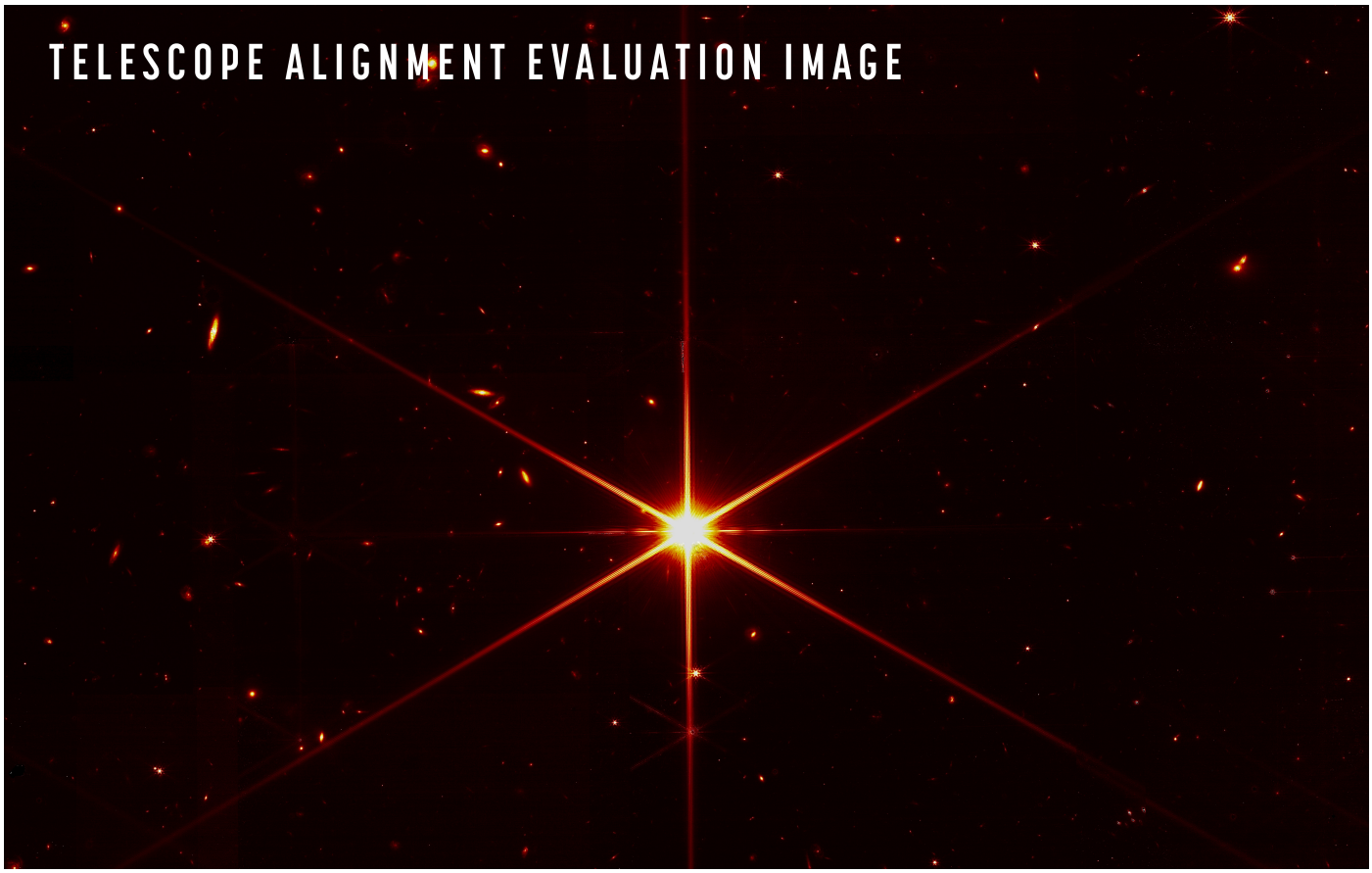
Physical properties of Stars

Anne-Sophie Chys

2023-03-13

Abstract

This document discusses the significant role that stars play in the Universe, providing vast sources of light and energy. The study of stars has been a central topic in astronomy for centuries, with researchers aiming to better understand their properties such as mass, size, temperature or luminosity. Through observations and measurements, scientists have developed a comprehensive understanding of stars, helping us gain a better understanding of the Universe as a whole. The investigation of star properties continues to be an essential area of study, with new insights being continually uncovered through advanced research and technologies. This project utilizes a comprehensive dataset on the properties of stars in our galaxy, offering information on various features to facilitate an in-depth study of stars. Thorough analysis and visualization of the dataset will yield a better understanding of star properties. This document is created as a part of a project to learn the R programming luanguage, undertaken as student at the University of Applied Sciences of Howest.



While the purpose of this image was to focus on the bright star at the center for alignment evaluation, Webb's optics and NIRCam are so sensitive that the galaxies and stars seen in the background show up. At this stage of Webb's mirror alignment, known as "fine phasing," each of the primary mirror segments have been adjusted to produce one unified image of the same star using only the NIRCam instrument. This image of the star, which is called 2MASS J17554042+6551277, uses a red filter to optimize visual contrast. Credits: NASA/STScI (https://www.nasa.gov/press-release/nasa-s-webb-reaches-alignment-milestone-optics-working-successfully)

Introduction

Stars are one of the most fascinating celestial bodies in the Universe, and their properties have captivated human interest for centuries. The study of stars has been a central topic in astronomy, contributing to our understanding of the cosmos and the laws of physics that govern it. The properties of stars, including their mass, size, temperature, and luminosity, provide valuable insights into their nature and behavior. Moreover, stars play a vital role in the Universe as sources of light and energy, making them an essential area of study.

Through observations and measurements, scientists have developed a comprehensive understanding of the properties of stars, which has

helped us gain a better understanding of the Universe as a whole. The investigation of star properties continues to be a crucial area of study, with ongoing advances in technology and research leading to new insights.

The dataset on the physical properties of stars in our galaxy offers a comprehensive and detailed examination of various features, including mass, radius, distance from Earth, and other identifying factors. The dataset is designed to facilitate an in-depth study of the stars in our galaxy and provides several columns such as the features described before. The properties of stars will be thoroughly analyzed and visualized through various plots to yield a better understanding of the dataset.

In this context, the investigation of star properties is an exciting and dynamic field, with new discoveries and insights being continually uncovered. The study of stars has a significant impact on our understanding of the Universe, making it a crucial area of research for astronomers and astrophysicists alike.

