Competitive Pokemon Type Balance

amcooper2

7 August 2023

Has the Pokemon Company successfully balanced each type competitively?

Research Question

In the most recent generation of Pokemon games (Generation 9), they have attempted to improve the balance of each of the types competitively, as well as giving Pokemon the ability to change their type once in battle, to sure up weaknesses. Have these changes combined helped to actually increase type diversity across the first set of competitive Pokemon tournaments of this generation compared to the previous?

Data Origins

The website "limitlessvgc.com" has kept track of the number of times each Pokemon has reached the top 8 of a major competitive tournament in any team. This project uses web scraping to gather both this data and the data from the following website.

The website "pokemondb.net" has a full list of Pokemon and the types that they belong to. This project uses this resource to create a lookup table to be able to work out the types of a Pokemon from the named stored in the tournament data.

Unfortunately certain Pokemon with unique forms have had different naming conventions across different websites and so creating a perfect lookup table became difficult, meaning some manual input would be necessary for 100% of the data to be usable. This could greatly affect the newer generation as well, as there have been fewer tournaments so far, meaning the data is alredy more limited in comparison.

On a longer scale project I would consider applying a best match algorithm as a part of the lookup search in order to find the closest matching name in the lookup table and help to remove any need for manual input without losing data.

But for the sake of this project the visualisation has been done without this, but with graphs still being created using manually entered data to ensure that the conclusions are not misleading.

Using web scraping in order to collect the data from "limitlessvgc.com":

```
# POINTS EARNED DATA

link_old <-
  "https://limitlessvgc.com/pokemon/?rank=points&time=all&type=all&region=all&format=vgc22&show=100"
page_old <- read_html(link_old)

pokemon_old <- page_old %>%
  html_nodes(".pokemon-link") %>%
  html_text()
points_old <- page_old %>%
```

```
  html_nodes("td:nth-child(4)") %>%
  html_text()
share_old <- page_old %>%
  html_nodes("td:nth-child(5)") %>%
  html_text()

data_old <- data.frame(pokemon_old, points_old, share_old)
# write.csv(data_old, "data_gen_8.csv")

# Prints out the length and the head
print(dim(data_old))
```

```
## [1] 99  3
```

```
print(head(data_old))
```

```
##   pokemon_old points_old share_old
## 1  Incineroar       3199    81.30%
## 2      Zacian       2465    62.64%
## 3      Kyogre       1543    39.21%
## 4   Thundurus       1307    33.21%
## 5  Grimmsnarl       1242    31.56%
## 6   Rillaboom       1201    30.52%
```

Then we do the same for the data from the new generation:

```
## [1] 64  3
```

```
##   pokemon_new points_new share_new
## 1   Gholdengo        629    61.79%
## 2 Meowscarada        445    43.71%
## 3   Amoonguss        426    41.85%
## 4   Kingambit        344    33.79%
## 5    Pelipper        253    24.85%
## 6     Dondozo        241    23.67%
```

Then finally the type lookup chart:

```
## [1] 1194    2
```

```
##           pokemon_names pokemon_types
## 1             Bulbasaur   GrassPoison
## 2               Ivysaur   GrassPoison
## 3              Venusaur   GrassPoison
## 4 VenusaurMega Venusaur   GrassPoison
## 5            Charmander          Fire
## 6            Charmeleon          Fire
```

Variables The number of points represents the number of top 8 appearences that Pokemon had, the share the proportion of teams it appeared in, which was unused in this project, but interesting data none the less. (With 6 Pokemon on each team, the share could sometimes be very high).

Data Preparation

Here is what will become the final dataframe, though there will be alternative versions of it down the line for when certain visual representations will require them.

```
data_final <- data.frame (
  types = c("Normal", "Fire", "Water", "Grass", "Electric", "Ice", "Fighting",
            "Poison", "Ground", "Flying", "Psychic", "Bug", "Rock", "Ghost",
            "Dragon", "Dark", "Steel", "Fairy"),
  old_values = c(0),
  new_values = c(0)
)
```

The data was first prepared by using the lookup table and assigning each Pokemon their respective types where possible.

```
types_old <- c()
for (x in data_old$pokemon_old) {
  type_row <- data_types[data_types$pokemon_names == x,]
  type <- type_row[1,2]
  #print(type)
  types_old <- append(types_old, type)
}

data_old$types <- types_old
print(head(data_old))
```

```
##   pokemon_old points_old share_old     types
## 1  Incineroar       3199    81.30%  FireDark
## 2      Zacian       2465    62.64%      <NA>
## 3      Kyogre       1543    39.21%     Water
## 4   Thundurus       1307    33.21%      <NA>
## 5  Grimmsnarl       1242    31.56% DarkFairy
## 6   Rillaboom       1201    30.52%     Grass
```

Then we do the same for generation 9.

```
##   pokemon_new points_new share_new       types
## 1   Gholdengo        629    61.79%  SteelGhost
## 2 Meowscarada        445    43.71%   GrassDark
## 3   Amoonguss        426    41.85% GrassPoison
## 4   Kingambit        344    33.79%   DarkSteel
## 5    Pelipper        253    24.85% WaterFlying
## 6     Dondozo        241    23.67%       Water
```

As you can see, there are already some values that can be seen in the old data.

