HW-2-utkarshapatil01

Utkarsha Patil

Transforming like a Data... Transformer

Required Setup

```r
# Sets the number of significant figures to two - e.g., 0.01
options(digits = 2)

# Required package for quick package downloading and loading
if (!require(pacman))
  install.packages("pacman")
```

```
Loading required package: pacman
```

```r
# Downloads and load required packages
pacman::p_load(dlookr, # Exploratory data analysis
               forecast, # Needed for Box-Cox transformations
               formattable, # HTML tables from R outputs
               here, # Standardizes paths to data
               kableExtra, # Alternative to formattable
               knitr, # Needed to write HTML reports
               missRanger, # To generate NAs
               tidyverse) # Powerful data wrangling package suite
```

Load and Examine a Data Set

```
# Let's load a data set from the squirrel data set
ages <- read.csv("age_gaps.csv")
  # Add a categorical group
ages_modified <-
ages %>%
mutate(Age_difference_group = ifelse(age_difference >= 0 & age_difference <= 15, "small",
                              ifelse(age_difference > 15 & age_difference <= 35, "Middle",
                                     "large")),
         Age_difference_group = fct_rev(Age_difference_group))

# What does the data look like?
ages |>
  head(20) |>
  formattable()
```

movie_name

release_year

director

age_difference

couple_number

actor_1_name

actor_2_name

character_1_gender

character_2_gender

actor_1_birthdate

actor_2_birthdate

actor_1_age

actor_2_age

Harold and Maude

1971

Hal Ashby

52

1

Ruth Gordon

Bud Cort

woman

man

1896-10-30

1948-03-29

75

23

Venus

2006

Roger Michell

50

1

Peter O'Toole

Jodie Whittaker

man

woman

1932-08-02

1982-06-03

74

24

The Quiet American

2002

Phillip Noyce

49

1

Michael Caine

Do Thi Hai Yen

man

woman

1933-03-14

1982-10-01

69

20

The Big Lebowski

1998

Joel Coen

45

1

David Huddleston

Tara Reid

man

woman

1930-09-17

1975-11-08

68

23

Beginners

2010

Mike Mills

43

1

Christopher Plummer

Goran Visnjic

man

man

1929-12-13

1972-09-09

81

38

Poison Ivy

1992

Katt Shea

42

1

Tom Skerritt

Drew Barrymore

man

woman

1933-08-25

1975-02-22

59

17

Whatever Works

2009

Woody Allen

40

1

Larry David

Evan Rachel Wood

man

woman

1947-07-02

1987-09-07

62

22

Entrapment

1999

Jon Amiel

39

1

Sean Connery

Catherine Zeta-Jones

man

woman

1930-08-25

1969-09-25

69

30

Husbands and Wives

1992

Woody Allen

38

1

Woody Allen

Juliette Lewis

man

woman

1935-12-01

1973-06-21

57

19

Magnolia

1999

Paul Thomas Anderson

38

1

Jason Robards

Julianne Moore

man

woman

1922-07-26

1960-12-03

77

39

Indiana Jones and the Last Crusade

1989

Steven Spielberg

36

1

Sean Connery

Alison Doody

man

woman

1930-08-25

1966-03-09

59

23

Mr. Peabody and the Mermaid

1948

Irving Pichel

36

1

William Powell

Ann Blyth

man

woman

1892-06-29

1928-08-16

56

20

First Knight

1995

Jerry Zucker

35

1

Sean Connery

Julia Ormond

man

woman

1930-08-25

1965-01-04

65

30

Something's Gotta Give

2003

Nancy Meyers

35

1

Jack Nicholson

Amanda Peet

man

woman

1937-04-22

1972-01-11

66

31

Eternal Sunshine of the Spotless Mind

2004

Michel Gondry

34

1

Tom Wilkinson

Kirsten Dunst

man

woman

1948-02-05

1982-04-30

56

22

Lost in Translation

2003

Sofia Coppola

34

1

Bill Murray

Scarlett Johansson

man

woman

1950-09-21

1984-11-22

53

19

Shopgirl

2005

Anand Tucker

34

1

Steve Martin

Claire Danes

man

woman

1945-08-14

1979-04-12

60

26

Wild Target

2010

Jonathan Lynn

34

1

Bill Nighy

Emily Blunt

man

woman

1949-12-12

1983-02-23

61

27

Fort Apache, The Bronx

1981

Daniel Petrie

33

1

Paul Newman

Rachel Ticotin

man

woman

1925-01-26

1958-11-01

56

23

Hollywood Ending

2002

Woody Allen

33

1

Woody Allen

Debra Messing

man

woman

1935-12-01

1968-08-15

67

34

Data Normality

Data normality, in statistics, refers to the assumption or property that data follows a normal distribution, also known as a Gaussian distribution. The normal distribution is a specific probability distribution characterized by a symmetric, bell-shaped curve.

Describing Properties of our Data (Refined)

Skewness is a statistical measure that describes the asymmetry or lack of symmetry in a data set's distribution. It quantifies the degree to which the data deviates from a perfectly symmetrical distribution.

```
ages_modified |>
  select(actor_1_age, actor_2_age, age_difference) |> #check skewness of the actor's ages
  describe() |>
  select(described_variables, skewness) |>
  formattable()
```

described_variables

skewness

actor_1_age

0.59

actor_2_age

0.98

age_difference

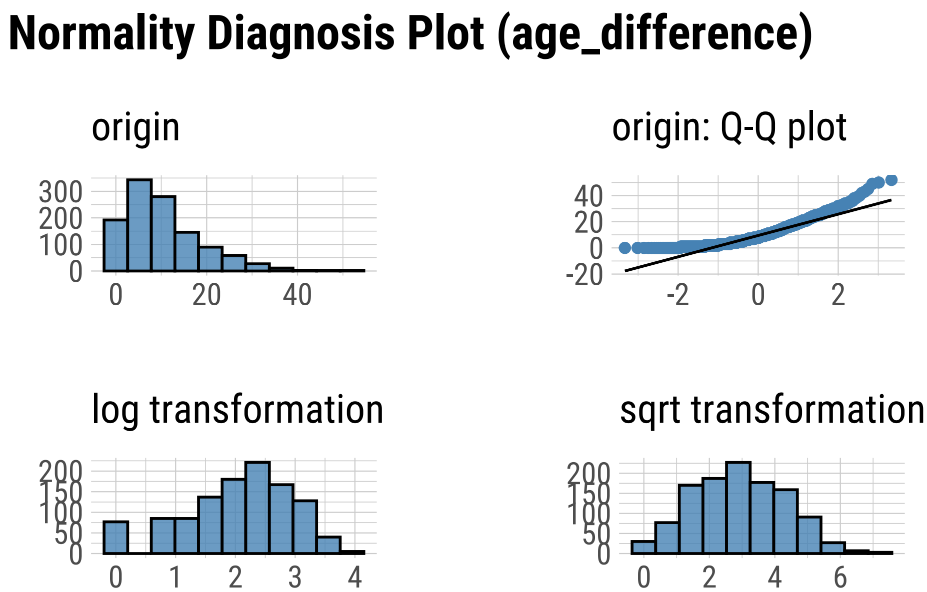
1.20

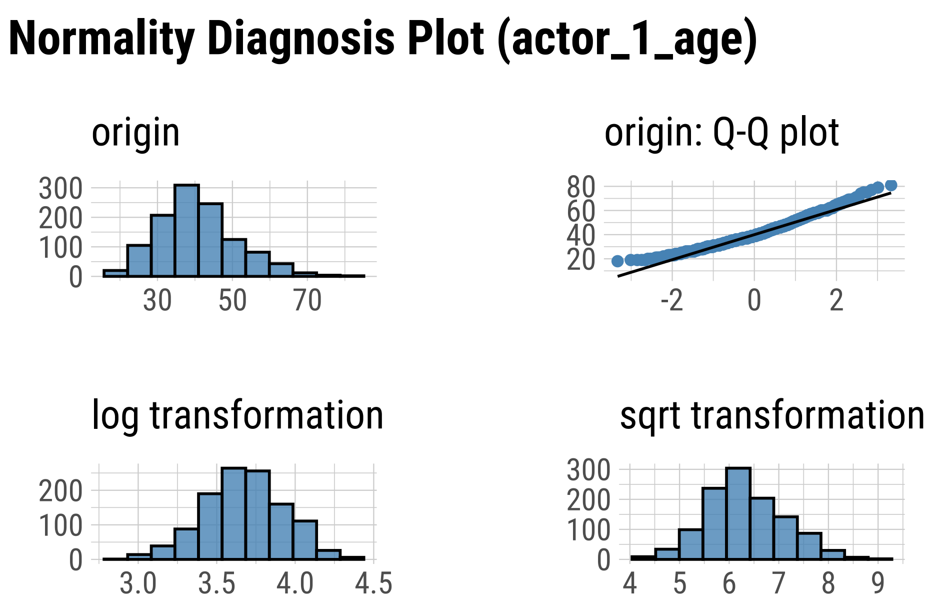
Testing Normality (Accelerated)

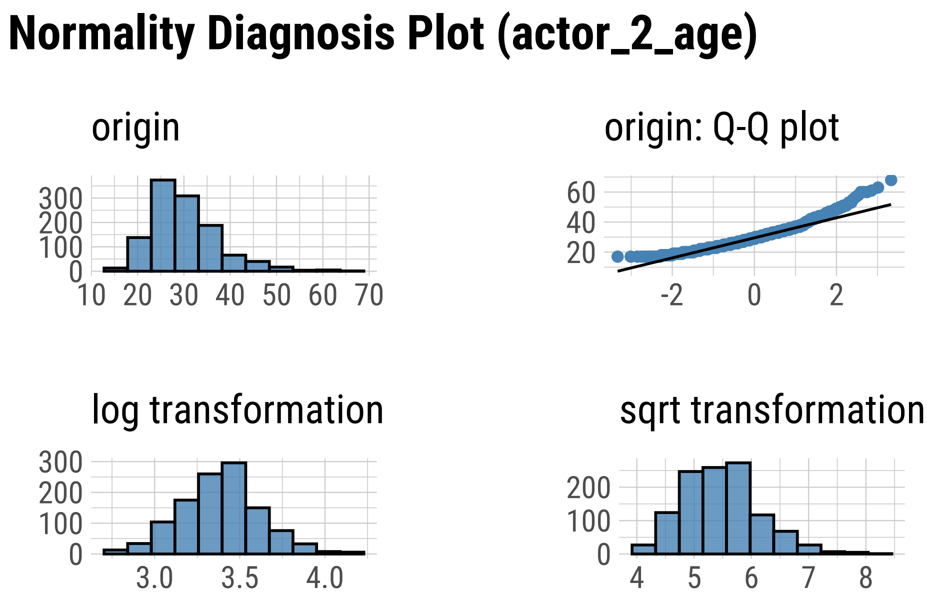
Q-Q Plots

A Quantile-Quantile plot, commonly known as a Q-Q plot, is a graphical tool used in statistics to assess whether a dataset follows a particular theoretical distribution, typically the normal distribution.

```
ages_modified |>
plot_normality(age_difference,actor_1_age, actor_2_age) # a Q-Q plot for 'age_difference'
```


Normality Diagnosis Plot (age_difference)
origin
origin: Q-Q plot
log transformation
sqrt transformation


Normality Diagnosis Plot (actor_1_age)
origin
origin: Q-Q plot
log transformation
sqrt transformation