Script

Installing and loading packages

Initially, import the required packages that are not currently installed. Upon executing these lines, the installation of necessary packages will occur, while previously installed packages will remain unaffected. Below there is given a description for what each package is used:

- `Hmisc` : provides several useful functions for data analysis, including tools for data manipulation, descriptive statistics, and modeling.
- `ggplot2` : a popular data visualization tool in R that allows for creating high-quality graphics using a “grammar of graphics” syntax.
- `ggrepel` : provides geom functions for ggplot2 that allow labels to be placed outside the plot area so that they don’t overlap with the data.
- `rgl` : is used for creating interactive 3D plots and offers a wide range of tools for creating, manipulating, and visualizing 3D data.
- `fmsb` : provides tools for creating radar charts (known as spider charts), used to compare multivariate data across multiple variables.
- `dplyr` : provides a fast and efficient way to manipulate data in R, including tools for filtering, arranging, summarizing, and joining data sets.
- `grid` : provides low-level functions for creating and manipulating plots, including tools for customizing plot layout and adding annotations.
- `cowplot` : provides functions for combining multiple plots into a single plot and customizing plot themes, among other things.

```
pkg <- installed.packages()[, "Package"]
if(!('Hmisc' %in% pkg)) {install.packages("Hmisc")}
if(!('ggplot2' %in% pkg)) {install.packages("ggplot2")}
if(!('ggrepel' %in% pkg)) {install.packages("ggrepel")}
if(!('rgl' %in% pkg)) {install.packages("rgl")}
if(!('fmsb' %in% pkg)) {install.packages("fmsb")}
if(!('dplyr' %in% pkg)) {install.packages("dplyr")}
if(!('grid' %in% pkg)) {install.packages("grid")}
if(!('cowplot' %in% pkg)) {install.packages("cowplot")}
```

The following code will then be used to load the packages inside the code.

```
library("Hmisc")
library("ggplot2")
library("ggrepel")
library("rgl")
library("fmsb")
library("dplyr")
library("grid")
library("cowplot")
```

Import the data

The datasets used in this script are available on multiple platforms, including Kaggle and GitHub. The `final_data.csv` file is used as Dataset 1, while the `cleaned.csv` file is used as Dataset 2, both from Kaggle (https://www.kaggle.com/datasets/thedevastator/properties-of-stars-in-our-galaxy). Dataset 3, titled `Physical Properties of Stars`, can be found on GitHub under the name of vincentarelbundock (https://vincentarelbundock.github.io/Rdatasets/articles/data.html). To get started, download the necessary files and set the working directory accordingly. Under this GitHub (https://github.com/Anne-SophieChys/Project_Of_The_Stars.git) platform the files can be found and downloaded to get started!

Below there is an example given that can be modified to the desired directory where the files are downloaded. This tilde indicates the start point of the home directory. In this example the `/Downloads` directory is used.

```
setwd("~/Downloads")
```

Below the code is given for loading the data into this script. There are 3 datasets used and will be loaded separately from each other in a new variable. Dataset 1 will be used for the features 'Name', 'Distance' (from the point of Earth), 'Mass' and 'Radius' of the stars. Next, Dataset 2 will be used for the 'Luminosity' features. At last Dataset 3 will be added for features 'Type', 'Magnitude' and the 'Temperature' of the stars. These are also available on GitHub (https://github.com/Anne-SophieChys/Project_Of_The_Stars.git).

```
# Read CSV into Data Frame - Dataset 1
# Na(me), Di(stance), Ma(ss), Ra(dius)
star_NaDiMaRa <- read.csv(file="StarFile1_NameDistanceMassRadius", header = TRUE)

# Read CSV into Data Frame - Dataset 2
# Na(me), Lu(minosity)
star_NaLu <- read.csv(file="StarFile2_NameDistanceMassRadiusLuminosity", header = TRUE)

# Read CSV into Data Frame - Dataset 3
# Na(me), Ty(pe), Ma(gnitude), Te(mp)
star_NaTyMaTe <- read.csv(file="StarFile3_NameMagTempType", header = TRUE)
```

Get information about the data

Represent the summary

The next step is to get information for this loaded data For each file there will be performed a `summary()` where there will be displayed informative information about all the columns from the file. Namely the length, class and mode for character types, and for numeric values the minimum, maximum, mean and first-, second- (median) and third quantile are displayed.

Dataset 1 - star_NaDiMaRa

```
summary(star_NaDiMaRa)
```

```
##        X        Star_name           Distance             Mass
##  Min.   :  0   Length:253         Length:253         Min.   :  0.00099
##  1st Qu.: 63   Class :character   Class :character   1st Qu.:  0.04009
##  Median :126   Mode  :character   Mode  :character   Median :  0.06587
##  Mean   :126                                         Mean   :  5.84210
##  3rd Qu.:189                                         3rd Qu.:  1.90000
##  Max.   :252                                         Max.   :290.00000
##     Radius
##  Length:253
##  Class :character
##  Mode  :character
##
##
##
```

Dataset 2 - star_NaLu

```
summary(star_NaLu)
```

```
##        X           Star_name           Distance            Mass
##  Min.   :  0.00   Length:254         Length:254         Length:254
##  1st Qu.: 63.25   Class :character   Class :character   Class :character
##  Median :126.50   Mode  :character   Mode  :character   Mode  :character
##  Mean   :126.50
##  3rd Qu.:189.75
##  Max.   :253.00
##     Radius           Luminosity
##  Length:254         Length:254
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Dataset 3 - star_NaTyMaTe

```
summary(star_NaTyMaTe)
```

```
##        X             star             magnitude          temp
##  Min.   : 1.00   Length:96          Min.   :-8.000   Min.   : 2500
##  1st Qu.:24.75   Class :character   1st Qu.:-1.800   1st Qu.: 3168
##  Median :48.50   Mode  :character   Median : 2.400   Median : 5050
##  Mean   :48.50                      Mean   : 4.257   Mean   : 8752
##  3rd Qu.:72.25                      3rd Qu.:11.325   3rd Qu.: 9900
##  Max.   :96.00                      Max.   :17.000   Max.   :33600
##      type
##  Length:96
##  Class :character
##  Mode  :character
##
##
##
```

Represent the structure

Following step is to review the amount of rows (observations) and columns (variables) of the files. The rows correspond also with the 'Length' description from above. The function that will be used for this, is the `str()` function to display the internal structure of an R object. It stands for 'structure' and provides a concise summary of the object's type and contents, including per variable, the datatype with the first amount of values.

Dataset 1 - star_NaDiMaRa

```
str(star_NaDiMaRa)
```

```
## 'data.frame':    253 obs. of  5 variables:
##  $ X        : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Star_name: chr  "Sun" "Sirius" "Canopus" "Alpha Centauri" ...
##  $ Distance : chr  "0.000015813" "8.6" "310" "4.4" ...
##  $ Mass     : num  1 2.1 15 1.1 1.1 2.2 2.6 23 1.5 20 ...
##  $ Radius   : chr  "1" "1.71" "71" "1.2" ...
```

Dataset 2 - star_NaLu

```
str(star_NaLu)
```

```
## 'data.frame':    254 obs. of  6 variables:
##  $ X         : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Star_name : chr  "Sun" "Sirius" "Canopus" "Alpha Centauri" ...
##  $ Distance  : chr  "0.000015813" "0008.6" "0310" "0004.4" ...
##  $ Mass      : chr  "1" "2.1" "15" "1.1" ...
##  $ Radius    : chr  "1" "1.71Â´" "71" "1.2" ...
##  $ Luminosity: chr  "1" "25.4" "13,500" "2" ...
```

Dataset 3 - star_NaTyMaTe

