Mark Peters

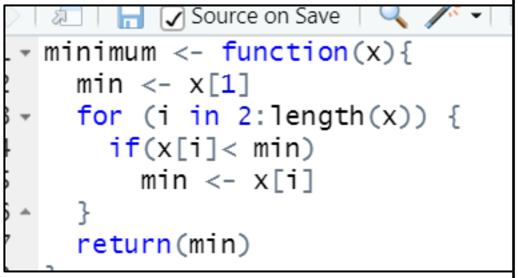
R Progammering Inlämningsuppgift

Once the game begins, who knows when it will stop?

1. Basic algorithmic thinking in R:

I started by designing each of the functions needed to run the program. It was stressed that it should be our focus, so I wanted to get it right.

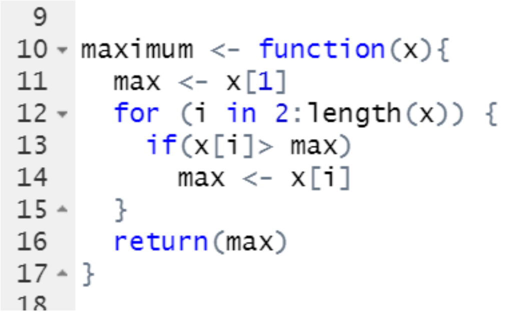
Minimum function:



```
minimum <- function(x){
  min <- x[1]
  for (i in 2:length(x)) {
    if(x[i]< min)
      min <- x[i]
  }
  return(min)
```

The function accepts a vector and assigns the first value in the list to ‘min’. It then cycles the rest of the values through a ‘for’ loop, from the second to the last in the list (length giving the function the point at which to stop), replacing the min value if a smaller one is found. This value is then returned to the parent program.

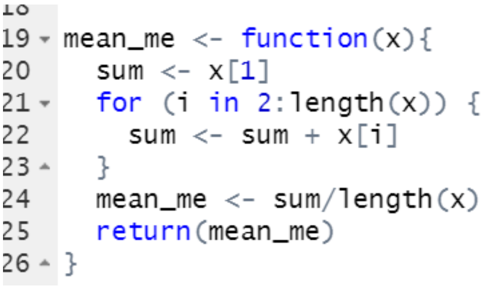
Maximum function:



```
maximum <- function(x){
  max <- x[1]
  for (i in 2:length(x)) {
    if(x[i]> max)
      max <- x[i]
  }
  return(max)
}
```

The maximum function applies almost the same process as the minimum except, naturally, replacing the ‘less than’ sign with the ‘greater than’. It cycles through all supplied values and replaces ‘max’ with an element that is larger than it.

Mean function:

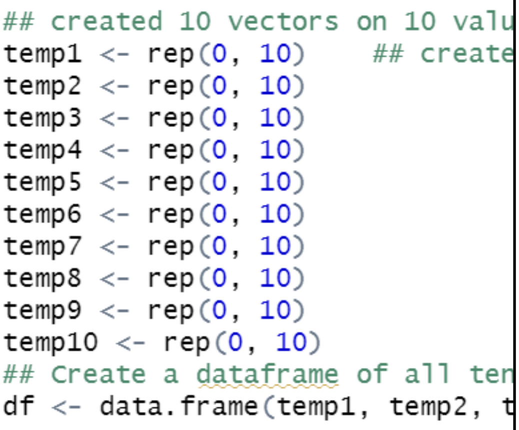


```
mean_me <- function(x){
  sum <- x[1]
  for (i in 2:length(x)) {
    sum <- sum + x[i]
  }
  mean_me <- sum/length(x)
  return(mean_me)
}
```

The mean function – which I christened ‘mean-me’ to avoid confusion – uses the same structure as the previous two functions (setting an initial value and then cycling through the rest with a for loop) but has an extra stage.

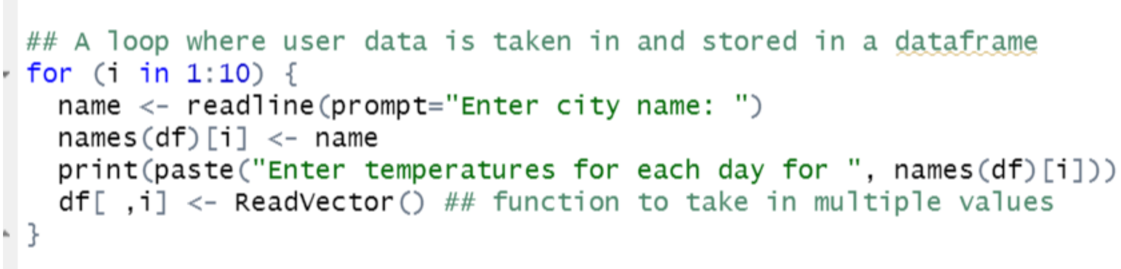
Sum becomes that number plus the next on the list. After it has collated all the numbers in the list it gets divided by the vector length. This gives the mean value for the entire vector.

Dataframe creation:



```
## created 10 vectors on 10 valu
temp1 <- rep(0, 10)     ## create
temp2 <- rep(0, 10)
temp3 <- rep(0, 10)
temp4 <- rep(0, 10)
temp5 <- rep(0, 10)
temp6 <- rep(0, 10)
temp7 <- rep(0, 10)
temp8 <- rep(0, 10)
temp9 <- rep(0, 10)
temp10 <- rep(0, 10)
## Create a dataframe of all ten
df <- data.frame(temp1, temp2, t
```

No doubt far from the most efficient way to create a dataframe, but I created each vector individually and filled them with zeros. I did this because my program asks for the city name first and I needed values in the vector for the name to ‘stick’. All the vectors were placed into a dataframe named ‘df’.



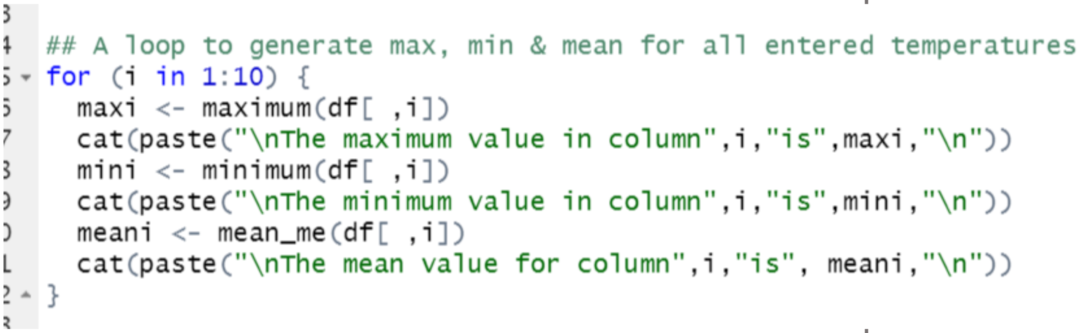
```r
## A loop where user data is taken in and stored in a dataframe
for (i in 1:10) {
  name <- readline(prompt="Enter city name: ")
  names(df)[i] <- name
  print(paste("Enter temperatures for each day for ", names(df)[i]))
  df[ ,i] <- ReadVector() ## function to take in multiple values
}
```

Input User Data:

I used another ‘for’ loop to take input data from the user. I know this isn’t a very ‘user friendly’ way to generate data, but for the purposes of this exercise it works.

The user enters the city name first which is added to the first vector with the ‘names’ function. The ‘ReadVector’ algorithm takes in each temperature value, one at a time, and fills the vector.

The for loop then moves on to the next city, and so on until all ten cities’ values have been entered.