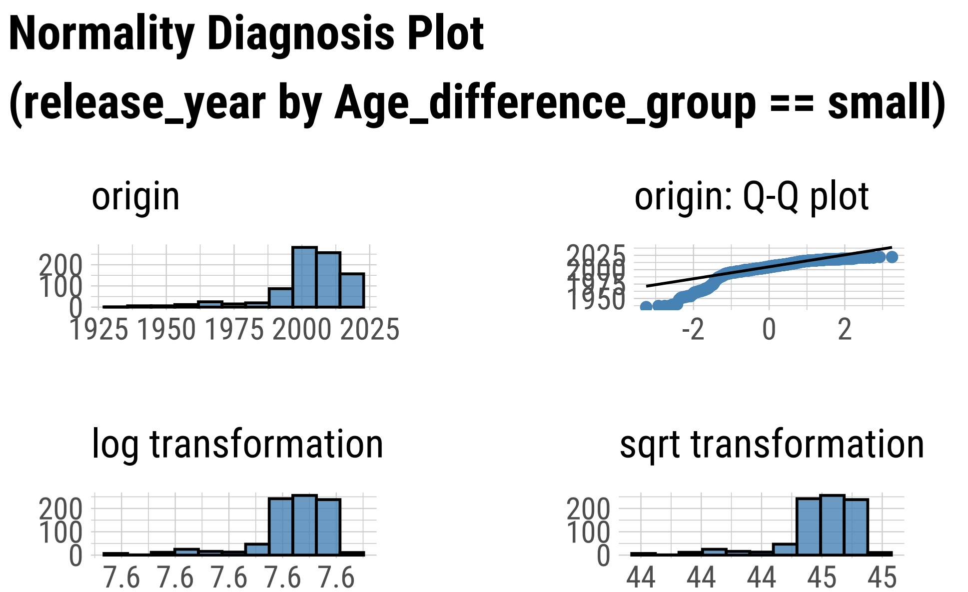
Normality within Groups

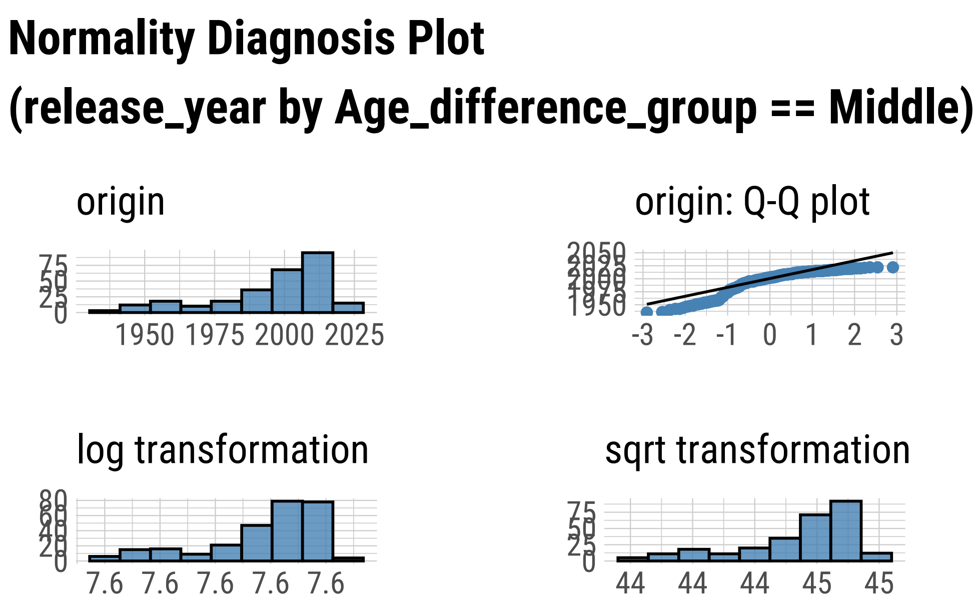
When you want to assess the normality of data within groups, you are typically dealing with data that is organized into subgroups or categories, and you want to determine if the data within each subgroup follows a normal distribution.

Looking within Age_group at the subgroup normality

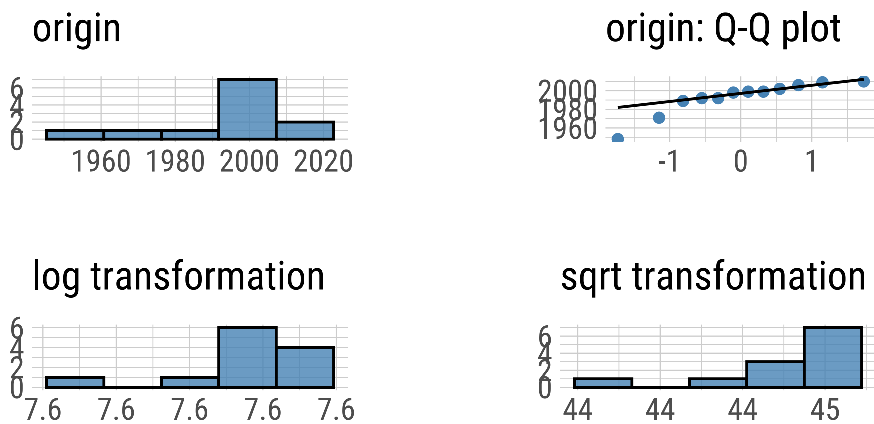
Q-Q Plots

```
ages_modified %>%
  group_by(Age_difference_group) %>% #plotting the graphs according to age group categorie
  select(release_year, couple_number) %>%
  plot_normality()
```

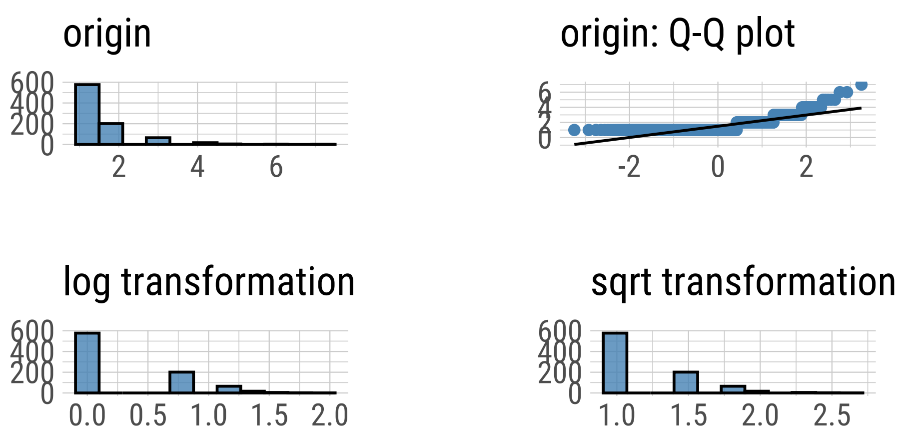

Normality Diagnosis Plot
(release_year by Age_difference_group == small)
origin
origin: Q-Q plot
log transformation
sqrt transformation
200
100
0
1925 1950 1975 2000 2025
-2 0 2
7.6 7.6 7.6 7.6 7.6
44 44 44 45 45


Normality Diagnosis Plot
(release_year by Age_difference_group == Middle)
origin
origin: Q-Q plot
log transformation
sqrt transformation
1950 1975 2000 2025
-3 -2 -1 0 1 2 3
7.6 7.6 7.6 7.6 7.6
44 44 44 45 45

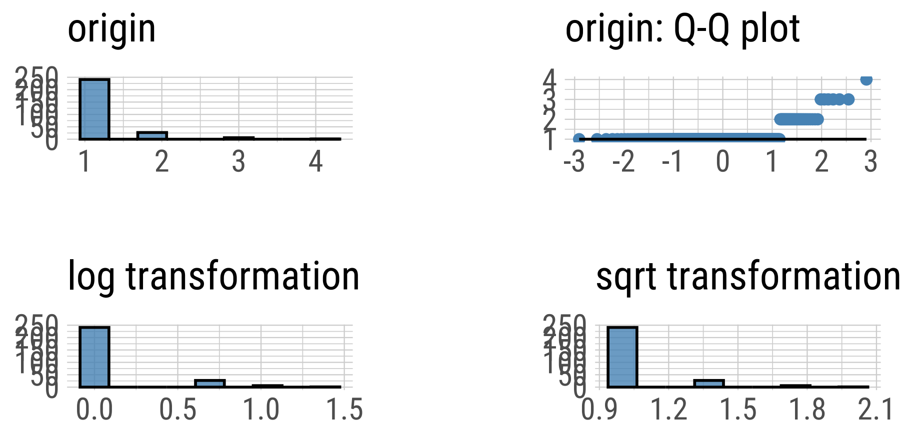
Normality Diagnosis Plot
(release_year by Age_difference_group == large)



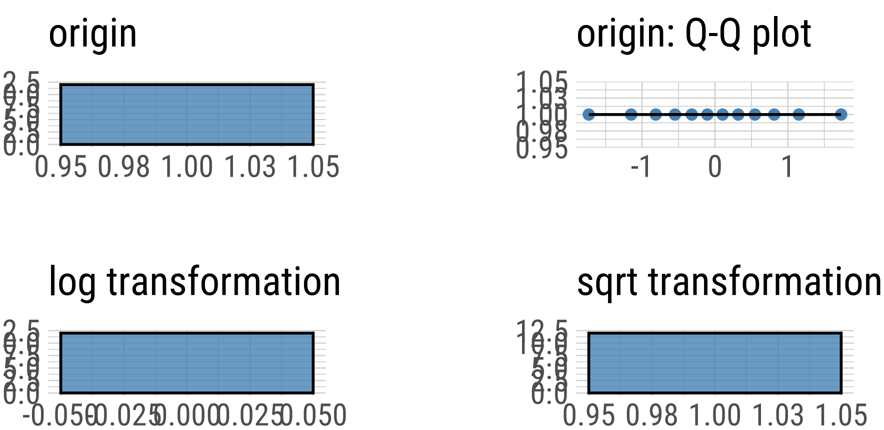
Normality Diagnosis Plot
(couple_number by Age_difference_group == small)



Normality Diagnosis Plot
(couple_number by Age_difference_group == Middl



Normality Diagnosis Plot
(couple_number by Age_difference_group == large)



Transforming Data

We will try to transform the age_difference column with through several approaches and discuss the pros and cons of each. First however, we will remove `0` values, because age_difference values.

```
InsMod <- ages_modified |>
  filter(age_difference > 0)
```

Square-root Transformation

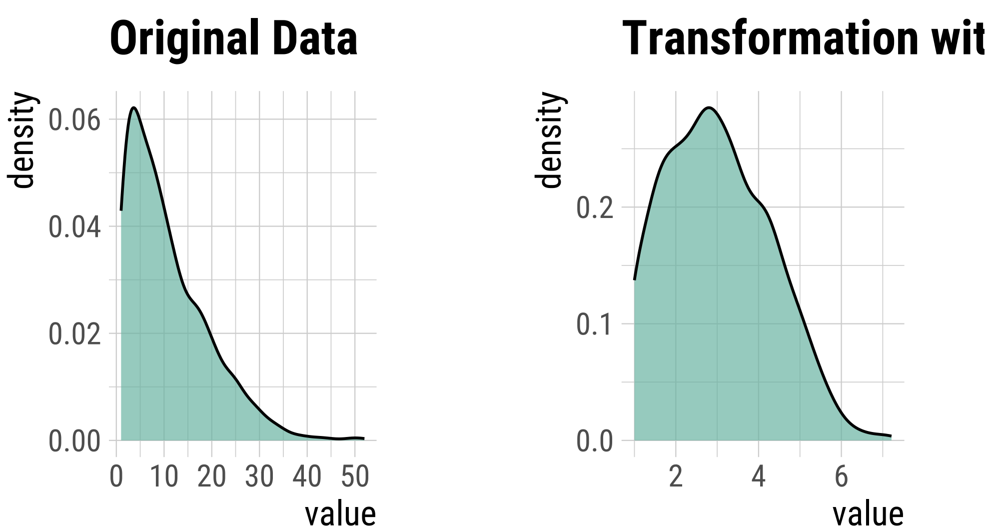
In R, you can perform a square root transformation on a variable in your data set to make its distribution closer to normal or to stabilize variance. This transformation is often used when dealing with data that exhibits a right-skewed distribution.

```
# Transforming the age_difference column using Square-root Transformation
sqrtIns <- transform(InsMod$age_difference, method = "sqrt")

summary(sqrtIns)
```

```
* Resolving Skewness with sqrt

* Information of Transformation (before vs after)
            Original Transformation
n            1125.00       1125.000
na              0.00          0.000
mean           10.70          3.016
sd              8.45          1.269
se_mean         0.25          0.038
IQR            12.00          2.000
skewness        1.22          0.362
kurtosis        1.62         -0.487
p00             1.00          1.000
p01             1.00          1.000
p05             1.00          1.000
p10             2.00          1.414
p20             3.00          1.732
p25             4.00          2.000
p30             5.00          2.236
p40             7.00          2.646
p50             8.00          2.828
```

```
p60          11.00           3.317
p70          14.00           3.742
p75          16.00           4.000
p80          17.00           4.123
p90          23.00           4.796
p95          27.00           5.196
p99          35.76           5.980
p100         52.00           7.211
```

```r
sqrtIns |>
  plot() # plotting the transformed data by using square root transformation
```



