Linear Models in R (M1–MIDO)

Lab Session 1 — 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.

> **Note**
>
> 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(rstatix)
library(matrixTests)
library(ggfortify)
library(qqplotr)
library(collapse)
library(tidyverse)
```

Question 1. Loading dataset

Import the `data_pokemon.csv` file with `read_csv()`. Save the data in an object called `pok`.

- Quickly examine the data using `glimpse()` from `{dplyr}`
- Display the first 10 rows of `pok` using `head()` or `slice()`.

Solutions

- Loading `data_pokemon.csv`

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

- `glimpse()` on `pok`

```r
glimpse(pok)
```

```
Rows: 800
Columns: 12
$ id         <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2~
$ name       <chr> "Bulbasaur", "Ivysaur", "Venusaur", "Mega Venusaur", "Charmander", "Charmele~
$ type_1     <chr> "Grass", "Grass", "Grass", "Grass", "Fire", "Fire", "Fire", "Fire", "Fire", ~
$ type_2     <chr> "Poison", "Poison", "Poison", "Poison", "None", "None", "Flying", "Dragon", ~
$ hp         <dbl> 45, 60, 80, 80, 39, 58, 78, 78, 78, 44, 59, 79, 79, 45, 50, 60, 40, 45, 65, ~
$ attack     <dbl> 49, 62, 82, 100, 52, 64, 84, 130, 104, 48, 63, 83, 103, 30, 20, 45, 35, 25, ~
$ defense    <dbl> 49, 63, 83, 123, 43, 58, 78, 111, 78, 65, 80, 100, 120, 35, 55, 50, 30, 50, ~
$ sp_attack  <dbl> 65, 80, 100, 122, 60, 80, 109, 130, 159, 50, 65, 85, 135, 20, 25, 90, 20, 25~
$ sp_def     <dbl> 65, 80, 100, 120, 50, 65, 85, 85, 115, 64, 80, 105, 115, 20, 25, 80, 20, 25,~
$ speed      <dbl> 45, 60, 80, 80, 65, 80, 100, 100, 100, 43, 58, 78, 78, 45, 30, 70, 50, 35, 7~
$ generation <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
$ legendary  <chr> "No", "No", "No", "No", "No", "No", "No", "No", "No", "No", "No", "No", "No"~
```

- `head()` and `slice()` on pok

```r
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
```

```r
slice(pok, 1:10)
```

```
# 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. Summary statistics

For the variables `attack`, `speed`, `defense`, `hp`, compute summary statistics: number of missing values, number of distinct values, mean, median, and standard deviation.

Hint: `summary()`, `descr()`, `describe_distribution()`, `get_summary_stats()` `summarise()`, `mean()`, `sd()`, `median()`, `n_distinct()`, `is.na()`

Solutions

- `summary()` on selected variables `attack`, `speed`, `defense`, `hp`

```
select(pok, attack, speed, defense, hp) |>
  summary()
```

```
     attack          speed           defense             hp       
 Min.   :  5   Min.   :  5.0   Min.   :  5.0   Min.   :  1.0  
 1st Qu.: 55   1st Qu.: 45.0   1st Qu.: 50.0   1st Qu.: 50.0  
 Median : 75   Median : 65.0   Median : 70.0   Median : 65.0  
 Mean   : 79   Mean   : 68.3   Mean   : 73.8   Mean   : 69.3  
 3rd Qu.:100   3rd Qu.: 90.0   3rd Qu.: 90.0   3rd Qu.: 80.0  
 Max.   :190   Max.   :180.0   Max.   :230.0   Max.   :255.0  
```

- `descr()` (`{collapse}`)

```
select(pok, attack, speed, defense, hp) |>
  descr(Ndistinct = TRUE, Qprobs = c(0.25, 0.5, 0.75)) |>
  as_tibble()
```

```
# A tibble: 4 x 13
  Variable Class       N Ndist  Mean    SD   Min   Max  Skew  Kurt `25%` `50%` `75%`
  <chr>    <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 attack   numeric   800   111  79.0  32.5     5   190 0.551  3.16    55    75   100
2 speed    numeric   800   108  68.3  29.1     5   180 0.357  2.76    45    65    90
3 defense  numeric   800   103  73.8  31.2     5   230 1.15   5.70    50    70    90
4 hp       numeric   800    94  69.3  25.5     1   255 1.57  10.2     50    65    80
```

- `describe_distribution()` (`{datawizard}`)

```r
select(pok, attack, speed, defense, hp) |>
  describe_distribution(centrality = c("mean", "median"), quartiles = TRUE) |>
  as_tibble()
```

```
# A tibble: 4 x 14
  Variable Median   MAD  Mean    SD   IQR   Min   Max    Q1    Q3 Skewness Kurtosis     n
  <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl> <int>
1 attack       75  29.7  79.0  32.5    45     5   190    55   100    0.552    0.170   800
2 speed        65  31.1  68.3  29.1    45     5   180    45    90    0.358   -0.236   800
3 defense      70  29.7  73.8  31.2    40     5   230    50    90    1.16     2.73    800
4 hp           65  22.2  69.3  25.5    30     1   255    50    80    1.57     7.23    800
```

- `get_summary_stats()` (`{rstatix}`)

```r
select(pok, attack, speed, defense, hp) |>
  get_summary_stats(show = c("n", "mean", "sd", "median", "q1", "q3"))
```

```
# A tibble: 4 x 7
  variable     n  mean    sd median    q1    q3
  <fct>    <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
1 attack     800  79.0  32.5     75    55   100
2 speed      800  68.3  29.1     65    45    90
3 defense    800  73.8  31.2     70    50    90
4 hp         800  69.3  25.5     65    50    80
```

- `summarise()` (`{dplyr}`). More complicated
- First we create a list of functions

```r
myfunctions <- list(
  n = length, nmiss = \(x) sum(is.na(x)), ndistinct = n_distinct,
  mean = mean, sd = sd, median = median
)
```

- We use `summarise()` with `across()` and `myfunctions`. Then we apply `pivot_longer()` (`{tidyr}`)

```
pok |>
  summarise(across(c("attack", "speed", "defense", "hp"), myfunctions)) |>
  pivot_longer(
    cols = everything(),
    names_to = c("Variable", ".value"),
    names_sep = "_"
  )
```

```
# A tibble: 4 x 7
  Variable     n nmiss ndistinct  mean    sd median
  <chr>    <int> <int>     <int> <dbl> <dbl>  <dbl>
1 attack     800     0       111  79.0  32.5     75
2 speed      800     0       108  68.3  29.1     65
3 defense    800     0       103  73.8  31.2     70
4 hp         800     0        94  69.3  25.5     65
```

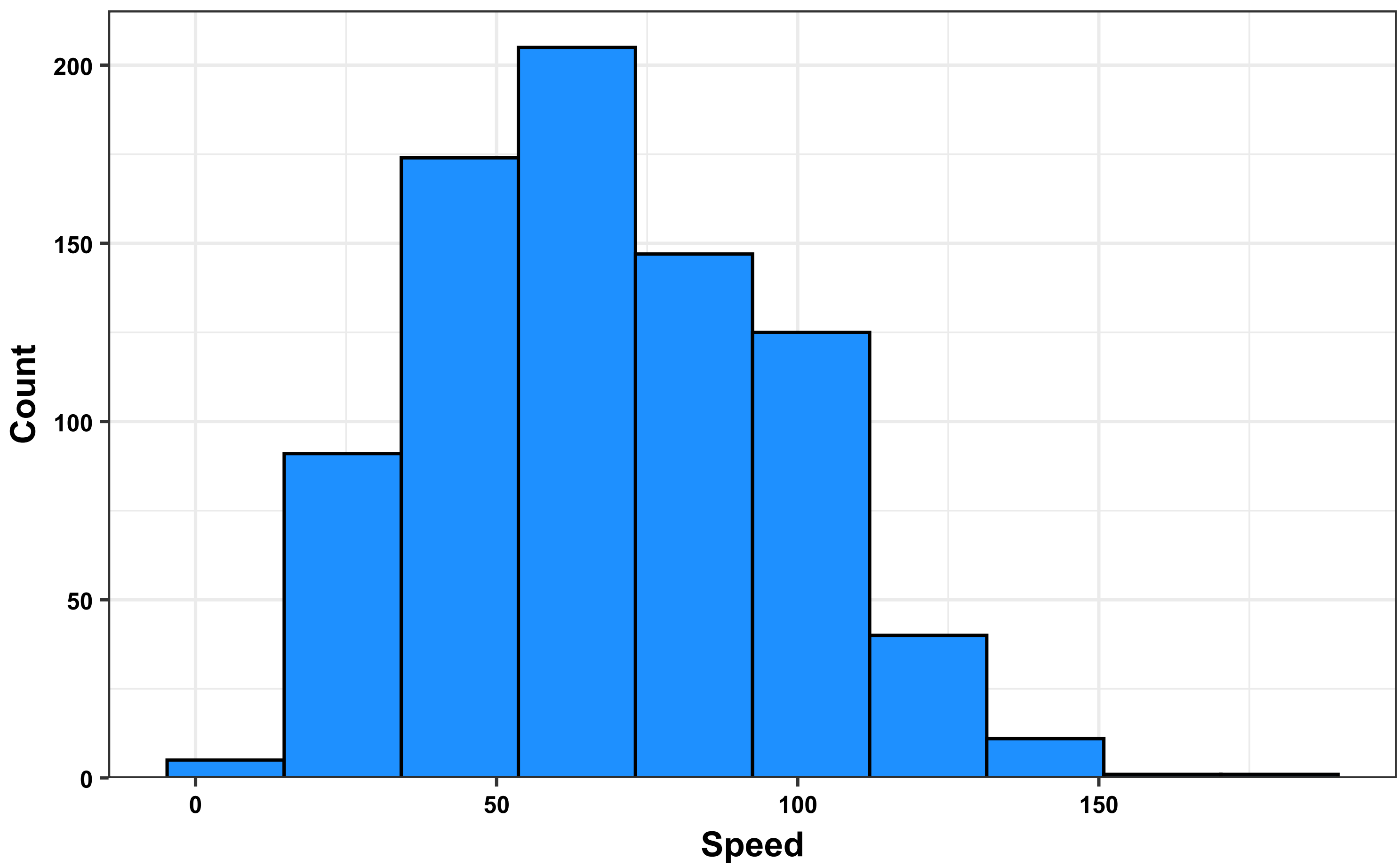
Question 3. Histogram and Scatter plots

1. Plot histogram of `attack` and `speed`. **Hint:** `geom_histogram()`
2. Create scatter plots of `attack` against each numeric predictor `speed`, `defense`, `hp`. **Hint:** `geom_point()`, with `geom_smooth(method = "lm")`

