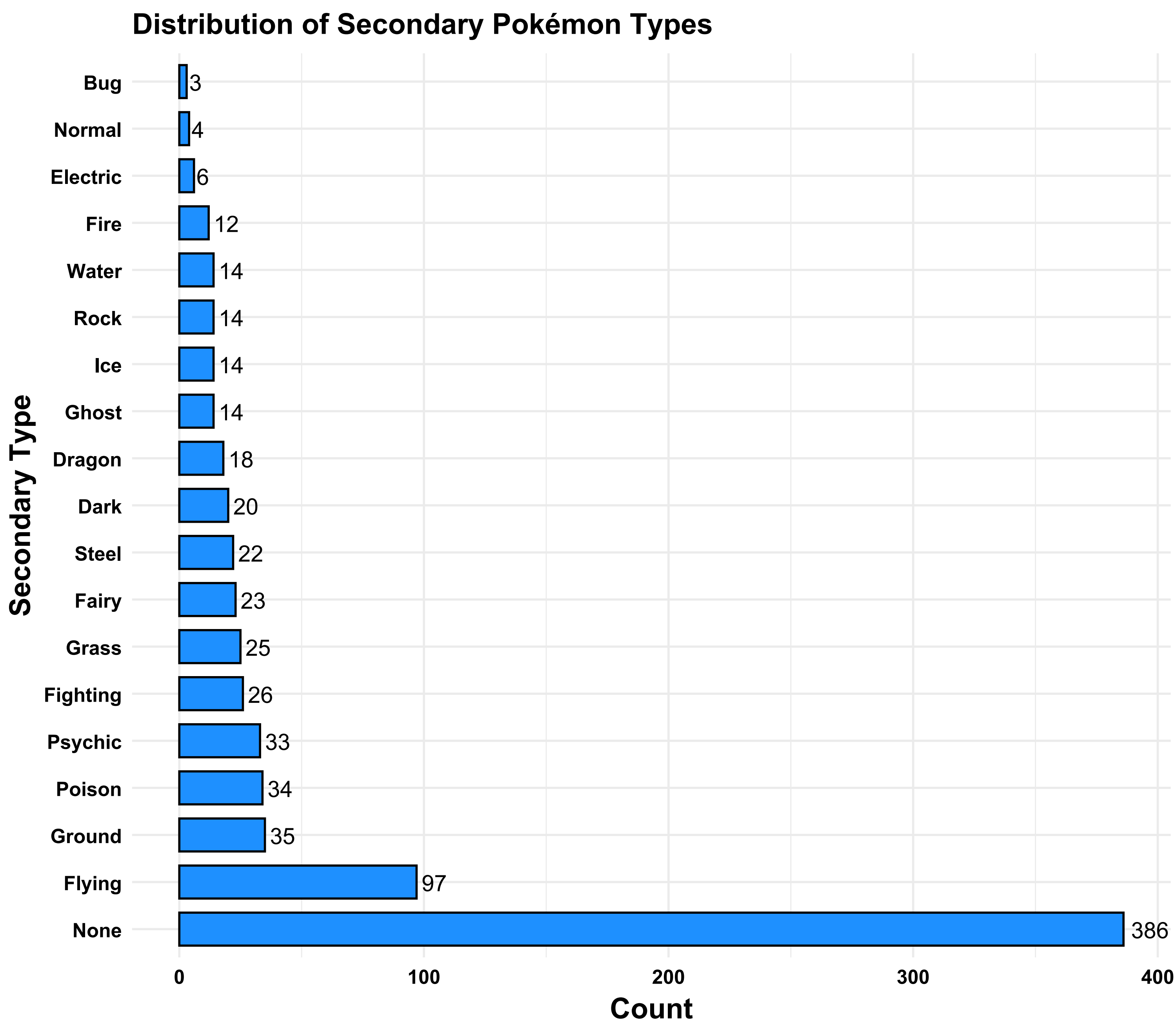
- Bar plots for the distribution of `type_1` with `geom_bar()`

```
pok |>
  mutate(type_1 = fct_infreq(type_1)) |>    # order by descending frequency
  ggplot(aes(x = type_1)) +
  geom_bar(fill = "dodgerblue", color = "black", width = 0.7) +
  labs(
    title = "Distribution of Primary Pokémon Types",
    x = "Primary Type",  y = "Count"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs_pubr()
```



- Bar plots for the distribution of `type_1` with `geom_col()`

```
pok |>
  mutate(type_2 = fct_infreq(type_2)) |>    # order by descending frequency
  count(type_2) |>  # get the count
  ggplot(aes(y = type_2, x = n, label = n)) +
  geom_col(fill = "dodgerblue", color = "black", width = 0.7) +
  geom_text(hjust = -0.2) +
  labs(
    title = "Distribution of Secondary Pokémon Types",
    x = "Count",  y = "Secondary Type"
  ) +
  theme_minimal() +
  labs_pubr()
```



Question 3: Data management, variable creation

The variable (`type_1`), has 18 levels, which can be too many to include directly in an OLS regression. To simplify the analysis, we want to group these 18 types into 3 broader, meaningful groups:

- 1.Elemental/Environmental: Fire, Water, Grass, Electric, Ice, Flying, Poison
- 2.Physical/Material: Bug, Fighting, Ground, Rock, Steel, Normal
- 3.Mystical/Supernatural: Psychic, Ghost, Dragon, Fairy, Dark

1. In the `pok` dataframe, create a new `factor` variable called `type_group3` that assigns each Pokémon to one of the 3 groups above based on its primary type (`type_1`).

2. Create a new binary variable called `has_secondary_type` defined as follows:
   - "Yes" if the Pokémon has a secondary type (`type_2` is not "None")
   - "No" if the Pokémon does not have a secondary type (`type_2` is "None")

3. Transform the variables `legendary` and `generation` into `factors`.

4. Verify that the new variables are well created

Solutions

Creation of the variable `type_group3` and `has_secondary_type`

- Define the 3-level grouping map

```
type_map3 <- list(
  elemental_env = c("Fire", "Water", "Grass", "Electric", "Ice", "Flying", "Poison"),
  physical_material = c("Bug", "Fighting", "Ground", "Rock", "Steel", "Normal"),
  mystical_supernatural = c("Psychic", "Ghost", "Dragon", "Fairy", "Dark")
)
```

- Create variables with `mutate()` from `{dplyr}`

```
pok <- mutate(
  pok,
  type_group3 = case_when(
    type_1 %in% type_map3$elemental_env ~ "1.Elemental/Environmental",
    type_1 %in% type_map3$physical_material ~ "2.Physical/Material",
    type_1 %in% type_map3$mystical_supernatural ~ "3.Mystical/Supernatural",
    .default = NA_character_
  ),
  has_secondary_type = ifelse(type_2 == "None", 0, 1) |> factor(labels = c("No", "Yes"))
) |>
  mutate(type_group3 = as.factor(type_group3), legendary = as.factor(legendary)) |>
  mutate(generation = factor(generation, labels = paste0("Gen", 1:6))) |>
  relabel(
    type_group3 = "Primary Type", has_secondary_type = "Has a secondary type",
    legendary = "Legendary", generation = "Pokemon generation"
  )
```

- `type_group3`: variable checking

```r
tabyl(pok, type_group3) |>
  as_tibble()
```

```
# A tibble: 3 x 3
  type_group3                     n percent
  <fct>                       <int>   <dbl>
1 1.Elemental/Environmental     334   0.418
2 2.Physical/Material           297   0.371
3 3.Mystical/Supernatural       169   0.211
```

```r
tabyl(pok, type_1, type_group3) |>
  as_tibble()
```

```
# A tibble: 18 x 4
   type_1   `1.Elemental/Environmental` `2.Physical/Material` `3.Mystical/Supernatural`
   <chr>                          <dbl>                 <dbl>                     <dbl>
 1 Bug                                0                    69                         0
 2 Dark                               0                     0                        31
 3 Dragon                             0                     0                        32
 4 Electric                          44                     0                         0
 5 Fairy                              0                     0                        17
 6 Fighting                           0                    27                         0
 7 Fire                              52                     0                         0
 8 Flying                             4                     0                         0
 9 Ghost                              0                     0                        32
10 Grass                             70                     0                         0
11 Ground                             0                    32                         0
12 Ice                               24                     0                         0
13 Normal                             0                    98                         0
14 Poison                            28                     0                         0
15 Psychic                            0                     0                        57
16 Rock                               0                    44                         0
17 Steel                              0                    27                         0
18 Water                            112                     0                         0
```