```
str(star_NaTyMaTe)
```

```
## 'data.frame':    96 obs. of  5 variables:
##  $ X        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ star     : chr  "Sun" "SiriusA" "Canopus" "Arcturus" ...
##  $ magnitude: num  4.8 1.4 -3.1 -0.4 4.3 0.5 -0.6 -7.2 2.6 -5.7 ...
##  $ temp     : int  5840 9620 7400 4590 5840 9900 5150 12140 6580 3200 ...
##  $ type     : chr  "G" "A" "F" "K" ...
```

Represent the column names

At last, the column names will be presented with the `colnames()` function for each file. Each column starts with an `index[1]`.

Dataset 1 - star_NaDiMaRa

```
colnames(star_NaDiMaRa)
```

```
## [1] "X"         "Star_name" "Distance"  "Mass"      "Radius"
```

Dataset 2 - star_NaLu

```
colnames(star_NaLu)
```

```
## [1] "X"          "Star_name"  "Distance"   "Mass"       "Radius"
## [6] "Luminosity"
```

Dataset 3 - star_NaTyMaTe

```
colnames(star_NaTyMaTe)
```

```
## [1] "X"         "star"      "magnitude" "temp"      "type"
```

Rearrange the DataFrames

Now the datasets are loaded and explored, they can be rearranged for preparation to merge the datasets together. Each step will be discussed below. Each loaded file has been assigned a variable name that reflects the contents of its respective data frame. In the previous step 'Import the data', a description was provided of what each variable name represents based on the loaded data in the comments of the code.

Remove the row with index 63 out of the `star_NaLu` dataframe. This dataframe corresponds then with the same amount and the same order of the stars there indices that is available in the `star_NaDiMaRa` dataframe.

```
star_NaLu <- star_NaLu[-63,]
```

Reorganize the 'X' column so the indices will go from zero till the end of the amount of rows without breaks between the counting in the `star_NaLu` dataframe.

```
star_NaLu$X <- 0:252
```

Retain the 'X' and the 'Luminosity' column out of the `star_NaLu` dataframe. The data that still is available in this dataframe will not further be

used, but will be used out of another dataframe, namely `star_NaDiMaRa` .

```
star_NaLu <- star_NaLu[,c(1,6)]
```

Remove the 'X' column for the `star_NaTyMaTe` dataframe. Otherwise during the merging of the dataframes, there will be two times a 'X' column.

```
star_NaTyMaTe <- star_NaTyMaTe[,c(2:5)]
```

Rename the column 'Name' to 'Star_name' for the `star_NaTyMaTe` dataframe. Then the dataframe can be merged with the other already merged dataframes so they have the same colomn names to merge on.

```
colnames(star_NaTyMaTe)[1] <- "Star_name"
```

Now the dataframes are ready to merge together. The `star_NaDiMaRa` dataframe can be merged with the `star_NaLu` dataframe based on the similar 'X' column. This merged dataframe can then be merged with the third dataframe `star_NaTyMaTe` based on the similar 'Star_name' column.

```
# Merging Dataframe 1 with Dataframe 2
star_merge <- merge(star_NaDiMaRa, star_NaLu, by = "X", all = TRUE)

# Merging the merged dataframes from above with Dataframe 3
filesmerged <- merge(star_merge, star_NaTyMaTe, by = "Star_name", all = TRUE)
```

The `colnames()` function can be used for displaying the column names for checking if the merge is correctly performed.

```
colnames(filesmerged)
```

```
## [1] "Star_name"  "X"          "Distance"   "Mass"       "Radius"
## [6] "Luminosity" "magnitude"  "temp"       "type"
```

Hertszprung-Russell Diagram

After successful merging of the files, the next step involves transforming the data to the appropriate format for plotting using the `ggplot()` function. This data transformation step is crucial in ensuring that the data is structured correctly to enable seamless integration into the desired Hertzsprung-Russell plot.

Transform the data

First make a copy of the merged files so this can't be overwritten.

```
HertzsprungRussell <- filesmerged
```

Remove the `<NA>` values for columns 'Luminosity' and 'temp' for this plot.

```
HertzsprungRussell <- HertzsprungRussell[complete.cases(HertzsprungRussell$Luminosity),]
HertzsprungRussell <- HertzsprungRussell[complete.cases(HertzsprungRussell$Distance),]
HertzsprungRussell <- HertzsprungRussell[complete.cases(HertzsprungRussell$temp),]
```

Remove commas out of the 'Luminosity' column so this also will not give an error.

```
HertzsprungRussell$Luminosity <- gsub(",", "", HertzsprungRussell$Luminosity)
```

Set all the data that will be used for this HertzsprungRussel diagram as a numeric value with the `as.numeric()` function.

```
HertzsprungRussell$Distance <- as.numeric(HertzsprungRussell$Distance)
HertzsprungRussell$Luminosity <- as.numeric(HertzsprungRussell$Luminosity)
HertzsprungRussell$Mass <- as.numeric(HertzsprungRussell$Mass)
```

still `<NA>` values are available in the `Distance` column that needs to be removed. Set also this values again numeric with the `as.numeric()` function.

```
HertzsprungRussell <- HertzsprungRussell[complete.cases(HertzsprungRussell$Distance),]
HertzsprungRussell$Distance <- as.numeric(HertzsprungRussell$Distance)
```

The amount of rows and columns can be reviewed for controlling how many rows where removed through the `<NA>` values.

```
dim(HertzsprungRussell)
```

```
## [1] 36  9
```

Review the first ten lines with the `head()` function of the HertzsprungRussell dataframe. So there can be displayed which columns will be used in the diagram that will be made.

```
head(HertzsprungRussell,10)
```

```
##     Star_name  X Distance Mass Radius Luminosity magnitude  temp type
## 126     Acrux 13      320 18.0    8.9    25000.0      -4.0 28000    B
## 127    Adhara 22      430 12.5     14    39000.0      -5.2 23000    B
## 129 Aldebaran 14       65  1.5     44      520.0      -0.8  4130    K
## 130    Alhena 43      109  2.8    3.3      120.0       0.0  9900    A
## 131    Alioth 32       81  2.9    4.2      110.0       0.4  9900    A
## 132    Alkaid 40      104  6.1    3.4     1350.0      -1.7 20500    B
## 137   Alnitak 33     1050 33.0     20   120000.0      -5.9 33600    O
## 144    Altair 12       17  1.8    1.8       10.5       2.2  8060    A
## 145    Altair 12       17  1.8    1.8       10.5       2.2  8060    A
## 146   Antares 15      600 12.0    680    75000.0      -5.2  3340    M
```

In the `head(10)` function from above can be seen that on index 9 and 10 there stands two times the same star. Remove the star from index 9.

```
HertzsprungRussell <- HertzsprungRussell[-9,]
```

Control of the changes went through, again with the `head(10)` function.