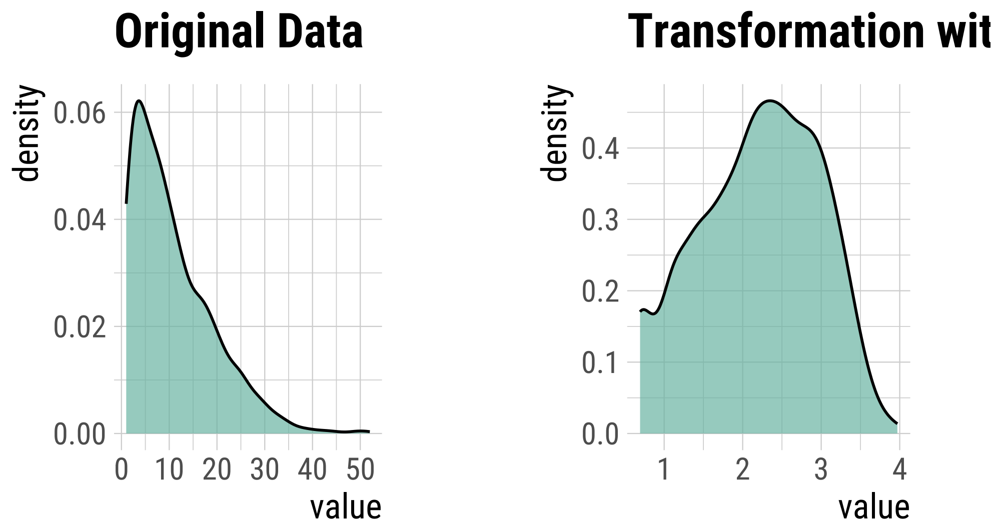
Logarithmic (+1) Transformation

A logarithmic transformation with a "+1" added to each value is a common data transformation used to address issues related to skewness or to stabilize variance in data. It's particularly useful when dealing with data that has positive values, including zero. The "+1" addition is used to handle cases where the data contains zero values because the logarithm of zero is undefined.

```r
# Transforming the age_difference column using Logarithmic Transformation
Log1Ins <- transform(InsMod$age_difference, method = "log+1")

summary(Log1Ins)
```

```
* Resolving Skewness with log+1

* Information of Transformation (before vs after)
         Original Transformation
n         1125.00       1125.000
na           0.00          0.000
mean        10.70          2.187
sd           8.45          0.773
se_mean      0.25          0.023
IQR         12.00          1.224
skewness     1.22         -0.231
kurtosis     1.62         -0.787
p00          1.00          0.693
p01          1.00          0.693
p05          1.00          0.693
p10          2.00          1.099
p20          3.00          1.386
p25          4.00          1.609
p30          5.00          1.792
p40          7.00          2.079
p50          8.00          2.197
p60         11.00          2.485
p70         14.00          2.708
p75         16.00          2.833
p80         17.00          2.890
p90         23.00          3.178
p95         27.00          3.332
p99         35.76          3.604
p100        52.00          3.970
```

```r
Log1Ins |>
  plot()
```



Squared Transformation

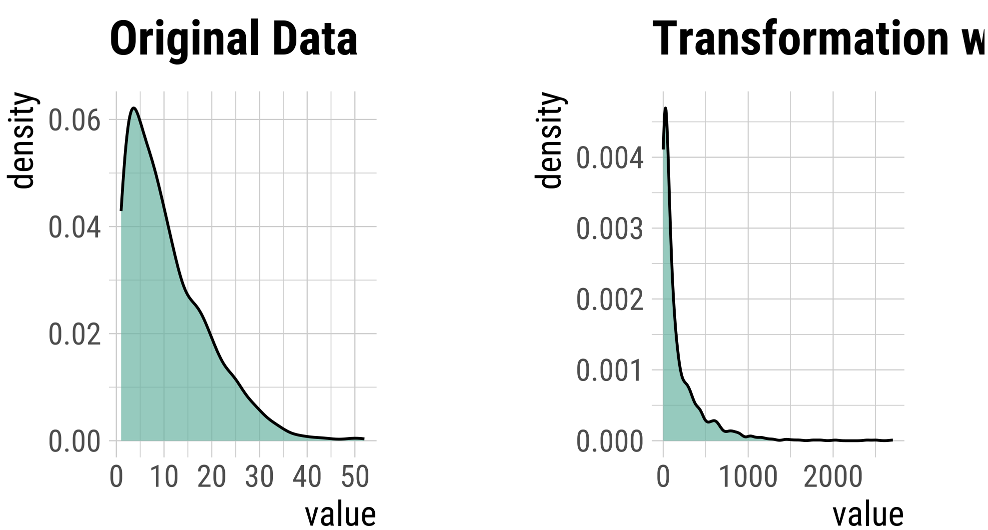
A squared transformation is a data transformation that involves taking the square of each value in a data set. This transformation is often used to emphasize the differences between values and can be useful in various statistical analyses and modeling techniques.

```
# Transforming the age_difference column using Squared Transformation
SqrdIns <- transform(InsMod$age_difference, method = "x^2")

summary(SqrdIns)
```

```
* Resolving Skewness with x^2

* Information of Transformation (before vs after)
         Original Transformation
n         1125.00         1125.0
na           0.00            0.0
mean        10.70          185.9
sd           8.45          287.0
se_mean      0.25            8.6
IQR         12.00          240.0
```

```
skewness       1.22           3.4
kurtosis       1.62          17.1
p00            1.00           1.0
p01            1.00           1.0
p05            1.00           1.0
p10            2.00           4.0
p20            3.00           9.0
p25            4.00          16.0
p30            5.00          25.0
p40            7.00          49.0
p50            8.00          64.0
p60           11.00         121.0
p70           14.00         196.0
p75           16.00         256.0
p80           17.00         289.0
p90           23.00         529.0
p95           27.00         729.0
p99           35.76        1279.0
p100          52.00        2704.0
```

```
SqrdIns |>
  plot()
```



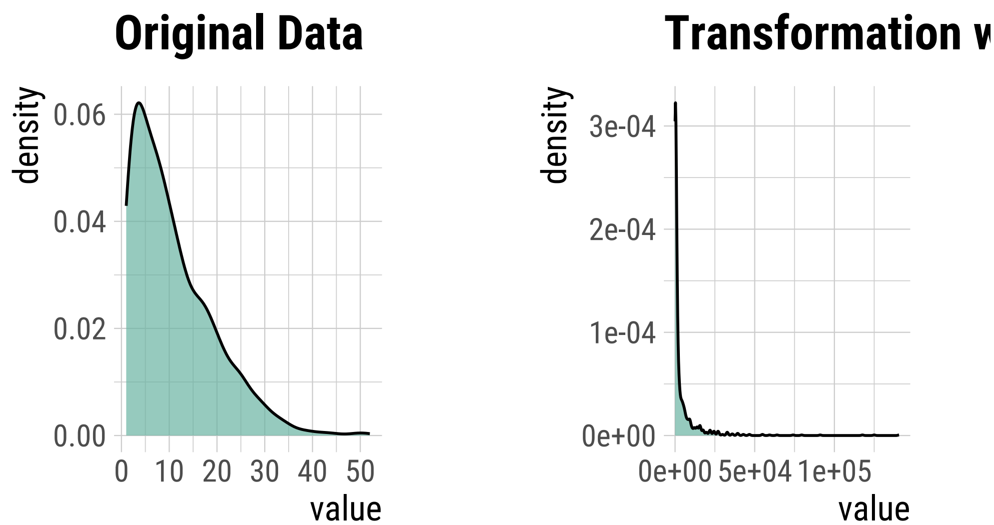
Cubed Transformation

A cubed transformation is a data transformation that involves taking the cube of each value in a data set. This transformation is used to emphasize nonlinear relationships between variables or to create more pronounced distinctions between values. Similar to squared transformations, cubed transformations can be applied to variables for various purposes, including modeling, data normalization, or addressing data skewness.

```r
# Transforming the age_difference column using Cubed Transformation
CubeIns  <- transform(InsMod$age_difference, method = "x^3")

summary(CubeIns)
```

```
* Resolving Skewness with x^3

* Information of Transformation (before vs after)
         Original Transformation
n         1125.00        1.1e+03
na           0.00        0.0e+00
mean        10.70        4.2e+03
sd           8.45        1.1e+04
se_mean      0.25        3.2e+02
IQR         12.00        4.0e+03
skewness     1.22        6.5e+00
kurtosis     1.62        5.9e+01
p00          1.00        1.0e+00
p01          1.00        1.0e+00
p05          1.00        1.0e+00
p10          2.00        8.0e+00
p20          3.00        2.7e+01
p25          4.00        6.4e+01
p30          5.00        1.2e+02
p40          7.00        3.4e+02
p50          8.00        5.1e+02
p60         11.00        1.3e+03
p70         14.00        2.7e+03
p75         16.00        4.1e+03
p80         17.00        4.9e+03
p90         23.00        1.2e+04
p95         27.00        2.0e+04
p99         35.76        4.6e+04
p100        52.00        1.4e+05
```

```
CubeIns |>
  plot()
```



Box-cox Transformation

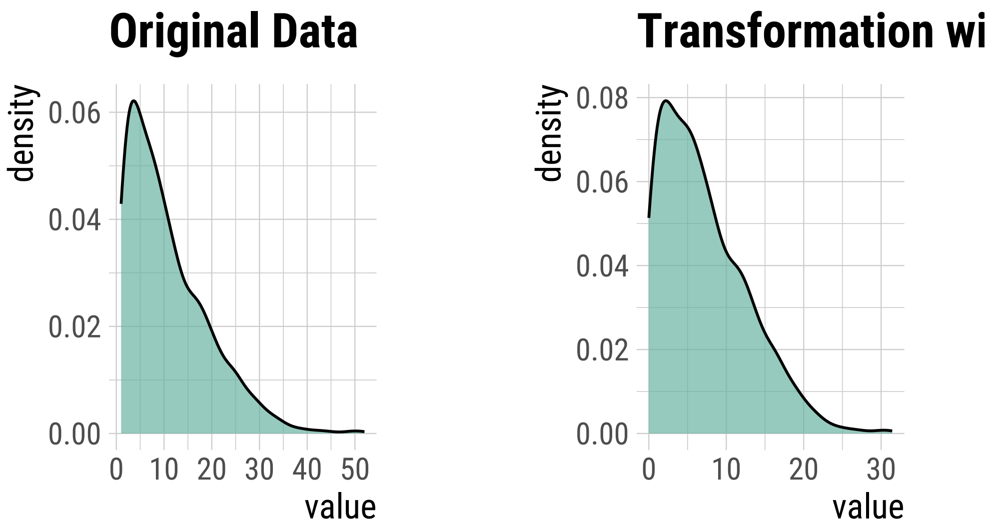
The Box-Cox transformation is a family of power transformations that are used to stabilize variance and make a data set more closely approximate a normal distribution. It is particularly useful when dealing with data that exhibits heteroscedasticity (varying levels of variance across different levels of the independent variable) or data that does not meet the assumptions of normality.

```
# Transforming the age_difference column using Box-cox Transformation
BoxCoxIns <- transform(InsMod$age_difference, method = "Box-Cox")

summary(BoxCoxIns)
```

```
* Resolving Skewness with Box-Cox

* Information of Transformation (before vs after)
            Original Transformation
n            1125.00        1125.00
```

```
na           0.00         0.00
mean        10.70         7.14
sd           8.45         5.60
se_mean      0.25         0.17
IQR         12.00         8.35
skewness     1.22         0.93
kurtosis     1.62         0.66
p00          1.00         0.00
p01          1.00         0.00
p05          1.00         0.00
p10          2.00         0.94
p20          3.00         1.80
p25          4.00         2.62
p30          5.00         3.40
p40          7.00         4.90
p50          8.00         5.62
p60         11.00         7.70
p70         14.00         9.68
p75         16.00        10.97
p80         17.00        11.60
p90         23.00        15.29
p95         27.00        17.65
p99         35.76        22.65
p100        52.00        31.43
```

```
BoxCoxIns |>
  plot()
```