- `has_secondary_type`: variable checking

```r
tabyl(pok, has_secondary_type) |>
  as_tibble()
```

```
# A tibble: 2 x 3
  has_secondary_type     n percent
  <fct>              <int>   <dbl>
1 No                   386   0.482
2 Yes                  414   0.518
```

```r
tabyl(pok, type_2, has_secondary_type) |>
  as_tibble()
```

```
# A tibble: 19 x 3
   type_2      No   Yes
   <chr>    <dbl> <dbl>
 1 Bug          0     3
 2 Dark         0    20
 3 Dragon       0    18
 4 Electric     0     6
 5 Fairy        0    23
 6 Fighting     0    26
 7 Fire         0    12
 8 Flying       0    97
 9 Ghost        0    14
10 Grass        0    25
11 Ground       0    35
12 Ice          0    14
13 None       386     0
14 Normal       0     4
15 Poison       0    34
16 Psychic      0    33
17 Rock         0    14
18 Steel        0    22
19 Water        0    14
```

- Cross tabulation between `type_group3` and `has_secondary_type`

```r
tabyl(pok, type_group3, has_secondary_type) |>
  adorn_totals(where = c("row", "col")) |>
  adorn_percentages(denominator = "row") |> # or "col"
  adorn_pct_formatting(digits = 1) |>
  adorn_ns(position = "front") |>
  as_tibble()
```

```
# A tibble: 4 x 4
  type_group3                No          Yes         Total
  <fct>                      <chr>       <chr>       <chr>
1 1.Elemental/Environmental 177 (53.0%) 157 (47.0%) 334 (100.0%)
2 2.Physical/Material       125 (42.1%) 172 (57.9%) 297 (100.0%)
3 3.Mystical/Supernatural   84 (49.7%)  85 (50.3%)  169 (100.0%)
4 Total                     386 (48.2%) 414 (51.7%) 800 (100.0%)
```

- `legendary` and `generation` : variable checking

```
count(pok, legendary)
```

```
# A tibble: 2 x 2
  legendary     n
  <fct>     <int>
1 No          735
2 Yes          65
```

```
count(pok, generation)
```

```
# A tibble: 6 x 2
  generation     n
  <fct>      <int>
1 Gen1         166
2 Gen2         106
3 Gen3         160
4 Gen4         121
5 Gen5         165
6 Gen6          82
```

Question 4: Box plot

We now want to explore how the Pokémon attack distribution varies across several categorical variables in the dataset. Boxplots are useful for comparing the distribution of a numeric variable across groups.

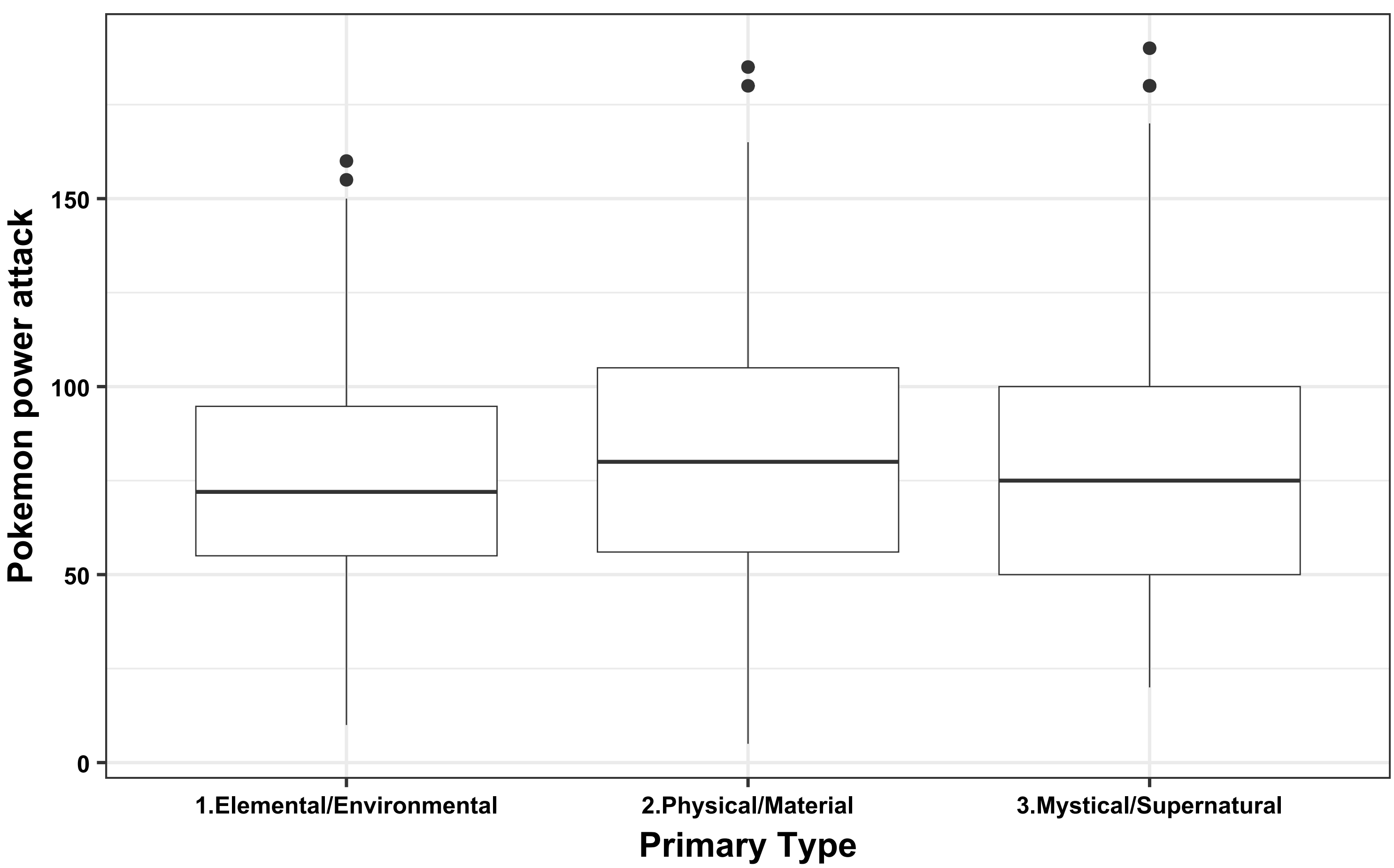
Using the Pokémon dataset, create four separate boxplots where the response variable is `attack`, and the grouping variables are `type_group3` , `has_secondary_type`, `legendary`, `generation`.

Produce the four boxplots either separately or arranged in a multi-panel layout (your choice).

Solutions

- Boxplots of `attack` by `type_group3`

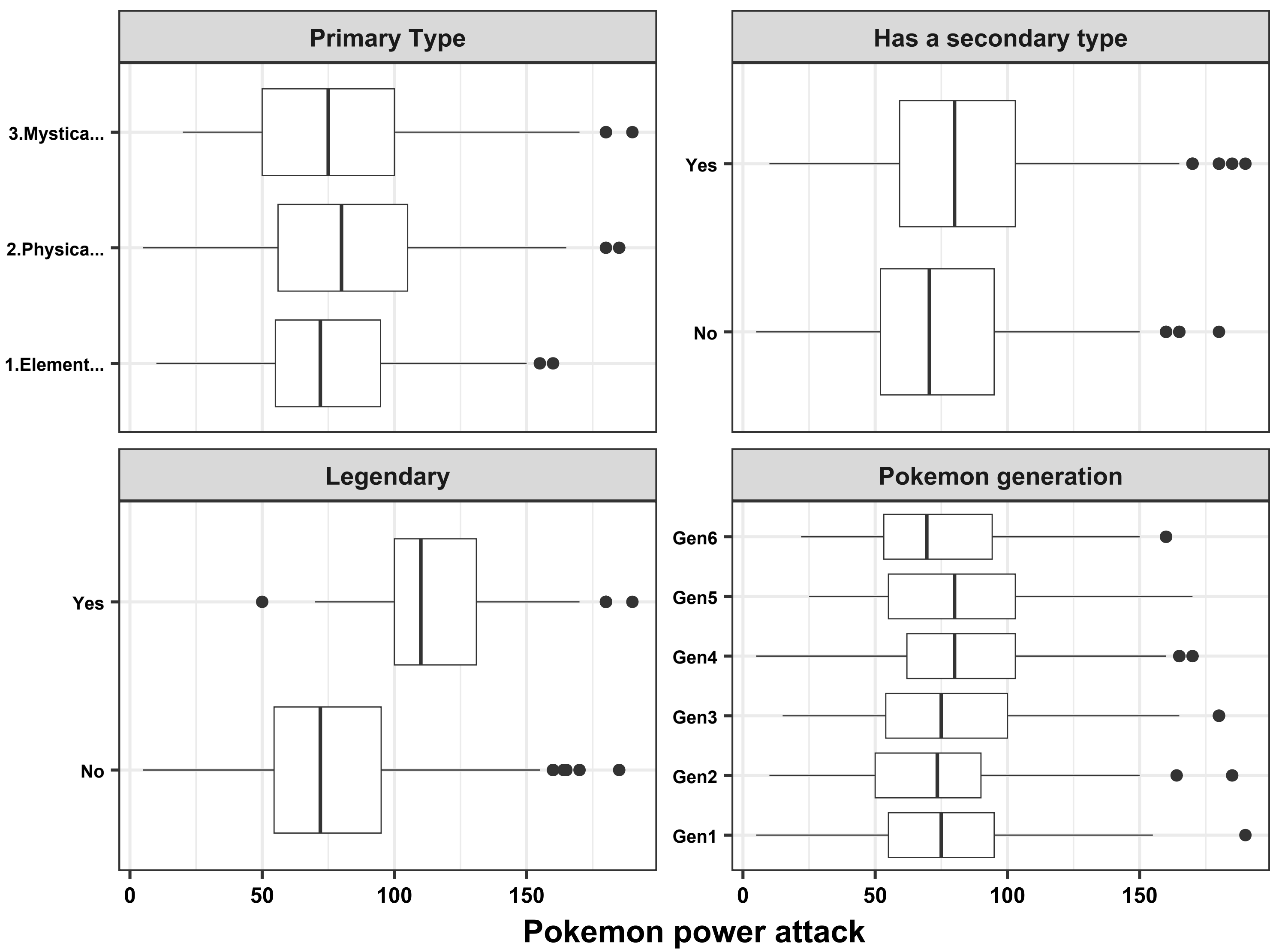
```
ggplot(pok, aes(x = type_group3, y = attack)) +
  geom_boxplot(linewidth = 0.25, median.linewidth = 0.75) +
  labs(y = "Pokemon power attack", x = vlabels(pok$type_group3)) +
  theme_bw(base_size = 14) +
  labs_pubr()
```