```
head(HertzsprungRussell,10)
```

```
##     Star_name  X Distance Mass Radius Luminosity magnitude  temp type
## 126     Acrux 13      320 18.0    8.9    25000.0      -4.0 28000    B
## 127    Adhara 22      430 12.5     14    39000.0      -5.2 23000    B
## 129 Aldebaran 14       65  1.5     44      520.0      -0.8  4130    K
## 130    Alhena 43      109  2.8    3.3      120.0       0.0  9900    A
## 131    Alioth 32       81  2.9    4.2      110.0       0.4  9900    A
## 132    Alkaid 40      104  6.1    3.4     1350.0      -1.7 20500    B
## 137   Alnitak 33     1050 33.0     20   120000.0      -5.9 33600    O
## 144    Altair 12       17  1.8    1.8       10.5       2.2  8060    A
## 146   Antares 15      600 12.0    680    75000.0      -5.2  3340    M
## 147  Arcturus  4       37  1.1     26      170.0      -0.4  4590    K
```

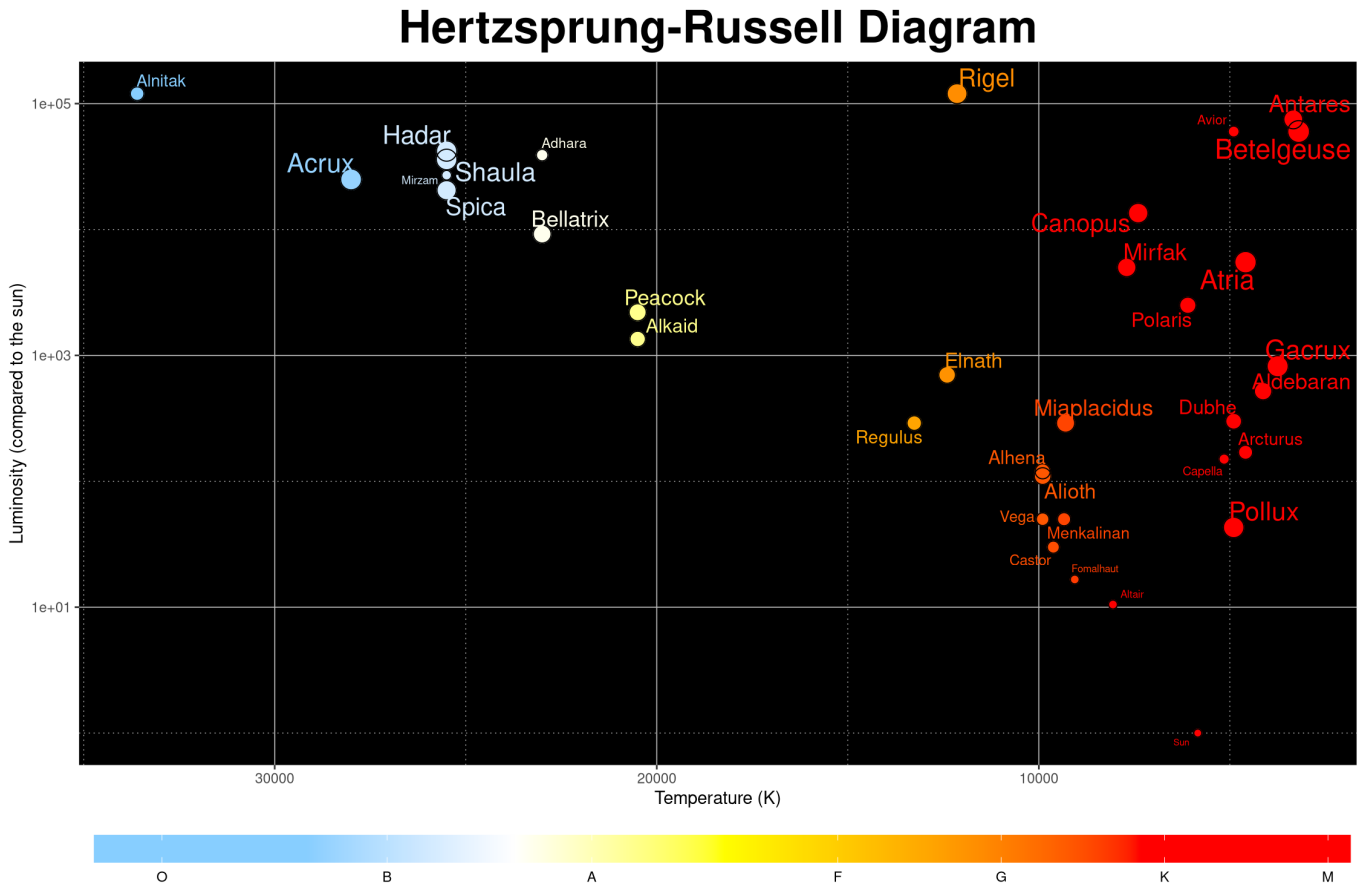
Create the Hertszprung-Russell Diagram

First choose the colors for the diagram.

```
colorsHertz = c("red", "orange", "yellow", "white", "skyblue1")
```

Create the plot

```
HRDiagram <- ggplot(data = HertzsprungRussell,
                    aes(x = HertzsprungRussell$temp,
                        y = HertzsprungRussell$Luminosity,
                        color = HertzsprungRussell$temp,
                        size = HertzsprungRussell$Radius)) +

  ggtitle("Hertzsprung-Russell Diagram") +
  geom_point(shape = 19) +
  geom_point(shape = 21, colour = "black") +
  geom_text_repel(aes(label = HertzsprungRussell$Star_name)) +


  theme(plot.title = element_text(size = 28, face="bold", hjust = 0.5),
        panel.background = element_rect(fill = "black"),
        panel.grid.major = element_line(color = "grey", size = 0.3),
        panel.grid.minor = element_line(color = "grey", linetype = "dotted"),
        # scale_x_discrete(position = "top"),
        legend.position = "bottom",
        legend.title = element_blank(),
        legend.key.width = unit(5.5, "cm"),
        # Can be same length as plot?
        plot.margin = unit(c(0.5, 0.5, 0.5, 0.5), "cm")) +

  xlab("Temperature (K)") +
  ylab("Luminosity (compared to the sun)")

HRDiagram +
  scale_y_log10() +
  scale_x_log10() +
  scale_x_reverse() +
  scale_color_gradientn(colors = c("red", "red", "orange", "yellow",
                                   "white", "skyblue1", "skyblue1"),
                        breaks = c(32000, 26533, 21567,
                                   15600, 11633, 7667, 3700),
                        labels = c("O", "B", "A",
                                   "F", "G", "K", "M")) +

  guides(size = FALSE,
    color = guide_colorbar(reverse = TRUE))
```



Discuss the Hertzsprung-Russell Diagram

The Hertzsprung-Russell diagram is a fundamental tool in astronomy used to plot the various properties of stars. The plot displays a graph with temperature on the x-axis and luminosity on the y-axis, with each star represented by a single point. The color of the stars on the plot is compared from red to blue, where blue stars are hotter, and red stars are colder. A scale bar is provided with all the different classes (O, B, A, F, G, K, M) for star classification based on their temperature and spectral class.

The luminosity of each star is compared to that of the Sun, which is set as the reference point with a luminosity of 1 (the lowest star on the plot). The radius of stars is also visible in the plot, allowing for a comparison of the size of different stars. For example, Betelgeuse appears as the largest star in the plot. Overall, the Hertzsprung-Russell diagram is a powerful tool for understanding the properties and behavior of stars.

Lollipop Plot

A lollipop plot is a type of data visualization that is used to display the values of a numerical variable for different categories or groups.

Transform the data

First make a copy of the merged files so this can't be overwritten.