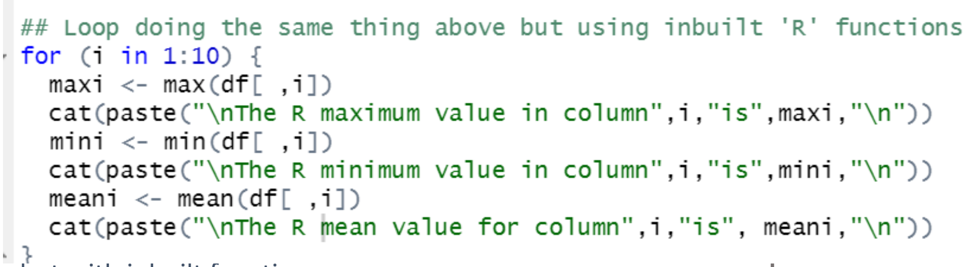


```r
## A loop to generate max, min & mean for all entered temperatures
for (i in 1:10) {
  maxi <- maximum(df[ ,i])
  cat(paste("\nThe maximum value in column",i,"is",maxi,"\n"))
  mini <- minimum(df[ ,i])
  cat(paste("\nThe minimum value in column",i,"is",mini,"\n"))
  meani <- mean_me(df[ ,i])
  cat(paste("\nThe mean value for column",i,"is", meani,"\n"))
}
```

Print out max, min & mean:

Another ‘for’ loop (bit of a one-trick-pony I am) is used to generate min, max and mean values for each of the cities entered by the user. The loop takes the column number and then prints out the values for it (including the city’s name) to the screen.

After printing the three values for the first city it moves onto the second until all ten have been printed to screen.



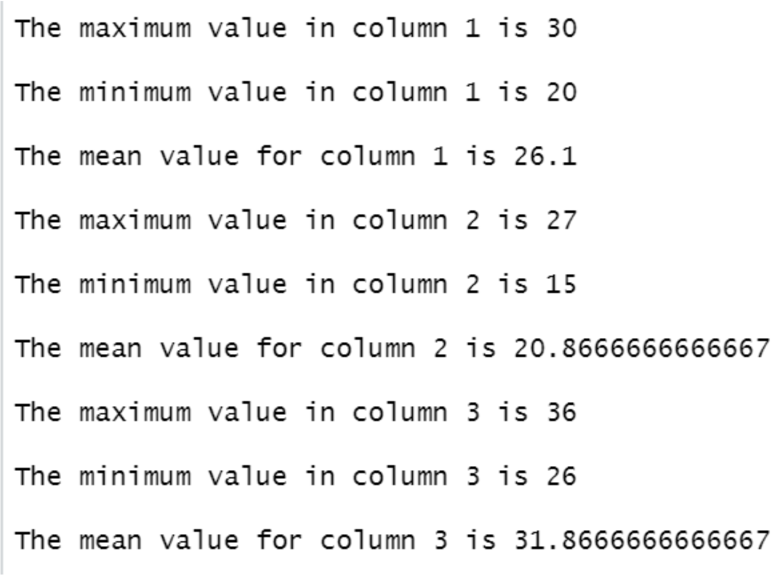
```r
## Loop doing the same thing above but using inbuilt 'R' functions
for (i in 1:10) {
  maxi <- max(df[ ,i])
  cat(paste("\nThe R maximum value in column",i,"is",maxi,"\n"))
  mini <- min(df[ ,i])
  cat(paste("\nThe R minimum value in column",i,"is",mini,"\n"))
  meani <- mean(df[ ,i])
  cat(paste("\nThe R mean value for column",i,"is", meani,"\n"))
}
```

Same results but with inbuilt functions:

I repeat the same process as the previous but generate results with the inbuild R functions instead of my own.

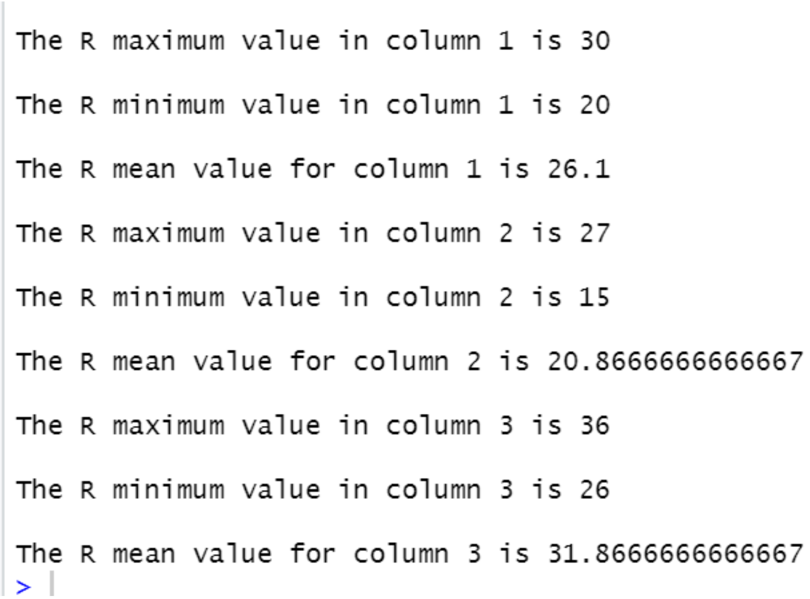
Results of the program using a second test program:

This is a print out of all the values that came from my own functions:



```
The maximum value in column 1 is 30

The minimum value in column 1 is 20

The mean value for column 1 is 26.1

The maximum value in column 2 is 27

The minimum value in column 2 is 15

The mean value for column 2 is 20.8666666666667

The maximum value in column 3 is 36

The minimum value in column 3 is 26

The mean value for column 3 is 31.8666666666667
```

This is a printout of the values generated form the R functions:



```
The R maximum value in column 1 is 30

The R minimum value in column 1 is 20

The R mean value for column 1 is 26.1

The R maximum value in column 2 is 27

The R minimum value in column 2 is 15

The R mean value for column 2 is 20.8666666666667

The R maximum value in column 3 is 36

The R minimum value in column 3 is 26

The R mean value for column 3 is 31.8666666666667
> 
```

Happily, the results were the same.

2.Data Pre-Processing:

For this question I worked with the suggested dataset, mainly because it needed a good amount of processing in order to be usable, and that wasn’t the same for many of the ones that I came across.

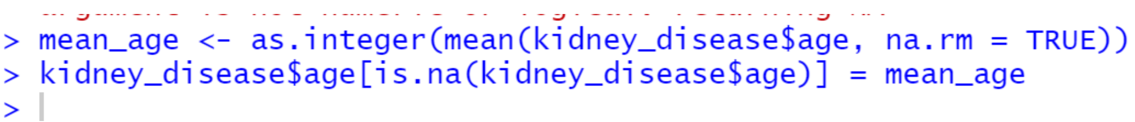
Here is how the chronic kidney disease dataset was described on Kaggle:

The data was taken over a 2-month period in India with 25 features (eg, red blood cell count, white blood cell count, etc). The target is the 'classification', which is either 'ckd' or 'notckd' - ckd=chronic kidney disease.

Here is a full list of each column in the dataset:

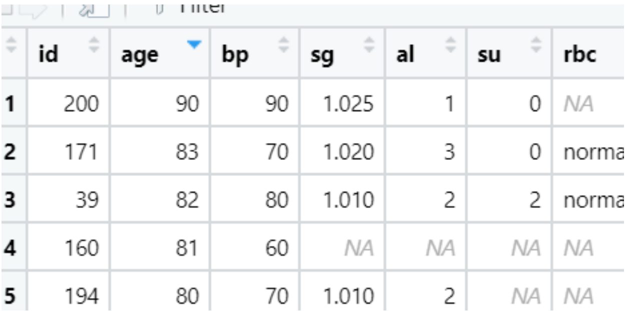
age - age	sod - sodium
bp - blood pressure	pot - potassium
sg - specific gravity	hemo - hemoglobin
al - albumin	pcv - packed cell volume
su - sugar	wc - white blood cell count
rbc - red blood cells	rc - red blood cell count
pc - pus cell	htn - hypertension
pcc - pus cell clumps	dm - diabetes mellitus
ba - bacteria	cad - coronary artery disease
bgr - blood glucose random	appet - appetite
bu - blood urea	pe - pedal edema
sc - serum creatinine	ane - anemia
	class - classification