Solutions

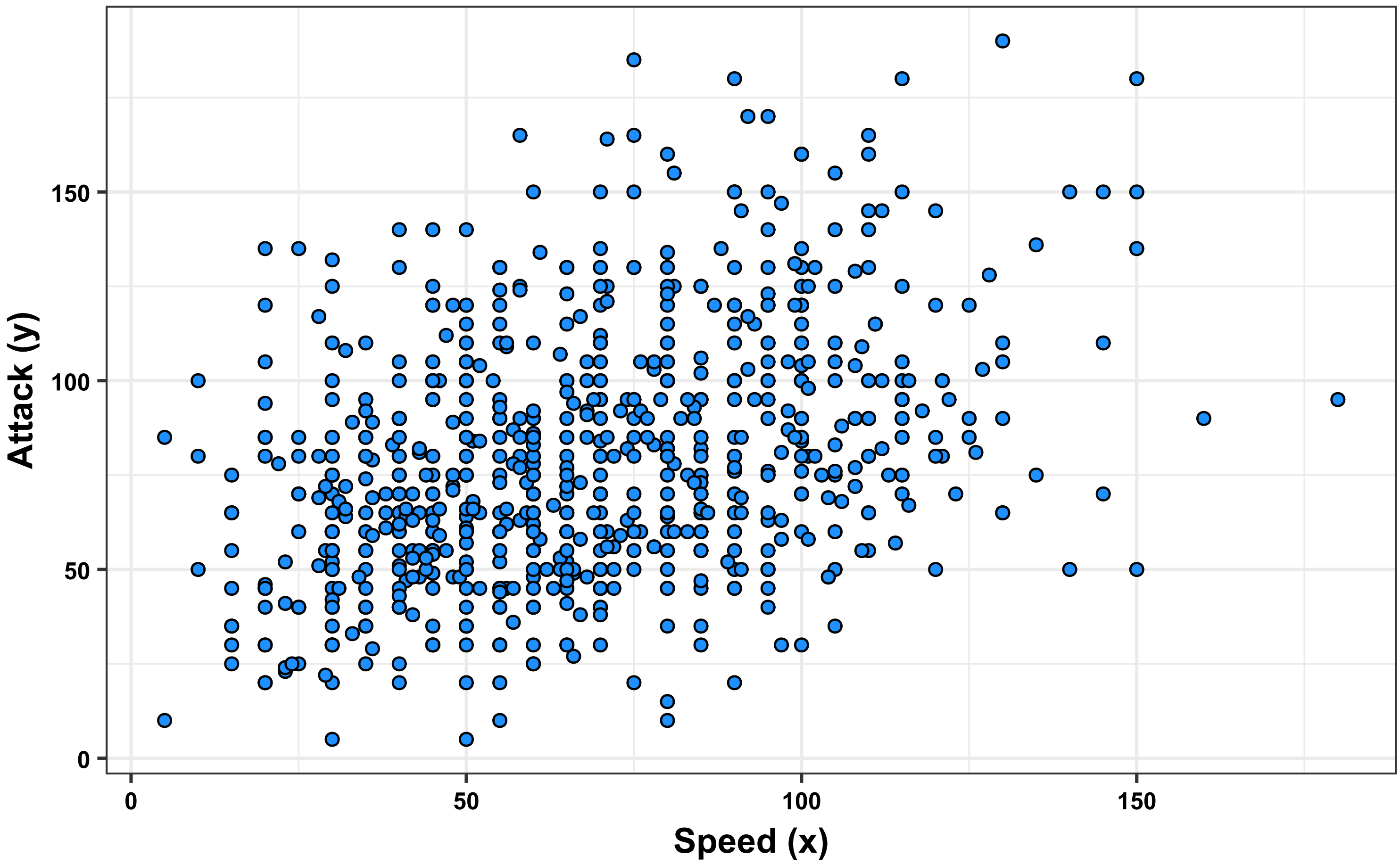
- `ggplot()` and `geom_histogram()` on the vector `attack` from the dataframe `pok`

```
ggplot(pok, aes(x = speed)) +
  geom_histogram(bins = 10, color = "black", fill = "dodgerblue") +
  scale_y_continuous(expand = expansion(c(0, 0.05))) +
  labs(x = "Speed", y = "Count") +
  theme_bw(base_size = 14) +
  labs_pubr()
```



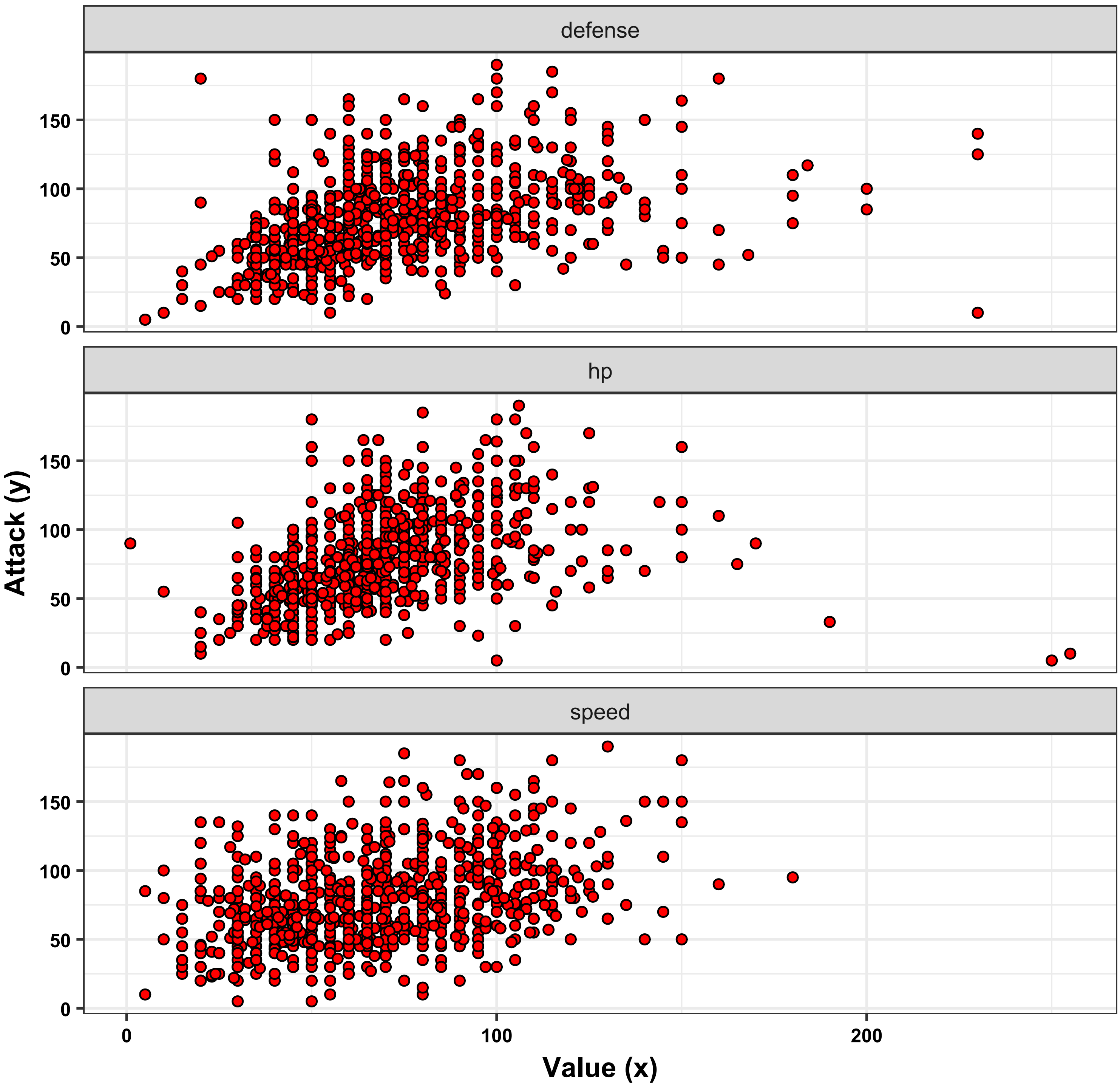
- Scatter plot of `attack` against `speed`

```
pok |>
  ggplot(aes(x = speed, y = attack)) +
  geom_point(size = 2, shape = 21, fill = "dodgerblue", color = "black") +
  labs(x = "Speed (x)", y = "Attack (y)") +
  theme_bw(base_size = 14) +
  labs_pubr()
```



- Scatter plots of `attack` against each numeric predictor `speed`, `defense`, `hp`

```
select(pok, attack, speed, defense, hp) |>
  pivot_longer(cols = c(speed, defense, hp)) |>
  ggplot(aes(x = value, y = attack)) +
  facet_wrap(vars(name), ncol = 1) +
  geom_point(size = 2, shape = 21, fill = "red", color = "black", alpha = 1) +
  labs(x = "Value (x)", y = "Attack (y)") +
  theme_bw(base_size = 14) +
  labs_pubr()
```



Question 4. Correlation matrix

Compute and interpret the correlation matrix of the predictors `speed`, `defense`, `hp`.

Hint: `cor()`, `correlation()`

Solutions

- With `cor()`

```
select(pok, speed, defense, hp) |>
  cor(method = "pearson")
```

```
           speed  defense      hp
speed   1.000000 0.015227 0.17595
defense 0.015227 1.000000 0.23962
hp      0.175952 0.239622 1.00000
```

- With `correlation()` from `{correlation}`

```
select(pok, speed, defense, hp) |> correlation(method = "pearson")
```

```
# Correlation Matrix (pearson-method)

Parameter1 | Parameter2 |    r |        95% CI | t(798) |         p
-------------------------------------------------------------------
speed      |    defense | 0.02 | [-0.05, 0.08] |   0.43 | 0.667
speed      |         hp | 0.18 | [ 0.11, 0.24] |   5.05 | < .001***
defense    |         hp | 0.24 | [ 0.17, 0.30] |   6.97 | < .001***

p-value adjustment method: Holm (1979)
Observations: 800
```

```
select(pok, speed, defense, hp) |>
  correlation(method = "pearson") |>
  summary(redundant = TRUE)
```