```
lollipop <- HertzsprungRussell
```

There will be made use of a formula of the **distance modulus**.

`M = m + 5 - 5log10(d)`

* M = The absolute magnitude (how bright a star shines when a star stands on a 10 parsec distance of the Earth)
* m = The relative magnitude (how bright a star shines when they're observed from the Earth)
* d = The distance in parsec (1 parsec = 3.26 light years)

This formula is transformed so `'m'`, the relative magnitude can be calculated.

`m = M - 5 + 5log10(d)`

For this formula, the `d` needs to stand in parsec instead of light years (how it is given in the dataframe). 1 parsec is the same as 3.26 light years, so to transform the `Distance` column, this needs to be divided through 3.26.

```
lollipop$DistanceLY <- HertzsprungRussell$Distance/3.26
```

Next, the formula can be filled in. So the relative magnitude of each star can be calculated.

```
lollipop$RelativeMag <- lollipop$magnitude -5 + 5*log10(lollipop$DistanceLY)
dim(lollipop)
```

```
## [1] 35 11
```

At last, before the plot creation, the Sun will be removed from the dataset. Below, the values of the Sun is reviewed, but the value is to large to place this also in the plot. This is the reason why the Sun will be removed from the dataset. What can be reviewed is that the relative magnitude of the Sun is -26.7. This is a very compatible approach value of the real, already calculated and know value.

```
lollipop[lollipop$Star_name == "Sun", ]
```

```
##     Star_name X   Distance Mass Radius Luminosity magnitude temp type
## 273       Sun 0 1.5813e-05    1      1          1       4.8 5840    G
##       DistanceLY RelativeMag
## 273 4.850613e-06   -26.77102
```

After reviewing the 'Sun' values, the Sun will be removed from the data. the `tail()` functions is a control if the Sun (that normally stood on the second last line) is removed from the dataframe.

```
lollipop <- subset(lollipop, !(Star_name == "Sun"))
tail(lollipop)
```

```
##     Star_name  X Distance  Mass Radius Luminosity magnitude  temp type
## 231    Pollux 17       34  1.90    8.8         43       1.0  4900    K
## 241   Regulus 21       77  3.80    3.1        290      -0.8 13260    B
## 243     Rigel  7      860 23.00   78.9     120000      -7.2 12140    B
## 269    Shaula 23      700 14.50    8.8      36000      -3.4 25500    B
## 272     Spica 16      260 11.43   7.47      20512      -3.4 25500    B
## 294      Vega  5       25  2.20    2.7         50       0.5  9900    A
##     DistanceLY RelativeMag
## 231  10.429448  1.09130658
## 241  23.619632  1.06636563
## 243 263.803681 -0.09359574
## 269 214.723926  3.25940220
## 272  79.754601  1.10877874
## 294   7.668712 -0.07638796
```

Next, specific columns are stored in the X, y and z columns. This will be stored in a dataframe `lollipop_df` .

```
# Create data
x = lollipop$Star_name
y = lollipop$RelativeMag
z = lollipop$type

lollipop_df <- data.frame(x, y, z)
```

There will be add a column at the end (the colour column) that corresponds with the class types of the stars.

```
lollipop_df$ColourClass <- ifelse(lollipop_df$z == "M","red",
                                  ifelse(lollipop_df$z == "K","#FF5D00",
                                         ifelse(lollipop_df$z == "G","orange",
                                                ifelse(lollipop_df$z == "F", "yellow",
                                                       ifelse(lollipop_df$z == "A", "white",
                                                              ifelse(lollipop_df$z == "B", "lightblue1",
                                                                     ifelse(lollipop_df$z == "O", "blue",
                                                                            NA)))))))
```

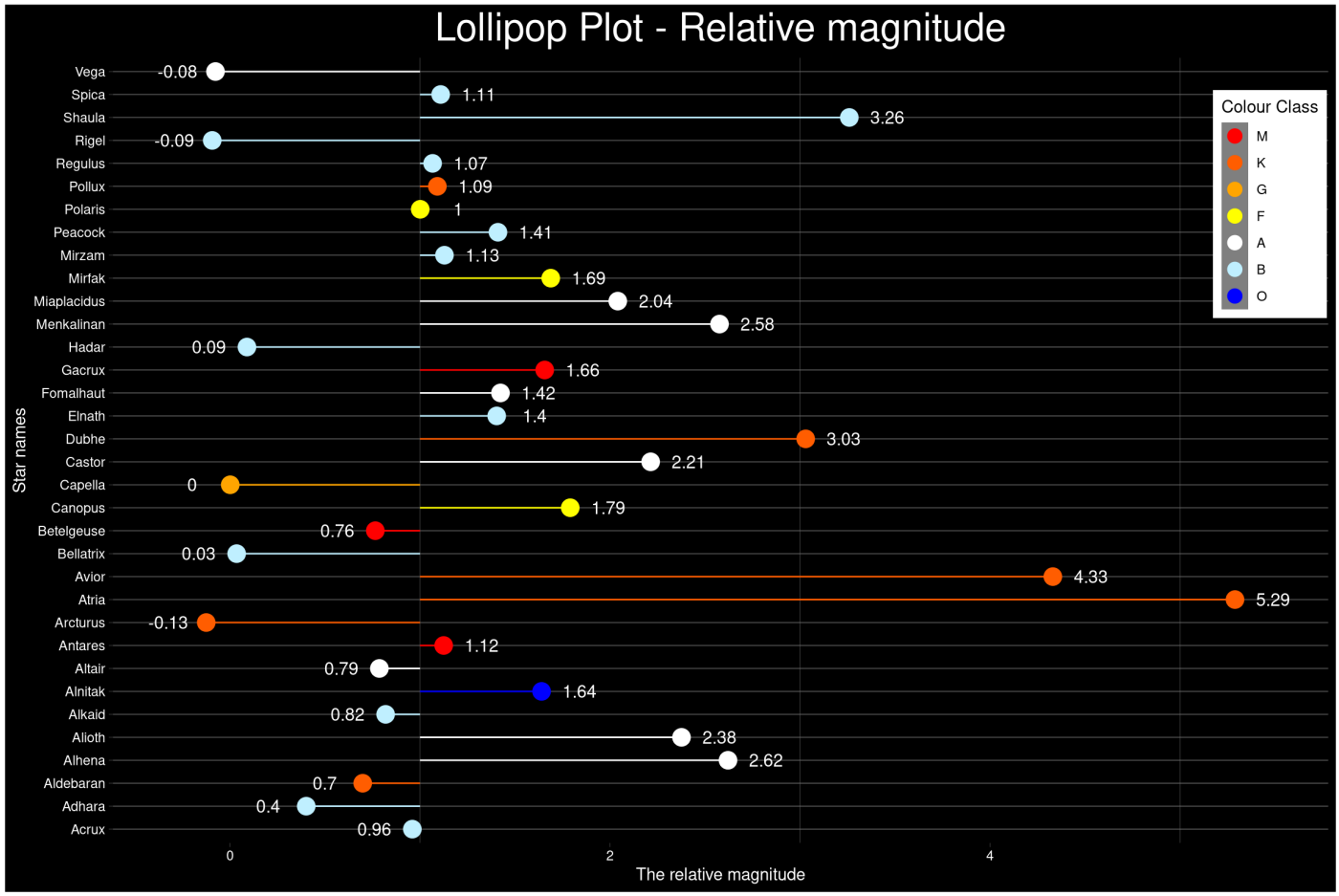
Create the Lollipop Plot and a legend

First the plot will be made in step 1. Next, in step 2 the legend will be created. This will be created with the `cowplot()` function. This will store the whole plot and convert it to a legend. This legend will be used to set the second layer on the already made `ggplot()`.