- Boxplots in a multi-panel layout

```
varcats <- c("type_group3", "has_secondary_type", "legendary", "generation")
```

```
boxplots <- select(pok, id, attack, all_of(varcats)) |>
  pivot_longer(all_of(varcats), names_to = "var") |>
  mutate(var = factor(var, levels = varcats, labels = vlabels(pok[, varcats]))) |>
  mutate(value = fct_relabel(value, \(x) str_trunc(x, 12))) |>
  ggplot(aes(x = attack, y = value)) +
  geom_boxplot(linewidth = 0.25, median.linewidth = 0.75) +
  facet_wrap(vars(var), scales = "free_y") +
  labs(x = "Pokemon power attack", y = NULL) +
  theme_bw(base_size = 14) +
  labs_pubr() +
  theme(
    strip.text = element_text(size = 11, face = "bold"),
    axis.text.y = element_text(size = 8, face = "bold")
  )

boxplots
```



Question 5: Attack mean over group variables

You have explored the distribution of the variable `attack` using boxplots. We now want to summarize these differences numerically by computing the mean Attack score for several grouping variables.

Using the updated Pokémon dataset, compute the mean value of `attack` for each level of the following categorical variables:

- `type_group3` (3-level grouped primary type)
- `has_secondary_type` ("No" / "Yes")
- `legendary` ("No" / "Yes")
- `generation` ("Gen1" to "Gen6")

For each variable, produce a summary table showing at least:

- the group name
- the mean of `attack`
- the standard deviation of `attack`
- the number of observations in each group

Hint: Use `mean_by_group()` from `helper_functions3.R`

Solutions

- We use `mean_by_group()`

```
mean_by_group(data = pok, x = "attack", by = "type_group3")
```

```
# A tibble: 4 x 3
  Variable                       N `Mean (SD)`
  <chr>                      <int> <chr>
1 Primary Type                  NA <NA>
2  1.Elemental/Environmental   334 74.9 (26.8)
3  2.Physical/Material         297 82.0 (34.4)
4  3.Mystical/Supernatural     169 81.7 (38.0)
```

```
mean_by_group(pok, "attack", "has_secondary_type")
```

```
# A tibble: 3 x 3
  Variable                 N `Mean (SD)`
  <chr>                <int> <chr>
1 Has a secondary type    NA <NA>
2  No                    386 74.5 (30.5)
3  Yes                   414 83.2 (33.7)
```

```r
mean_by_group(pok, "attack", "legendary")
```

```
# A tibble: 3 x 3
  Variable      N `Mean (SD)`
  <chr>     <int> <chr>
1 Legendary    NA <NA>
2  No         735 75.7 (30.5)
3  Yes         65 116.7 (30.3)
```

```r
mean_by_group(pok, "attack", "generation", digits = 2)
```

```
# A tibble: 7 x 3
  Variable               N `Mean (SD)`
  <chr>              <int> <chr>
1 Pokemon generation    NA <NA>
2  Gen1                166 76.64 (30.74)
3  Gen2                106 72.03 (32.71)
4  Gen3                160 81.63 (36.59)
5  Gen4                121 82.87 (32.78)
6  Gen5                165 82.07 (30.37)
7  Gen6                 82 75.80 (29.18)
```

- We can use `mean_by_group()` on different groups with `split()` from Base R and `map()` from `{purrr}`

```r
split(pok, ~legendary) |>
  map(\(d) mean_by_group(d, x = "attack", by = "type_group3", digits = 2)) |>
  list_rbind(names_to = "Legendary Pokemon")
```

```
# A tibble: 8 x 4
  `Legendary Pokemon` Variable                          N `Mean (SD)`
  <chr>               <chr>                         <int> <chr>
1 No                  Primary Type                     NA <NA>
2 No                   1.Elemental/Environmental      314 73.25 (26.25)
3 No                   2.Physical/Material            283 79.89 (33.20)
4 No                   3.Mystical/Supernatural        138 72.51 (32.82)
5 Yes                 Primary Type                     NA <NA>
6 Yes                  1.Elemental/Environmental       20 101.40 (21.08)
7 Yes                  2.Physical/Material             14 125.29 (30.59)
8 Yes                  3.Mystical/Supernatural         31 122.65 (32.42)
```

- All in one table using `map()` from `{purrr}`

```
tabmeans <- map(varcats, \(by) mean_by_group(pok, "attack", by)) |>
  list_rbind()

tabmeans |>
  kable(align = "lcc", padding = 2) |>
  row_spec(c(1, 5, 8, 11), bold = TRUE, color = "white", background = "black")
```

Variable	N	Mean (SD)
Primary Type		
1.Elemental/Environmental	334	74.9 (26.8)
2.Physical/Material	297	82.0 (34.4)
3.Mystical/Supernatural	169	81.7 (38.0)
Has a secondary type		
No	386	74.5 (30.5)
Yes	414	83.2 (33.7)
Legendary		
No	735	75.7 (30.5)
Yes	65	116.7 (30.3)
Pokemon generation		
Gen1	166	76.6 (30.7)
Gen2	106	72.0 (32.7)
Gen3	160	81.6 (36.6)
Gen4	121	82.9 (32.8)
Gen5	165	82.1 (30.4)
Gen6	82	75.8 (29.2)

- To export the table in Excel format, you can use `write.xlsx()` from `{openxlsx}`

```
write.xlsx(tabmeans, "tabmeans.xlsx")
```

Question 6: Dummy variables creation

In the Pokémon dataset create these variables

$$\texttt{legend1} = \begin{cases} 1 & \text{if } \texttt{legendary = "Yes"} \\ 0 & \text{otherwise} \end{cases} \qquad \texttt{legend0} = \begin{cases} 1 & \text{if } \texttt{legendary = "No"} \\ 0 & \text{otherwise} \end{cases}$$

$$\texttt{typeg1} = \begin{cases} 1 & \text{if } \texttt{type\_group3 = "1.Elemental/Environmental"} \\ 0 & \text{otherwise} \end{cases}$$

$$\texttt{typeg2} = \begin{cases} 1 & \text{if } \texttt{type\_group3 = "2.Physical/Material"} \\ 0 & \text{otherwise} \end{cases}$$

$$\texttt{typeg3} = \begin{cases} 1 & \text{if } \texttt{type\_group3 = "3.Mystical/Supernatural"} \\ 0 & \text{otherwise} \end{cases}$$

Hint: use `ifelse()` or any other functions/methods

Solutions

- We create the requested variables

```
pok <- pok |>
  mutate(
    legend1 = ifelse(legendary == "Yes", 1, 0), legend0 = ifelse(legendary == "No", 1, 0)
  ) |>
  mutate(typeg1 = 1 * (type_group3 == "1.Elemental/Environmental")) |>
  mutate(typeg2 = 1 * (type_group3 == levels(type_group3)[2])) |>
  mutate(typeg3 = 1 * (type_group3 == levels(type_group3)[3]))
```

- Quick check with `count()` from **{dplyr}**

```
count(pok, legendary, legend0, legend1) |> as.data.frame()
```

```
  legendary legend0 legend1   n
1        No       1       0 735
2       Yes       0       1  65
```

```
count(pok, type_group3, typeg1, typeg2, typeg3) |> as.data.frame()
```

```
                type_group3 typeg1 typeg2 typeg3   n
1 1.Elemental/Environmental      1      0      0 334
2       2.Physical/Material      0      1      0 297
3   3.Mystical/Supernatural      0      0      1 169
```

Question 7: Using dummy variables in OLS regression

You have created the dummy variables `legend1`, `legend0`, `typeg1`, `typeg2`, and `typeg3`, which encode information about whether a Pokémon is Legendary and which primary type-group it belongs to.