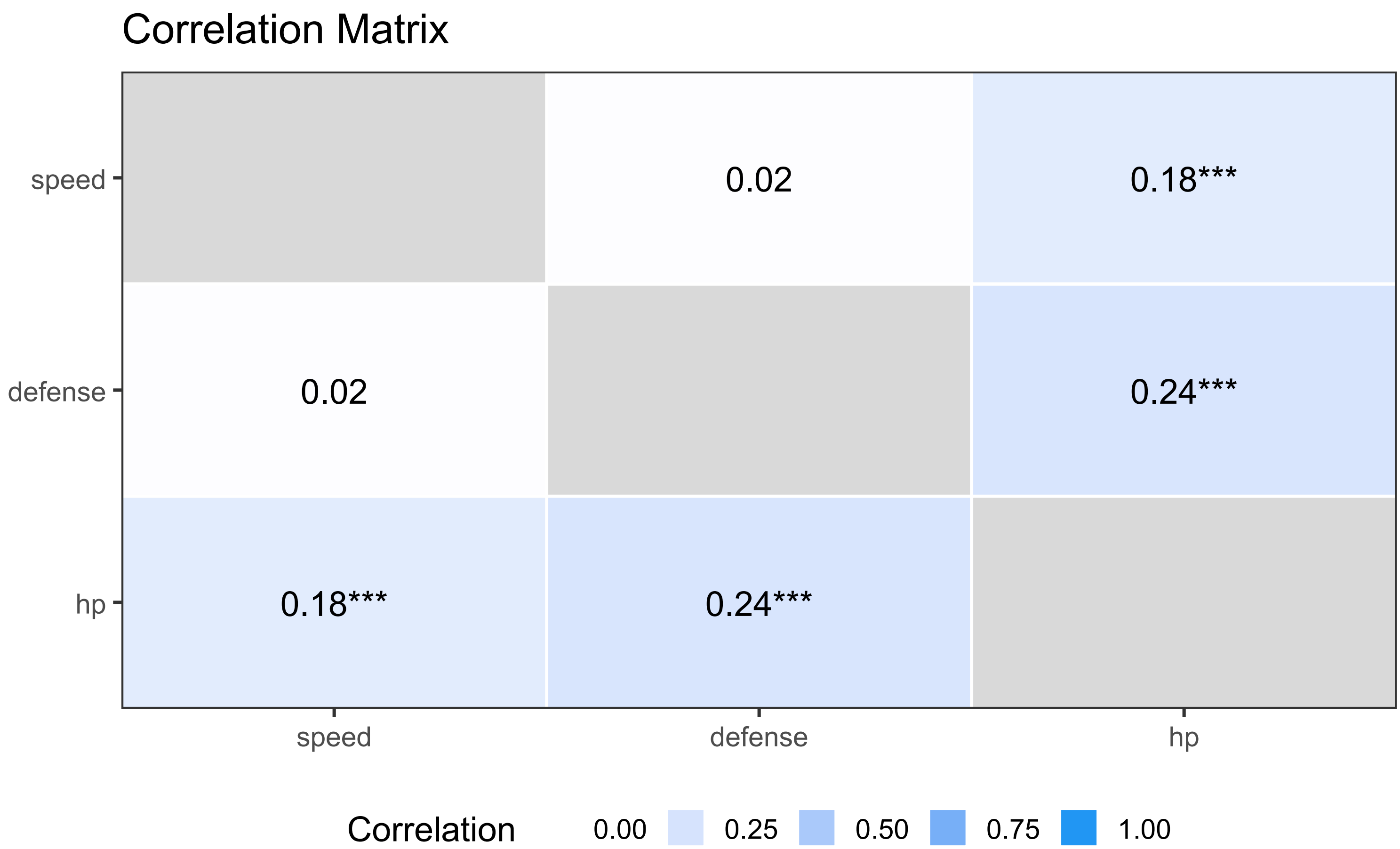
```
# Correlation Matrix (pearson-method)

Parameter |   speed | defense |      hp
---------------------------------------
speed     |         |    0.02 | 0.18***
defense   |    0.02 |         | 0.24***
hp        | 0.18*** | 0.24*** |

p-value adjustment method: Holm (1979)
```

- We can have a plot

```
select(pok, speed, defense, hp) |>
  correlation(method = "pearson") |>
  summary(redundant = TRUE) |>
  plot() +
  theme_bw(base_size = 14) +
  theme(legend.position = "bottom")
```



Question 5. Simple Linear Regression (SLR)

1. For each predictors `speed`, `defense`, `hp`, fit a SLR to explain the variable `attack`:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma^2).$$

$$\texttt{attack} = \beta_0 + \beta_1 \texttt{speed} + \varepsilon$$
$$\texttt{attack} = \beta_0 + \beta_1 \texttt{defense} + \varepsilon$$
$$\texttt{attack} = \beta_0 + \beta_1 \texttt{hp} + \varepsilon$$

Save the models in 3 objects: `slr_speed`, `slr_defense`, `slr_hp`. Interpret intercept and slope.
Hint: `lm()`

2. Interpret the results of the three hypothesis tests $H_0 : \beta_j = 0$ versus $H_1 : \beta_j \neq 0$?

Hint: `summary()` or `coeftest()` on `slr_speed`, `slr_defense`, `slr_hp`

3. Compute 95% CIs for coefficients.

Hint: `confint()`, `tidy()`, `model_parameters()`

Solutions

- We use `lm()` to fit the simple regressions models

```r
slr_speed <- lm(formula = attack ~ speed, data = pok)
slr_speed
```

```
Call:
lm(formula = attack ~ speed, data = pok)

Coefficients:
(Intercept)        speed
     49.928        0.426
```

$$\widehat{\texttt{attack}} = 49.928 + 0.426 \times \texttt{speed}$$

```r
slr_defense <- lm(attack ~ defense, data = pok)
slr_defense
```

```

Call:
lm(formula = attack ~ defense, data = pok)

Coefficients:
(Intercept)      defense
     45.284        0.457
```

$$\widehat{\texttt{attack}} = 45.284 + 0.457 \times \texttt{defense}$$

```r
slr_hp <- lm(attack ~ hp, data = pok)
slr_hp
```

```

Call:
lm(formula = attack ~ hp, data = pok)

Coefficients:
(Intercept)           hp
     41.816        0.537
```

$$\widehat{\texttt{attack}} = 41.816 + 0.537 \times \texttt{hp}$$

- What kind of object does `lm()` create ?

```r
class(slr_hp)
```

```
[1] "lm"
```

```r
typeof(slr_hp)
```

```
[1] "list"
```

```r
names(slr_hp)
```

```
 [1] "coefficients"  "residuals"     "effects"       "rank"          "fitted.values" "assign"
 [7] "qr"            "df.residual"   "xlevels"       "call"          "terms"         "model"
```

```r
sum_lm_speed <- summary(slr_speed)
sum_lm_speed
```

```

Call:
lm(formula = attack ~ speed, data = pok)

Residuals:
   Min     1Q Median     3Q    Max 
-73.99 -21.55  -3.79  18.07 103.14 

Coefficients:
            Estimate Std. Error t value            Pr(>|t|)    
(Intercept)  49.9285     2.7120    18.4 <0.0000000000000002 ***
speed         0.4258     0.0366    11.6 <0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 30 on 798 degrees of freedom
Multiple R-squared:  0.145,	Adjusted R-squared:  0.144 
F-statistic:  136 on 1 and 798 DF,  p-value: <0.0000000000000002
```

- What kind of object does `summary()` create ?

```r
class(sum_lm_speed)
```

```
[1] "summary.lm"
```

```r
typeof(sum_lm_speed)
```

```
[1] "list"
```

```r
names(sum_lm_speed)
```

```
 [1] "call"          "terms"         "residuals"     "coefficients"  "aliased"       "sigma"        
 [7] "df"            "r.squared"     "adj.r.squared" "fstatistic"    "cov.unscaled" 
```

- Testw of $\beta = 0$ with `coeftest()` from `{lmtest()}`

```r
coeftest(slr_speed)
```

```
t test of coefficients:

            Estimate Std. Error t value            Pr(>|t|)
(Intercept)  49.9285     2.7120    18.4 <0.0000000000000002 ***
speed         0.4258     0.0366    11.6 <0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
coeftest(slr_defense)
```

```
t test of coefficients:

            Estimate Std. Error t value            Pr(>|t|)
(Intercept)  45.2842     2.6538    17.1 <0.0000000000000002 ***
defense       0.4566     0.0331    13.8 <0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
coeftest(slr_hp)
```

```
t test of coefficients:

            Estimate Std. Error t value            Pr(>|t|)
(Intercept)  41.8163     3.0104    13.9 <0.0000000000000002 ***
hp            0.5369     0.0408    13.2 <0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- 95% CIs for coefficients of `slr_speed` with `confint()`

```
confint(slr_speed, level = 0.95)
```

```
                2.5 %   97.5 %
(Intercept) 44.60493 55.25204
speed        0.35405  0.49755
```

- 95% CIs for coefficients of `slr_defense` with `tidy()` from `{broom}`

```
tidy(slr_defense, conf.int = TRUE, conf.level = 0.95)
```

```
# A tibble: 2 x 7
  term        estimate std.error statistic  p.value conf.low conf.high
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
1 (Intercept)   45.3      2.65        17.1 6.82e-56   40.1      50.5  
2 defense        0.457    0.0331      13.8 5.86e-39    0.392     0.522
```

- 95% CIs for coefficients of `slr_hp` with `model_parameters()` from `{parameters}`

```
model_parameters(slr_hp, ci = 0.95, ci_method = "residual", digits = 3)
```

```
Parameter   | Coefficient |    SE |           95% CI | t(798) |      p
----------------------------------------------------------------------
(Intercept) |      41.816 | 3.010 | [35.907, 47.726] | 13.891 | < .001
hp          |       0.537 | 0.041 | [ 0.457,  0.617] | 13.164 | < .001
```

Question 6. Fitted values, residuals

1. In the pok database, create the following variables

- yhat_speed, which represents the fitted values of the slr_speed model: $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$
- res_speed, which represents the residuals of the slr_speed model: $\hat{\varepsilon}_i = y_i - \hat{y}_i$

Hint: retrieve $\hat{\beta}_0$ and $\hat{\beta}_1$ with `coef()`

Display the first 10 rows of pok with the fitted values and residuals added.

2. Compute the fitted values and residuals using the functions `predict()` (or `fitted`) and `residuals()` (or `resid()`) on slr_speed. Also try the handy function `augment()` on slr_speed

Solutions

- We retrieve $\hat{\beta}_0$ and $\hat{\beta}_1$ with `coef()`

```
betas_speed <- coef(slr_speed)
betas_speed
```

```
(Intercept)       speed
    49.9285      0.4258
```

- We create yhat_speed and res_speed in pok with `mutate()` from **{dplyr}**

```
pok <- pok |>
  mutate(yhat_speed = betas_speed[1] + betas_speed[2] * speed) |>
  mutate(res_speed = attack - yhat_speed)
```

```
select(pok, id, name, attack, speed, yhat_speed, res_speed) |> head(10)
```

```
# A tibble: 10 x 6
      id name             attack speed yhat_speed res_speed
   <dbl> <chr>             <dbl> <dbl>      <dbl>     <dbl>
 1     1 Bulbasaur            49    45       69.1    -20.1 
 2     2 Ivysaur              62    60       75.5    -13.5 
 3     3 Venusaur             82    80       84.0     -1.99
 4     4 Mega Venusaur       100    80       84.0     16.0 
 5     5 Charmander           52    65       77.6    -25.6 
 6     6 Charmeleon           64    80       84.0    -20.0 
 7     7 Charizard            84   100       92.5     -8.51
 8     8 Mega Charizard X    130   100       92.5     37.5 
 9     9 Mega Charizard Y    104   100       92.5     11.5 
10    10 Squirtle             48    43       68.2    -20.2 
```

- With `fitted()` and `residuals()`. 10 first values

```
fitted(slr_speed)[1:10]
```

```
     1      2      3      4      5      6      7      8      9     10
69.090 75.477 83.993 83.993 77.606 83.993 92.509 92.509 92.509 68.238
```

```
residuals(slr_speed)[1:10]
```

```
       1        2        3        4        5        6        7        8        9       10
-20.0896 -13.4767  -1.9927  16.0073 -25.6057 -19.9927  -8.5088  37.4912  11.4912 -20.2380
```

- With `augment()` from **{broom}**