```
# Step 1: make the plot
ggplot(lollipop_df, aes(x = x, y = y, label = round(y, 2),
                        color = lollipop_df$ColourClass)) +
  geom_segment(aes(x = x, xend = x, y = 1, yend = y),
               color = lollipop_df$ColourClass, size = 0.5) +
  geom_point(color = lollipop_df$ColourClass, size = 5) +
  geom_text(nudge_y = ifelse(y < 1, -0.2, 0.2),
            colour = "white",
            size = 4) +
  theme_dark() +
  coord_flip() +

  theme(
    panel.background = element_rect(fill = "black"),
    plot.background = element_rect(fill = "black"),
    panel.grid.major.x = element_blank(),
    panel.border = element_blank(),
    axis.text = element_text(color = "white"),
    axis.title = element_text(colour = "white"),
    plot.title = element_text(hjust = 0.5, color = "white", size = 25)) +

  xlab("Star names") +
  ylab("The relative magnitude") +
  ggtitle("Lollipop Plot - Relative magnitude") +
  guides(color = guide_legend(title = "ColourClass"))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
colorlegend <- c("red", "#FF5D00", "orange", "yellow", "white", "lightblue1", "blue")
legendtext <- c("M", "K", "G", "F", "A", "B", "O")

# Step 2: create a legend
# First set the plot that will be used for the legend (with minimal theme options)
mylollipopplot <- ggplot(data = lollipop_df,
                         aes(x = x, y = y, color = ColourClass)) +
  geom_segment(aes(x = x, xend = x, y = 1, yend = y),
               color = lollipop_df$ColourClass, size = 0.5) +
  geom_point(size = 4) +
  theme_dark() +
  coord_flip() +
  guides(color = guide_legend(title = "Colour Class")) +
  scale_color_manual(values = colorlegend,
                     labels = legendtext) +

  theme(legend.position = "right")

# Second, store this minimal plot in cowplot - legend
legend <- cowplot::get_legend(mylollipopplot)

# Set the legend options
xleft <- 0.75  # x coordinate of left edge of viewport
ybottom <- 0.4  # y coordinate of bottom edge of viewport
width <- 0.4  # width of viewport
height <- 0.75  # height of viewport
vp <- viewport(x = xleft, y = ybottom, width = width, height = height,
               just = c("left", "bottom"), name = "legend_vp")

# Push the new viewport onto the grid stack and draw the legend within it
pushViewport(vp)
grid.draw(legend)

# Pop the viewport off the grid stack
popViewport()
```



Discuss the Lollipop Plot

The relative magnitude can be reviewed in this Lollipop plot on the x-axis and the star names on the y-axis. The relative magnitudes are values that indicates how bright stars are, referred from upon the Earth. How lower that this value is, the brighter the star based from the view from upon Earth. Normally this is the star Sirius, but through a lack off values in the dataset, this star is not available anymore through the filtering of the dataset. Out of the graph, Arcturus is the brightest star (-0.13), but out of previous experiments is suggested that Canopus comes after the brightness of Sirius. This star is available in the dataset with a value of 1.79, but this an incorrect value. This value needs to be around -0.74. Also the stars Vega (-0.08), Capella (0) and Rigel (-0.09) are also in the top 10 list of brightest stars. This can also be reviewed in next reference [review: Star Ranking (https://en.wikipedia.org/wiki/List_of_brightest_stars)].

Plotframe3d

The `rgl` package in R provides a wide range of tools for creating interactive 3D plots. The `plot3d()` function is a commonly used function within the `rgl` package that allows users to create 3D scatterplots and surface plots, as well as custom 3D objects. It offers a range of customization options including the ability to change lighting, color, and opacity of the plotted objects, as well as the ability to add axes, text labels, and titles. The `rgl` package is often used in scientific and engineering fields for visualizing complex data in a more intuitive way. Also here is the data prepossessed and will be discussed below which steps are performed.

Transform the data

First make a copy of the merged files so this can't be overwritten.

```
plotframe3d <- HertzsprungRussell
```

The two biggest stars are removed from the dataframe, because these are to big in comparison of the other stars and to place this nicely in a graph.

```
plotframe3d <- plotframe3d[-c(9, 14),]
```

Also the column 'Radius' will be set as numeric.

```
plotframe3d$Radius <- as.numeric(plotframe3d$Radius)
```

The last step before performing the `plot3d()` is to order the radius from low to high