We now want to explore how these characteristics relate to the `attack` variable using simple and multiple linear regressions.

Using the Pokémon dataset and the dummy variables you created, estimate the following **three OLS regression models**, each with `attack` as the dependent variable:

$$\texttt{attack} = \beta_0 + \beta_1 \texttt{legend1} + \varepsilon \qquad \text{(Model 1)}$$

$$\texttt{attack} = \beta_0 + \beta_1 \texttt{typeg2} + \beta_2 \texttt{typeg3} + \varepsilon \qquad \text{(Model 2)}$$

$$\texttt{attack} = \beta_0 + \beta_1 \texttt{legend1} + \beta_2 \texttt{typeg2} + \beta_3 \texttt{typeg3} + \varepsilon \qquad \text{(Model 3)}$$

1. For each model, report the regression output and interpret the coefficients.
2. Compare the 3 models with `compare_performance()` from `{performance}`.
3. Refit the 3 models using the `factor` variables `legendary` and `type_group3`.

Solutions

- We fit the 3 models

```
mod1 <- lm(attack ~ legend1, data = pok)
```

- This specification implies that non-legendary Pokémon is the reference group

```
mod2 <- lm(attack ~ typeg2 + typeg3, data = pok)
```

- This specification implies that `1.Elemental/Environmental` Pokémon is the reference group

```
mod3 <- lm(attack ~ legend1 + typeg2 + typeg3, data = pok)
```

- This specification implies that non-legendary Pokémon and `1.Elemental/Environmental` are the reference groups

Model 1 with `model_parameters()` from `{parameters}`

```r
model_parameters(mod1, ci_method = "residual", digits = 1)
```

```
Parameter   | Coefficient |  SE |         95% CI | t(798) |      p
------------------------------------------------------------------
(Intercept) |        75.7 | 1.1 | [73.5, 77.9]   |   67.3 | < .001
legend1     |        41.0 | 3.9 | [33.3, 48.7]   |   10.4 | < .001
```

$\texttt{attack} = \beta_0 + \beta_1 \texttt{legend1} + \varepsilon$	
Profile	$\mathbb{E}(\texttt{attack} \mid \cdot)$
Non-legendary Pokémon	β_0
Legendary Pokémon	$\beta_0 + \beta_1$

- `Intercept`: $\hat{\beta}_0 = 75.7$: mean `attack` for non-legendary Pokémon.
- `legend1`: $\hat{\beta}_1 = 41.0$: difference in mean `attack` between legendary and non-legendary Pokémon is on average equal to 41.0 $(p < 0.001)$.

Model 2

```r
model_parameters(mod2, ci_method = "residual", digits = 1)
```

```
Parameter   | Coefficient |  SE |         95% CI | t(797) |      p
------------------------------------------------------------------
(Intercept) |        74.9 | 1.8 | [71.5, 78.4]   |   42.4 | < .001
typeg2      |         7.1 | 2.6 | [ 2.0, 12.1]   |    2.8 | 0.006
typeg3      |         6.8 | 3.1 | [ 0.8, 12.8]   |    2.2 | 0.027
```

$\texttt{attack} = \beta_0 + \beta_1 \texttt{typeg2} + \beta_2 \texttt{typeg3} + \varepsilon$	
Profile	$\mathbb{E}(\texttt{attack} \mid \cdot)$
1.Elemental/Environmental	β_0
2.Physical/Material	$\beta_0 + \beta_1$
3.Mystical/Supernatural	$\beta_0 + \beta_2$

- `Intercept`: $\hat{\beta}_0 = 74.9$: mean `attack` for Pokémon in the `1.Elemental/Environmental` group (reference).

- `typeg2`: $\hat{\beta}_1 = 7.1$: difference in mean `attack` between the `2.Physical/Material` group and the `1.Elemental/Environmental` group is on average equal to 7.1 $(p = 0.006)$.

- `typeg3`: $\hat{\beta}_2 = 6.8$: difference in mean `attack` between the `3.Mystical/Supernatural` group and the `1.Elemental/Environmental` group is on average equal to 6.8 $(p = 0.027)$.

Model 3

```r
model_parameters(mod3, ci_method = "residual", digits = 1)
```

```
Parameter   | Coefficient |  SE |        95% CI | t(796) |      p
-----------------------------------------------------------------
(Intercept) |        72.4 | 1.7 | [69.1, 75.7]  |   43.2 | < .001
legend1     |        41.8 | 4.0 | [34.0, 49.7]  |   10.5 | < .001
typeg2      |         7.6 | 2.4 | [ 2.9, 12.4]  |    3.2 | 0.002
typeg3      |         1.6 | 2.9 | [-4.1,  7.3]  |    0.5 | 0.583
```

$$\texttt{attack} = \beta_0 + \beta_1\texttt{legend1} + \beta_2\texttt{typeg2} + \beta_3\texttt{typeg3} + \varepsilon$$

Profile	$\mathbb{E}(\texttt{attack} \mid \cdot)$
Non-legendary, 1.Elemental/Environmental	β_0
Non-legendary, 2.Physical/Material	$\beta_0 + \beta_2$
Non-legendary, 3.Mystical/Supernatural	$\beta_0 + \beta_3$
Legendary, 1.Elemental/Environmental	$\beta_0 + \beta_1$
Legendary, 2.Physical/Material	$\beta_0 + \beta_1 + \beta_2$
Legendary, 3.Mystical/Supernatural	$\beta_0 + \beta_1 + \beta_3$

- `Intercept`: $\hat{\beta}_0 = 72.4$: mean `attack` for non-legendary Pokémon in the `1.Elemental/Environmental` group (reference category).

- `legend1`: $\hat{\beta}_1 = 41.8$: difference in mean `attack` between the legendary and non-legendary Pokémon is on average equal to 41.8 $(p < 0.001)$, **for all primary type**.

- `typeg2`: $\hat{\beta}_2 = 7.6$: difference in mean `attack` between the `2.Physical/Material` group and the `1.Elemental/Environmental` group is on is on average equal to 7.6 $(p = 0.002)$, **for all legendary status**.

- `typeg3`: $\hat{\beta}_3 = 1.6$: difference in mean `attack` between the `3.Mystical/Supernatural` group and the `1.Elemental/Environmental` group is on is on average equal to 1.6 $(p = 0.583)$, **for all legendary status**.

- We compare the 3 models

```r
compare_performance(
  mod1, mod2, mod3,
  metrics = c("AIC", "BIC", "R2", "R2_adj", "SIGMA", "RMSE")
)
```

```
# Comparison of Model Performance Indices

Name | Model |  AIC (weights) |  BIC (weights) |    R2 | R2 (adj.) |   RMSE |  Sigma
--------------------------------------------------------------------------------------
mod1 |    lm | 7741.5 (0.038) | 7755.6 (0.811) | 0.119 |     0.118 | 30.441 | 30.479
mod2 |    lm | 7836.1 (<.001) | 7854.9 (<.001) | 0.011 |     0.009 | 32.254 | 32.315
mod3 |    lm | 7735.1 (0.962) | 7758.5 (0.189) | 0.131 |     0.127 | 30.242 | 30.318
```

We perform nested F tests

- Is `mod3` better than `mod1` ?

```r
linearHypothesis(mod3, c("typeg2", "typeg3"))
```

```r
anova(mod1, mod3)
```

```
Analysis of Variance Table

Model 1: attack ~ legend1
Model 2: attack ~ legend1 + typeg2 + typeg3
  Res.Df    RSS Df Sum of Sq    F Pr(>F)
1    798 741307
2    796 731682  2      9624 5.24 0.0055 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Is `mod3` better than `mod2` ?