Below is a quick calculation of how much of the data is missing due to this:

```
## Old error: 0.2984329
```

```
## New error: 0.1591356
```

(This is the proportion of the representation that is missing in comparison to the maximum amount of representation which could be measured).

The final dataframe, however, can be seen below:

```
# FINAL DATA

old_total = 0
new_total = 0
old_error = 0
new_error = 0

for (y in rownames(data_old)) {
  y <- as.numeric(y)

  current_type <- data_old$types[y]
  current_points <- data_old$points_old[y]
  current_name <- data_old$pokemon_old[y]

  current_points <- as.numeric(current_points)
  for (z in rownames(data_final)) {
    z <- as.numeric(z)
    test_type <- data_final$types[z]
    if (!is.na(current_type)) {
      old_total <- old_total + current_points
      statement <- grepl(test_type, current_type, fixed=TRUE)
      #print(test_type)
      #print(statement)
      if (grepl(test_type, current_type, fixed=TRUE)) {
        data_final$old_values[z] <- data_final$old_values[z] + current_points
      }

    } else {
      old_error <- old_error + current_points
    }
  }
}


for (y in rownames(data_new)) {
  y <- as.numeric(y)

  current_type <- data_new$types[y]
  current_points <- data_new$points_new[y]

  current_points <- as.numeric(current_points)
  for (z in rownames(data_final)) {
    z <- as.numeric(z)
    test_type <- data_final$types[z]
    if (!is.na(current_type)) {
      new_total <- new_total + current_points
      statement <- grepl(test_type, current_type, fixed=TRUE)
      #print(test_type)
      #print(statement)
      if (grepl(test_type, current_type, fixed=TRUE)) {
```

```
        data_final$new_values[z] <- data_final$new_values[z] + current_points
      }
    } else {
      new_error <- new_error + current_points
    }
  }
}

print(head(data_final))
```

```
##      types old_values new_values
## 1   Normal        524         31
## 2     Fire       4178        580
## 3    Water       3253        714
## 4    Grass       3406        930
## 5 Electric       1175        232
## 6      Ice         10        181
```

Alternative Dataframes Two more alternative versions of this dataframe were also made.

One which uses a scalar factor and a separate column to state which generation the data came from, to mitigate the difference in sample size for a future visualisation:

```
# DATA SCALING AND SORTING

# Scale factor
scale_factor <- mean(data_final$old_values)/mean(data_final$new_values)

merged_data <- data.frame(status=rep(c("Generation 8", "Generation 9"), each=18),
                          type=rep(c(data_final$types),2),
                          representation=c(data_final$old_values,
                                           data_final$new_values*scale_factor))

# write.csv(merged_data, "merged_data.csv")
print(head(merged_data))
```

```
##         status     type representation
## 1 Generation 8   Normal            524
## 2 Generation 8     Fire           4178
## 3 Generation 8    Water           3253
## 4 Generation 8    Grass           3406
## 5 Generation 8 Electric           1175
## 6 Generation 8      Ice             10
```

And another which contains the data sorted in order from least to most represented:

```
# Sorting Data
sorted_old_data <- data.frame (
  type = c(data_final$types),
  representation = c(data_final$old_values)
)
```

```r
sorted_old_data <- sorted_old_data[order(sorted_old_data$representation),]

sorted_new_data <- data.frame (
  type = c(data_final$types),
  representation = c(data_final$new_values)
)

sorted_new_data <- sorted_new_data[order(sorted_new_data$representation),]

sorted_data <- data.frame(generation=rep(c("Generation 8", "Generation 9"), each=18),
                          rank=rep(1:18,2),
                          representation=c(sorted_old_data$representation,
                                sorted_new_data$representation*scale_factor),
                          type=c(sorted_old_data$type, sorted_new_data$type))

# write.csv(sorted_data, "sorted_data.csv")
print(head(sorted_new_data))
```