```
augment_speed <- augment(slr_speed)
head(augment_speed, 10)
```

```
# A tibble: 10 x 8
   attack speed .fitted .resid    .hat .sigma    .cooksd .std.resid
    <dbl> <dbl>   <dbl>  <dbl>   <dbl>  <dbl>      <dbl>      <dbl>
 1     49    45    69.1 -20.1  0.00205   30.0 0.000461      -0.670
 2     62    60    75.5 -13.5  0.00135   30.0 0.000137      -0.449
 3     82    80    84.0  -1.99 0.00145   30.0 0.00000321    -0.0664
 4    100    80    84.0  16.0  0.00145   30.0 0.000207       0.534
 5     52    65    77.6 -25.6  0.00127   30.0 0.000462      -0.853
 6     64    80    84.0 -20.0  0.00145   30.0 0.000323      -0.666
 7     84   100    92.5  -8.51 0.00274   30.0 0.000111      -0.284
 8    130   100    92.5  37.5  0.00274   30.0 0.00215        1.25
 9    104   100    92.5  11.5  0.00274   30.0 0.000202       0.383
10     48    43    68.2 -20.2  0.00220   30.0 0.000501      -0.675
```

Question 7. Residual diagnostics

Note: Standardized vs Studentized residuals

$$\hat{\varepsilon} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbb{X}\hat{\boldsymbol{\beta}} = \mathbf{y} - P_{\mathbb{X}}\mathbf{y} = (\mathbf{I}_n - P_{\mathbb{X}})\mathbf{y} = (\mathbf{I}_n - P_{\mathbb{X}})\boldsymbol{\varepsilon}$$

- Let denote by h_{ij} the element of the projector $P_{\mathbb{X}} = H_{\mathbb{X}}$ such that $P_{\mathbb{X}} = H_{\mathbb{X}} = [h_{ij}]$
- The diagonal elements $h_{ii} \in [0, 1]$ are called the *leverages*
- If $h_{ii} > 2p/n$ (sometines $h_{ii} > 3p/n$), then the observation i is consider an *outlier*

- We have $\mathrm{Cov}(\boldsymbol{\varepsilon}) = \sigma^2\mathbf{I}_n$ but $\mathrm{Cov}(\hat{\boldsymbol{\varepsilon}}) = \sigma^2(\mathbf{I}_n - H_{\mathbb{X}})$

- The residuals are not independant, however, in many cases, especially if n is large, the h_{ii}'s tend to be small.
  The impact of this is usually small and **diagnostics can reasonably be applied to the residuals in order to check the assumptions on the error** but we can also modify the residuals to adjust for this effect.

- **Standardized residuals** (from `rstandard()`)
  Raw residuals are rescaled by their estimated standard deviation, taking into account leverage.

$$\hat{r}_i = \frac{\hat{\varepsilon}_i}{\hat{\sigma}\sqrt{1 - h_{ii}}}$$

where $\hat{\varepsilon}_i$ is the raw residual and h_{ii} is the leverage of observation i.
These make residuals roughly comparable across observations.

- **Studentized residuals** (from `rstudent()`)
  Go one step further: each residual is scaled using a variance estimate that **excludes the** i-th observation.
  This gives more accurate standard errors and makes large outliers easier to detect.

$$t_i^* = \frac{\hat{\varepsilon}_i}{\hat{\sigma}_{(-i)}\sqrt{1 - h_{ii}}}$$

where $\hat{\sigma}_{(-i)}$ is the error standard deviation estimated without observation i.

In practice:

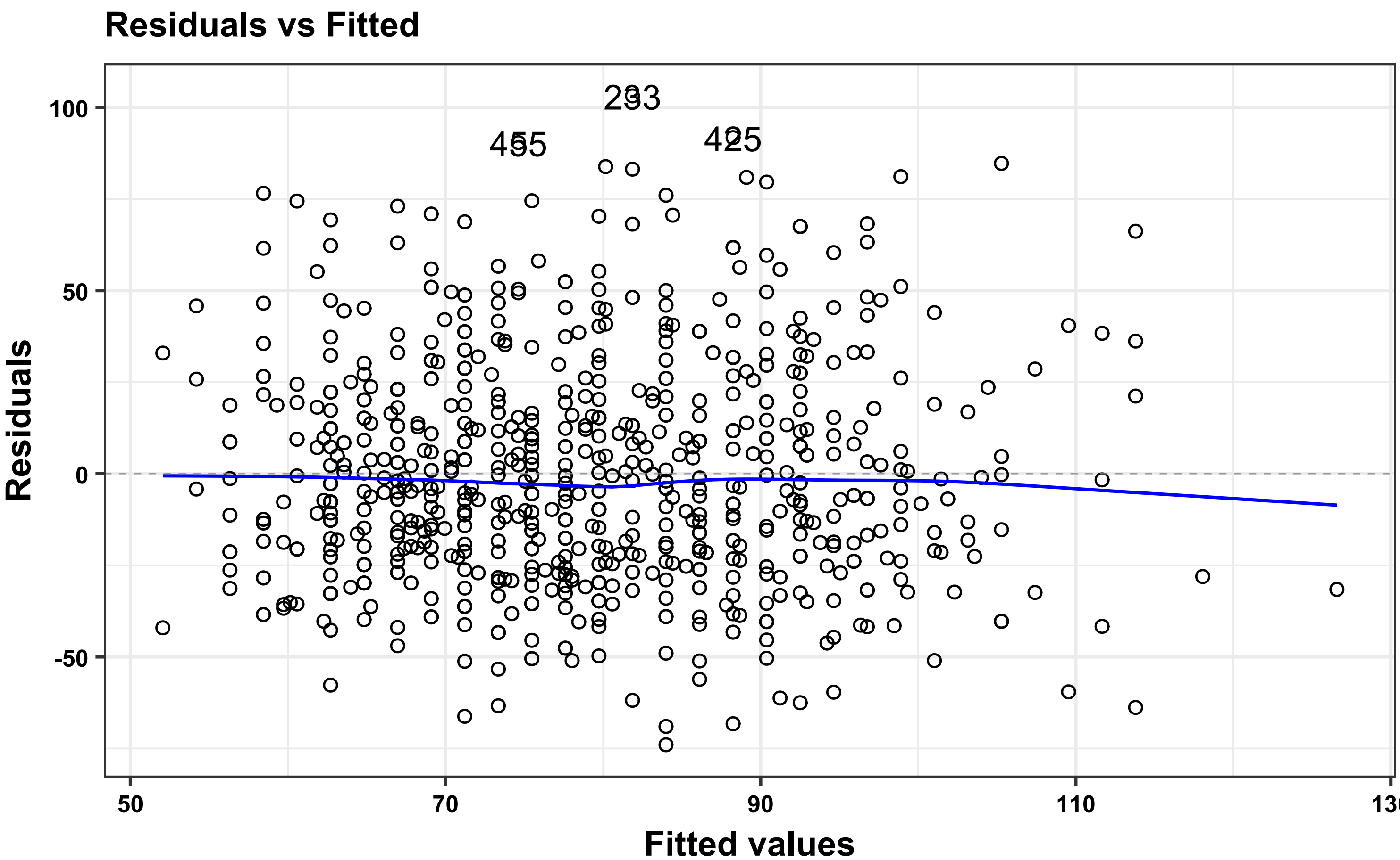
- Use **standardized residuals** for quick checks.
- Use **studentized residuals** when formally testing for outliers or influential observations.

1. For the `slr_speed` model, with the help of `autoplot()`, plot residuals vs fitted values and $\sqrt{|\text{Standardized residuals}|}$ vs fitted values
2. Plot *studentized residuals* vs fitted: **Hint:** `fitted()`, `rstudent()`
3. Plot residuals vs `speed` and vs `defense`
4. Plot residuals in the order of observation (to detect time dependence). **Hint:** `augment()`
5. Plot an histogram of the standardized residuals.
6. Plot a normal Q-Q plot of the standardized residuals. **Hint:** `qqnorm()`, `qqline()`, `augment()`, `stat_qq()`, `stat_qq_line()`
7. Performs the Breusch–Pagan test for heteroskedasticity: **Hint:** `bptest()`, `ncvTest()`
8. Perform the Durbin–Watson test on the residuals: **Hint:** `dwtest()`, `durbinWatsonTest()`
9. Perform test for normality on standardized residuals. **Hint:** `shapiro.test()`, `shapiro_test()`, `col_jarquebera()`

Solutions

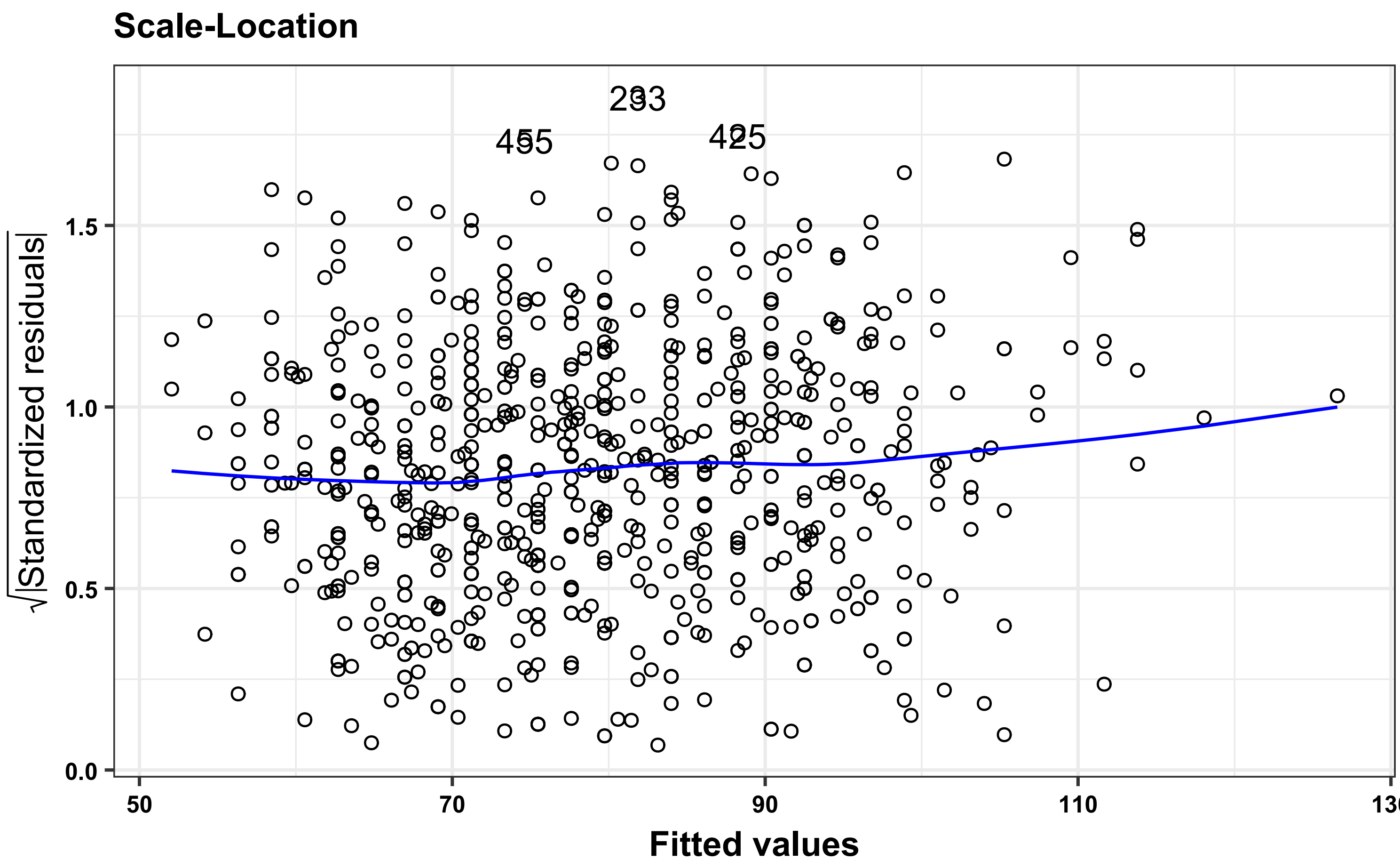
- Residuals vs fitted values with `autoplot()` (need to load `{ggfortify}`)