```r
anova(mod2, mod3)
```

```
Analysis of Variance Table

Model 1: attack ~ typeg2 + typeg3
Model 2: attack ~ legend1 + typeg2 + typeg3
  Res.Df    RSS Df Sum of Sq   F               Pr(>F)
1    797 832263
2    796 731682  1    100580 109 <0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- We refit the 3 models using the `factor` variables `legendary` and `type_group3`

```r
mod1bis <- lm(attack ~ legendary, data = pok)
model_parameters(mod1bis, digits = 1, include_reference = TRUE) |>
  format_table()
```

```
        Parameter Coefficient  SE        95% CI t(798)      p
1     (Intercept)        75.7 1.1 [73.5, 77.9]   67.3 < .001
2  legendary [No]         0.0
3 legendary [Yes]        41.0 3.9 [33.3, 48.7]   10.4 < .001
```

```r
mod2bis <- lm(attack ~ type_group3, data = pok)
model_parameters(mod2bis, digits = 1, include_reference = TRUE) |>
  format_table()
```

```
                                 Parameter Coefficient  SE        95% CI t(797)      p
1                              (Intercept)        74.9 1.8 [71.5, 78.4]   42.4 < .001
2 type group3 [1.Elemental/Environmental]         0.0
3       type group3 [2.Physical/Material]         7.1 2.6 [ 2.0, 12.1]    2.8 0.006
4   type group3 [3.Mystical/Supernatural]         6.8 3.1 [ 0.8, 12.8]    2.2 0.027
```

```r
mod3bis <- lm(attack ~ legendary + type_group3, data = pok)
model_parameters(mod3bis, digits = 1, include_reference = TRUE) |>
  format_table()
```

```
                                 Parameter Coefficient  SE        95% CI t(796)      p
1                              (Intercept)        72.4 1.7 [69.1, 75.7]   43.2 < .001
2                           legendary [No]         0.0
3                          legendary [Yes]        41.8 4.0 [34.0, 49.7]   10.5 < .001
4 type group3 [1.Elemental/Environmental]         0.0
5       type group3 [2.Physical/Material]         7.6 2.4 [ 2.9, 12.4]    3.2 0.002
6   type group3 [3.Mystical/Supernatural]         1.6 2.9 [-4.1,  7.3]    0.5 0.583
```

Using `factor` variables in the formula is better

- `Anova()` from `{car}` gives the global significance test of every variable in the model.

```r
Anova(mod3bis, type = 3)
```

```
Anova Table (Type III tests)

Response: attack
             Sum Sq  Df F value              Pr(>F)
(Intercept) 1716647   1 1867.55 <0.0000000000000002 ***
legendary    100580   1  109.42 <0.0000000000000002 ***
type_group3    9624   2    5.24              0.0055 **
Residuals    731682 796
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- You can change the reference group with `relevel()` or with `C()`. For example, if you want `3.Mystical/Supernatural` as the reference for primary type (`type_group3`)

```
lm(attack ~ legendary + relevel(type_group3, ref = 3), data = pok) |>
  model_parameters() |>
  format_table(select = "{estimate} [{ci}]|{p}", digits = 1)
```

```
                                          Parameter  Coefficient [CI]       p
1                                       (Intercept) 74.0 [69.2, 78.8] <0.001
2                                   legendary [Yes] 41.8 [34.0, 49.7] <0.001
3 relevel(type group3, ref = 3)1 Elemental/Environmental -1.6 [-7.3,  4.1]  0.583
4       relevel(type group3, ref = 3)2 Physical/Material  6.0 [ 0.2, 11.9]  0.043
```

```
lm(attack ~ legendary + C(type_group3, base = 3), data = pok) |>
  model_parameters() |>
  format_table(select = "{estimate} [{ci}]|{p}", digits = 1)
```

```
                                             Parameter  Coefficient [CI]       p
1                                          (Intercept) 74.0 [69.2, 78.8] <0.001
2                                      legendary [Yes] 41.8 [34.0, 49.7] <0.001
3 C(type group3, base = 3) [1.Elemental/Environmental] -1.6 [-7.3,  4.1]  0.583
4       C(type group3, base = 3) [2.Physical/Material]  6.0 [ 0.2, 11.9]  0.043
```

- We can also use the constraint $\sum_{j=1}^{J} \alpha_j = 0$ (course notation).

```
lm(attack ~ legendary + C(type_group3, contr = sum), data = pok) |>
  model_parameters() |>
  format_table(select = "{estimate} [{ci}]|{p}", digits = 1)
```

```
                        Parameter  Coefficient [CI]       p
1                     (Intercept) 75.5 [73.2, 77.8] <0.001
2                 legendary [Yes] 41.8 [34.0, 49.7] <0.001
3 C(type group3, contr = sum) [1] -3.1 [-6.0, -0.2]  0.038
4 C(type group3, contr = sum) [2]  4.6 [ 1.6,  7.5]  0.003
```

- See the course for the interpretation of coefficients. Some statisticians dislike this parameterization.

Question 8: Testing equality of coefficients

Consider OLS model (`mod3`):

$$\texttt{attack} = \beta_0 + \beta_1 \texttt{legend1} + \beta_2 \texttt{typeg2} + \beta_3 \texttt{typeg3} + \varepsilon$$

```
Parameter   | Coefficient |    SE |           95% CI | t(796) |      p
---------------------------------------------------------------------
(Intercept) |      72.435 | 1.676 | [69.145, 75.725] | 43.215 | < .001
legend1     |      41.840 | 4.000 | [33.988, 49.691] | 10.460 | < .001
typeg2      |       7.623 | 2.419 | [ 2.876, 12.371] |  3.152 | 0.002
typeg3      |       1.595 | 2.904 | [-4.106,  7.296] |  0.549 | 0.583
```

Perform the following test

$$H_0 : \beta_2 = \beta_3 \qquad \text{versus} \qquad H_1 : \beta_2 \neq \beta_3.$$

Solutions

- We can perform the test with `linearHypothesis()` from **{car}**

```
linearHypothesis(mod3, c("typeg2 = typeg3")) |> as_tibble()
```

```
# A tibble: 2 x 6
  Res.Df     RSS    Df `Sum of Sq`     F `Pr(>F)`
   <dbl>   <dbl> <dbl>       <dbl> <dbl>    <dbl>
1    797 735465.    NA          NA  NA       NA
2    796 731682.     1       3783.  4.12   0.0428
```

- **Conclusion**: β_2 is significantly different from β_3.
  There is difference in power between `2.Physical/Material` and `3.Mystical/Supernatural`.
  But we do not get the estimated difference and CI.

- We can perform the test with `glht()` from **{multcomp}**. It provides the estimated difference and CI

```
glht(mod3, linfct = c("typeg2 - typeg3 = 0")) |>
  model_parameters(digits = 3, verbose = FALSE)
```

```
# Fixed Effects

Parameter              | Coefficient |    SE |          95% CI | t(796) |     p
-------------------------------------------------------------------------------
typeg2 - typeg3 == 0 |       6.029 | 2.972 | [0.195, 11.862] |  2.029 | 0.043
```

- What if we use `factor` variables in the model ?

```
model_parameters(mod3bis, digits = 1, verbose = FALSE)
```

```
Parameter                             | Coefficient |  SE |       95% CI | t(796) |      p
--------------------------------------------------------------------------------------------
(Intercept)                           |        72.4 | 1.7 | [69.1, 75.7] |   43.2 | < .001
legendary [Yes]                       |        41.8 | 4.0 | [34.0, 49.7] |   10.5 | < .001
type group3 [2.Physical/Material]     |         7.6 | 2.4 | [ 2.9, 12.4] |    3.2 | 0.002
type group3 [3.Mystical/Supernatural] |         1.6 | 2.9 | [-4.1,  7.3] |    0.5 | 0.583
```

- Name of the model coefficients with `find_parameters()` from `{insight}`

```
name_param <- find_parameters(mod3bis, flatten = TRUE)
name_param
```

```
[1] "(Intercept)"                      "legendaryYes"
[3] "type_group32.Physical/Material"   "type_group33.Mystical/Supernatural"
```

- Test with `linearHypothesis()`

```
linearHypothesis(mod3bis, glue("{name_param[3]} = {name_param[4]}")) |>
  as.data.frame()
```

```
  Res.Df    RSS Df Sum of Sq      F   Pr(>F)
1    797 735465 NA        NA     NA       NA
2    796 731682  1      3783 4.1155 0.042824
```

- Test with `glht()`. With define the contrast vector C.