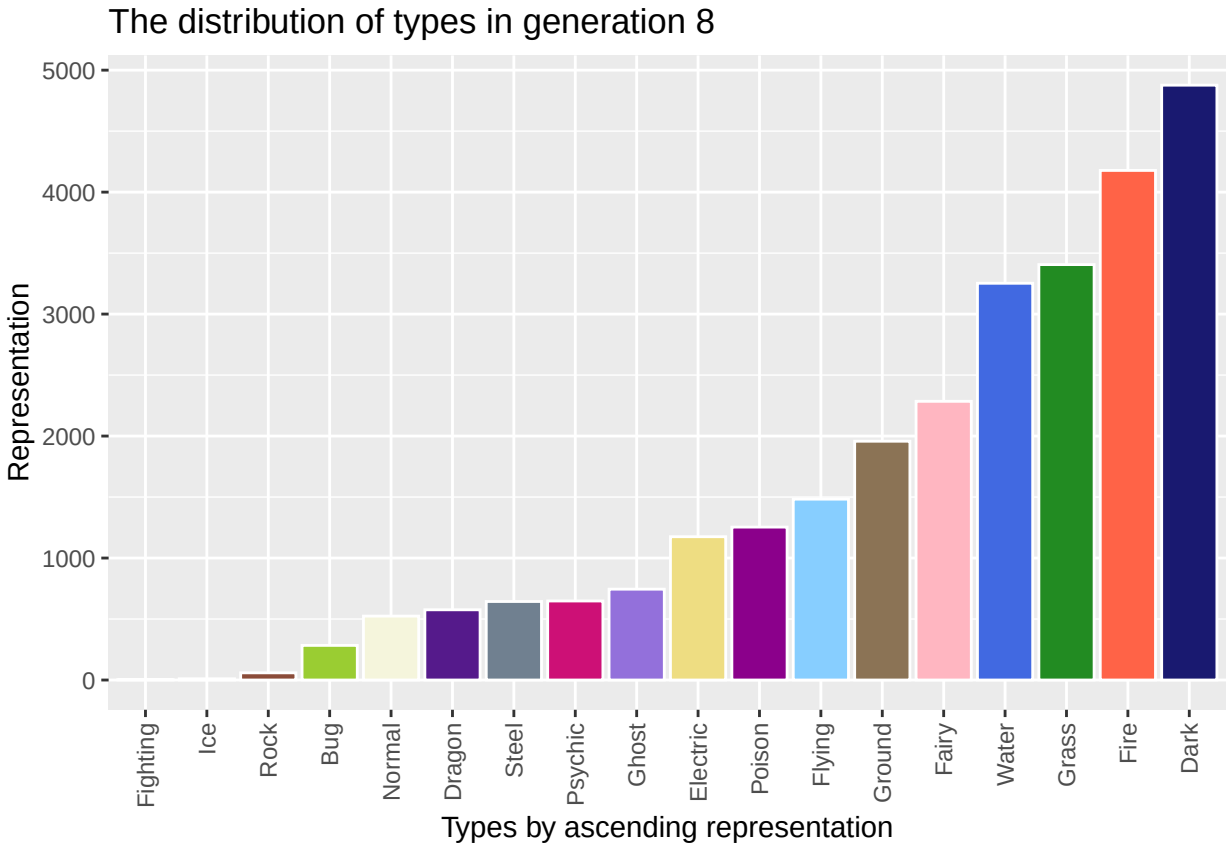
```
##        type representation
## 1    Normal             31
## 12      Bug            161
## 9    Ground            172
## 6       Ice            181
## 11  Psychic            197
## 5  Electric            232
```

Visualisations

The first two visualisations are the sorted distributions of each type in each generation.

Generation 8 Type Representation

```r
# Generation 8
ggplot(data = data_final, aes(x = reorder(types, +old_values), y = old_values,
                              fill = types)) +
  geom_bar(stat="identity", color="white") +
  scale_fill_manual(values=type_colourings) +
  scale_color_manual(values=) +
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5),
        legend.position = "none") +
  labs (title="The distribution of types in generation 8",
        x ="Types by ascending representation", y = "Representation")
```