```
plotframe3d <- plotframe3d[order(plotframe3d$Radius), ]
```

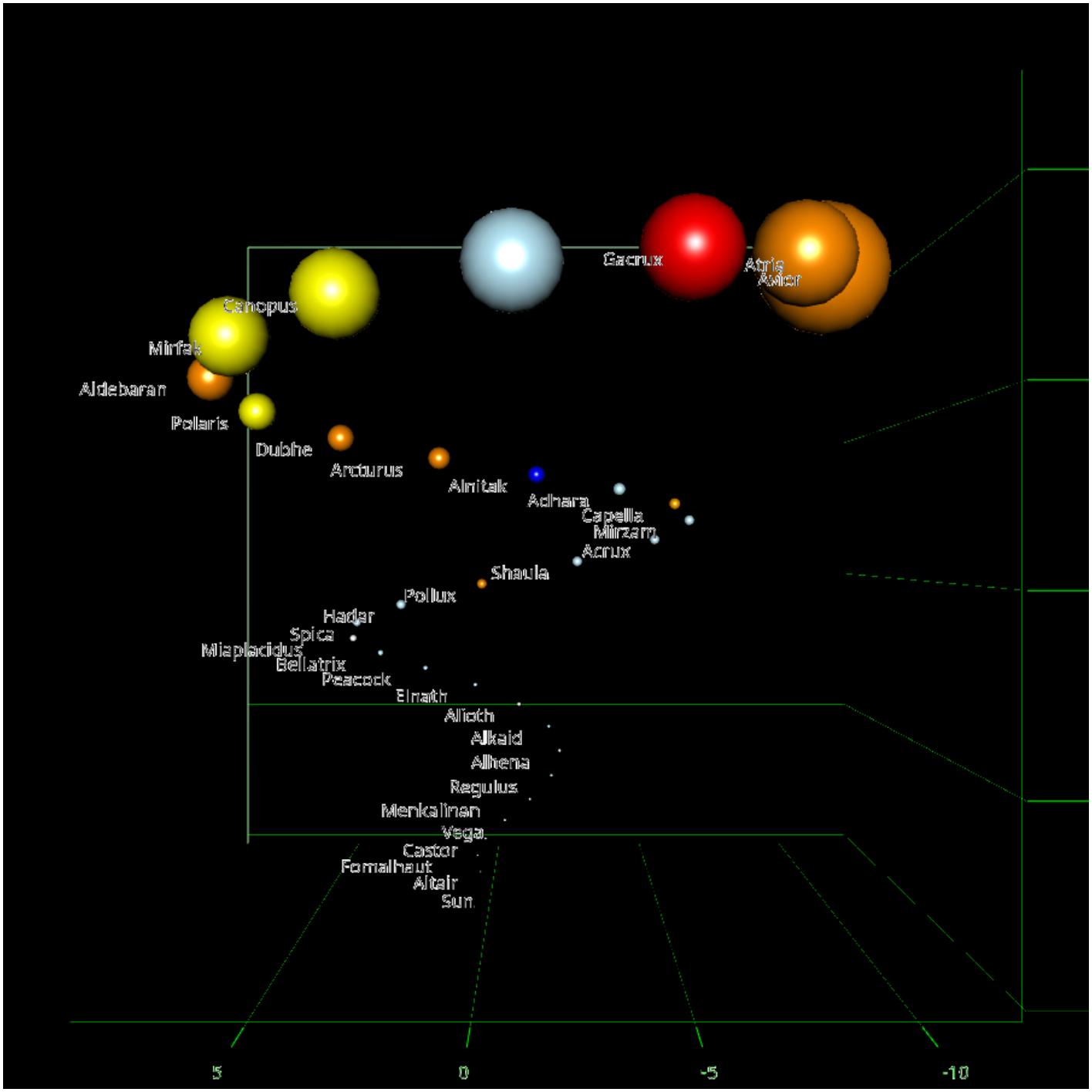
Create the plot3d

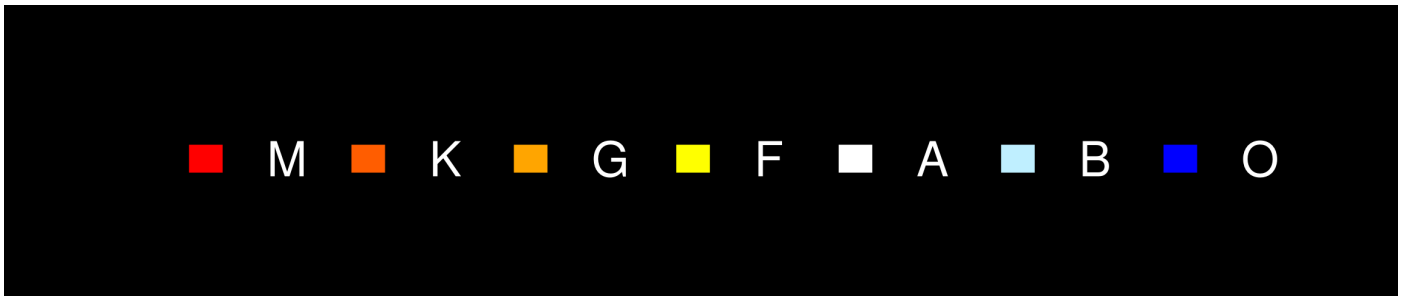
Write an extra column to the dataframe with the colours of the classes

```
plotframe3d$ColourClass <- ifelse(plotframe3d$type == "M", "red",
ifelse(plotframe3d$type == "K", "darkorange",
       ifelse(plotframe3d$type == "G", "orange",
              ifelse(plotframe3d$type == "F", "yellow",
                     ifelse(plotframe3d$type == "A", "white",
                            ifelse(plotframe3d$type == "B", "lightblue1",
                                   ifelse(plotframe3d$type == "O", "blue",
                                          NA)))))))
```

Plot out the 3d plot in an interactive way. This plot can be manipulated in an interactive way, namely zoom in till the smallest star (the sun), and rotate the graph for the possibility to read all the starnames properly.

```
# Set the values in 3d
n = nrow(plotframe3d)
theta <- seq(0, 5*pi, length.out = n)
r <- seq(0, 10, length.out = n)
x <- r * cos(theta)
y <- r * sin(theta)
z <- seq(-10, 10, length.out = n)

# Select 35 points on this spiral
indices <- seq(1, n, length.out = n)
x <- x[indices]
y <- y[indices]
z <- z[indices]

# Plot the datapoints
plot3d(x, y, z, type="s",
       col = (plotframe3d$ColourClass),
       size = plotframe3d$Radius/20)

# Adjust the plot settings
# Adjust the viewpoint
rgl.viewpoint(theta=180, phi=120, fov=60, zoom=0.6)
bg3d("black")
axes3d(col = "green", width = 2)
grid3d(side = c("x", "y", "z"), col = "green",
       lwd = 0.5, lty = 1)

# Add labels to the datapoints
labels <- plotframe3d$Star_name
text3d(x = x, y = y, z = z,
       texts = labels,
       col="white",
       pos = NULL,
       adj = c(1.5, 1, 25))

# Adjust the widget
rglwidget(elementId = "test-rgl", width = 900, height = 900)
```





This code can be performed for adding the legend by the `plot3d` from above.

```
# Create empty plot
plot(0, type = "n", xlim = c(0, 1), ylim = c(0, 1),
     axes = FALSE, xlab = "", ylab = "")

# Add legend
legend("bottom", legend = legendtext,
       fill = colorlegend, horiz = TRUE, cex = 0.8)
```

Discuss the 3D plot

Out of this plot the different stars can be reviewed in a 3D plot. This plot is performed to look at all the stars there colors (that is also linked to there star type (see the legend) and the radii of the stars. These stars are plotted in a spiral so this takes less space. Also this is plotted in a 3D plot that is interactive to zoom in and turn around. So the colors and radii of the smaller stars can be displayed better. In this pdf it is not possible to be interactive with this plot. This document is also available as a `.html` on GitHub (https://github.com/Anne-SophieChys/Project_Of_The_Stars.git) so the interactivity can be used.

Radarchart

Transform the data

First make a copy of the merged files so this can't be overwritten.