```
C <- rbind("beta2 - beta3" = c(0, 0, 1, -1))
C
```

```
              [,1] [,2] [,3] [,4]
beta2 - beta3    0    0    1   -1
```

```
glht(mod3bis, linfct = C) |>
  model_parameters(digits = 3, verbose = FALSE)
```

```
# Fixed Effects

Parameter          | Coefficient |    SE |          95% CI | t(796) |     p
---------------------------------------------------------------------------
beta2 - beta3 == 0 |       6.029 | 2.972 | [0.195, 11.862] |  2.029 | 0.043
```

Question 9: Predicting mean `attack` for all category combinations

Consider the following regression model estimated using the Pokémon dataset (`mod3bis`):

$$\texttt{attack} = \beta_0 + \beta_1 \texttt{legendary}_{\texttt{Yes}} + \beta_2 \texttt{type\_group3}_{\texttt{2.Physical/Material}} + \beta_3 \texttt{type\_group3}_{\texttt{3.Mystical/Supernatural}} + \varepsilon$$

```
Parameter                              | Coefficient |  SE |        95% CI | t(796) |      p
----------------------------------------------------------------------------------------------
(Intercept)                            |       72.4 | 1.7 | [69.1, 75.7] |   43.2 | < .001
legendary [Yes]                        |       41.8 | 4.0 | [34.0, 49.7] |   10.5 | < .001
type group3 [2.Physical/Material]      |        7.6 | 2.4 | [ 2.9, 12.4] |    3.2 | 0.002
type group3 [3.Mystical/Supernatural]  |        1.6 | 2.9 | [-4.1,  7.3] |    0.5 | 0.583
```

We want to compute the **predicted mean Attack** for every combination of these two variables.

1. Create a prediction grid containing all $2 \times 3 = 6$ combinations of
   - `legendary` $\in$ {Yes, No}
   - `type_group3` $\in$ {1.Elemental/Environmental, 2.Physical/Material, 3.Mystical/Supernatural}

2. Compute the predicted mean of `attack` for each of the six combinations.

3. Present the results in a table showing:
   - the combination of categories
   - the predicted mean Attack with standard error or 95% confidence interval

Solutions

- Create all $2 \times 3 = 6$ combinations of `legendary` and `type_group3`

```
newdata_grid <- expand_grid(
  legendary = fct_unique(pok$legendary),
  type_group3 = fct_unique(pok$type_group3)
) |>
  arrange(legendary, type_group3)

newdata_grid
```

```
# A tibble: 6 x 2
  legendary type_group3
  <fct>     <fct>
1 No        1.Elemental/Environmental
2 No        2.Physical/Material
3 No        3.Mystical/Supernatural
4 Yes       1.Elemental/Environmental
5 Yes       2.Physical/Material
6 Yes       3.Mystical/Supernatural
```

- Get predictions with standard errors and CI

```r
predictions <- predict(
  mod3bis,
  newdata = newdata_grid, se.fit = TRUE, interval = "confidence"
)

pred_means_tidy <- bind_cols(newdata_grid, predictions[["fit"]]) |>
  mutate(across(is.numeric, \(x) style_number(x, digits = 1))) |>
  mutate("Mean [95% CI]" = glue("{fit} [{lwr}, {upr}]")) |>
  select(Legendary = legendary, "Primary Type" = type_group3, "Mean [95% CI]")
```

```r
pred_means_tidy
```

```
# A tibble: 6 x 3
  Legendary `Primary Type`             `Mean [95% CI]`
  <fct>     <fct>                      <glue>
1 No        1.Elemental/Environmental 72.4 [69.1, 75.7]
2 No        2.Physical/Material       80.1 [76.6, 83.5]
3 No        3.Mystical/Supernatural   74.0 [69.2, 78.8]
4 Yes       1.Elemental/Environmental 114.3 [106.2, 122.3]
5 Yes       2.Physical/Material       121.9 [113.7, 130.1]
6 Yes       3.Mystical/Supernatural   115.9 [108.0, 123.7]
```

- With `estimate_expectation()` from **{modelbased}**

```r
estimate_expectation(mod3bis, data = newdata_grid, ci = 0.95) |>
  rename(Legendary = 1, "Primary Type" = type_group3)
```

```r
means <- estimate_expectation(mod3bis, by = c("type_group3", "legendary"), ci = 0.95) |>
  rename(Legendary = 1, "Primary Type" = type_group3)

means
```

```
Model-based Predictions

Primary Type              | legendary | Predicted |   SE |           95% CI
---------------------------------------------------------------------------
1.Elemental/Environmental | No        |     72.43 | 1.68 | [ 69.14,  75.72]
2.Physical/Material       | No        |     80.06 | 1.77 | [ 76.58,  83.53]
3.Mystical/Supernatural   | No        |     74.03 | 2.44 | [ 69.23,  78.83]
1.Elemental/Environmental | Yes       |    114.27 | 4.11 | [106.21, 122.34]
2.Physical/Material       | Yes       |    121.90 | 4.20 | [113.66, 130.14]
3.Mystical/Supernatural   | Yes       |    115.87 | 4.01 | [107.99, 123.75]

Variable predicted: attack
Predictors modulated: type_group3, legendary
```

- Contrast analysis can be achieved through `estimate_contrasts()` from `{modelbased}`