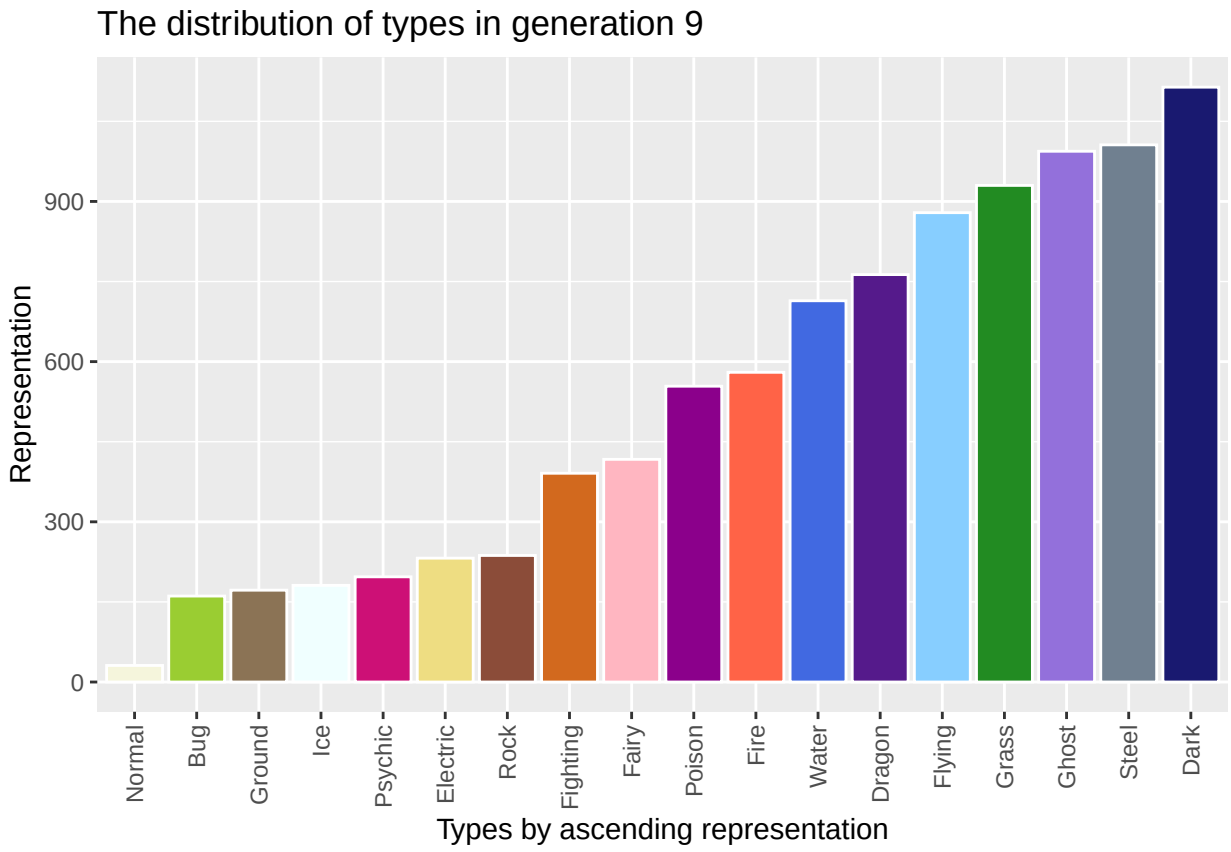


As you can see, some types are not even represented in generation 8, though it is worth noting that by manually entering in the missing data we can see that they should all in fact have some representation, even if it is still low.

(Alternative graphs with 100% representation are shown at the end)

Generation 9 Type Representation

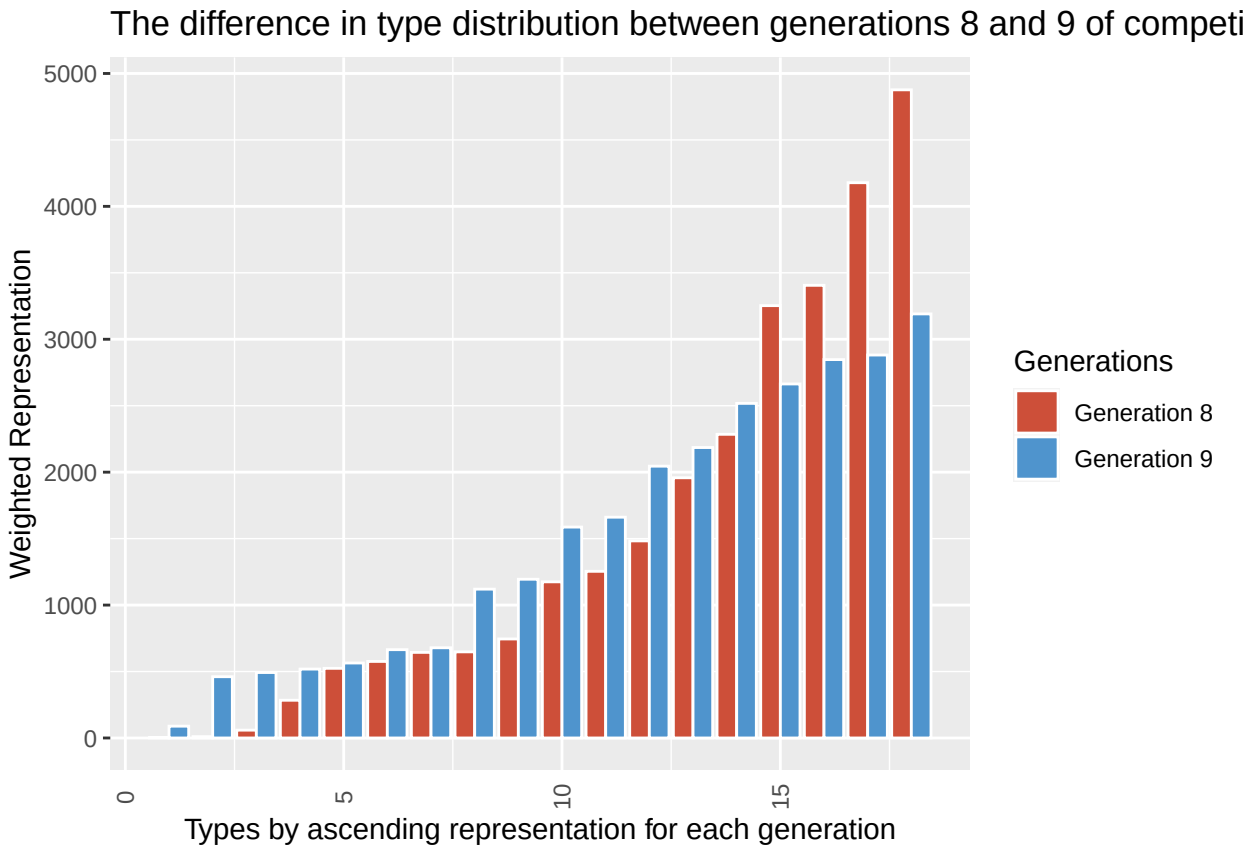
```
ggplot(data = data_final, aes(x = reorder(types, +new_values), y = new_values,
                              fill = types)) +
  geom_bar(stat="identity", color="white") +
  scale_fill_manual(values=type_colourings) +
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5),
        legend.position = "none") +
  labs (title="The distribution of types in generation 9",
        x ="Types by ascending representation", y = "Representation")
```