I worked from left to right. I got the mean of all ages after removing the ‘NA’s’ and them with the mean age of all participants.



```
> mean_age <- as.integer(mean(kidney_disease$age, na.rm = TRUE))
> kidney_disease$age[is.na(kidney_disease$age)] = mean_age
> 
```

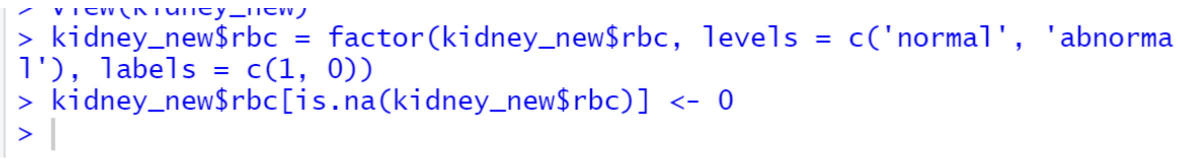
I repeated the same steps for each of the following columns until I got to ‘rbc’ (red blood cells).



	id	age	bp	sg	al	su	rbc
1	200	90	90	1.025	1	0	NA
2	171	83	70	1.020	3	0	norma
3	39	82	80	1.010	2	2	norma
4	160	81	60	NA	NA	NA	NA
5	194	80	70	1.010	2	NA	NA

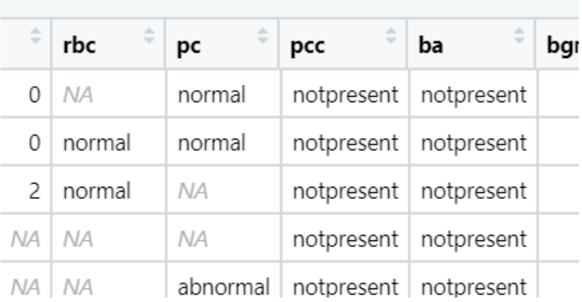
In ‘rbc’ we meet ‘characters’ for the first time. My approach here was what was presented to us in class – that in order to make the code machine readable the bivalent values should be changed to either a numerical ‘1 and 0’ or a logical ‘True and False’.

I assigned ‘normal’ to ‘1’ and ‘abnormal’ to a ‘0’ and changed the values for the whole column, and subsequently assigned ‘NA’ to ‘0’.



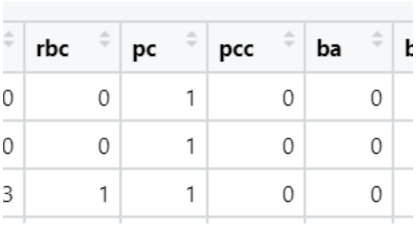
```
> View(kidney_new)
> kidney_new$rbc = factor(kidney_new$rbc, levels = c('normal', 'abnorma
l'), labels = c(1, 0))
> kidney_new$rbc[is.na(kidney_new$rbc)] <- 0
> 
```

This process was repeated for another four columns, the only difference being the nature of the character strings being replaced:



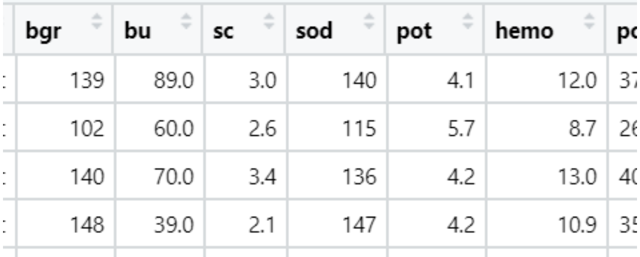
	rbc	pc	pcc	ba	bgr
0	NA	normal	notpresent	notpresent	
0	normal	normal	notpresent	notpresent	
2	normal	NA	notpresent	notpresent	
NA	NA	NA	notpresent	notpresent	
NA	NA	abnormal	notpresent	notpresent	

the ‘pcc’ and ‘ba’ columns (0,1) columns were ‘notpresent’ and ‘present’ respectively.



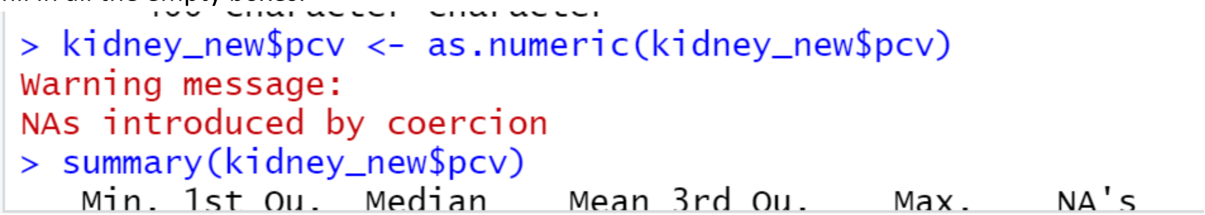
	rbc	pc	pcc	ba	b
0	0	1	0	0	
0	0	1	0	0	
3	1	1	0	0	

The next six columns were treated the same way as the first five, by calculating the mean value of each column and replacing the ‘NA’ values with the result.



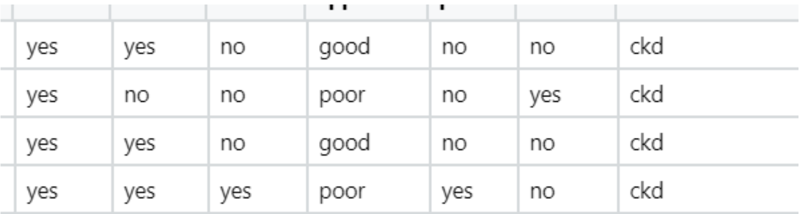
bgr	bu	sc	sod	pot	hemo	pc
139	89.0	3.0	140	4.1	12.0	37
102	60.0	2.6	115	5.7	8.7	26
140	70.0	3.4	136	4.2	13.0	40
148	39.0	2.1	147	4.2	10.9	35

Interestingly, the ‘pcv’ column threw me a curveball, as though it was a list of integer values they were listed as being ‘characters. I assume that was because they were entered into the dataset from user input. This meant I had to change all the values to ‘numeric’ before I could find the mean and so fill in all the empty boxes.



```
> kidney_new$pcv <- as.numeric(kidney_new$pcv)
Warning message:
NAs introduced by coercion
> summary(kidney_new$pcv)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
```

The next two columns were numerical and the remaining eight were all characters – all of them bivalent in nature. They had different permutations but were all either true or false in nature – ‘yes/no’, ‘good/poor’ and ‘ckd/notckd’.



yes	yes	no	good	no	no	ckd
yes	no	no	poor	no	yes	ckd
yes	yes	no	good	no	no	ckd
yes	yes	yes	poor	yes	no	ckd

All columns were assigned ones and zeros and NA’s were removed.

By the end of this process I had a dataset that was machine ready with all NA values removed and all columns being ‘numeric’ in form.