```r
predictors <- c("legendary", "type_group3")
```

```r
mod3bis |>
  estimate_contrasts(contrast = predictors, comparison = "pairwise", p_adjust = "bonferroni") |>
  format_table(select = "{estimate} [{ci}]|{p}", digits = 1) |>
  kable(format = "pipe", align = "l")
```

Level1	Level2	Difference [CI]	p
No, 2.Physical/Material	No, 1.Elemental/Environmental	7.6 [2.9, 12.4]	0.025
No, 3.Mystical/Supernatural	No, 1.Elemental/Environmental	1.6 [-4.1, 7.3]	> .999
Yes, 1.Elemental/Environmental	No, 1.Elemental/Environmental	41.8 [34.0, 49.7]	<0.001
Yes, 2.Physical/Material	No, 1.Elemental/Environmental	49.5 [40.2, 58.7]	<0.001
Yes, 3.Mystical/Supernatural	No, 1.Elemental/Environmental	43.4 [34.6, 52.3]	<0.001
No, 3.Mystical/Supernatural	No, 2.Physical/Material	-6.0 [-11.9, -0.2]	0.642
Yes, 1.Elemental/Environmental	No, 2.Physical/Material	34.2 [25.1, 43.3]	<0.001
Yes, 2.Physical/Material	No, 2.Physical/Material	41.8 [34.0, 49.7]	<0.001
Yes, 3.Mystical/Supernatural	No, 2.Physical/Material	35.8 [26.9, 44.7]	<0.001
Yes, 1.Elemental/Environmental	No, 3.Mystical/Supernatural	40.2 [29.8, 50.7]	<0.001
Yes, 2.Physical/Material	No, 3.Mystical/Supernatural	47.9 [37.3, 58.5]	<0.001
Yes, 3.Mystical/Supernatural	No, 3.Mystical/Supernatural	41.8 [34.0, 49.7]	<0.001
Yes, 2.Physical/Material	Yes, 1.Elemental/Environmental	7.6 [2.9, 12.4]	0.025
Yes, 3.Mystical/Supernatural	Yes, 1.Elemental/Environmental	1.6 [-4.1, 7.3]	> .999
Yes, 3.Mystical/Supernatural	Yes, 2.Physical/Material	-6.0 [-11.9, -0.2]	0.642

Question 10. Residual diagnostics

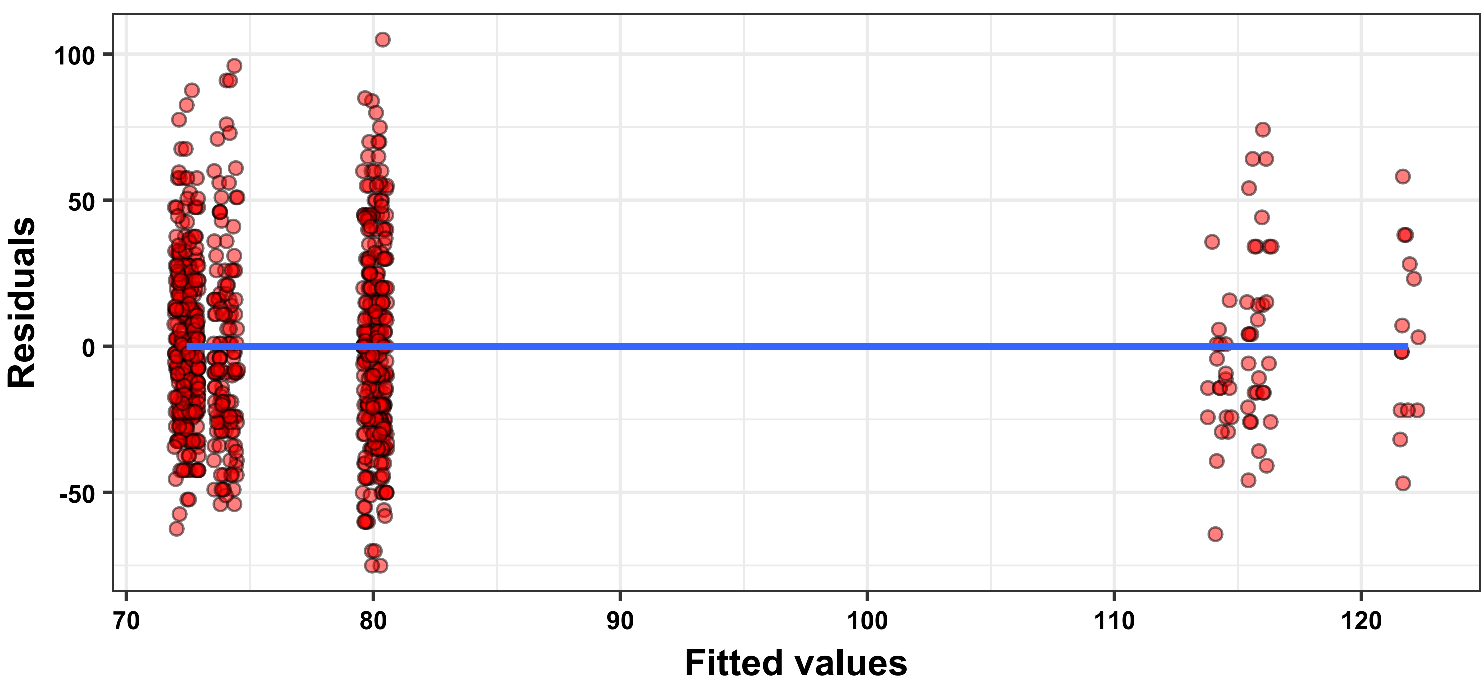
Using `mod3bis` and functions from the file `helper_functions3.R`:

1. Plot residuals vs fitted values and vs each predictor.
2. Plot $\sqrt{|\text{Standardized residuals}|}$ vs fitted values and vs each predictor.
3. Plot studentized residuals vs fitted values and vs each predictor.
4. Plot residuals vs order of observation.
5. Plot a histogram and normal Q-Q plot of the standardized residuals.

Solutions

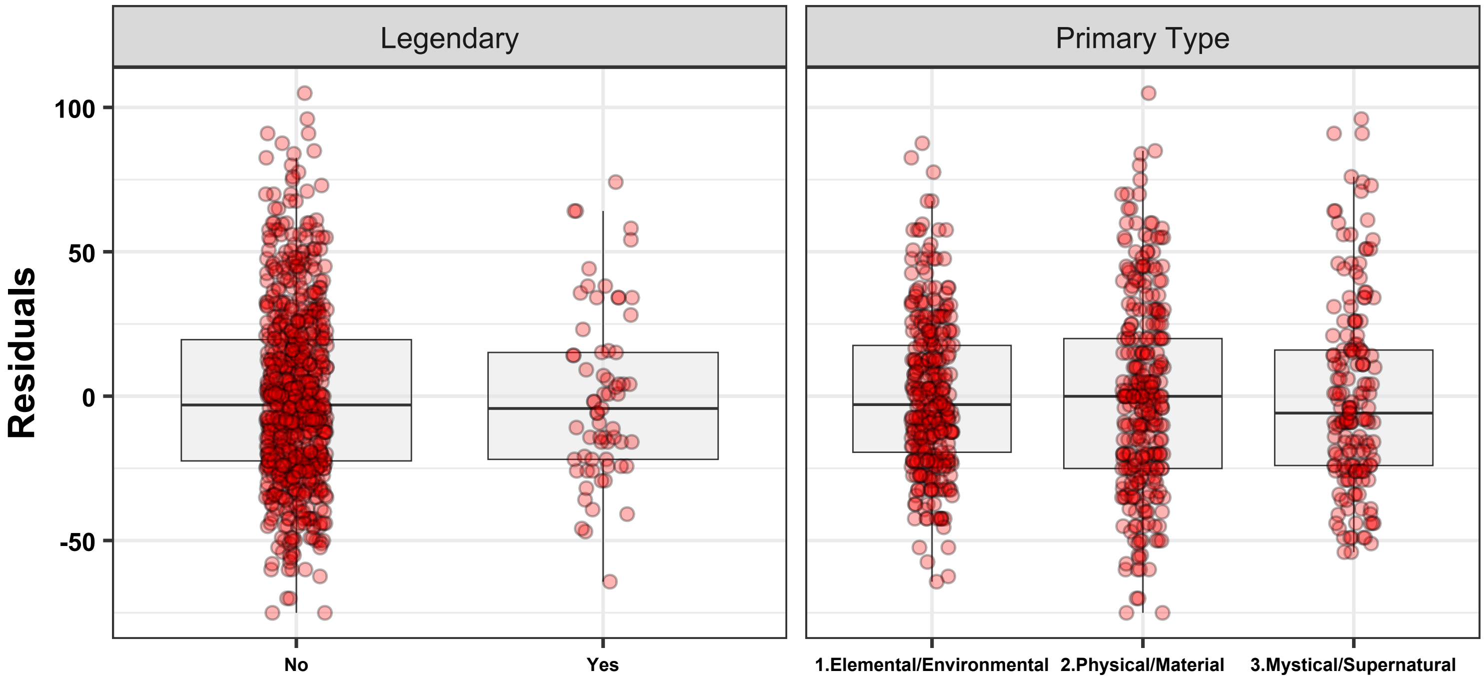
- Residuals vs fitted

```
resid_vs_fit(mod3bis, which = "res", jitter = TRUE)
```



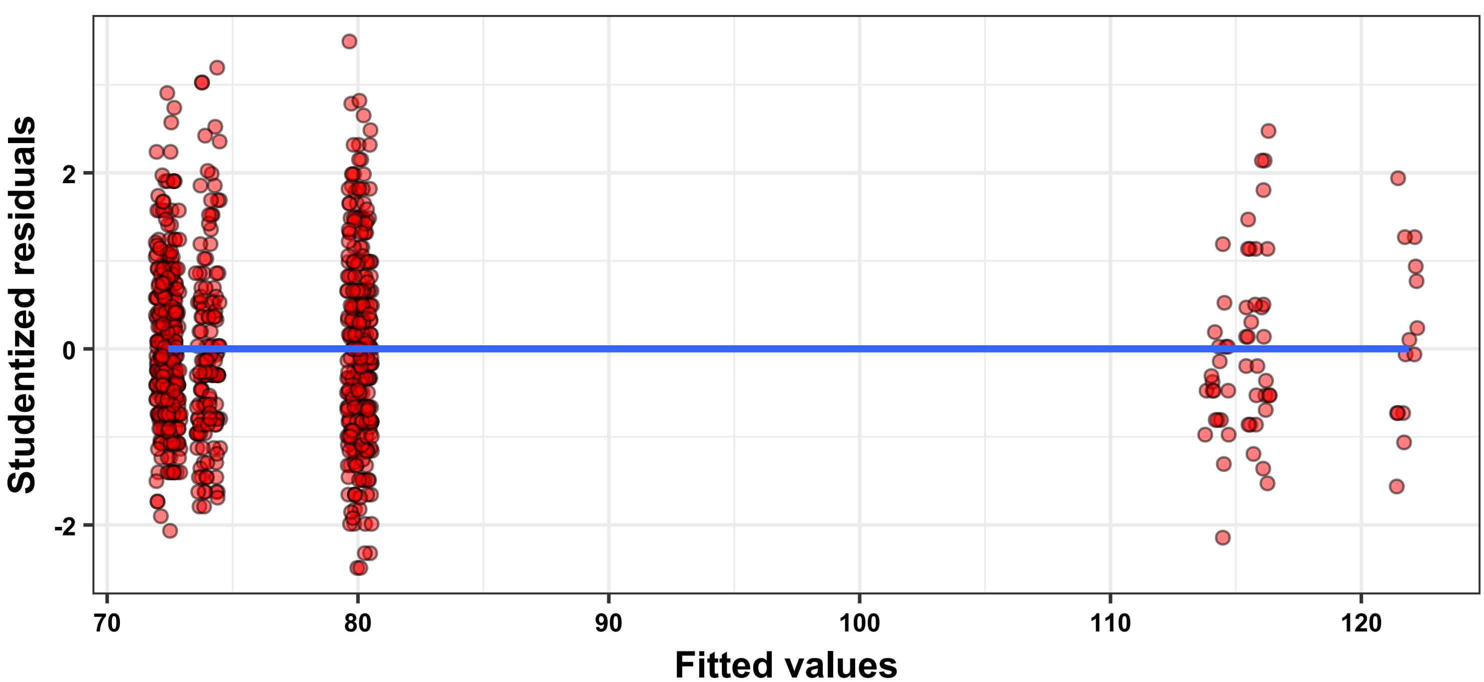
- Residuals vs predictors

