**Background**

Given present circumstances in in the world, I thought it might be nice to write a post on a lighter subject.

Recently, I came across an interesting Kaggle dataset that features [the power levels of Dragon Ball characters at different points in the franchise](https://www.kaggle.com/shiddharthsaran/dragon-ball-dataset). Whilst the dataset itself is quite simple with only four columns (Character, Power\_Level, Saga\_or\_Movie, Dragon\_Ball\_Series), I noticed that you do need to do a fair amount of data and string manipulation before you can perform any meaningful data analysis with it. Therefore, if you’re a fan of Dragon Ball and interested in learning about string manipulation in R, this post is definitely for you!

Functions or techniques that are covered in this post:

* Basic regular expression (regex) matching
* stringr::str\_detect()
* stringr::str\_remove\_all() or stringr::str\_remove()
* dplyr::anti\_join()
* Example of ‘dark mode’ ggplot in themes

**Getting started**

You can download the dataset from [Kaggle](https://www.kaggle.com/shiddharthsaran/dragon-ball-dataset), which you’ll need to register an account in order to do so. I would highly recommend doing so if you still haven’t, since they’ve got tons of datasets available on the website which you can practise on.

I’ll use the {here} workflow for this example, which is generally good practice as here::here implicitly sets the path root to the path to the top-level of they current project.

Let’s load our packages and explore the data using glimpse():

library(tidyverse)

library(here)

dball\_data <- read\_csv(here("datasets", "Dragon\_Ball\_Data\_Set.csv"))

dball\_data %>% glimpse()

## Observations: 1,244

## Variables: 4

## $ Character "Goku", "Bulma", "Bear Thief", "Master Roshi", "...

## $ Power\_Level "10", "1.5", "7", "30", "5", "8.5", "4", "8", "2...

## $ Saga\_or\_Movie "Emperor Pilaf Saga", "Emperor Pilaf Saga", "Emp...

## $ Dragon\_Ball\_Series "Dragon Ball", "Dragon Ball", "Dragon Ball", "Dr...

…and also tail() to view the last five rows of the data, just so we get a more comprehensive picture of what some of the other observations in the data look like:

dball\_data %>% tail()

## # A tibble: 6 x 4

## Character Power\_Level Saga\_or\_Movie Dragon\_Ball\_Seri~

##

## 1 Goku (base with SSJG ~ 448,000,000,000 Movie 14: Battle o~ Dragon Ball Z

## 2 Goku (MSSJ with SSJG'~ 22,400,000,000,0~ Movie 14: Battle o~ Dragon Ball Z

## 3 Goku (SSJG) 224,000,000,000,~ Movie 14: Battle o~ Dragon Ball Z

## 4 Goku 44,800,000,000 Movie 14: Battle o~ Dragon Ball Z

## 5 Beerus (full power, n~ 896,000,000,000,~ Movie 14: Battle o~ Dragon Ball Z

## 6 Whis (full power, nev~ 4,480,000,000,00~ Movie 14: Battle o~ Dragon Ball Z

**Who does the strongest Kamehameha? **

In the Dragon Ball series, there is an energy attack called *Kamehameha*, which is a signature (and perhaps the most well recognised) move by the main character **Goku**. This move is however not unique to him, and has also been used by other characters in the series, including his son **Gohan** and his master **Muten Roshi**.

Goku and Muten Roshi – image from Giphy

As you’ll see, this dataset includes observations which detail the power level of the notable occasions when this attack was used. Our task here is get some understanding about this attack move from the data, and see if we can figure out whose kamehameha is actually the strongest out of all the characters.