3. DataVisualization:

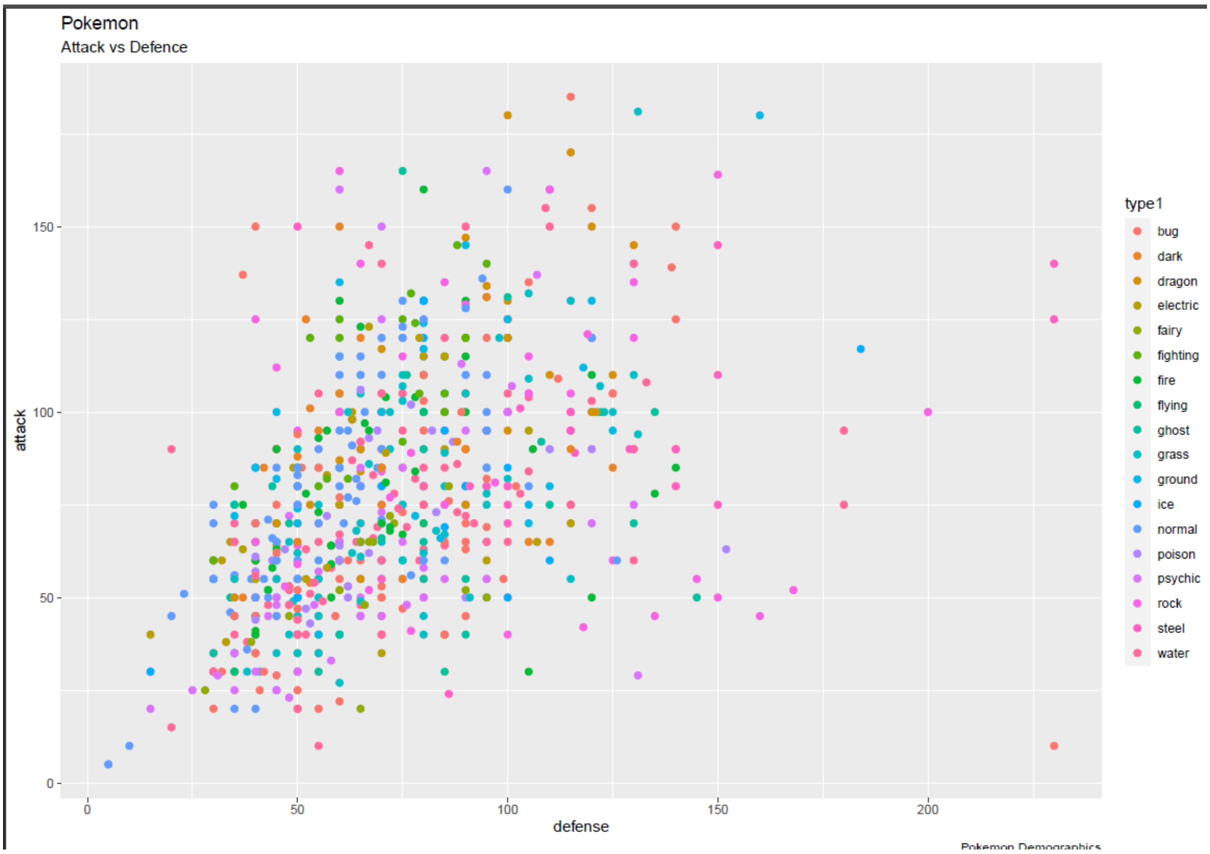
As I approach the end of a very long and intense couple of weeks, the goal very nearly in sight, I have already begun to critique my approach to the tasks. Maybe this is premature since I haven't finished yet, but I can already admit that I spent (and wasted) way too much time trying to find a dataset to work with, and even more time trying to decide what kind of graph would best portray the data. Hours were spent trying things out and then binning the results.

It's not clear but it seems you may have wanted to see all those abortive attempts to produce something of value. I regret my petulance in trashing the vast majority of what I did, if for no other reason than to illustrate the amount of time I've put into these tasks.

Not a time to stress the material I didn't keep and focus on what I did.

Practice:

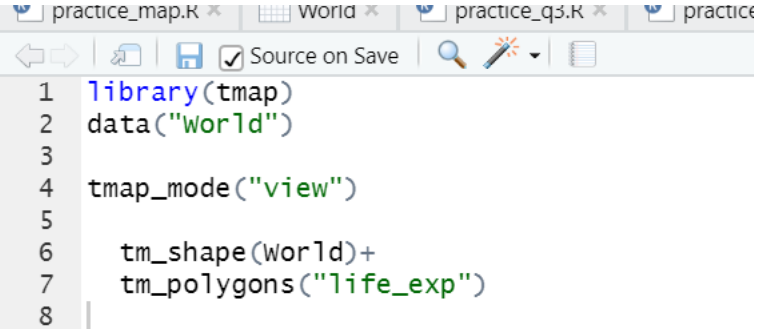
I made a scatterplot of the attack vs defence capabilities of all (currently) discovered Pokemon for my kids. There were too many to name, so it's divided by colour into the various classes. They were mildly amused though not enlightened so I gave up on the dataset there and then.



(full-sized pdf versions can be found at the end of the document)

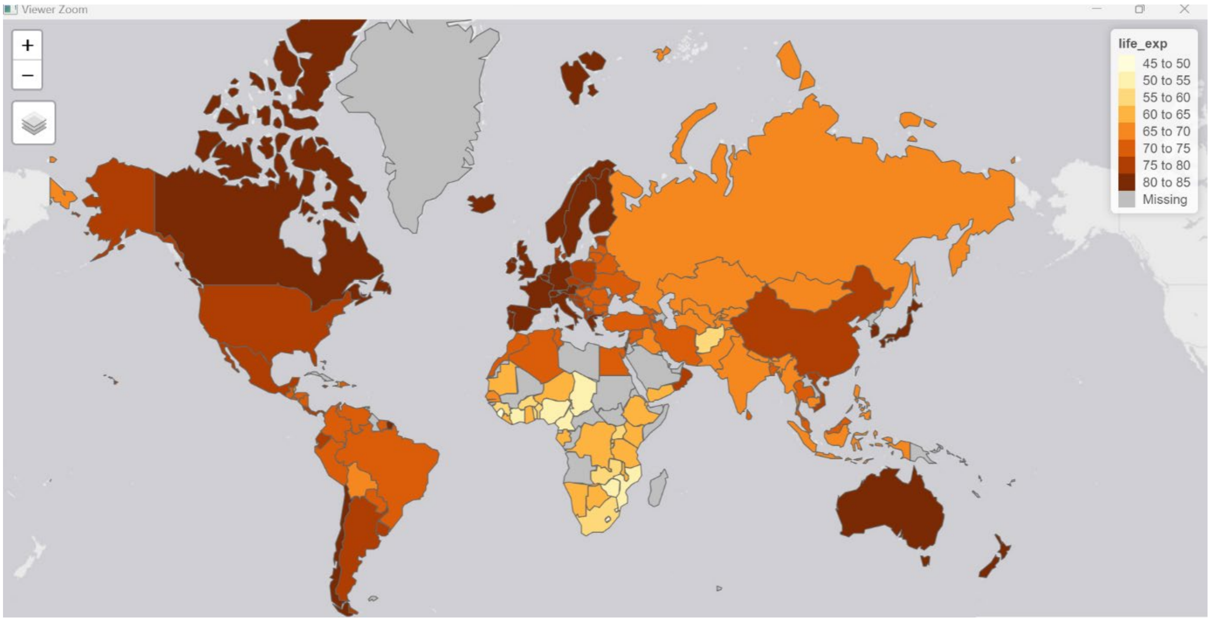
World Maps:

I eventually settled on a 'World' database which contained coordinates so I could plot on a world map. I wasted far too much time playing with all the various functions available, and I hope to develop this interest more when I have free time (somewhere around 2025 by my reckoning).

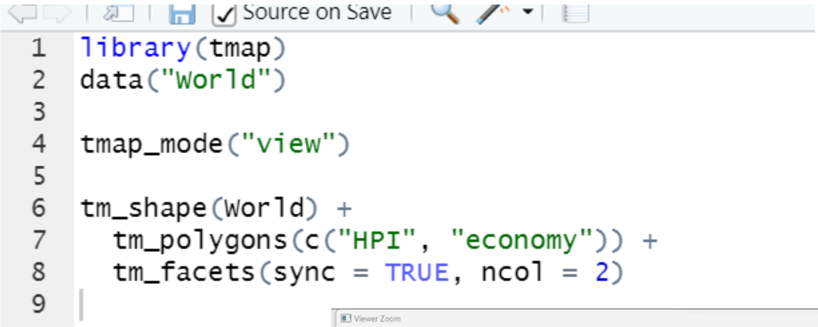


```r
library(tmap)
data("World")

tmap_mode("view")

  tm_shape(World)+
  tm_polygons("life_exp")
```

With the inbuilt view mode and ‘polygon’ function one can instantly create informative and interactive maps. I created a map showing life expectancy around the world which made for some sober viewing.