Imputing like a Data Scientist

Required Setup

```
pacman::p_load(colorblindr, # Colorblind friendly pallettes
               cluster, # K cluster analyses
               dlookr, # Exploratory data analysis
               formattable, # HTML tables from R outputs
               ggfortify, # Plotting tools for stats
               ggpubr, # Publishable ggplots
               here, # Standardizes paths to data
               kableExtra, # Alternative to formattable
               knitr, # Needed to write HTML reports
               missRanger, # To generate NAs
               plotly, # Visualization package
               rattle, # Decision tree visualization
               rpart, # rpart algorithm
               tidyverse, # Powerful data wrangling package suite
               visdat) # Another EDA visualization package
```

```
# Set global ggplot() theme
# Theme pub_clean() from the ggpubr package with base text size = 16
theme_set(theme_pubclean(base_size = 16))
# All axes titles to their respective far right sides
theme_update(axis.title = element_text(hjust = 1))
# Remove axes ticks
theme_update(axis.ticks = element_blank())
# Remove legend key
theme_update(legend.key = element_blank())
```

Diagnose your Data

diagnose() allows you to diagnose variables on a data frame. Like any other `dplyr` functions, the first argument is the tibble (or data frame). The second and subsequent arguments refer to variables within the data frame.

The variables of the `tbl_df` object returned by **diagnose()** are as

- `variables` : variable names
- `types` : the data type of the variables
- `missing_count` : number of missing values
- `missing_percent` : percentage of missing values
- `unique_count` : number of unique values
- `unique_rate` : rate of unique value. unique_count / number of observation

```
# What are the properties of the data
ages_modified |>
  diagnose() |>
  formattable()
```

variables

types

missing_count

missing_percent

unique_count

unique_rate

movie_name

character

0

0

830

0.7186

release_year

integer

0

0

82

0.0710

director

character

0

0

510

0.4416

age_difference

integer

0

0

46

0.0398

couple_number

integer

0

0

7

0.0061

actor_1_name

character

0

0

567

0.4909

actor_2_name

character

0

0

647

0.5602

character_1_gender

character

0

0

2

0.0017

character_2_gender

character

0

0

2

0.0017

actor_1_birthdate

character

0

0

562

0.4866

actor_2_birthdate

character

0

0

640

0.5541

actor_1_age

integer

0

0

59

0.0511

actor_2_age

integer

0

0

45

0.0390

Age_difference_group

factor

0

0

3

0.0026

Diagnose Outliers

The diagnose_outlier() produces outlier information for diagnosing the quality of the numerical data.

```
# Table showing outliers
ages_modified |>
  diagnose_outlier() |>
  filter(outliers_ratio > 0) |>
  mutate(rate = outliers_mean / with_mean) |>
  arrange(desc(rate)) |>
  select(-outliers_cnt) |>
  formattable()
```

variables

outliers_ratio

outliers_mean

with_mean

without_mean

rate

age_difference

2.3

37.2

10.4

9.8

3.57

couple_number

2.3

4.5

1.4

1.3

3.19

actor_1_age

1.1

73.0

40.6

40.3

1.80

actor_2_age

2.7

53.2

30.2

29.6

1.76

release_year

9.3

1958.7

2000.8

2005.1

0.98

```
# Boxplots and histograms of data with and without outliers
ages_modified|>
  select(find_outliers(ages_modified)) |>
          plot_outlier()
```

```
There is no numeric variable in the data or variable list.
```

```
#There is no numeric value in the data set
```

Basic Exploration of Missing Values (NAs)

this code takes an existing data set, introduces missing values into it with a 30% probability, and stores the resulting data set with missing values in a new variable called na.ages_modified.

```r
# Randomly generate NAs for 30
na.ages_modified <- ages_modified |>
  generateNA(p = 0.3) #roughly 30% of the values in dataset will be replaced with missing

# First six rows
na.ages_modified |>
head() |>
  formattable()
```

movie_name

release_year

director

age_difference

couple_number

actor_1_name

actor_2_name

character_1_gender

character_2_gender

actor_1_birthdate

actor_2_birthdate

actor_1_age

actor_2_age

Age_difference_group

Harold and Maude

1971

NA

NA

1

Ruth Gordon

Bud Cort

woman

man

NA

1948-03-29

NA

23

large

Venus

2006

Roger Michell

50

1

Peter O'Toole

Jodie Whittaker

NA

NA

1932-08-02

1982-06-03

74

NA

NA

The Quiet American

2002

NA

49

NA

Michael Caine

Do Thi Hai Yen

man

NA

1933-03-14

NA

69

20

large

The Big Lebowski

NA

Joel Coen

45

1

David Huddleston

Tara Reid

man

NA

NA

1975-11-08

68

23

large

Beginners

2010

Mike Mills

43

1

Christopher Plummer

Goran Visnjic

man

NA

1929-12-13

1972-09-09

81

38

NA

Poison Ivy

1992

Katt Shea

42

1

Tom Skerritt

NA

NA

NA

NA

1975-02-22

59

NA

large

```
# Create the NA table
na.ages_modified |>
  plot_na_pareto(only_na = TRUE, plot = FALSE) |>
  formattable() # Publishable table
```

variable

frequencies

ratio

grade

cumulative

Age_difference_group

346

0.3

Bad

7.1

actor_1_age

346

0.3

Bad

14.3

actor_1_birthdate

346

0.3

Bad

21.4

actor_1_name

346

0.3

Bad

28.6

actor_2_age

346

0.3

Bad

35.7

actor_2_birthdate

346

0.3

Bad

42.9

actor_2_name

346

0.3

Bad

50.0

age_difference

346

0.3

Bad

57.1

character_1_gender

346

0.3

Bad

64.3

character_2_gender

346

0.3

Bad

71.4

couple_number

346

0.3

Bad

78.6

director

346

0.3

Bad

85.7

movie_name

346

0.3

Bad

92.9

release_year

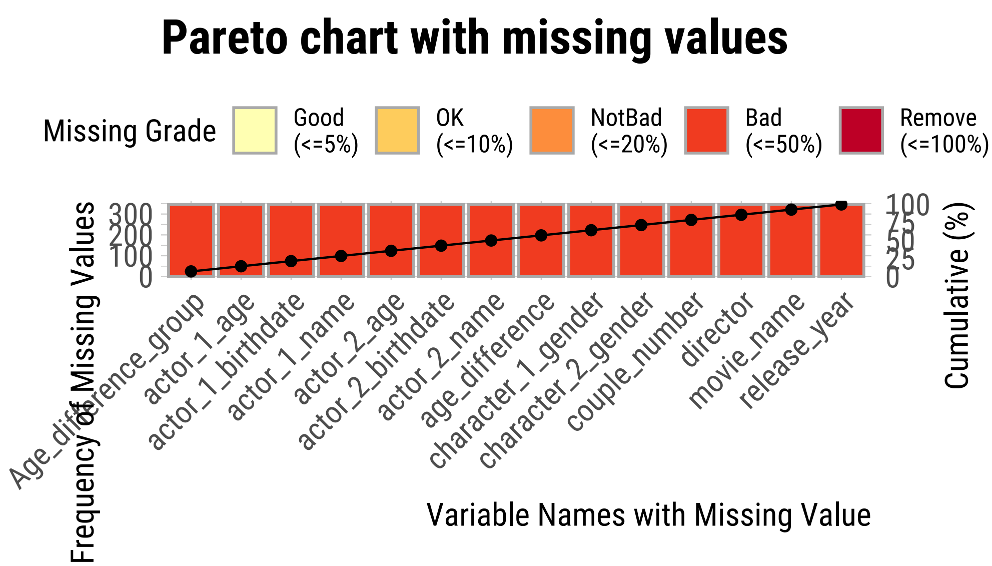
346

0.3

Bad

100.0

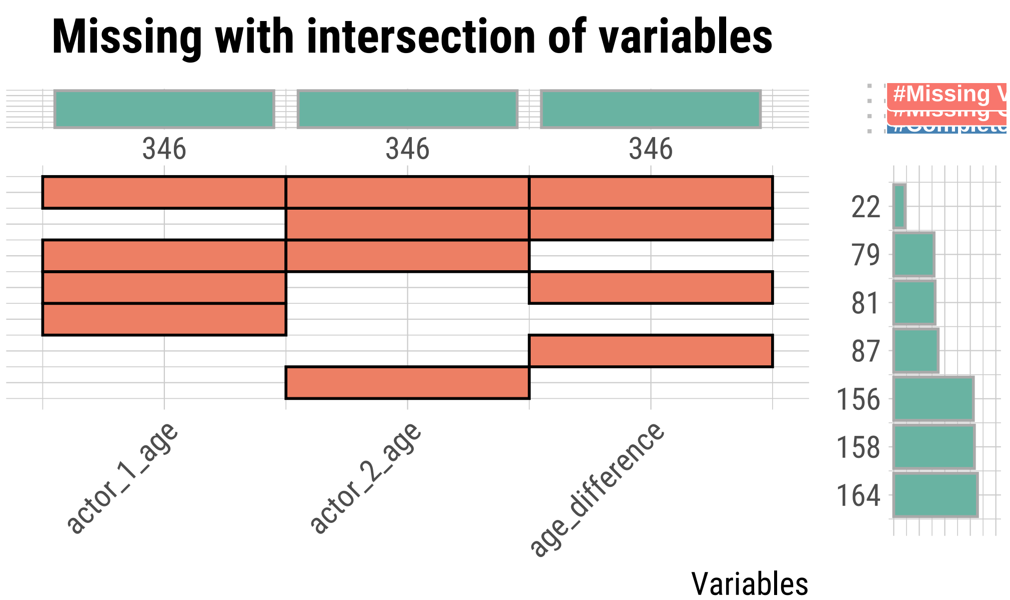
```
# Plot the insersect of the columns with missing values
# This plot visualizes the table above
na.ages_modified |>
  plot_na_pareto(only_na = TRUE)
```



Advanced Exploration of Missing Values (NAs)

The **vis_miss()** function is part of the **visdat** package in R, which is used for visualizing missing data in a data set. The **vis_miss()** function uses color-coding to represent missing values in your data, making it easier to identify patterns of missing values.

```
na.ages_modified |>
  select(actor_1_age, actor_2_age, age_difference) |>
  plot_na_intersect(only_na = TRUE)
```




```
# Interactive plotly() plot of all NA values to examine every row
#na.ages_modified |>
# select(actor_1_age, actor_2_age, Age_difference_group) |>
# vis_miss() |>
# ggplotly()
# This chunk is running properly but not able to render.
```