```
autoplot(slr_speed, which = 1, ncol = 1, colour = "black", shape = 21, size = 2) +
  theme_bw(base_size = 14) +
  labs_pubr()
```



- $\sqrt{|\text{Standardized residuals}|}$ vs fitted

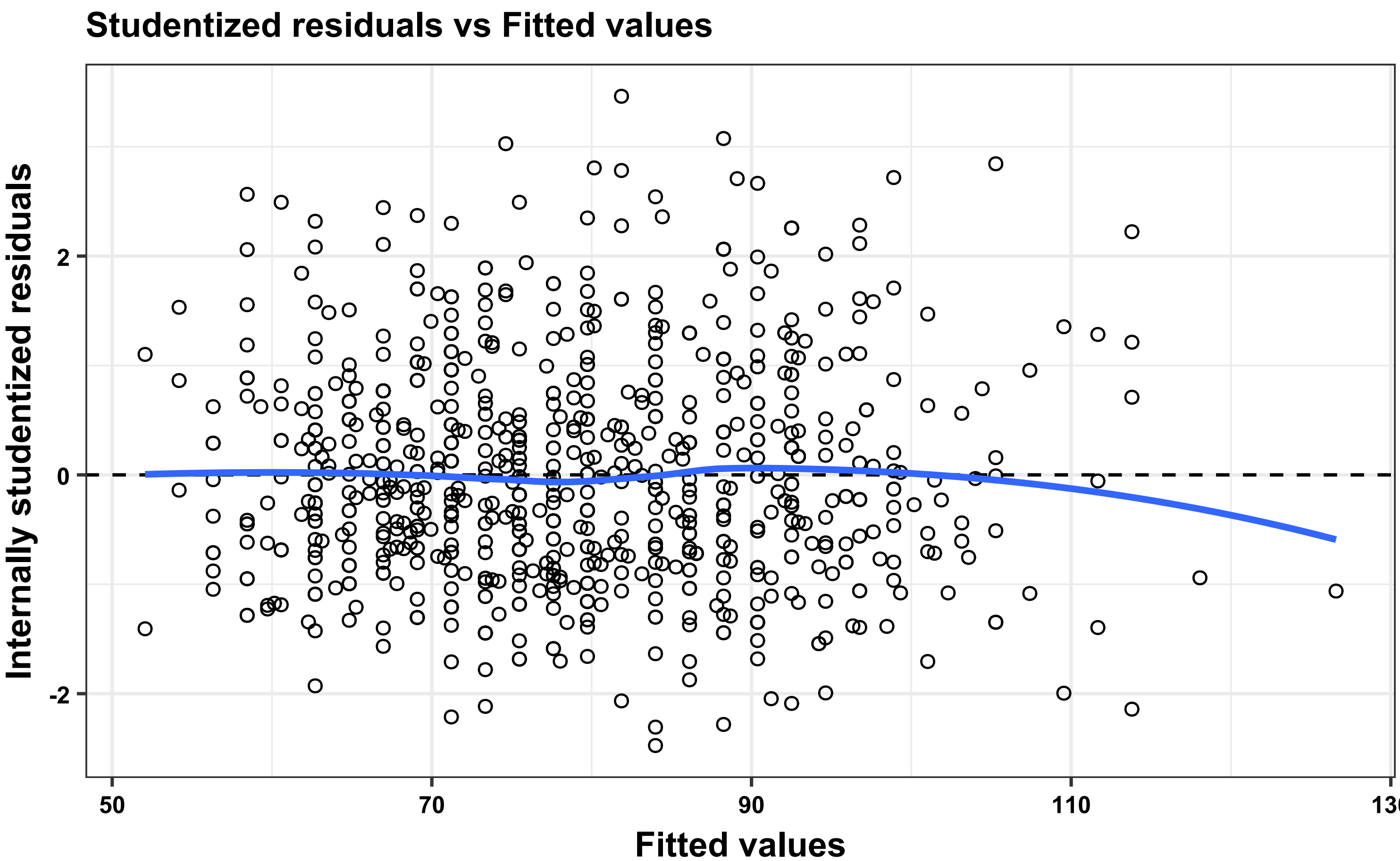
```
autoplot(slr_speed, which = 3, ncol = 1, colour = "black", shape = 21, size = 2) +
  theme_bw(base_size = 14) +
  labs_pubr()
```



- Homoscedasticity is satisfied

- We create a dataset with fitted values and *studentized residuals* (`rstudent()`)

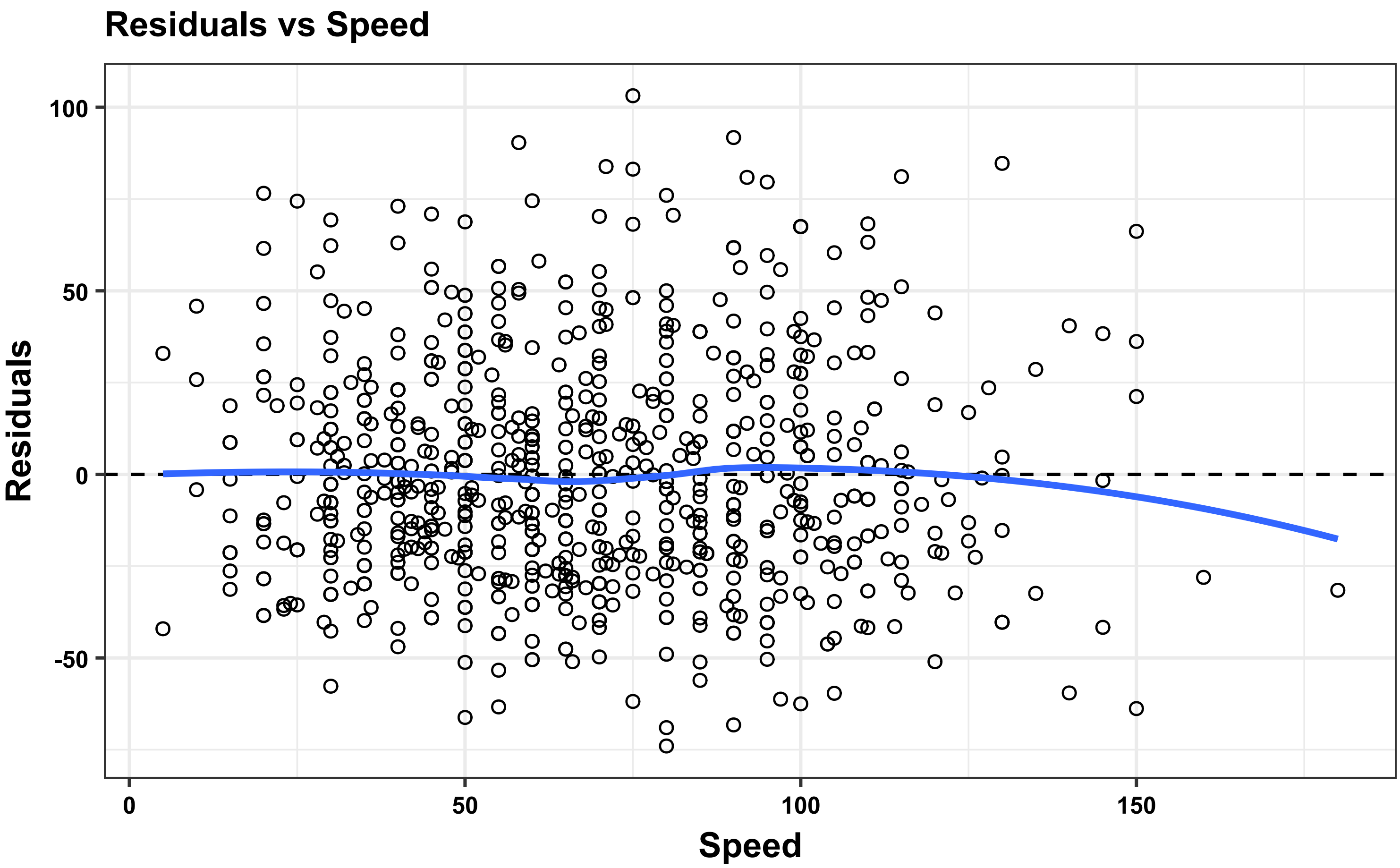
```
tibble(
  fitted = fitted(slr_speed), rstudent = rstudent(slr_speed)
) |>
  ggplot(aes(x = fitted, y = rstudent)) +
  geom_hline(yintercept = 0, linetype = 2) +
  geom_point(shape = 21, size = 2) +
  geom_smooth(method = "loess", se = FALSE)  +
  labs(x = "Fitted values", y = "Internally studentized residuals") +
  labs(title = "Studentized residuals vs Fitted values") +
  theme_bw(base_size = 14) +
  labs_pubr()
```



- Linearity and homoscedasticity are satisfied

- Residuals vs `speed`

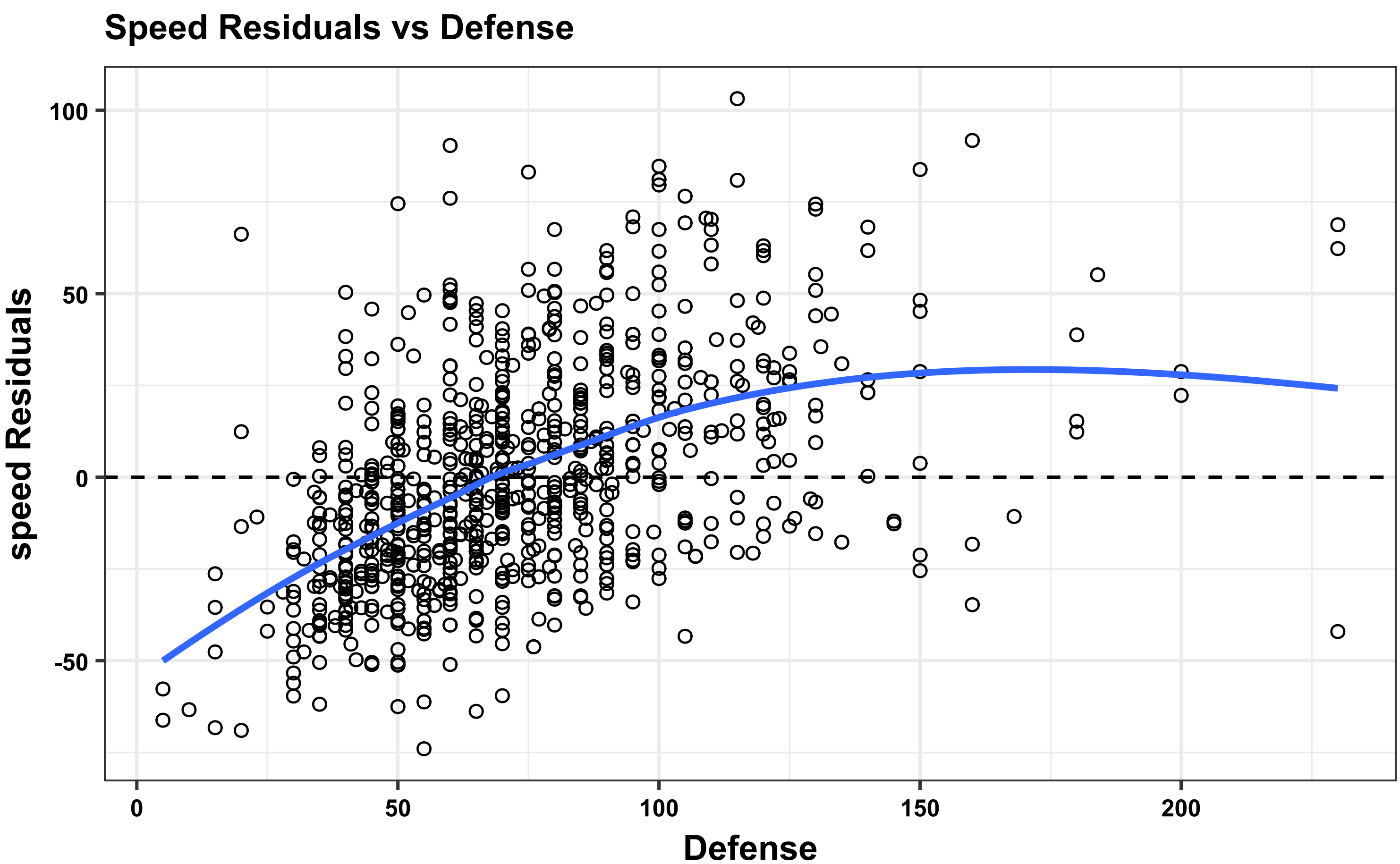
```
pok |>
  ggplot(aes(x = speed, y = res_speed)) +
  geom_hline(yintercept = 0, linetype = 2) +
  geom_point(shape = 21, size = 2) +
  geom_smooth(method = "loess", se = FALSE) +
  labs(x = "Speed", y = "Residuals") +
  labs(title = "Residuals vs Speed") +
  theme_bw(base_size = 14) +
  labs_pubr()
```



- Linearity and homoscedasticity are satisfied

- Speed Residuals vs `defense`

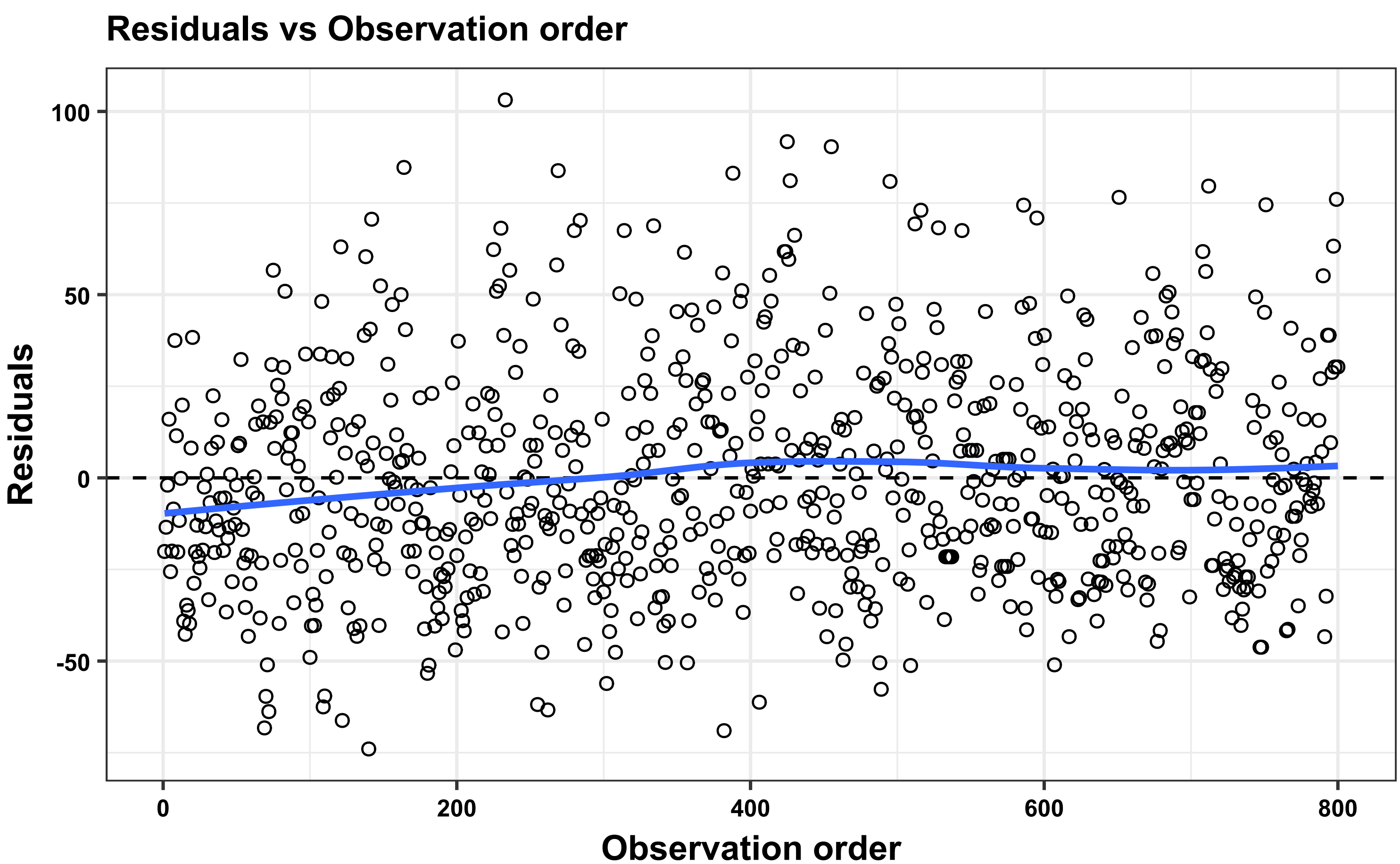
```
pok |>
  ggplot(aes(x = defense, y = res_speed)) +
  geom_hline(yintercept = 0, linetype = 2) +
  geom_point(shape = 21, size = 2) +
  geom_smooth(method = "loess", se = FALSE) +
  labs(x = "Defense", y = "speed Residuals") +
  labs(title = "Speed Residuals vs Defense") +
  theme_bw(base_size = 14) +
  labs_pubr()
```



- We should probably add `defense` to the model

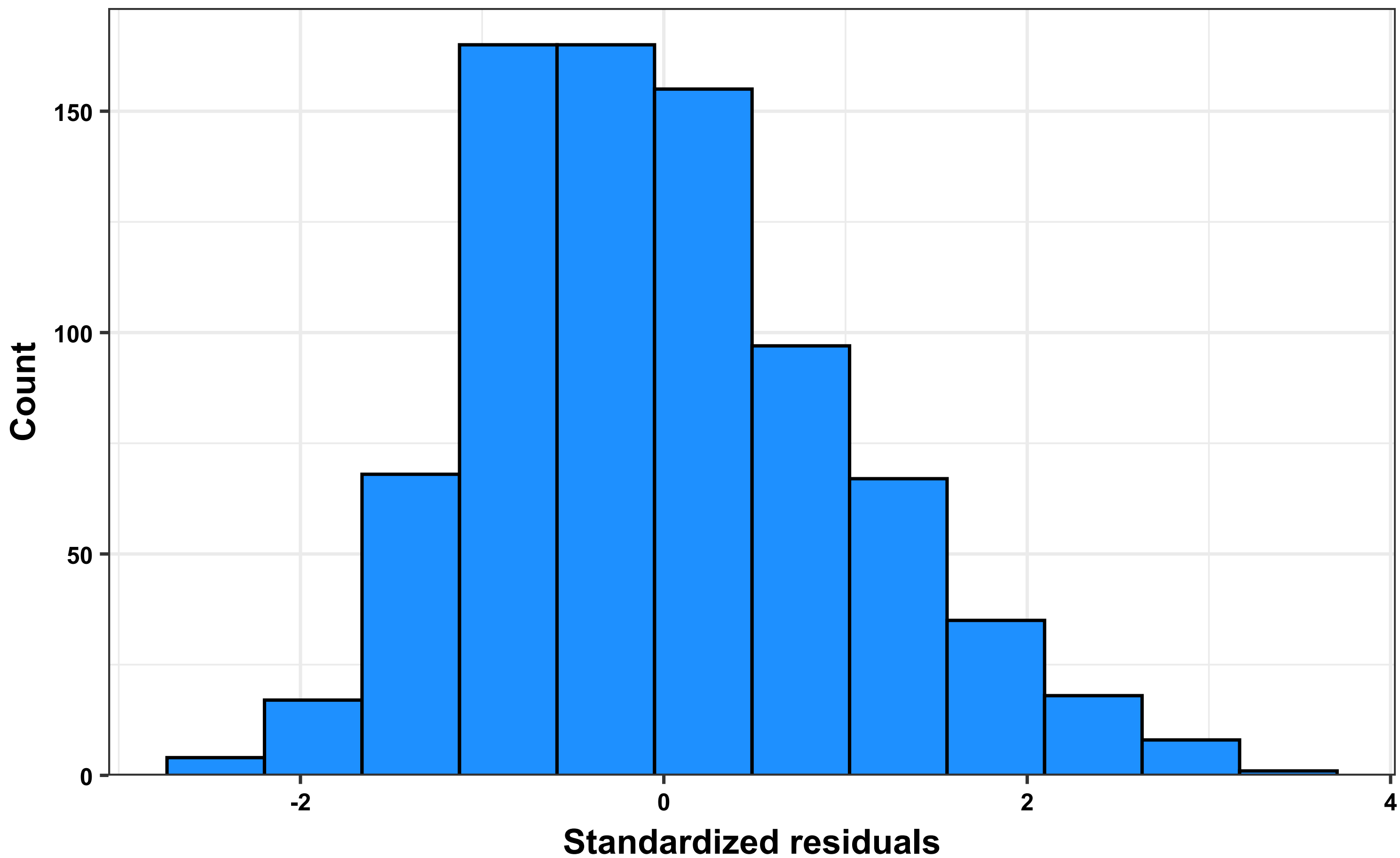
- Residuals vs the observation order