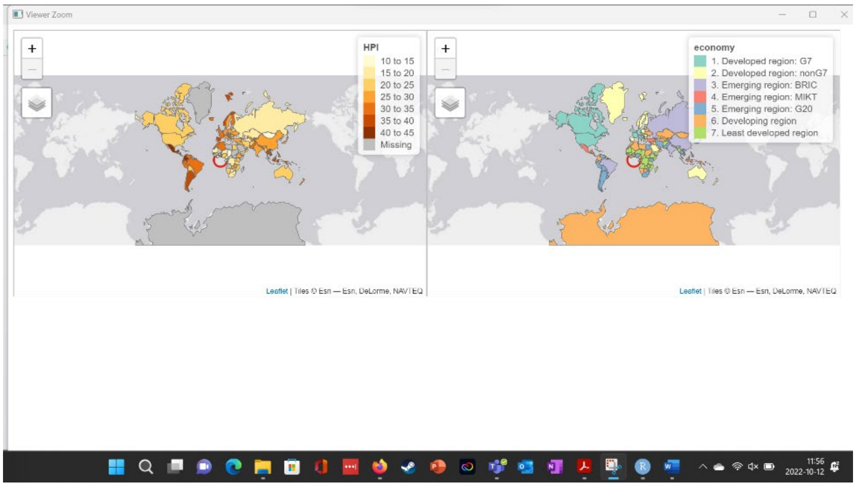


There were many fantastic functions, and the ones that stood out were the interactive ones, mainly as they were so easy to produce and are more pedagogic and memorable by their nature.



```r
library(tmap)
data("World")

tmap_mode("view")

tm_shape(World) +
  tm_polygons(c("HPI", "economy")) +
  tm_facets(sync = TRUE, ncol = 2)
```

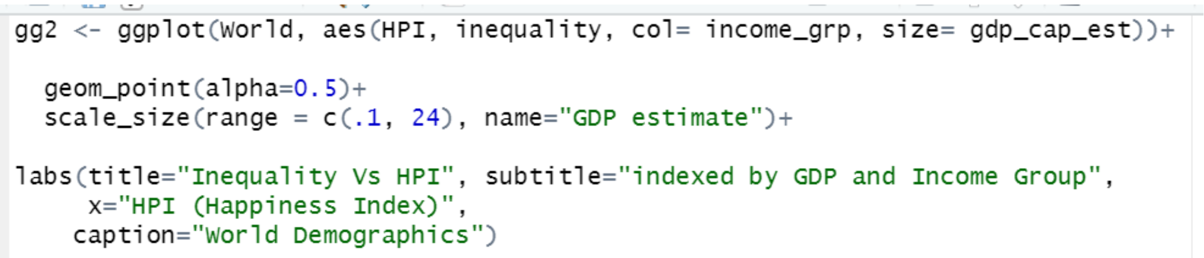
By using the ‘tm_facets’ function one can create maps that sit side by side and by ‘syncing’ them it’s possible to access more information on each scan.



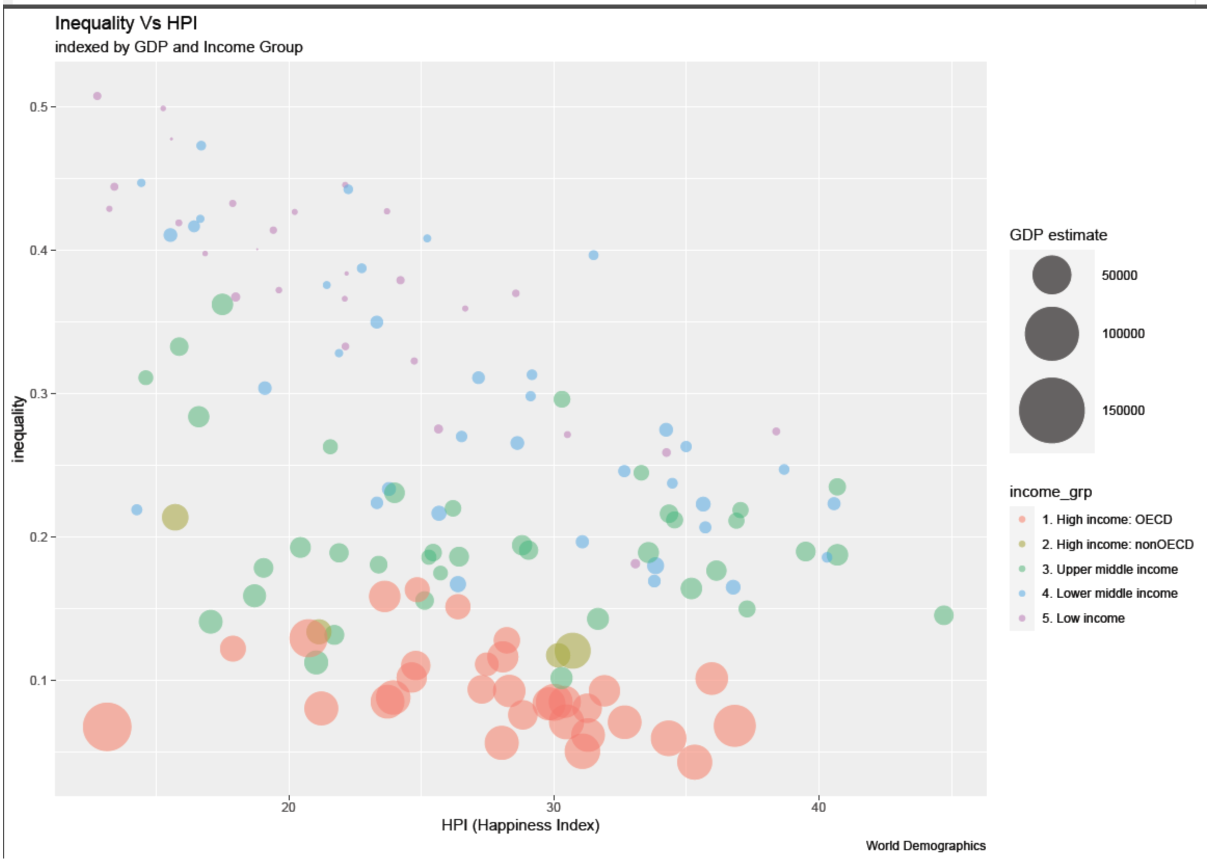
Scatterplot:

I wanted to create a scatterplot next in order to compare as many different factors as possible. In the limited time that I've been studying graphing with R, this has been the most powerful visual use of data that I've come across.

I chose to compare levels of inequality in countries to the HPI (happiness index) to see if racial or sexual discrimination was clearly reflected in the 'happiness' of each country's citizens. I added a legend for the income groups represented and the GDP of the countries as well.



```
gg2 <- ggplot(World, aes(HPI, inequality, col= income_grp, size= gdp_cap_est))+

  geom_point(alpha=0.5)+
  scale_size(range = c(.1, 24), name="GDP estimate")+

labs(title="Inequality Vs HPI", subtitle="indexed by GDP and Income Group",
     x="HPI (Happiness Index)",
    caption="World Demographics")
```