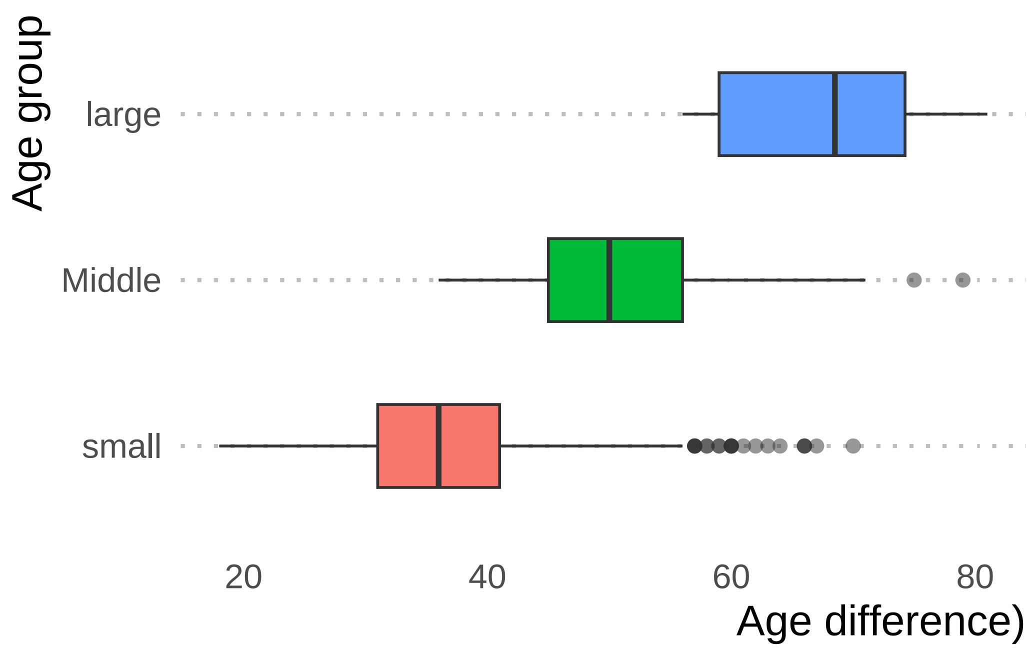
Impute Outliers and NAs

Classifying Outliers

Classifying outliers involves identifying data points that deviate significantly from the majority of the data. Outliers can be of different types, such as univariate outliers (outliers in a single variable) or multivariate outliers (outliers when considering multiple variables simultaneously).

Here we will use group_by operation to create group based on age difference

```
# Box plot
ages_modified %>% # Set the simulated normal data as a data frame
  ggplot(aes(x = actor_1_age, y = Age_difference_group, fill = Age_difference_group)) + #
  geom_boxplot(width = 0.5, outlier.size = 2, outlier.alpha = 0.5) +
  xlab("Age difference)") +
  ylab("Age group") +
  theme(legend.position = "none")
```



Mean Imputation

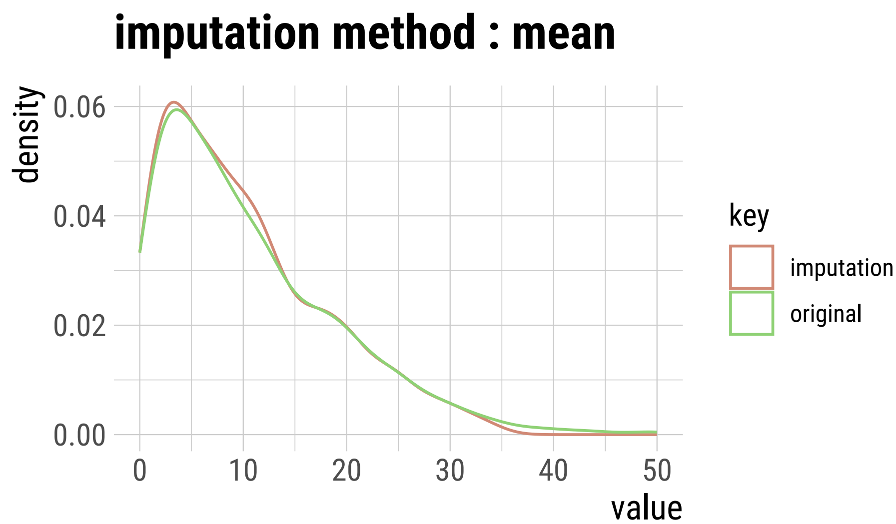
Mean imputation is a simple method for handling missing data in a dataset by replacing missing values with the mean (average) value of the non-missing values for that variable.

```
# Raw summary, output suppressed
mean_out_imp_age <- na.ages_modified |>
  select(age_difference) |>
  imputate_outlier(age_difference, method = "mean")

# Output showing the summary statistics of our imputation
mean_out_imp_age |>
  summary()
```

```
Impute outliers with mean

* Information of Imputation (before vs after)
                    Original Imputation
described_variables "value"  "value"
n                   "809"    "809"
na                  "346"    "346"
mean                "11"     "10"
sd                  "8.7"    "7.8"
se_mean             "0.31"   "0.28"
IQR                 "12"     "11"
skewness            "1.19"   "0.87"
kurtosis            "1.456"  "0.041"
p00                 "0"      "0"
p01                 "0"      "0"
p05                 "1"      "1"
p10                 "2"      "2"
p20                 "3"      "3"
p25                 "4"      "4"
p30                 "5"      "5"
p40                 "7"      "7"
p50                 "8"      "8"
p60                 "11"     "11"
p70                 "13"     "13"
p75                 "16"     "15"
p80                 "18"     "17"
p90                 "23"     "21"
p95                 "28"     "25"
```

```
p99                    "38"      "32"
p100                   "50"      "34"
```

```
# Visualization of the mean imputation
mean_out_imp_age |>
  plot()
```



Median Imputation

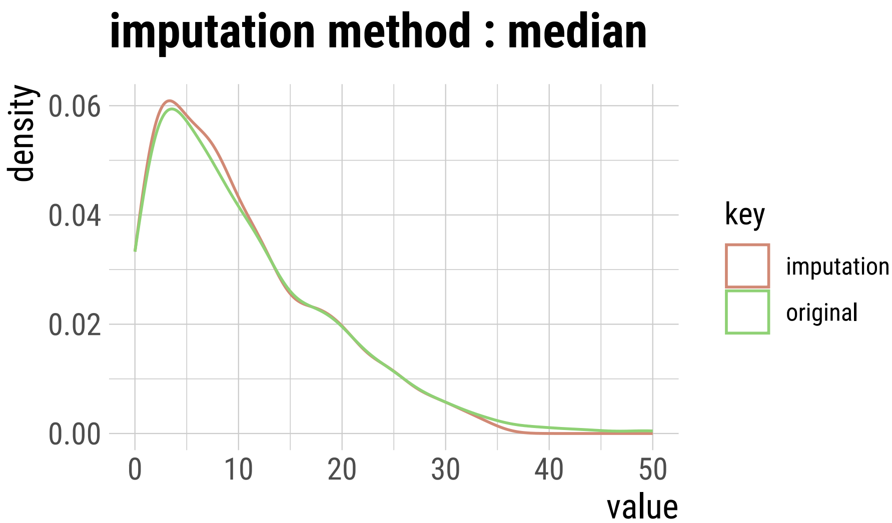
Median imputation is a method for handling missing data in a dataset by replacing missing values with the median value of the non-missing values for that variable. Median imputation is an alternative to mean imputation and can be useful when dealing with skewed or non-normally distributed data, as it is less sensitive to extreme values.

```
# Raw summary, output suppressed
med_out_imp_age <- na.ages_modified |>
  select(age_difference) |>
  imputate_outlier(age_difference, method = "median")

# Output showing the summary statistics of our imputation
med_out_imp_age |>
```

```r
  summary()
```

```
Impute outliers with median

* Information of Imputation (before vs after)
                    Original Imputation
described_variables "value"  "value"
n                   "809"    "809"
na                  "346"    "346"
mean                "11"     "10"
sd                  "8.7"    "7.8"
se_mean             "0.31"   "0.28"
IQR                 "12"     "11"
skewness            "1.19"   "0.88"
kurtosis            "1.456"  "0.052"
p00                 "0"      "0"
p01                 "0"      "0"
p05                 "1"      "1"
p10                 "2"      "2"
p20                 "3"      "3"
p25                 "4"      "4"
p30                 "5"      "5"
p40                 "7"      "7"
p50                 "8"      "8"
p60                 "11"     "10"
p70                 "13"     "13"
p75                 "16"     "15"
p80                 "18"     "17"
p90                 "23"     "21"
p95                 "28"     "25"
p99                 "38"     "32"
p100                "50"     "34"
```

```r
# Visualization of the median imputation
med_out_imp_age |>
  plot()
```



Mode Imputation

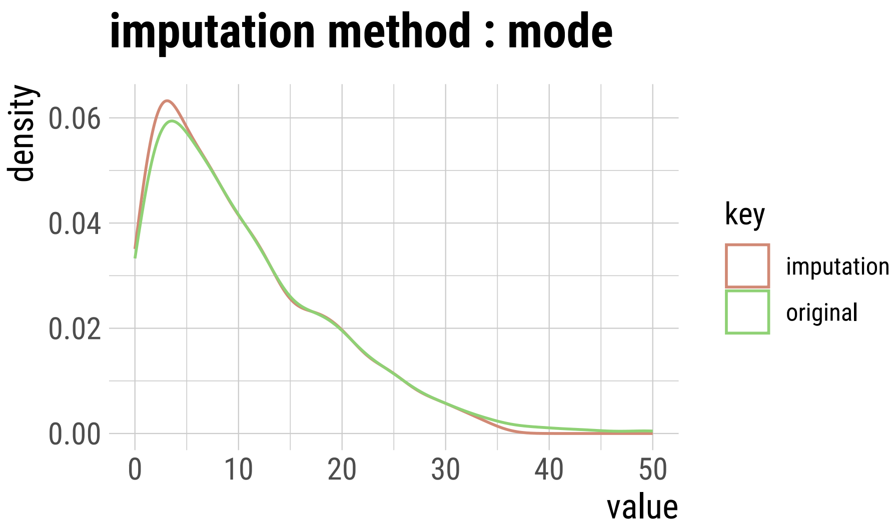
Mode imputation is a method for handling missing data in a dataset by replacing missing values with the mode, which is the most frequently occurring value, of the non-missing values for that variable. Mode imputation is typically used for categorical or nominal data where the concept of "average" (as in mean or median) does not apply.

```
# Raw summary, output suppressed
mode_out_imp_age <- na.ages_modified |>
  select(age_difference) |>
  imputate_outlier(age_difference, method = "mode")

# Output showing the summary statistics of our imputation
mode_out_imp_age |>
  summary()
```

```
Impute outliers with mode

* Information of Imputation (before vs after)
                     Original Imputation
described_variables "value"  "value"
```

```
n                    "809"    "809"
na                   "346"    "346"
mean                 "11"     "10"
sd                   "8.7"    "7.9"
se_mean              "0.31"   "0.28"
IQR                  "12"     "12"
skewness             "1.19"   "0.88"
kurtosis             "1.46"   "0.02"
p00                  "0"      "0"
p01                  "0"      "0"
p05                  "1"      "1"
p10                  "2"      "2"
p20                  "3"      "3"
p25                  "4"      "3"
p30                  "5"      "4"
p40                  "7"      "6"
p50                  "8"      "8"
p60                  "11"     "10"
p70                  "13"     "13"
p75                  "16"     "15"
p80                  "18"     "17"
p90                  "23"     "21"
p95                  "28"     "25"
p99                  "38"     "32"
p100                 "50"     "34"
```

```
# Visualization of the mode imputation
mode_out_imp_age |>
plot()
```



Capping Imputation (aka Winsorizing)