**Data cleaning**

Here, we use regex (regular expression) string matching to filter on the Character column. The str\_detect() function from the {stringr} package detects whether a pattern or expression exists in a string, and returns a logical value of either TRUE or FALSE (which is what dplyr::filter() takes in the second argument). I also used the stringr::regex() function and set the ignore\_case argument to TRUE, which makes the filter case-insensitive, such that cases of ‘Kame’ and ‘kAMe’ are also picked up if they do exist.

dball\_data %>%

filter(str\_detect(Character, regex("kameha", ignore\_case = TRUE))) -> dball\_data\_1

dball\_data\_1 %>% head()

## # A tibble: 6 x 4

## Character Power\_Level Saga\_or\_Movie Dragon\_Ball\_Seri~

##

## 1 Master Roshi's Max Power Kam~ 180 Emperor Pilaf Saga Dragon Ball

## 2 Goku's Kamehameha 12 Emperor Pilaf Saga Dragon Ball

## 3 Jackie Chun's Max power Kame~ 330 Tournament Saga Dragon Ball

## 4 Goku's Kamehameha 90 Red Ribbon Army S~ Dragon Ball

## 5 Goku's Kamehameha 90 Red Ribbon Army S~ Dragon Ball

## 6 Goku's Super Kamehameha 740 Piccolo Jr. Saga Dragon Ball

If this filter feels convoluted, it’s for a good reason. There is a variation of cases and spellings used in this dataset, which a ‘straightforward’ filter wouldn’t have picked up. So there are two of these:

dball\_data %>%

filter(str\_detect(Character, "Kamehameha")) -> dball\_data\_1b

## Show the rows which do not appears on BOTH datasets

dball\_data\_1 %>%

dplyr::anti\_join(dball\_data\_1b, by = "Character")

## # A tibble: 2 x 4

## Character Power\_Level Saga\_or\_Movie Dragon\_Ball\_Seri~

##

## 1 Jackie Chun's Max power Kameham~ 330 Tournament Saga Dragon Ball

## 2 Android 19 (Goku's kamehameha a~ 230,000,000 Android Saga Dragon Ball Z

Before we go any further with any analysis, we’ll also need to do something about Power\_Level, as it is currently in the form of character / text, which means we can’t do any meaningful analysis until we convert it to numeric. To do this, we can start with removing the comma separators with stringr::str\_remove\_all(), and then run as.numeric().

In ‘real life’, you often get data saved with *k* and *m* suffixes for thousands and millions, which will require a bit more cleaning to do – so here, I’m just thankful that all I have to do is to remove some comma separators.

dball\_data\_1 %>%

mutate\_at("Power\_Level", ~str\_remove\_all(., ",")) %>%

mutate\_at("Power\_Level", ~as.numeric(.)) -> dball\_data\_2

dball\_data\_2 %>% tail()

## # A tibble: 6 x 4

## Character Power\_Level Saga\_or\_Movie Dragon\_Ball\_Seri~

##

## 1 Goku's Super Kame~ 25300000000 OVA: Plan to Eradicate the ~ Dragon Ball Z

## 2 Family Kamehameha 300000000000 Movie 10: Broly- The Second~ Dragon Ball Z

## 3 Krillin's Kameham~ 8000000 Movie 11: Bio-Broly Dragon Ball Z

## 4 Goten's Kamehameha 950000000 Movie 11: Bio-Broly Dragon Ball Z

## 5 Trunk's Kamehameha 980000000 Movie 11: Bio-Broly Dragon Ball Z

## 6 Goten's Super Kam~ 3000000000 Movie 11: Bio-Broly Dragon Ball Z

Now that we’ve fixed the Power\_Level column, the next step is to isolate the information about the characters from the Character column. The reason why we have to do this is because, inconveniently, the column provides information for both the *character* and *the occasion* of when the kamehameha is used, which means we won’t be able to easily filter or group the dataset by the characters only.

One way to overcome this problem is to use the apostrophe (or single quote) as a delimiter to extract the characters from the column. Before I do this, I will take another manual step to remove the rows corresponding to absorbed kamehamehas, e.g. *Android 19 (Goku’s kamehameha absorbed)*, as it refers to the character’s power level *after* absorbing the attack, rather than the attack itself. (Yes, some characters are able to absorb kamehameha attacks and make themselves stronger..!)