Combined Distributions

Here is a framing of both distributions next to eachother over the same axis, using the sorted dataframe created earlier.

```
# Side By Side Bar Chart
# Coloured by generation
ggplot(sorted_data, aes(fill=generation, y=representation, x=rank)) +
  geom_bar(position="dodge", stat="identity", color="white") +
  scale_fill_manual(values=generation_colourings) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5),
        axis.ticks.x=element_blank()) +
  labs (title="The difference in type distribution between generations 8 and 9 of competitive pokemon",
        x ="Types by ascending representation for each generation",
        y = "Weighted Representation", fill="Generations")
```

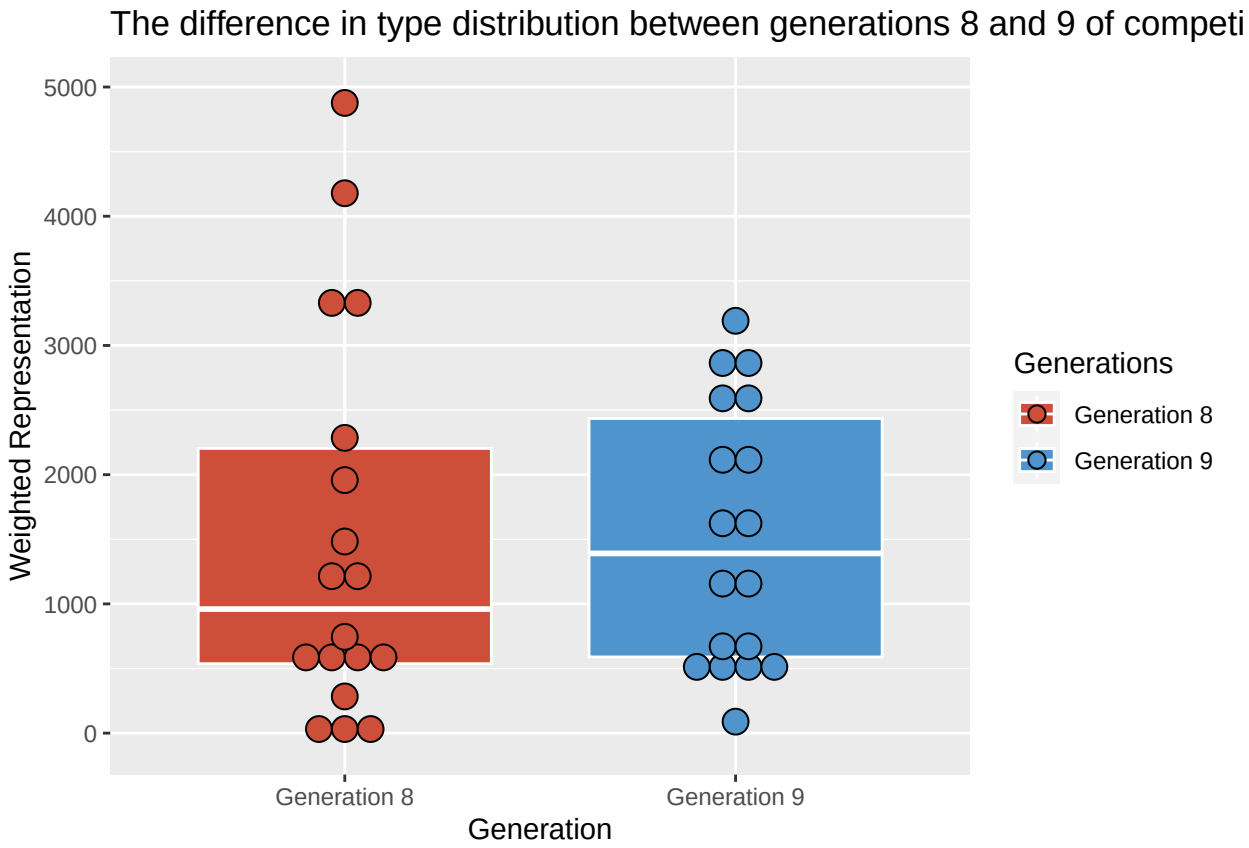


This can already begin to quite clearly show the much higher peaks and lower troughs found in generation 8.