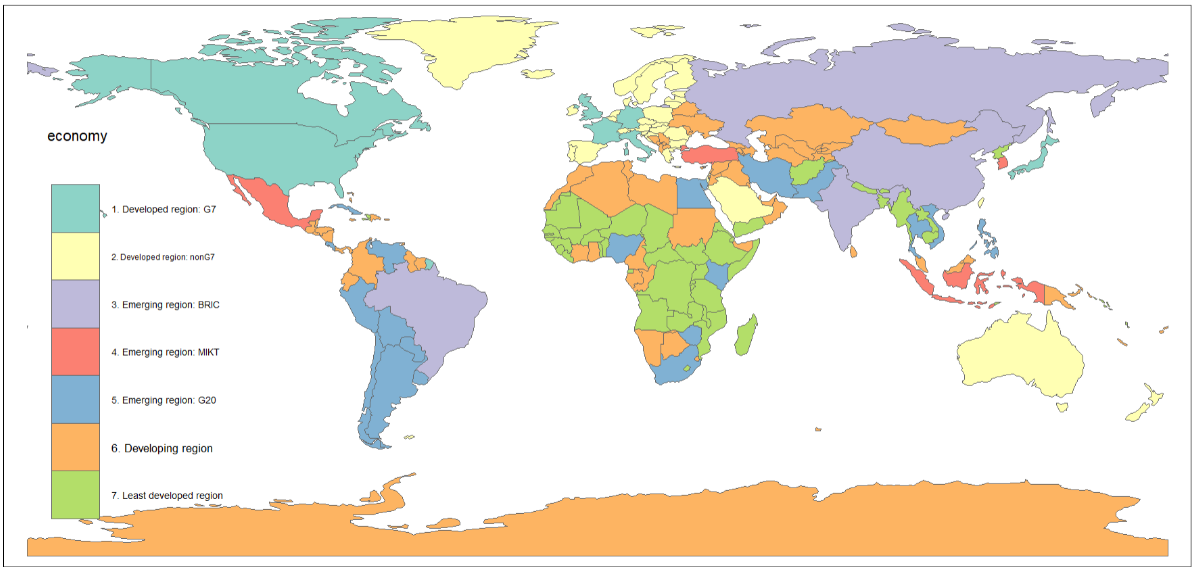
I really loved this graph. It clearly shows how without a stable economy and a relatively high standard of living it is hard for any country to achieve viable and visible equality (disregarding for now any cultural factors).

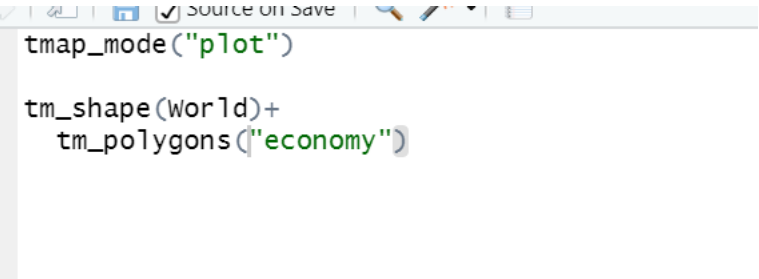
It is equally interested to note how spread out the HP index was among high earning countries. It raises many questions (for instance who is that country in the bottom left with lots of money, little inequality and with a miniscule amount of happiness to show for it. Tragic!

And the number of middle to high income countries that report very high indexes lends more credence to the cliché that 'money can't buy you everything'.

Boxplot:

Lastly, I created a boxplot to show the happiness of the different economic groups in the world. The world bank decides which group a country belongs to. Here is a map:

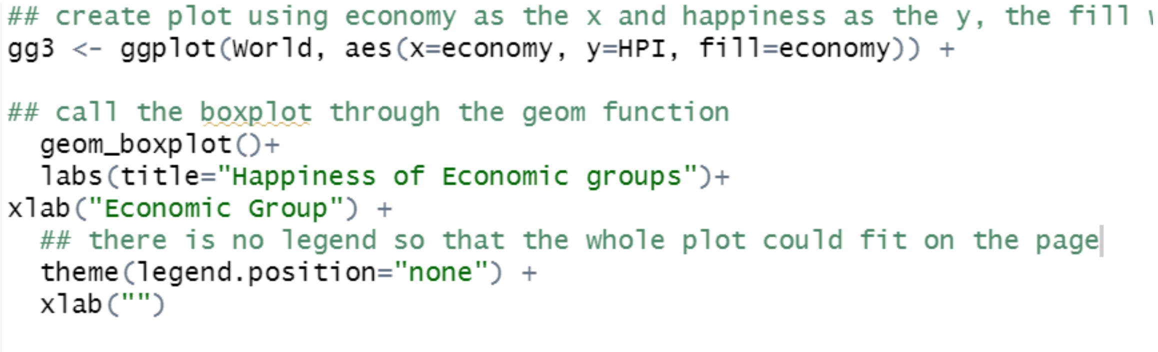




```
tmap_mode("plot")

tm_shape(World)+
  tm_polygons("economy")
```

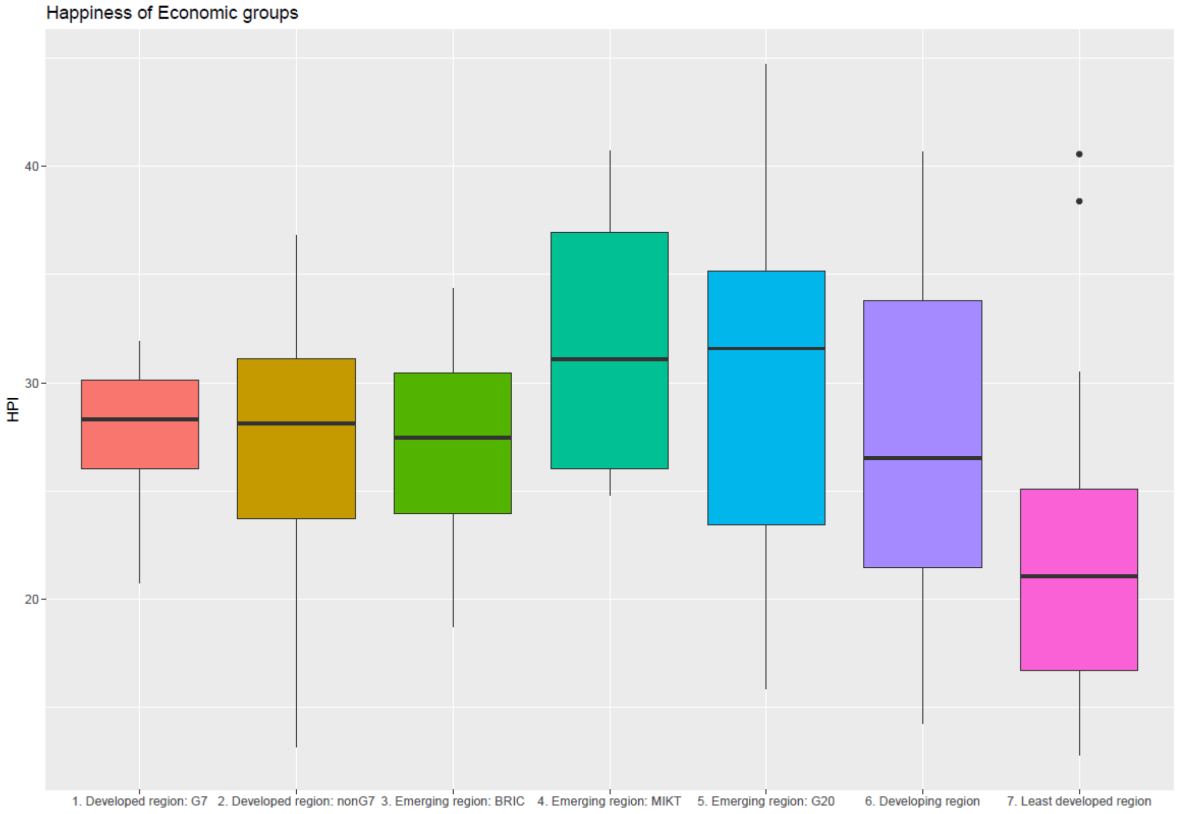
Again I used the tmap package to plot the map.

I created a boxplot with a focus on the economic groups, I wanted it to be as simple as possible.