After applying the filter, I use mutate() to create a new column called Character\_Single, and then str\_remove\_all() to remove all the characters that appear after the apostrophe:

dball\_data\_2 %>%

filter(!str\_detect(Character, "absorbed")) %>% # Remove 2 rows unrelated to kamehameha attacks

mutate(Character\_Single = str\_remove\_all(Character, "\\'.+")) %>% # Remove everything after apostrophe

select(Character\_Single, everything()) -> dball\_data\_3

## # A tibble: 10 x 5

## Character\_Single Character Power\_Level Saga\_or\_Movie Dragon\_Ball\_Ser~

##

## 1 Master Roshi Master Roshi's~ 180 Emperor Pilaf ~ Dragon Ball

## 2 Goku Goku's Kameham~ 12 Emperor Pilaf ~ Dragon Ball

## 3 Jackie Chun Jackie Chun's ~ 330 Tournament Saga Dragon Ball

## 4 Goku Goku's Kameham~ 90 Red Ribbon Arm~ Dragon Ball

## 5 Goku Goku's Kameham~ 90 Red Ribbon Arm~ Dragon Ball

## 6 Goku Goku's Super K~ 740 Piccolo Jr. Sa~ Dragon Ball

## 7 Goku Goku's Kameham~ 950 Saiyan Saga Dragon Ball Z

## 8 Goku Goku's Kameham~ 36000 Saiyan Saga Dragon Ball Z

## 9 Goku Goku's Kameham~ 44000 Saiyan Saga Dragon Ball Z

## 10 Goku Goku's Angry K~ 180000000 Frieza Saga Dragon Ball Z

Note that the apostrophe is a special character, and therefore it needs to be escaped by adding two forward slashes before it. The dot (.) matches all characters, and + tells R to match the preceding dot to match one or more times. Regex is a very useful thing to learn, and I would highly recommend just reading through the linked references below if you’ve never used regular expressions before.[1](https://martinctc.github.io/blog/data-cleaning-with-kamehamehas-in-r/#fn1)

**Analysis**

Now that we’ve got a clean dataset, what can we find out about the Kamehamehas?

The Kamehameha – image from Giphy

My approach is start with calculating the average power levels of Kamehamehas in R, grouped by Character\_Single. The resulting table tells us that on average, Goku’s Kamehameha is the most powerful, followed by Gohan:

dball\_data\_3 %>%

group\_by(Character\_Single) %>%

summarise\_at(vars(Power\_Level), ~mean(.)) %>%

arrange(desc(Power\_Level)) -> kame\_data\_grouped # Sort by descending

kame\_data\_grouped

## # A tibble: 11 x 2

## Character\_Single Power\_Level

##

## 1 Goku 3.46e14

## 2 Gohan 1.82e12

## 3 Family Kamehameha 3.00e11

## 4 Super Perfect Cell 8.00e10

## 5 Perfect Cell 3.02e10

## 6 Goten 1.98e 9

## 7 Trunk 9.80e 8

## 8 Krillin 8.00e 6

## 9 Student-Teacher Kamehameha 1.70e 4

## 10 Jackie Chun 3.30e 2

## 11 Master Roshi 1.80e 2

However, it’s not helpful to directly visualise this on a bar chart, as the Power Level of the strongest Kamehameha is 175,433 times greater than the median!

kame\_data\_grouped %>%

pull(Power\_Level) %>%

summary()

## Min. 1st Qu. Median Mean 3rd Qu. Max.

## 1.800e+02 4.008e+06 1.975e+09 3.170e+13 1.900e+11 3.465e+14

A way around this is to log transform the Power\_Level variable prior to visualising it, which I’ve saved the data into a new column called Power\_Index. Then, we can pipe the data directly into a ggplot chain, and set a dark mode using theme():