```
pok |>
  ggplot(aes(x = id, y = res_speed)) +
  geom_hline(yintercept = 0, linetype = 2) +
  geom_point(shape = 21, size = 2) +
  geom_smooth(method = "loess", se = FALSE) +
  labs(x = "Observation order", y = "Residuals") +
  labs(title = "Residuals vs Observation order") +
  theme_bw(base_size = 14) +
  labs_pubr()
```



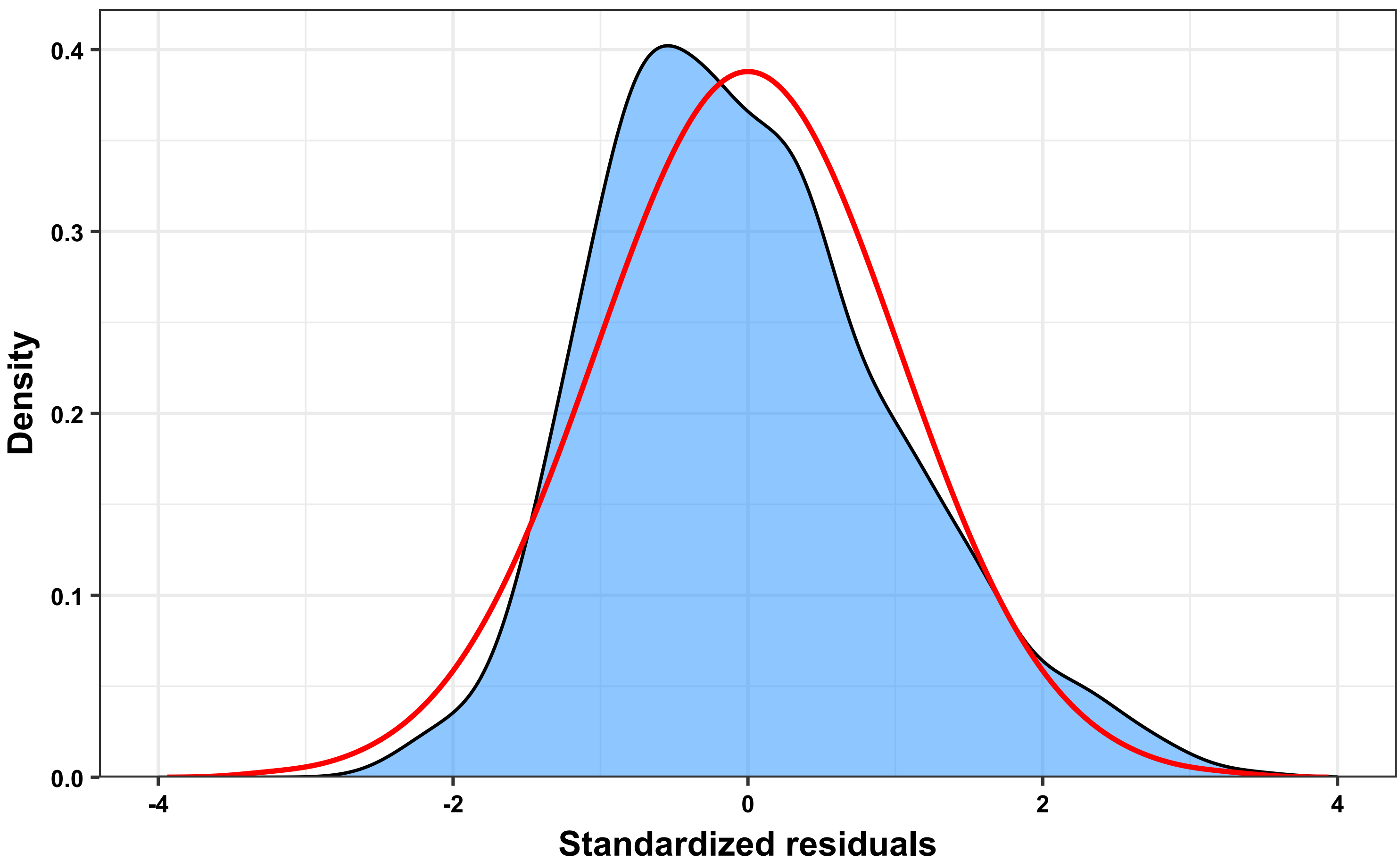
- Histogram of the standardized residuals