```r
resid_vs_var_factor(mod3bis, predictors, which = "res", fill = "red") +
  theme(axis.text.x = element_text(size = 7))
```



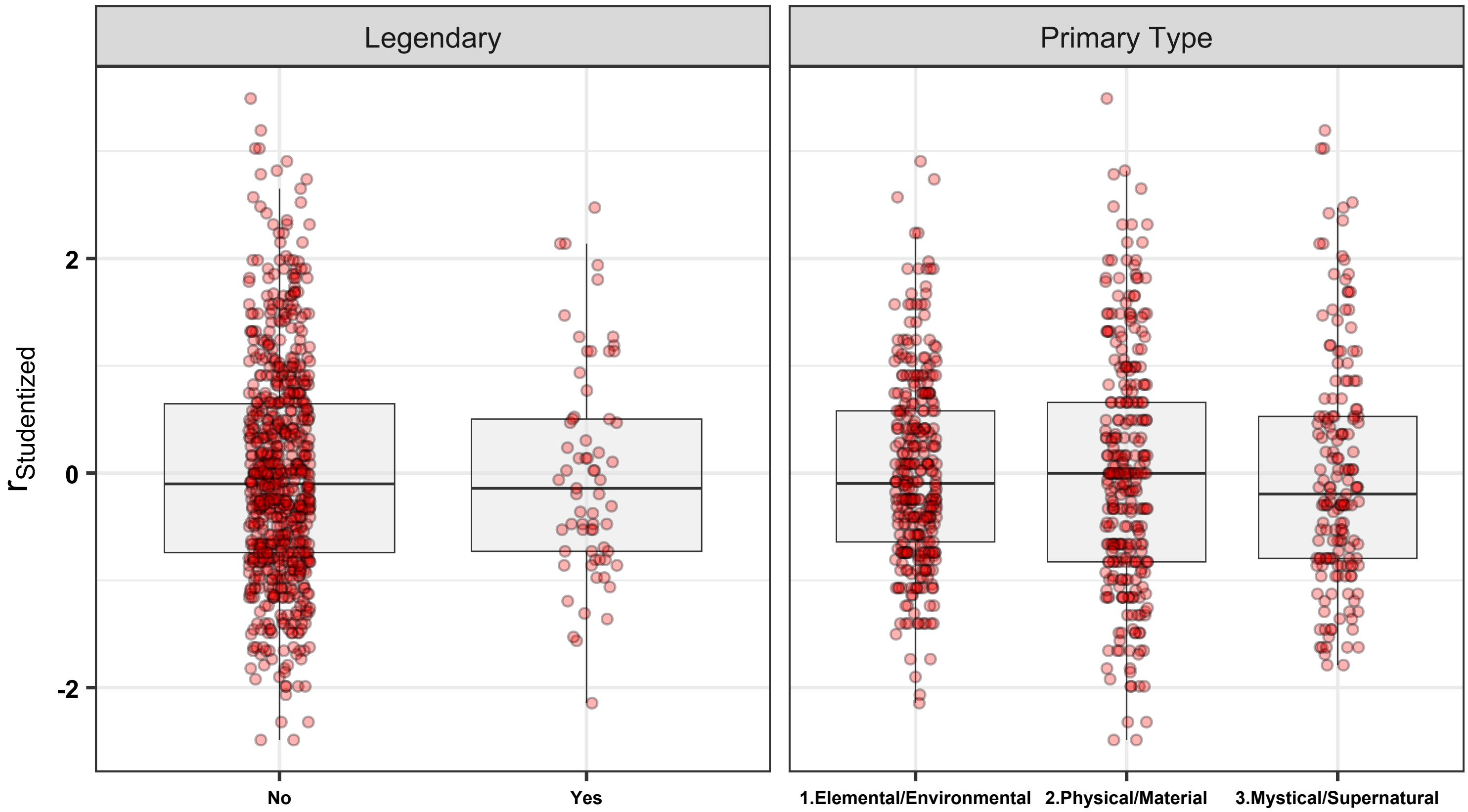
- Studentized residuals vs fitted values

```r
resid_vs_fit(mod3bis, which = "rstud", jitter = TRUE)
```



- Residuals vs predictors

```
resid_vs_var_factor(mod3bis, predictors, which = "rstud", fill = "red", size = 1.5) +
  theme(axis.text.x = element_text(size = 7))
```

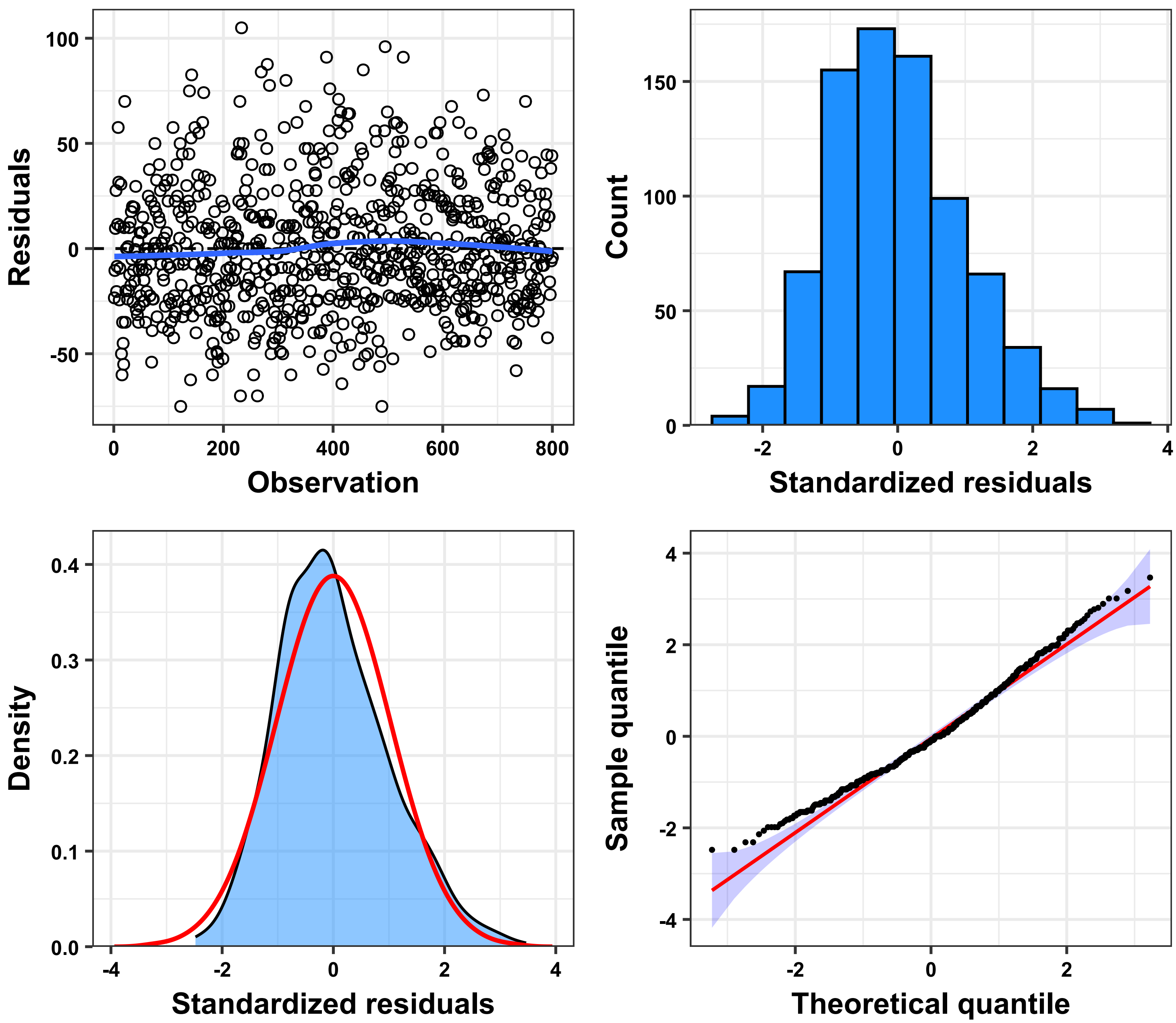


- Residuals vs observation order. Histogram, Density Q–Q plot of standardized residuals

```
p1 <- resid_vs_order(mod3bis)

p2 <- resid_stand_hist(mod3bis)

p3 <- resid_stand_dens(mod3bis)

p4 <- resid_stand_qq(mod3bis)
```

- We combine the 4 plots with the help of `+` from `{patchwork}`

```
p1 + p2 + p3 + p4
```



Session Info

Package	Version
broom	1.0.10
car	3.1-3
collapse	2.1.4
correlation	0.8.8
datawizard	1.3.0
effectsize	1.0.1
GGally	2.4.0
ggfortify	0.4.19
ggpubr	0.6.2
glue	1.8.0
gtsummary	2.4.0
insight	1.4.2
janitor	2.2.1
kableExtra	1.4.0
lmtest	0.9-40
marginaleffects	0.30.0
matrixTests	0.2.3.1
modelbased	0.13.0
multcomp	1.4-29
openxlsx	4.2.8
parameters	0.28.2
patchwork	1.3.2
performance	0.15.2
qqplotr	0.0.7
rstatix	0.7.3
scales	1.4.0
see	0.12.0
tidyverse	2.0.0