```
Featuresplot <- HertzsprungRussell
```

Remove the `<NA>` values for column 'Distance' for this plot.

```
Featuresplot <- Featuresplot[complete.cases(Featuresplot$Distance),]
```

Set all the data that will be used for this radarchart as a numeric value with the `as.numeric()` function.

```
Featuresplot$Radius <- as.numeric(Featuresplot$Radius)
Featuresplot$temp <- as.numeric(Featuresplot$temp)
```

Remove the column `Star_name` and `X` from the dataframe for this plot.

```
starchart <- Featuresplot[,c(-1,-2)]
```

Calculate the means for each column per star type.

```
means <- starchart %>% group_by(type) %>% summarize_if(is.numeric,mean)
```

Set the magnitude positive so this will not give an error, and this can be plotted in the radarchart.

```
means$magnitude <- abs(means$magnitude)
```

Calculate the `log10()` values of the means for each column. Add a value of 0.00001, so there can not become a `NULL` value, and this will not give an error if the `log10()` will be taken from this value.

```
means$Distance <- log10(means$Distance + 0.00001)
means$Mass <- log10(means$Mass + 0.00001)
means$Radius <- log10(means$Radius + 0.00001)
means$Luminosity <- log10(means$Luminosity + 0.00001)
means$magnitude <- log10(means$magnitude + 0.00001)
means$temp <- log10(means$temp + 0.00001)
```

Remove the character type 'type' column of the made `means` matrix Make then a new dataset with the name `TypesNr` where each type will get an other value from 1 till 7. This will be bind at the front of the `means` matrix.

```
data <- means[,-1]
TypesNr <- c(1, 2, 3, 4, 5, 6, 7)
data <- cbind(TypesNr, data)
```

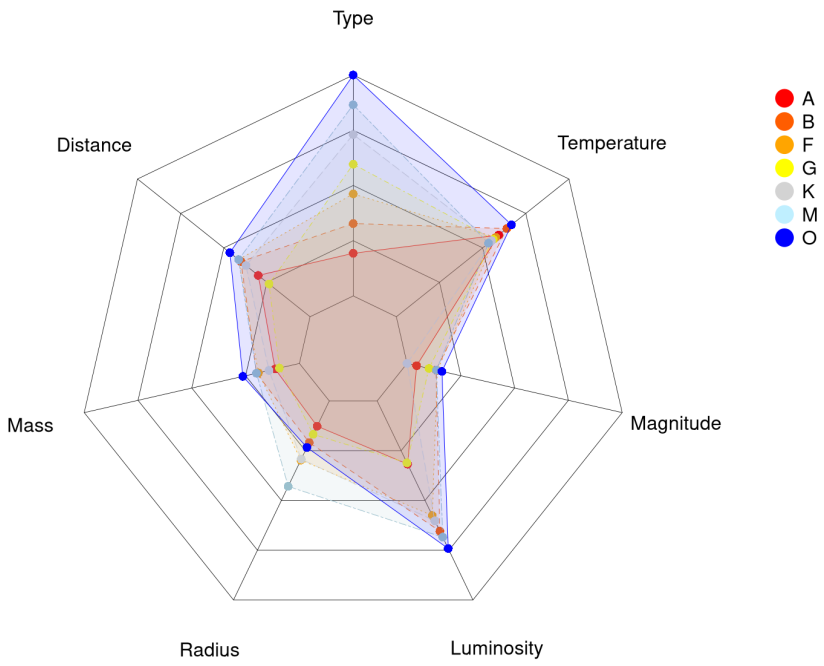
Set the data from a matrix to a dataframe.

```
data <- as.data.frame(data)
```

Create the radarchart

Next the radarchart will be created. In comments in the code the different steps will be explained.

```
# Set the colum names of the dataframe (this will appear on the plot)
colnames(data) <- c("Type", "Distance", "Mass", "Radius",
                    "Luminosity", "Magnitude", "Temperature")
rownames(data) <- means$type

# To use the fmsb package, Add 2 lines to the dataframe: the max and min of each variable to show on the plot
data <- rbind(max(data) , min(data) , data)

# plot
par(bg = "white")
radarchart(data,
           cglty = 1,              # the grid line type
           cglwd = 0.4,            # the grid width
           cglcol = "black",       # the grid color
           vlcex = 1,              # Custom lines of the data
           plwd = 0.5,            # Set the line width
           title = "Main values of physical properties of Stars",
           pcol = rev(c("blue", "lightblue3", "lightgrey",
                        "yellow", "orange", "#FF5D00", "red")),  # line colors data
           pfcol = scales::alpha(c("red", "#FF5D00", "orange", "yellow",
                                   "lightgrey", "lightblue3", "blue"),0.10)) # filling colors data


# Add a legend
legend(x = 1.5, y = 1, legend = means$type, bty = "n", pch=20 ,
       col = c("red", "#FF5D00", "orange", "yellow",
               "lightgrey", "lightblue1", "blue"),
       text.col = "black", cex = 1, pt.cex = 3)
```



Discuss the radarchart

What can be seen is the means of all numeric values of the stars. In this plot the 'Temperature', 'Magnitude', 'Luminosity', 'Radius', 'Mass', And 'Distance' are displayed. First the stars where grouped per star type, namely (A, B, F, G, K, M, O) from cold to hot stars. The mean per star type is calculated and plotted out in this radarchart. Of all the values the `log10()` value is used for making the data fit in this plot.
Here can be concluded that in the `Luminosity` the type of the star depends on the luminosity. How hotter the star (how more blue), how higher the luminosity. Also the `Mass` depends on the star type. How hotter the star (how more blue), how higher the mass. For the `Distance` there can be seen that how more massive the star is, the mostly how further they stand towards the Earth. In the `Radius` there can be concluded that the mean radius of the class `M` is the biggest, and the `A` the smallest. In the mean `Magnitude` the star types depends on the magnitude value, only the class `M` is switched to the lowest value instead of the second highest value if the order of the star types is followed. This can lay on maybe outliers in the dataset, also concluded before. At last the temperature needs to go from star type `A` upwards to star type `O`. This is also not the case so this can lay on also a possibility of outliers.

Conclusion

In conclusion, the Hertzsprung-Russell diagram is an essential tool in astronomy used to plot the various properties of stars, providing a clear visualization of the relationship between temperature, luminosity, and spectral class. It also allows for a comparison of the size of different stars. The Lollipop plot complements this diagram, showing the relative magnitudes of the stars and their brightness as viewed from Earth. The 3D plot provides a unique perspective of the stars, displaying their colors, radii, and spectral classes in an interactive manner.

Additionally, the Radarchart presents the mean values of different star properties, allowing for a comparison between different star types. Overall, these plots provide valuable insights into the behavior and characteristics of stars, enabling astronomers to better understand the cosmos. Normally the stars are classified based on their temperature from (A, B, F, G, K, M, O), however, it should be noted that outliers and inaccuracies in the whole data can affect the results obtained from these plots.