```
augment_speed |>
  ggplot(aes(x = .std.resid)) +
  geom_histogram(bins = 12, color = "black", fill = "dodgerblue") +
  scale_y_continuous(expand = expansion(c(0, 0.05))) +
  labs(x = "Standardized residuals", y = "Count") +
  theme_bw(base_size = 14) +
  labs_pubr()
```



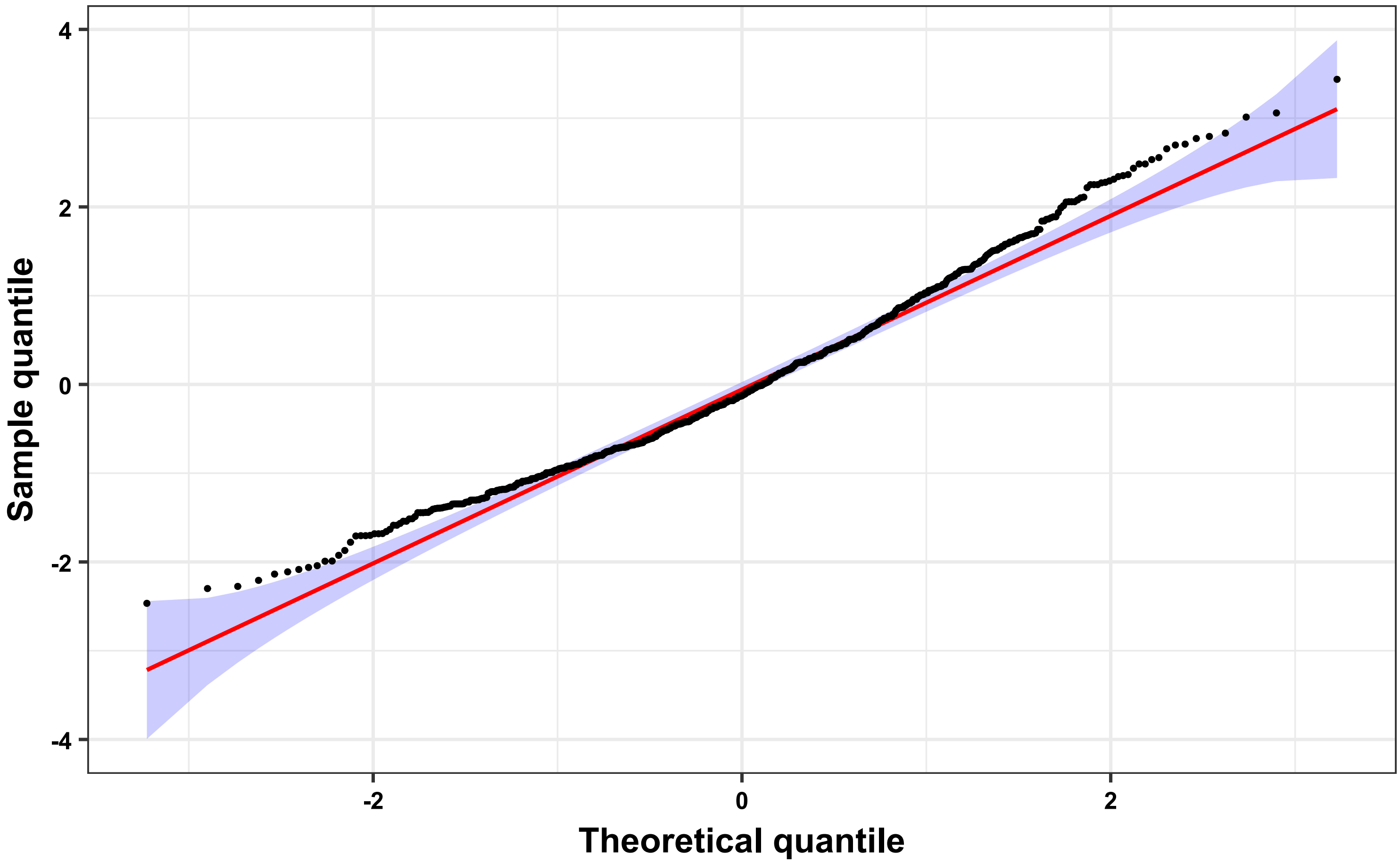
- Density of the standardized residuals

```r
ggdensity(augment_speed, x = ".std.resid", fill = "dodgerblue") +
  scale_x_continuous(limits = c(-4, 4)) +
  stat_overlay_normal_density(color = "red", linetype = 1, linewidth = 1) +
  scale_y_continuous(expand = expansion(c(0, 0.05))) +
  labs(x = "Standardized residuals", y = "Density") +
  theme_bw(base_size = 14) +
  labs_pubr()
```



- Normal Q-Q plot of the standardized residuals

```
augment_speed |>
  ggplot(aes(sample = .std.resid)) +
  stat_qq_band(alpha = 0.2, fill = "blue") + # du package {qqplotr}
  stat_qq_line(color = "red") + # version du package {qqplotr}
  stat_qq_point(size = 0.5) +
  labs(y = "Sample quantile", x = "Theoretical quantile") +
  theme_bw(base_size = 14) +
  labs_pubr()
```



- Breusch–Pagan test for heteroskedasticity with `bptest()` from `{lmtest}` or `ncvTest()` from `{car}`

```r
bptest(slr_speed, studentize = FALSE)
```

```
	Breusch-Pagan test

data:  slr_speed
BP = 7.92, df = 1, p-value = 0.0049
```

```r
ncvTest(slr_speed)
```

```
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 7.9215, Df = 1, p = 0.00489
```

- Durbin–Watson test with `dwtest()` from `{lmtest}` or `durbinWatsonTest()` from {car}

```r
dwtest(slr_speed)
```

```
	Durbin-Watson test

data:  slr_speed
DW = 1.4, p-value <0.0000000000000002
alternative hypothesis: true autocorrelation is greater than 0
```

```r
durbinWatsonTest(slr_speed)
```

```
 lag Autocorrelation D-W Statistic p-value
   1         0.29892        1.4003       0
 Alternative hypothesis: rho != 0
```

- Shapiro-Wilk Normality Test on standardized residuals