Comparing Quartiles

Finally is a box plot using weighted representation thanks to the scaled dataframe created earlier, which I believe to be the clearest visualisation of the improved representation in generation 9 compared to generation 8:

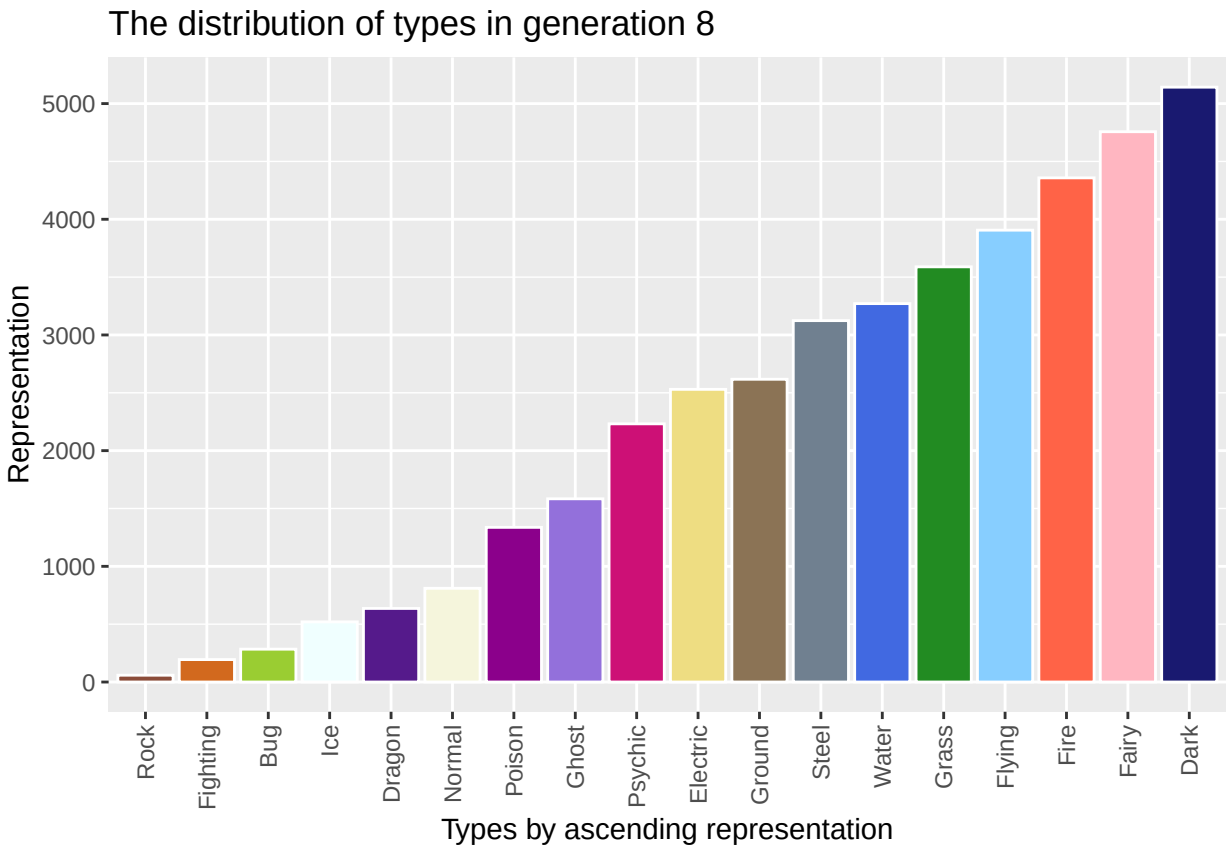
```
# Box Plot
ggplot(merged_data, aes(x=status, y=representation, fill=status)) +
  geom_boxplot(color="white") +
  scale_fill_manual(values=generation_colourings) +
  geom_dotplot(binaxis='y', stackdir='center', dotsize=1, binwidth=200) +
  labs (title="The difference in type distribution between generations 8 and 9 of competitive pokemon",
        x ="Generation", y = "Weighted Representation", fill="Generations")
```