kame\_data\_grouped %>%

mutate(Power\_Index = log(Power\_Level)) %>% # Log transform Power Levels

ggplot(aes(x = reorder(Character\_Single, Power\_Level),

y = Power\_Index,

fill = Character\_Single)) +

geom\_col() +

coord\_flip() +

scale\_fill\_brewer(palette = "Spectral") +

theme\_minimal() +

geom\_text(aes(y = Power\_Index,

label = round(Power\_Index, 1),

hjust = -.2),

colour = "#FFFFFF") +

ggtitle("Power Levels of Kamehamehas", subtitle = "By Dragon Ball characters") +

theme(plot.background = element\_rect(fill = "grey20"),

text = element\_text(colour = "#FFFFFF"),

panel.grid = element\_blank(),

plot.title = element\_text(colour="#FFFFFF", face="bold", size=20),

axis.line = element\_line(colour = "#FFFFFF"),

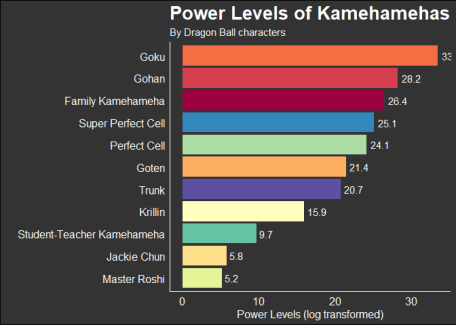
legend.position = "none",

axis.title = element\_text(colour = "#FFFFFF", size = 12),

axis.text = element\_text(colour = "#FFFFFF", size = 12)) +

ylab("Power Levels (log transformed)") +

xlab(" ")



So as it turns out, the results aren’t too surprising. Goku’s Kamehameha is the strongest of all the characters on average, although it has been referenced several times in the series that his son Gohan’s latent powers are beyond Goku’s.

Also, it is perhaps unsurprising that Master Roshi’s Kamehameha is the least powerful, given a highly powered comparison set of characters. Interestingly, Roshi’s Kamehameha is stronger as ‘Jackie Chun’ than as himself.

We can also see the extent to which Goku’s Kamehameha has grown more powerful across the series. This is available in the column Saga\_or\_Movie. In the same approach as above, we can do this by grouping the data by Saga\_or\_Movie, and pipe this into a ggplot bar chart:

dball\_data\_3 %>%

filter(Character\_Single == "Goku") %>%

mutate(Power\_Index = log(Power\_Level)) %>% # Log transform Power Levels

group\_by(Saga\_or\_Movie) %>%

summarise(Power\_Index = mean(Power\_Index)) %>%

ggplot(aes(x = reorder(Saga\_or\_Movie, Power\_Index),

y = Power\_Index)) +

geom\_col(fill = "#F85B1A") +

theme\_minimal() +

geom\_text(aes(y = Power\_Index,

label = round(Power\_Index, 1),

vjust = -.5),

colour = "#FFFFFF") +

ggtitle("Power Levels of Goku's Kamehamehas", subtitle = "By Saga/Movie") +

scale\_y\_continuous(limits = c(0, 40)) +

theme(plot.background = element\_rect(fill = "grey20"),

text = element\_text(colour = "#FFFFFF"),

panel.grid = element\_blank(),

plot.title = element\_text(colour="#FFFFFF", face="bold", size=20),

plot.subtitle = element\_text(colour="#FFFFFF", face="bold", size=12),

axis.line = element\_line(colour = "#FFFFFF"),

legend.position = "none",

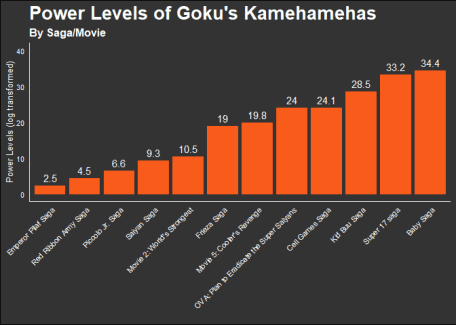
axis.title = element\_text(colour = "#FFFFFF", size = 10),

axis.text.y = element\_text(colour = "#FFFFFF", size = 8),

axis.text.x = element\_text(colour = "#FFFFFF", size = 8, angle = 45, hjust = 1)) +

ylab("Power Levels (log transformed)") +

xlab(" ")



I don’t have full knowledge of the chronology of the franchise, but I do know that *Emperor Pilaf Saga*, *Red Ribbon Army Saga*, and *Piccolo Jr. Saga* are the earliest story arcs where Goku’s martial arts abilities are still developing. It also appears that if I’d like to witness Goku’s most powerful Kamehameha attack, I should find this in the *Baby Saga*!

**Notes**

Hope this was an interesting read for you, and that this tells you something new about R or Dragon Ball.

There is certainly more you can do with this dataset, especially once it is processed into a usable, tidy format.

If you have any related datasets that will help make this analysis more interesting, please let me know!