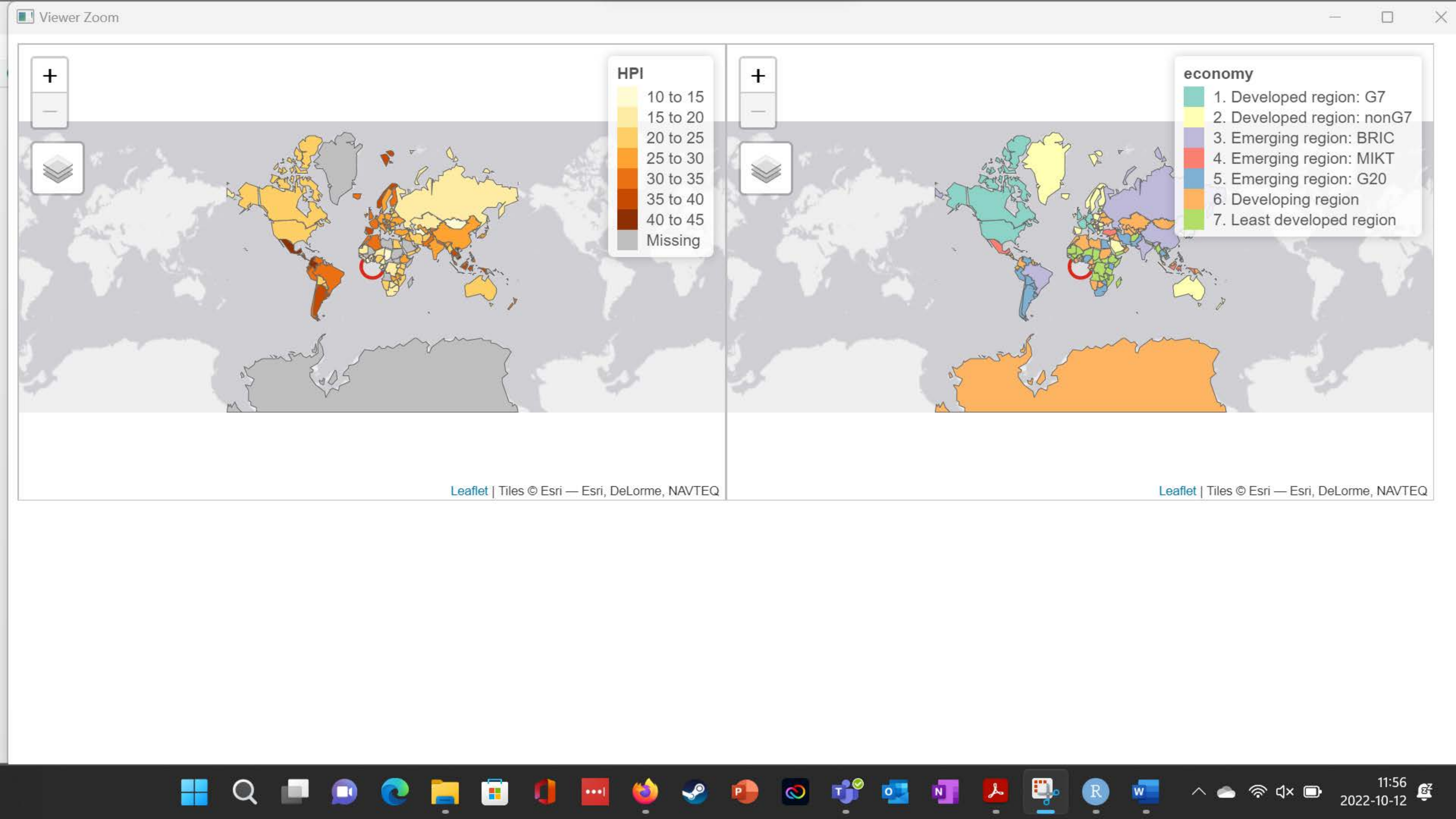


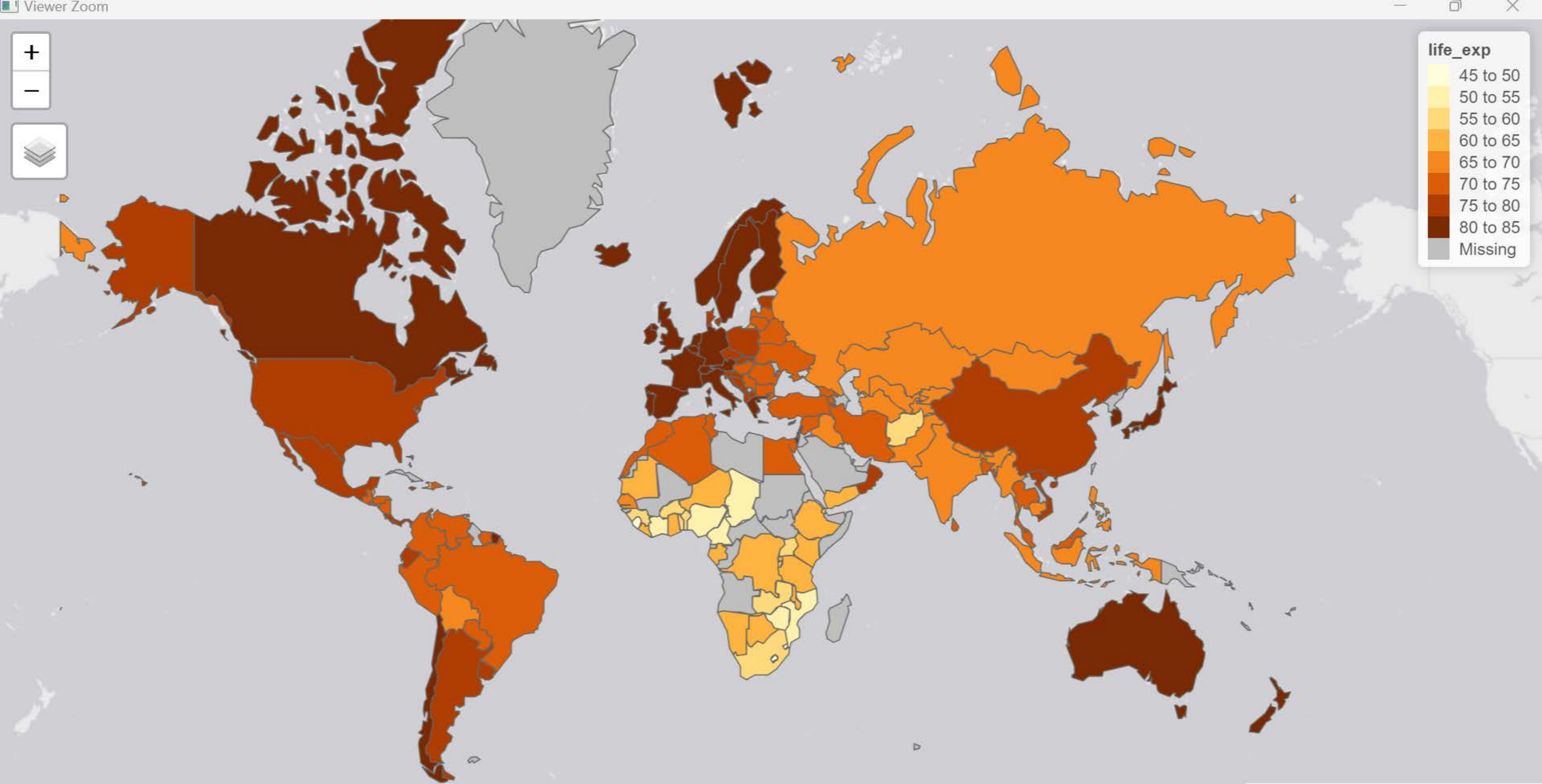
```
## create plot using economy as the x and happiness as the y, the fill
gg3 <- ggplot(World, aes(x=economy, y=HPI, fill=economy)) +

## call the boxplot through the geom function
  geom_boxplot()+
  labs(title="Happiness of Economic groups")+
xlab("Economic Group") +
  ## there is no legend so that the whole plot could fit on the page
  theme(legend.position="none") +
  xlab("")
```