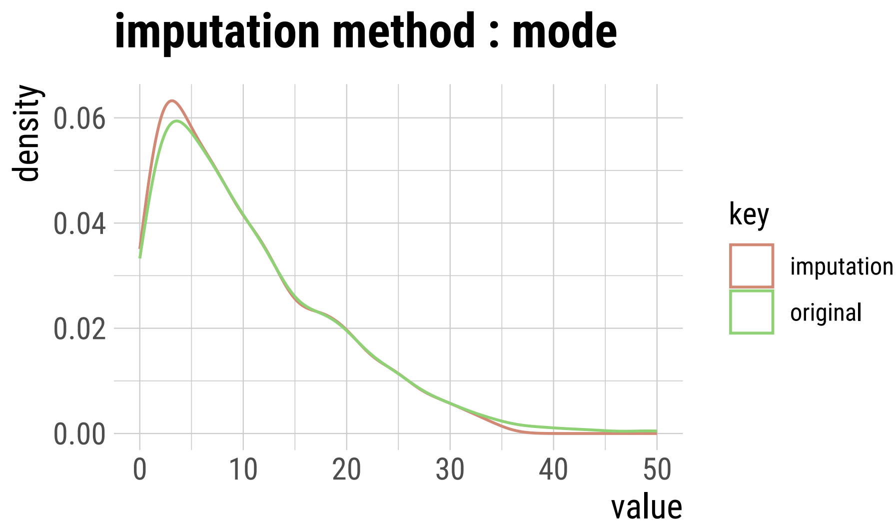
Capping imputation, also known as Winsorizing, is a data preprocessing technique used to handle outliers in a dataset by capping or limiting extreme values at a certain threshold. This method is particularly useful when you want to mitigate the impact of outliers without removing them entirely from the dataset.undefined.

```
# Raw summary, output suppressed
cap_out_imp_age <- na.ages_modified |>
  select(age_difference) |>
  imputate_outlier(age_difference, method = "mode")

# Output showing the summary statistics of our imputation
cap_out_imp_age |>
  summary()
```

```
Impute outliers with mode

* Information of Imputation (before vs after)
                   Original Imputation
described_variables "value"  "value"
```

```
n                     "809"    "809"
na                    "346"    "346"
mean                  "11"     "10"
sd                    "8.7"    "7.9"
se_mean               "0.31"   "0.28"
IQR                   "12"     "12"
skewness              "1.19"   "0.88"
kurtosis              "1.46"   "0.02"
p00                   "0"      "0"
p01                   "0"      "0"
p05                   "1"      "1"
p10                   "2"      "2"
p20                   "3"      "3"
p25                   "4"      "3"
p30                   "5"      "4"
p40                   "7"      "6"
p50                   "8"      "8"
p60                   "11"     "10"
p70                   "13"     "13"
p75                   "16"     "15"
p80                   "18"     "17"
p90                   "23"     "21"
p95                   "28"     "25"
p99                   "38"     "32"
p100                  "50"     "34"
```

```
# Visualization of the capping imputation
cap_out_imp_age |>
  plot()
```



K-Nearest Neighbor (KNN) Imputation

K-Nearest Neighbor (KNN) imputation is a technique used to fill in missing values in a dataset by estimating them based on the values of their nearest neighbors. This method is particularly useful when you want to impute missing values in a multivariate context, considering the relationships between variables.

```
if (!require(factoextra))
install.packages("factoextra")
```

```
Loading required package: factoextra
```

```
Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

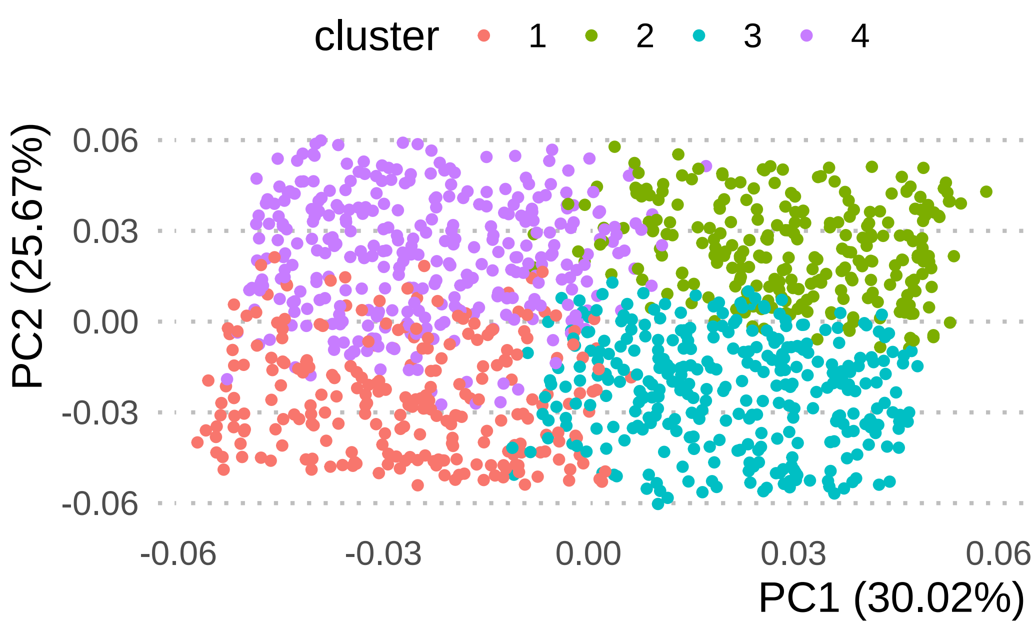
```
library(factoextra)
#check for missing values
any(is.na(ages_modified))
```

```
[1] FALSE
```

```r
#Check for infinite values
any(is.infinite(ages_modified$age_difference))
```

```
[1] FALSE
```

```r
#Impute missing values
ages_modified <- na.omit(ages_modified)
autoplot(clara(ages_modified[-14], 4))
```

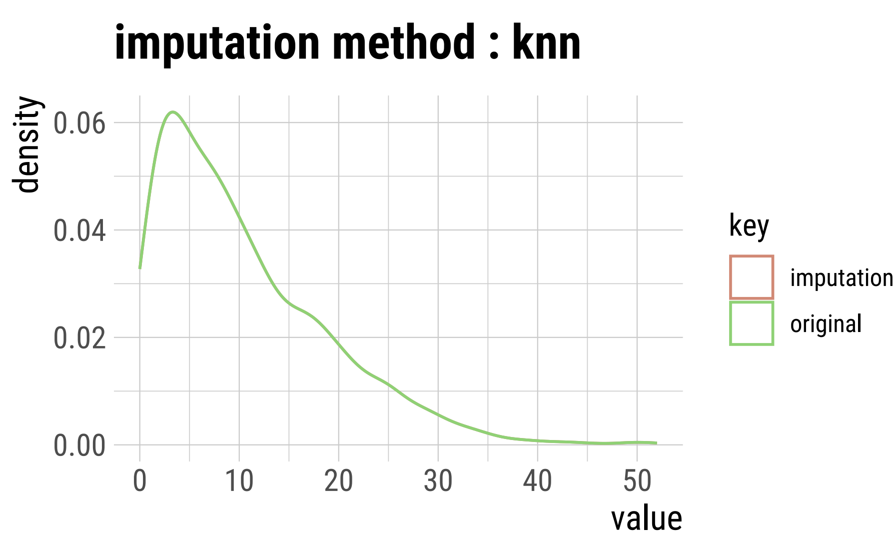



```r
library(magrittr)
```

```
Attaching package: 'magrittr'

The following object is masked from 'package:purrr':

    set_names
```

```
The following object is masked from 'package:tidyr':

    extract

The following object is masked from 'package:dlookr':

    extract
```

```
non_numeric <- ages_modified %>%
  select_if(is.numeric)
# Raw summary, output suppressed
knn_na_imp_age <- non_numeric %>%
  imputate_na(age_difference, method = "knn")

# Plot showing the results of our imputation
knn_na_imp_age %>%
  plot()
```



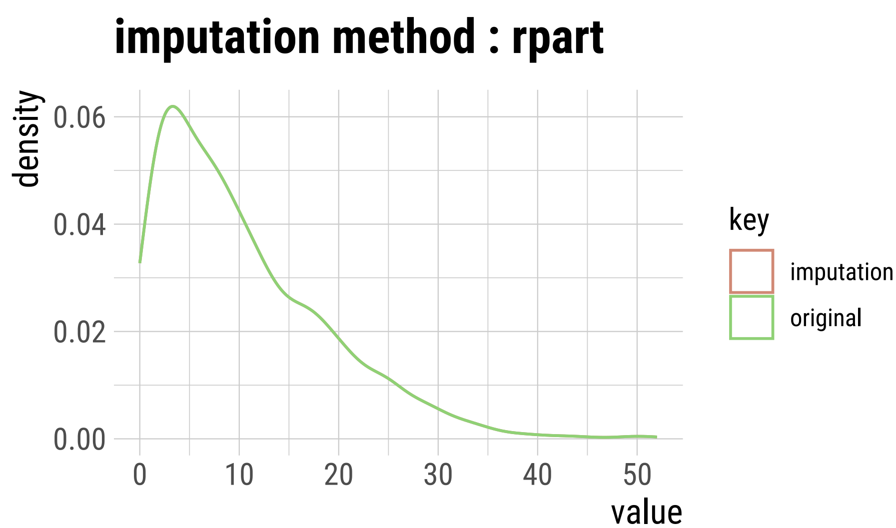
Recursive Partitioning and Regression Trees (rpart)

is a tree-based algorithm that recursively splits the data into subsets based on the values of predictor variables to make predictions about the target variable.

```r
library(magrittr)
non_numeric <- na.ages_modified %>%
  select_if(is.numeric)
# Raw summary, output suppressed
rpart_na_imp_age <- ages_modified |>
imputate_na(age_difference, method = "rpart")
```

```
Warning in imputate_na_impl(.data, vars, target, method, seed, print_flag, :
There are no missing values in age_difference.
```

```r
# Plot showing the results of our imputation
rpart_na_imp_age |>
 plot()
```



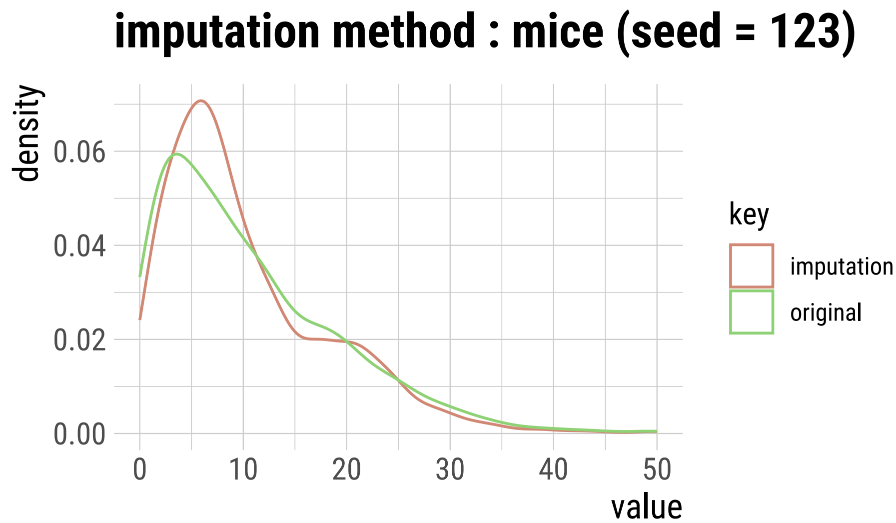
Multivariate Imputation by Chained Equations (MICE)

Multivariate Imputation by Chained Equations (MICE) is a statistical technique used for imputing missing data in multivariate datasets. It is particularly useful when you have missing values in multiple variables, and the relationships between these variables need to be considered when imputing missing data.