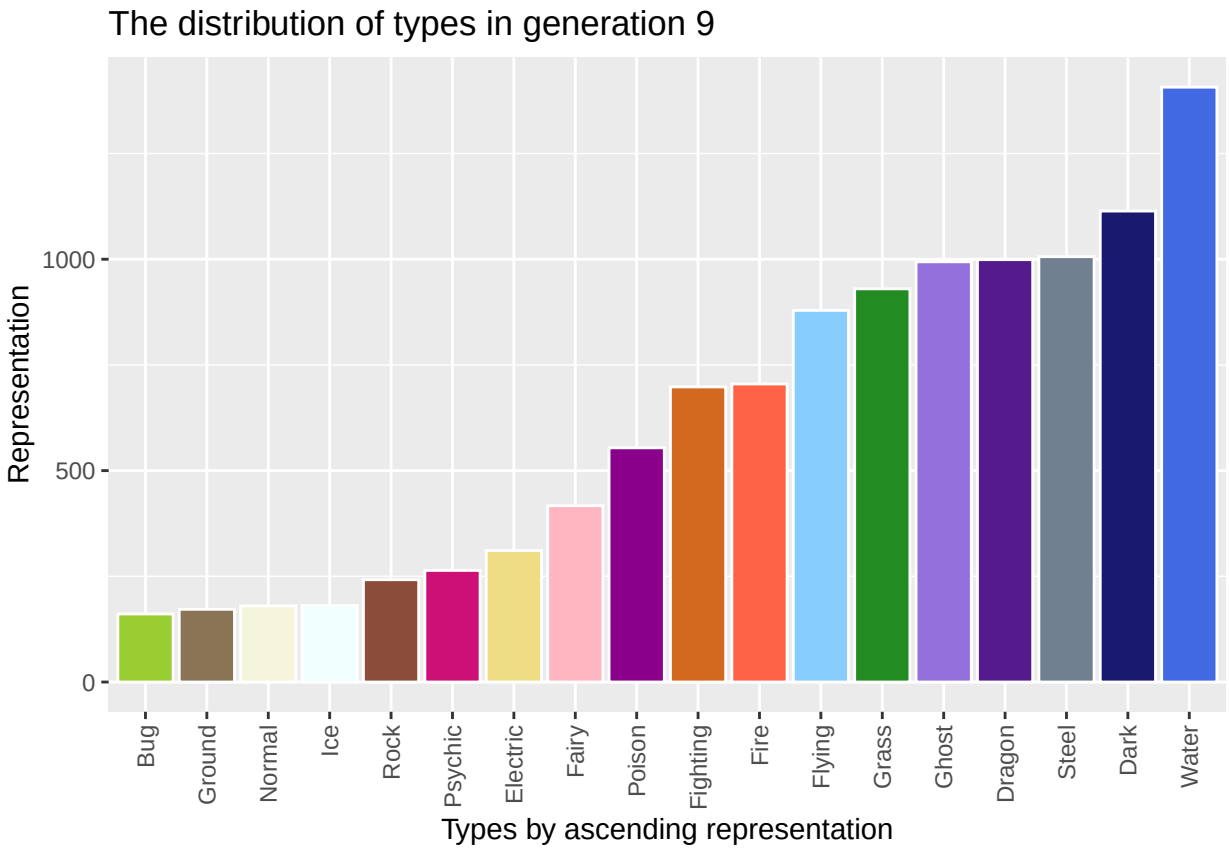
The improved representation here is represented by the more compact box plot shown in generation 9.

Alternative Visualisation (with manually entered data) Below are the versions of these same visualisations, but with 100% of the data through manual entering of the missing type values:

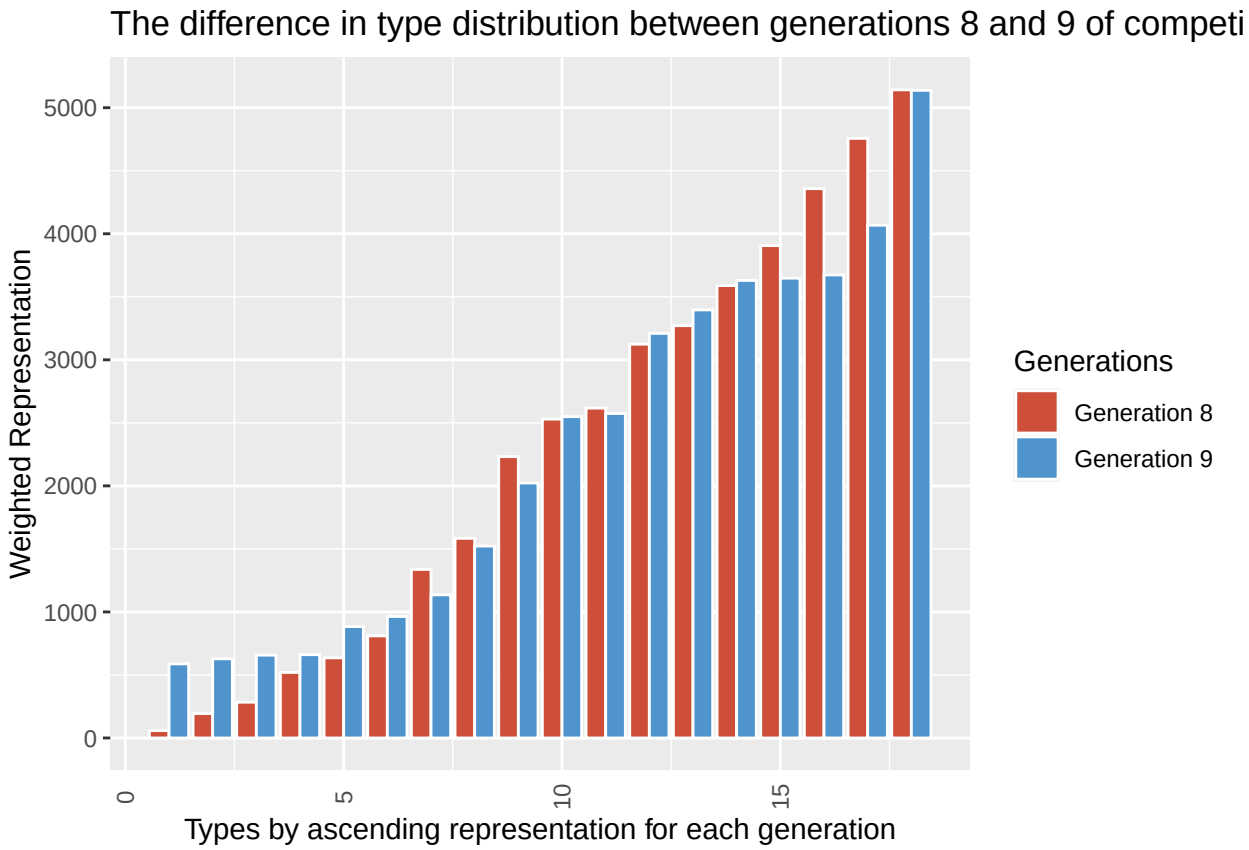
Generation 8:



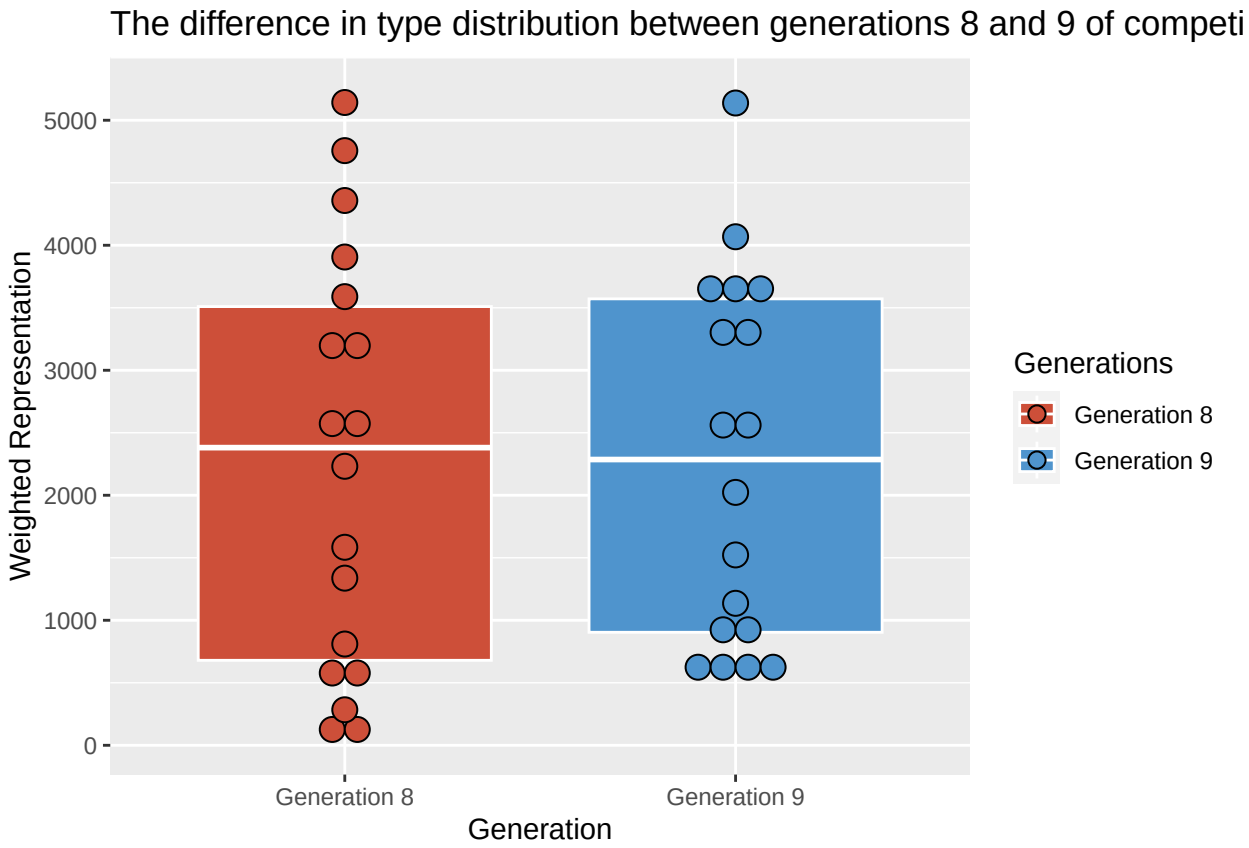
Generation 9:


The distribution of types in generation 9
Representation
1000
500
0
Bug
Ground
Normal
Ice
Rock
Psychic
Electric
Fairy
Poison
Fighting
Fire
Flying
Grass
Ghost
Dragon
Steel
Dark
Water
Types by ascending representation

Combined:



Quartiles box plot:



Summary

Overall there does appear to be some better representation between Pokemon types in the new generation.

It is important to note, however that when 100% of the data is used, this improvement does lessen.

Were I to continue to do this, I would like to improve how much data could be used without manual input through the use of a best match algorithm as a part of the lookup.

I might also look into the use of ggpattern in order to potentially add many new tools to the visualisation of these graphs such as textures.

Finally, I would wait until the end of the second year of the game's release to do it again as each generation tends to cover 2 yearly world tournaments each and so by waiting until after they were complete I would have access to much more data from generation 9 and could potentially even lessen the need for a scale factor.