```r
shapiro.test(augment_speed[[".std.resid"]])
```

```
	Shapiro-Wilk normality test

data:  augment_speed[[".std.resid"]]
W = 0.982, p-value = 0.000000025
```

```r
augment_speed |>
  shapiro_test(.std.resid)
```

```
# A tibble: 1 x 3
  variable   statistic            p
  <chr>          <dbl>        <dbl>
1 .std.resid     0.982 0.0000000245
```

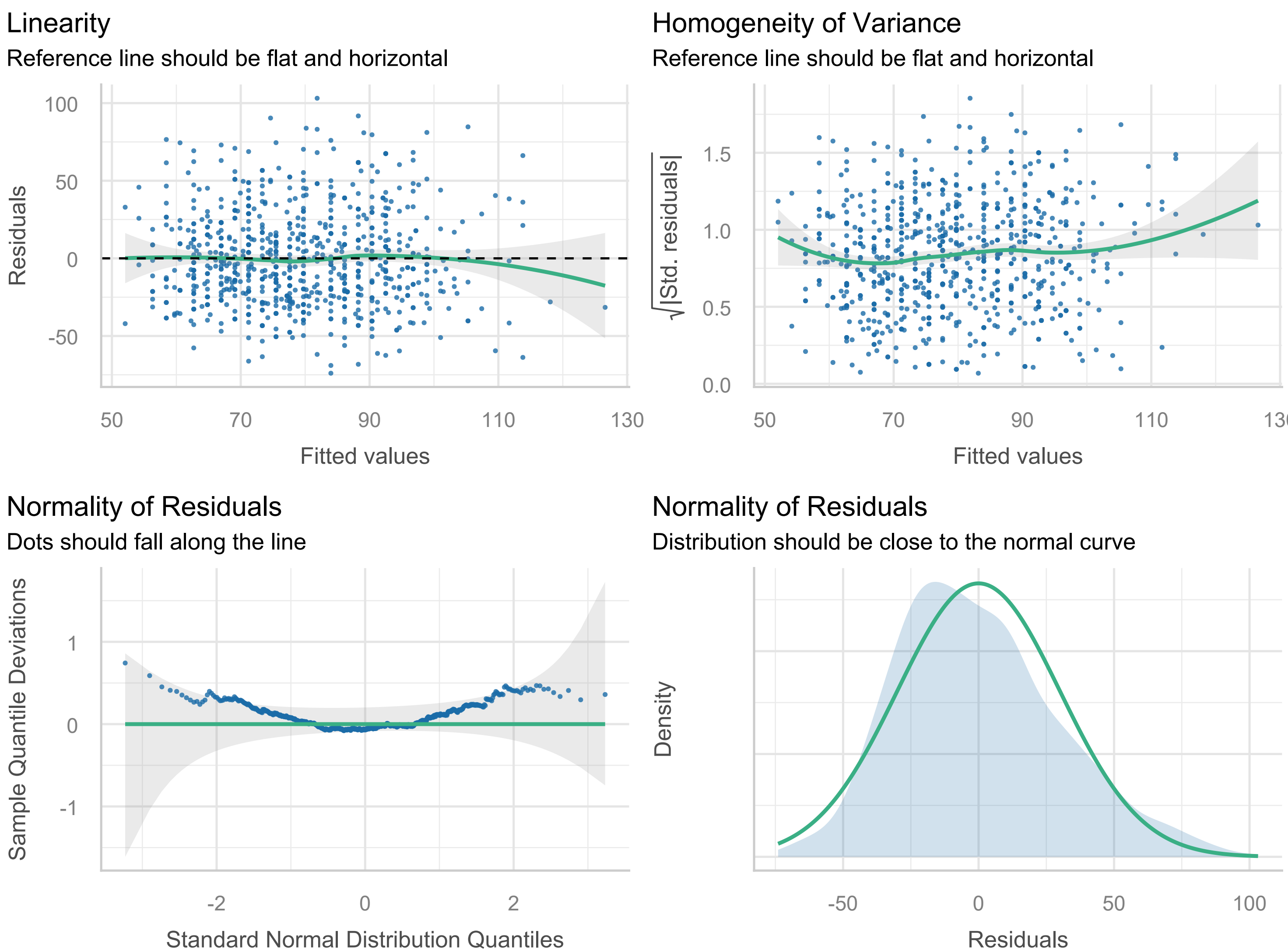
- Jarque-Bera test Test on standardized residuals

```r
select(augment_speed, .std.resid) |>
  col_jarquebera()
```

```
           obs skewness kurtosis df statistic         pvalue
.std.resid 800  0.50195   3.0528  2    33.686 0.000000048432
```

- `check_model()` from `{performance}`

```
check_model(slr_speed, check = c("qq", "normality", "linearity", "homogeneity"), size_dot = 1)
```



Question 8. Prediction & Intervals

Consider the model `slr_speed`

1. Compute 95% Confidence Interval for the mean $\mathbb{E}(\texttt{attack})$ at $\texttt{speed} = \texttt{30}, \texttt{70}, \texttt{110}, \texttt{150}$.

Hint: `predict(..., interval = "confidence")`, `estimate_expectation()`

2. Compute 95% Prediction Interval for new responses of `attack` at $\texttt{speed} = \texttt{20}, \texttt{60}, \texttt{100}, \texttt{140}$.

Hint: `predict(..., interval = "prediction")`, `estimate_prediction()`

3. On the graph representing the scatter plot between `attack` and `speed`, overlay the regression line and its 95% confidence band.

Solutions

- 95% Confidence Interval for the mean with `predict()`

```
grid_speed <- tibble(speed = seq(from = 30, to = 150, by = 40))
```

```
predict(slr_speed, newdata = grid_speed, interval = "confidence") |>
  as_tibble() |>
  mutate(speed = seq(30, 150, 40), .before = 1)
```

```
# A tibble: 4 x 4
  speed   fit   lwr   upr
  <dbl> <dbl> <dbl> <dbl>
1    30  62.7  59.3  66.1
2    70  79.7  77.6  81.8
3   110  96.8  93.1 100.
4   150 114.  108.  120.
```

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

```
estimate_expectation(slr_speed, by = "speed = seq(30, 150, 40)", ci = 0.95) |>
  as_tibble()
```

```
# A tibble: 4 x 5
  speed Predicted    SE CI_low CI_high
  <dbl>     <dbl> <dbl>  <dbl>   <dbl>
1    30      62.7  1.76   59.3    66.1
2    70      79.7  1.06   77.6    81.8
3   110      96.8  1.86   93.1   100.
4   150     114.   3.17  108.    120.
```

- 95% Prediction Interval for the new observations with `predict()`

```
predict(slr_speed, newdata = grid_speed, interval = "prediction") |>
  as_tibble() |>
  mutate(speed = seq(30, 150, 40), .before = 1)
```

```
# A tibble: 4 x 4
  speed   fit   lwr   upr
  <dbl> <dbl> <dbl> <dbl>
1    30  62.7  3.66  122.
2    70  79.7 20.8   139.
3   110  96.8 37.7   156.
4   150 114.  54.5   173.
```

- With `estimate_prediction()` from `{modelbased}`

```
estimate_prediction(slr_speed, by = "speed = seq(30, 150, 40)", ci = 0.95)
```

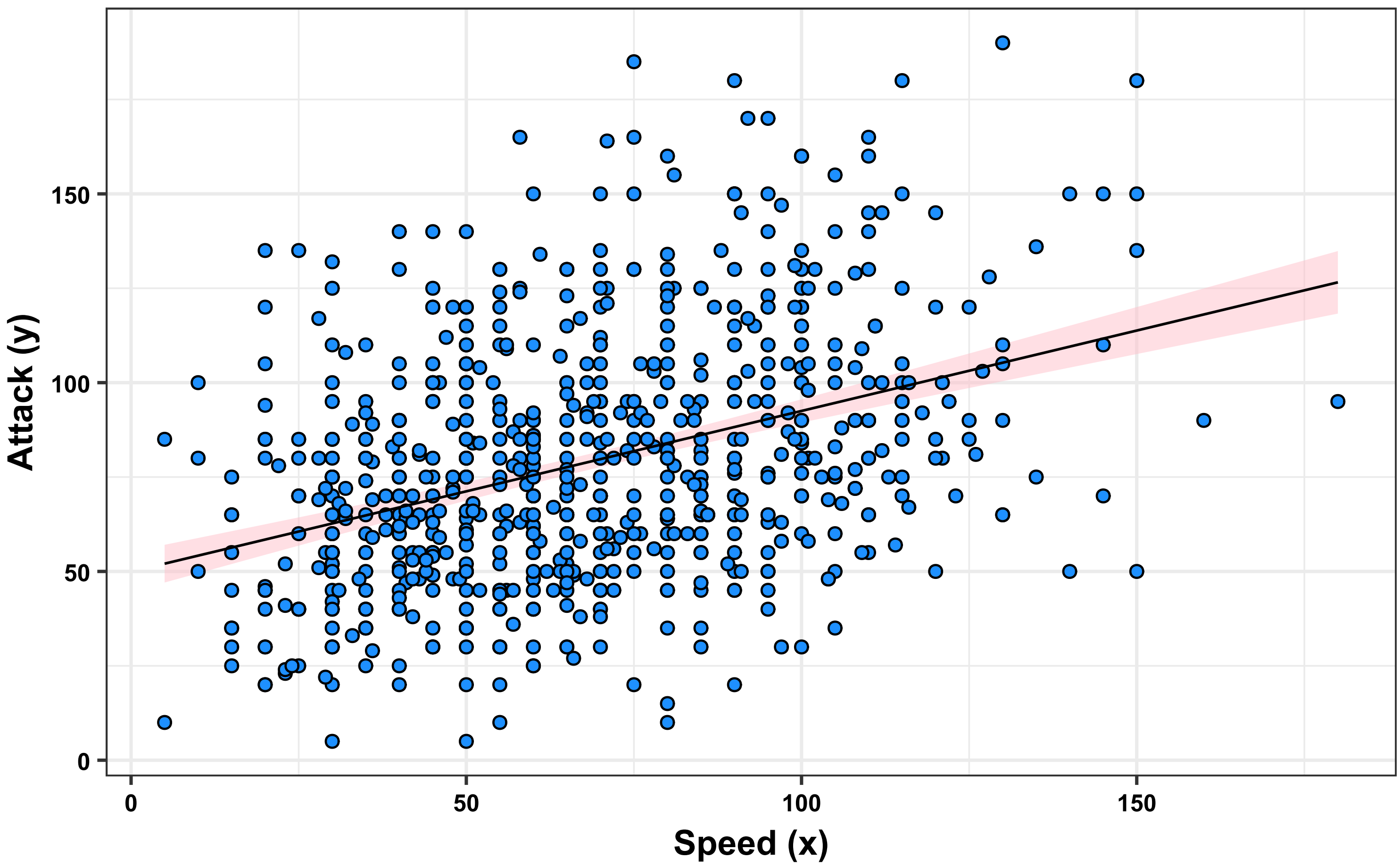
```
Model-based Predictions

speed | Predicted |    SE |          95% CI
--------------------------------------------
30    |     62.70 | 30.08 | [ 3.66, 121.74]
70    |     79.73 | 30.04 | [20.76, 138.71]
110   |     96.77 | 30.08 | [37.72, 155.82]
150   |    113.80 | 30.19 | [54.53, 173.06]

Variable predicted: attack
Predictors modulated: speed = seq(30, 150, 40)
```

- Regression line and the 95% confidence interval of the predicted values

```
predict(slr_speed, newdata = select(pok, speed), interval = "confidence") |>
  as_tibble() |>
  bind_cols(select(pok, attack, speed)) |>
  ggplot(aes(x = speed, y = attack)) +
  geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.5, fill = "pink") +
  geom_point(size = 2, shape = 21, fill = "dodgerblue", color = "black") +
  geom_line(aes(y = fit), color = "black", linewidth = 0.5) +
  labs(x = "Speed (x)", y = "Attack (y)") +
  theme_bw(base_size = 14) +
  labs_pubr()
```



Session Info

- R version 4.5.1 (2025-06-13 ucrt)
- Rstudio version 2025.9.1.401 (Cucumberleaf Sunflower)

Package	Version
broom	1.0.10
car	3.1-3
collapse	2.1.3
correlation	0.8.8
datawizard	1.3.0
effectsize	1.0.1
GGally	2.4.0
ggfortify	0.4.19
ggpubr	0.6.1
glue	1.8.0
insight	1.4.2
lmtest	0.9-40
matrixTests	0.2.3
modelbased	0.13.0
parameters	0.28.2
performance	0.15.2
qqplotr	0.0.7
rstatix	0.7.2
scales	1.4.0
see	0.12.0
tidyverse	2.0.0