```r
# Raw summary, output suppressed
mice_na_imp_age <- na.ages_modified |>
  imputate_na(age_difference, method = "mice", seed = 123)
```

```
iter imp variable
 1   1  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 1   2  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 1   3  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 1   4  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 1   5  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 2   1  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 2   2  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 2   3  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 2   4  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 2   5  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 3   1  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 3   2  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 3   3  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 3   4  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 3   5  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 4   1  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 4   2  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 4   3  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 4   4  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 4   5  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 5   1  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 5   2  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 5   3  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 5   4  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
 5   5  release_year  age_difference  couple_number  actor_1_age  actor_2_age  Age_differen
```

```r
# Plot showing the results of our imputation
mice_na_imp_age |>
  plot()
```



Correlating Like a Data Master

Required setup

```
if (!require(pacman))
  install.packages("pacman")

pacman::p_load(colorblindr,
       dlookr,
       formattable,
       GGally,
       ggdist,
       ggpubr,
       ggridges,
       here,
       tidyverse)

# Set global ggplot() theme
# Theme pub_clean() from the ggpubr package with base text size = 16
theme_set(theme_pubclean(base_size = 12))
```

```
# All axes titles to their respective far right sides
theme_update(axis.title = element_text(hjust = 1))
# Remove axes ticks
theme_update(axis.ticks = element_blank())
# Remove legend key
theme_update(legend.key = element_blank())
```

Describe and Visualize Correlations

Correlation measures are used to determine how changes in one variable are associated with changes in another variable.

Pearson correlation is used to measure the linear relationship between two continuous variables. A correlation coefficient value ranges from -1 (perfect negative correlation) to 1 (perfect positive correlation), with 0 indicating no linear correlation.

```
# Table of correlations between numerical variables (we are sticking to the default Pearso
correlate(ages_modified) |>
  formattable()
```

var1

var2

coef_corr

age_difference

release_year

-0.204

couple_number

release_year

0.029

actor_1_age

release_year

-0.017

actor_2_age

release_year

0.209

release_year

age_difference

-0.204

couple_number

age_difference

-0.246

actor_1_age

age_difference

0.704

actor_2_age

age_difference

-0.156

release_year

couple_number

0.029

age_difference

couple_number

-0.246

actor_1_age

couple_number

-0.100

actor_2_age

couple_number

0.140

release_year

actor_1_age

-0.017

age_difference

actor_1_age

0.704

couple_number

actor_1_age

-0.100

actor_2_age

actor_1_age

0.591

release_year

actor_2_age

0.209

age_difference

actor_2_age

-0.156

couple_number

actor_2_age

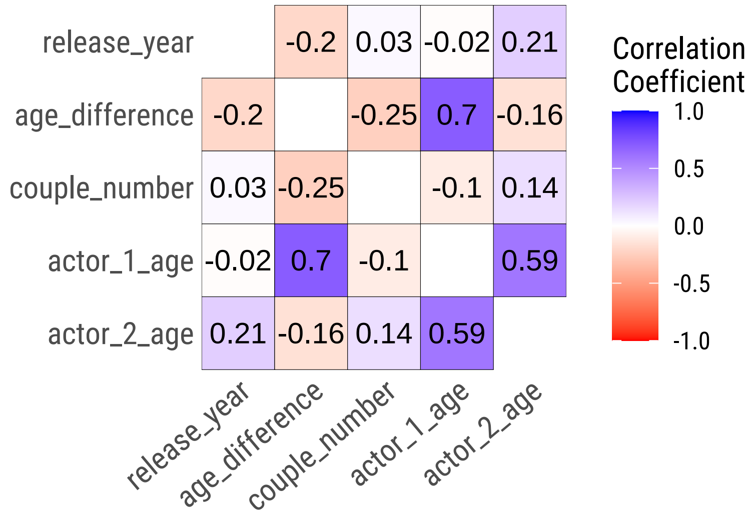
0.140

actor_1_age

actor_2_age

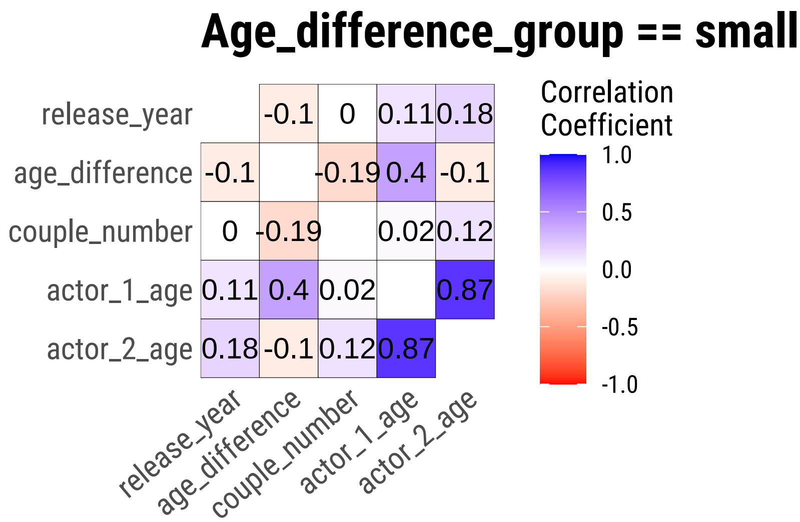
0.591

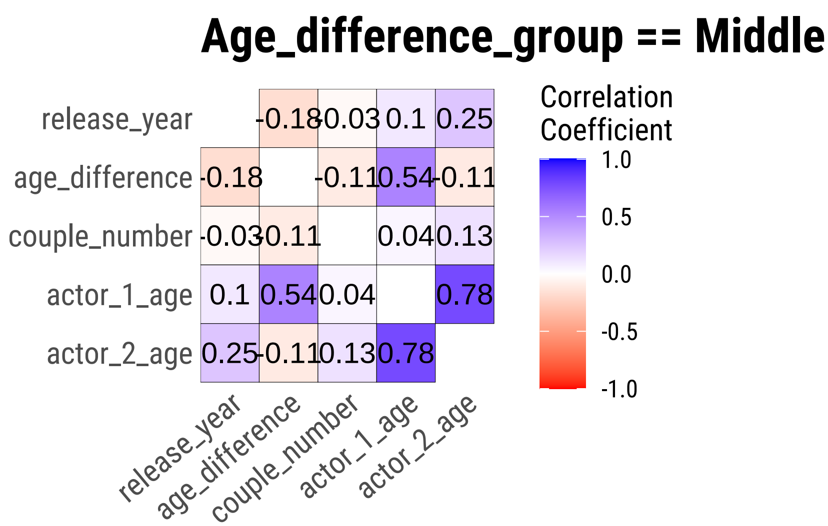
```
# Correlation matrix of numerical variables
ages_modified |>
plot_correlate()
```

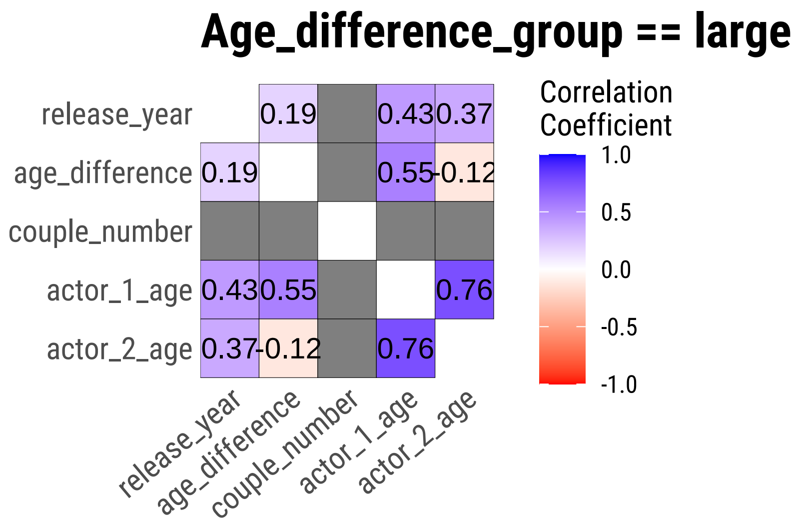


Visualize Correlations within Groups

```
ages_modified |>
  group_by(Age_difference_group) |>
  plot_correlate() # plotting co-relation in attributes
```

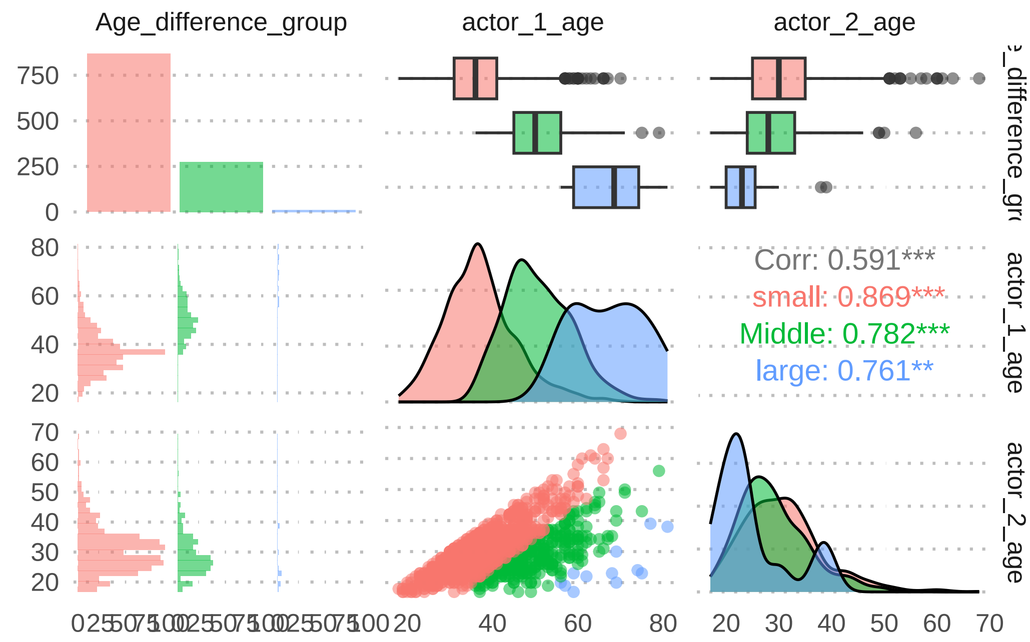

Age_difference_group == small
release_year
age_difference
couple_number
actor_1_age
actor_2_age
-0.1
0
0.11
0.18
-0.19
0.4
0.02
0.12
0.87
Correlation
Coefficient
1.0
0.5
0.0
-0.5
-1.0


Age_difference_group == Middle
release_year
age_difference
couple_number
actor_1_age
actor_2_age
-0.18
0.03
0.1
0.25
-0.11
0.54
0.04
0.13
0.78
Correlation
Coefficient
1.0
0.5
0.0
-0.5
-1.0




```
ages_modified  |>
  dplyr::select(Age_difference_group, actor_1_age, actor_2_age) |>
  ggpairs(aes(color = Age_difference_group, alpha = 0.5)) +
  theme(strip.background = element_blank())
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Describe and Visualize Relationships Based on Target Variables

Target Variables

The target variable is what you want your model to make predictions about based on the input features (independent variables).

Numerical Target Variables: Numerical Variable of Interest

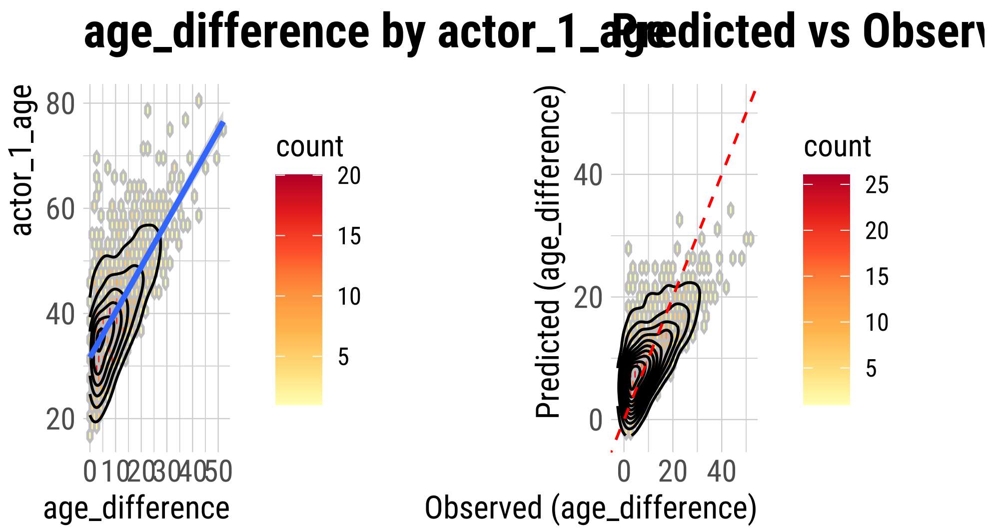
- Formula: actor_1_age(numerical response) ~ age_difference (numerical predictor)

```
# First, we need to remove NAs, they cause an error
dataset.noNA <- ages_modified |>
  drop_na()

# The numerical predictor variable that we want
num <- target_by(dataset.noNA, age_difference)

# Relating the variable of interest to the numerical target variable
num_num <- relate(num, actor_1_age)
```

```r
# Summary of the regression analysis - the same as the summary from lm(Formula)
summary(num_num)
```

```

Call:
lm(formula = formula_str, data = data)

Residuals:
    Min      1Q  Median      3Q     Max 
-25.302  -3.886  -0.059   3.988  22.273 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -12.9318     0.7164   -18.1   <2e-16 ***
actor_1_age   0.5748     0.0171    33.7   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6 on 1153 degrees of freedom
Multiple R-squared:  0.496, Adjusted R-squared:  0.495 
F-statistic: 1.13e+03 on 1 and 1153 DF,  p-value: <2e-16
```

```r
# Plotting the linear relationship
plot(num_num)
```



Numerical Target Variables: Categorical Variable of Interest

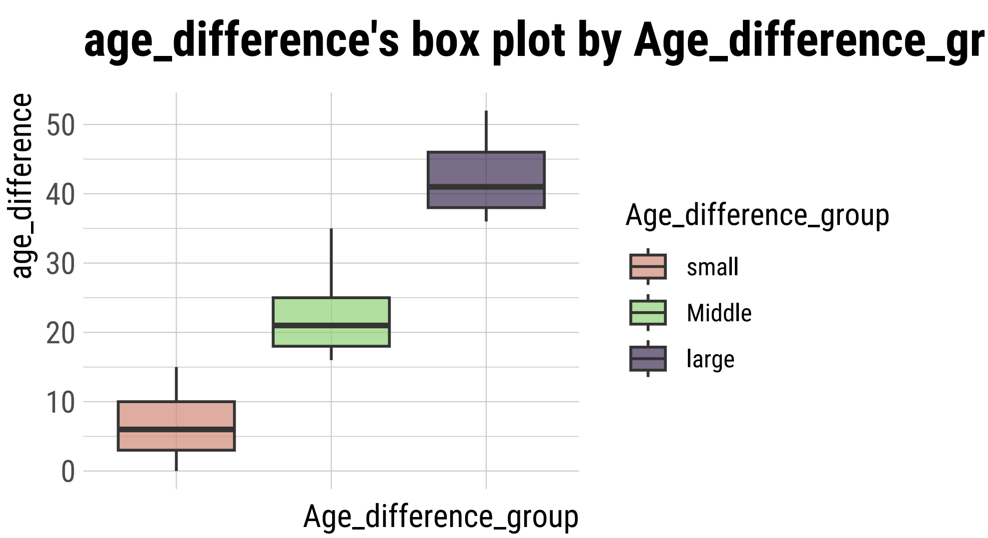
- Formula: age_difference(numerical response) ~ Age_difference_group(categorical predictor)

```
# The categorical predictor variable that we want
num <- target_by(ages_modified, age_difference)

# We need to change Group to a factor
num$Group <- as.factor(num$Age_difference_group)

# Relating the variable of interest to the numerical target variable
num_cat <- relate(num, Age_difference_group)

# Summary of the ANOVA analysis - the same as the summary from anova(lm(Formula))
summary(num_cat)
```

```
Call:
lm(formula = formula(formula_str), data = data)
```

```
Residuals:
   Min     1Q Median     3Q    Max 
-6.389 -3.578 -0.767  3.233 13.233 

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)    
(Intercept)                   6.389      0.148    43.2   <2e-16 ***
Age_difference_groupMiddle   15.378      0.301    51.0   <2e-16 ***
Age_difference_grouplarge    35.944      1.266    28.4   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.4 on 1152 degrees of freedom
Multiple R-squared:  0.738,	Adjusted R-squared:  0.738 
F-statistic: 1.63e+03 on 2 and 1152 DF,  p-value: <2e-16
```

```r
plot(num_cat) +
  theme(axis.text.x = element_blank())
```



Categorical Target Variables: Numerical Variable of Interest

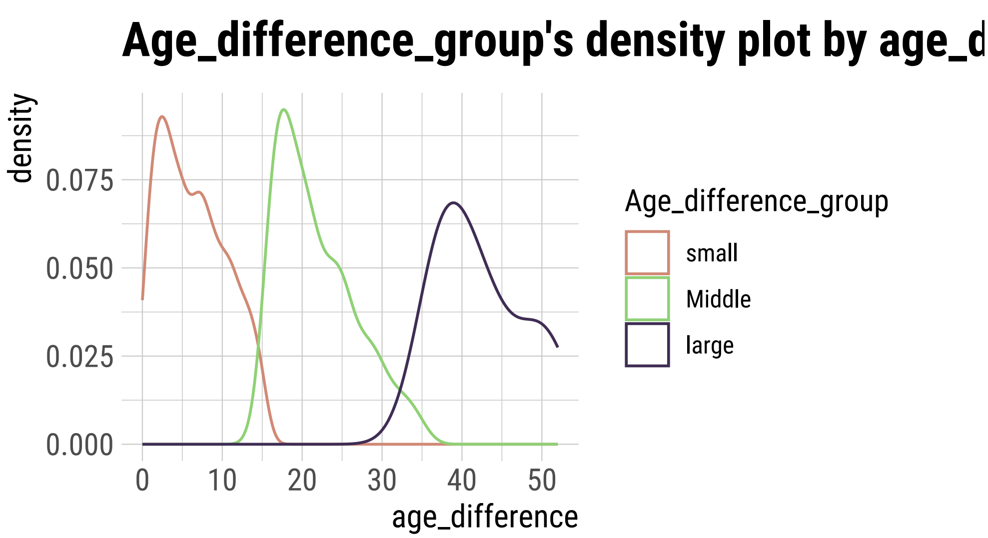
- Formula: Age_difference_group (categorical) ~ age_difference (numerical)

```
# The categorical predictor variable that we want
categ <- target_by(ages_modified, Age_difference_group)

# Relating the variable of interest to the numerical target variable
cat_num <- relate(categ, age_difference)

# Summary of descriptive statistics
summary(cat_num)
```

```
described_variables Age_difference_group       n               na
Length:4            small :1              Min.   :  12   Min.   :0
Class :character    Middle:1              1st Qu.: 209   1st Qu.:0
Mode  :character    large :1              Median : 572   Median :0
                    total :1              Mean   : 578   Mean   :0
                                          3rd Qu.: 940   3rd Qu.:0
                                          Max.   :1155   Max.   :0
     mean           sd           se_mean          IQR          skewness
Min.   : 6    Min.   :4.2   Min.   :0.14   Min.   : 7.0   Min.   :0.35
1st Qu.: 9    1st Qu.:4.7   1st Qu.:0.22   1st Qu.: 7.0   1st Qu.:0.54
Median :16    Median :5.2   Median :0.27   Median : 7.5   Median :0.69
Mean   :20    Mean   :5.8   Mean   :0.57   Mean   : 8.2   Mean   :0.73
3rd Qu.:27    3rd Qu.:6.3   3rd Qu.:0.62   3rd Qu.: 8.8   3rd Qu.:0.89
Max.   :42    Max.   :8.5   Max.   :1.60   Max.   :11.0   Max.   :1.20
   kurtosis           p00          p01          p05          p10
Min.   :-0.99   Min.   : 0   Min.   : 0   Min.   : 1   Min.   : 1
1st Qu.:-0.97   1st Qu.: 0   1st Qu.: 0   1st Qu.: 1   1st Qu.: 2
Median :-0.62   Median : 8   Median : 8   Median : 8   Median : 9
Mean   :-0.16   Mean   :13   Mean   :13   Mean   :14   Mean   :14
3rd Qu.: 0.19   3rd Qu.:21   3rd Qu.:21   3rd Qu.:21   3rd Qu.:21
Max.   : 1.59   Max.   :36   Max.   :36   Max.   :36   Max.   :36
     p20          p25          p30          p40          p50          p60
Min.   : 2   Min.   : 3   Min.   : 3   Min.   : 5   Min.   : 6   Min.   : 7
1st Qu.: 3   1st Qu.: 4   1st Qu.: 4   1st Qu.: 6   1st Qu.: 8   1st Qu.:10
Median :10   Median :11   Median :12   Median :12   Median :14   Median :16
Mean   :15   Mean   :16   Mean   :16   Mean   :17   Mean   :19   Mean   :21
3rd Qu.:22   3rd Qu.:23   3rd Qu.:23   3rd Qu.:24   3rd Qu.:26   3rd Qu.:27
Max.   :38   Max.   :38   Max.   :38   Max.   :39   Max.   :41   Max.   :43
     p70          p75          p80          p90          p95          p99
```

```
Min.   : 9   Min.   :10   Min.   :11   Min.   :13   Min.   :14   Min.   :15
1st Qu.:12   1st Qu.:14   1st Qu.:16   1st Qu.:20   1st Qu.:24   1st Qu.:29
Median :18   Median :20   Median :22   Median :26   Median :30   Median :35
Mean   :23   Mean   :24   Mean   :26   Mean   :29   Mean   :31   Mean   :34
3rd Qu.:29   3rd Qu.:30   3rd Qu.:32   3rd Qu.:34   3rd Qu.:37   3rd Qu.:40
Max.   :44   Max.   :46   Max.   :48   Max.   :50   Max.   :51   Max.   :52
     p100
Min.   :15
1st Qu.:30
Median :44
Mean   :38
3rd Qu.:52
Max.   :52
```

```r
plot(cat_num)
```



Here we will create new sub-category on the basis on age difference variable

```r
# Create new categorical column
cat_dataset <- ages_modified |>
  select(age_difference, Age_difference_group) |>
```

```
  drop_na() |>
  mutate(big_age_difference = ifelse(
    age_difference > (mean(age_difference + sd(age_difference))),
                              "Yes",
                              "No"))

# New dataset
cat_dataset |>
  head() |>
  formattable()
```

age_difference

Age_difference_group

big_age_difference

52

large

Yes

50

large

Yes

49

large

Yes

45

large

Yes

43

large

Yes

42

large

Yes

A **chi-square test** for independence, also known as a chi-square test of association, is a statistical test used to determine whether there is a significant association between two categorical variables.

```r
# The categorical predictor variable that we want
categ <- target_by(cat_dataset, big_age_difference)

# Relating the variable of interest to the categorical target variable
cat_cat <- relate(categ, Age_difference_group)

# Summary of the
summary(cat_cat)
```

```
Call: xtabs(formula = formula_str, data = data, addNA = TRUE)
Number of cases in table: 1155
Number of factors: 2
Test for independence of all factors:
    Chisq = 715, df = 2, p-value = 7e-156
    Chi-squared approximation may be incorrect
```

```r
plot(cat_cat)
```