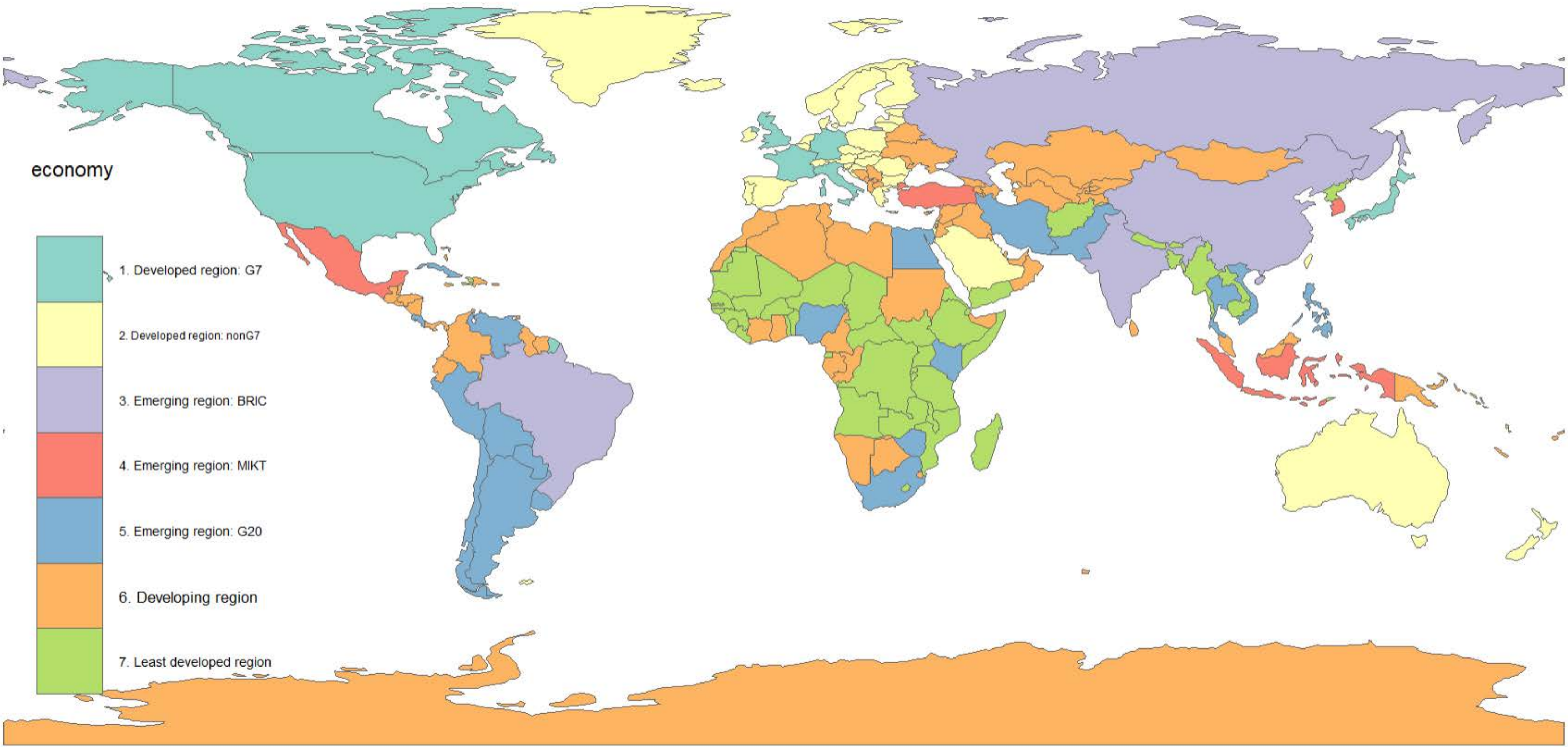
Here is how the boxplot of happiness in these regions came out:



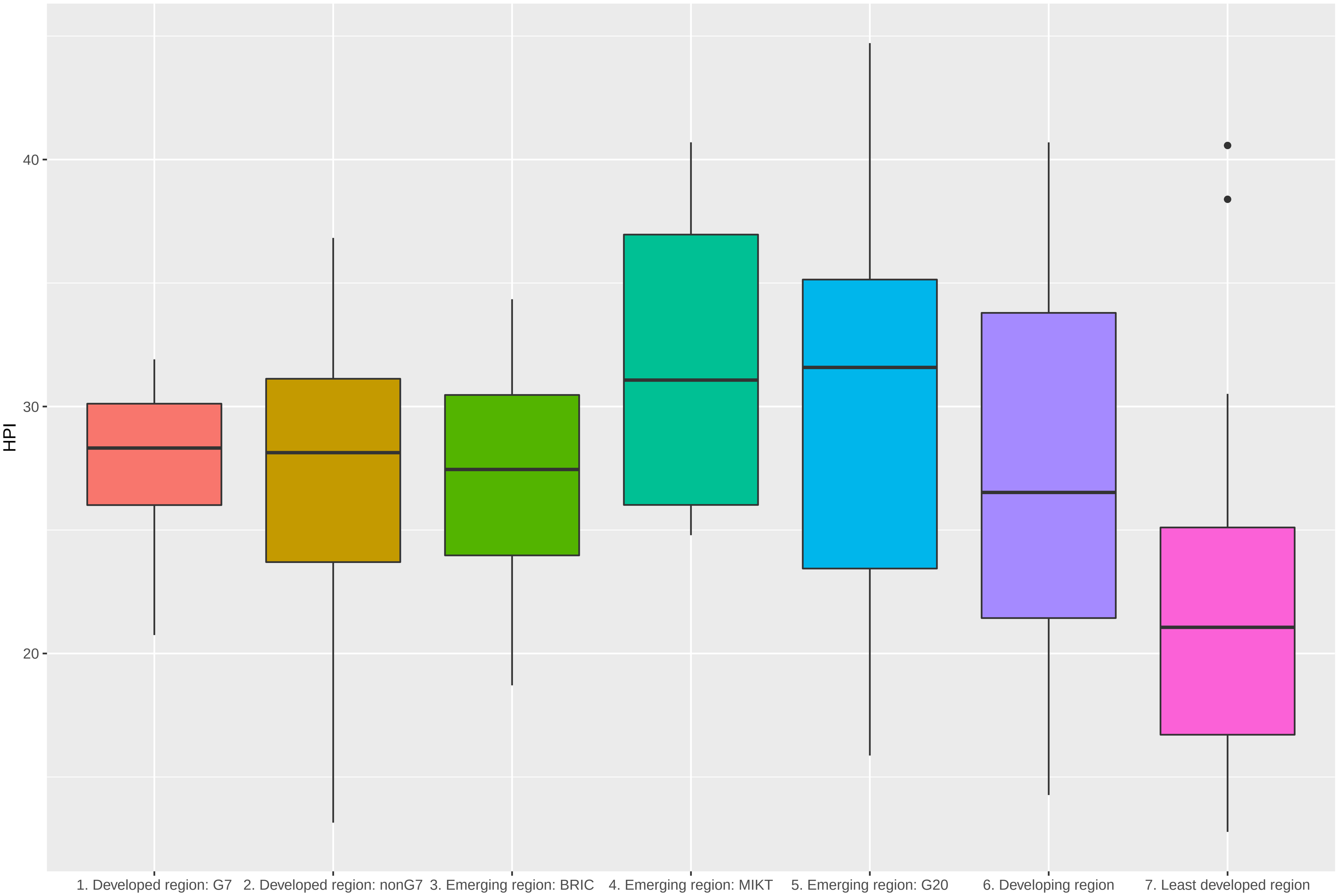
Not surprising that those with little money have little to be happy about, though it is surprising to see how middling and uninspiring the happiness measurement of the G7 countries was. There is something optimistic and heartening about seeing how high happiness is among emerging countries. If there is anything to conclude from this chart it is that how we measure happiness is fraught with difficulty, contradiction, and cultural relevance.

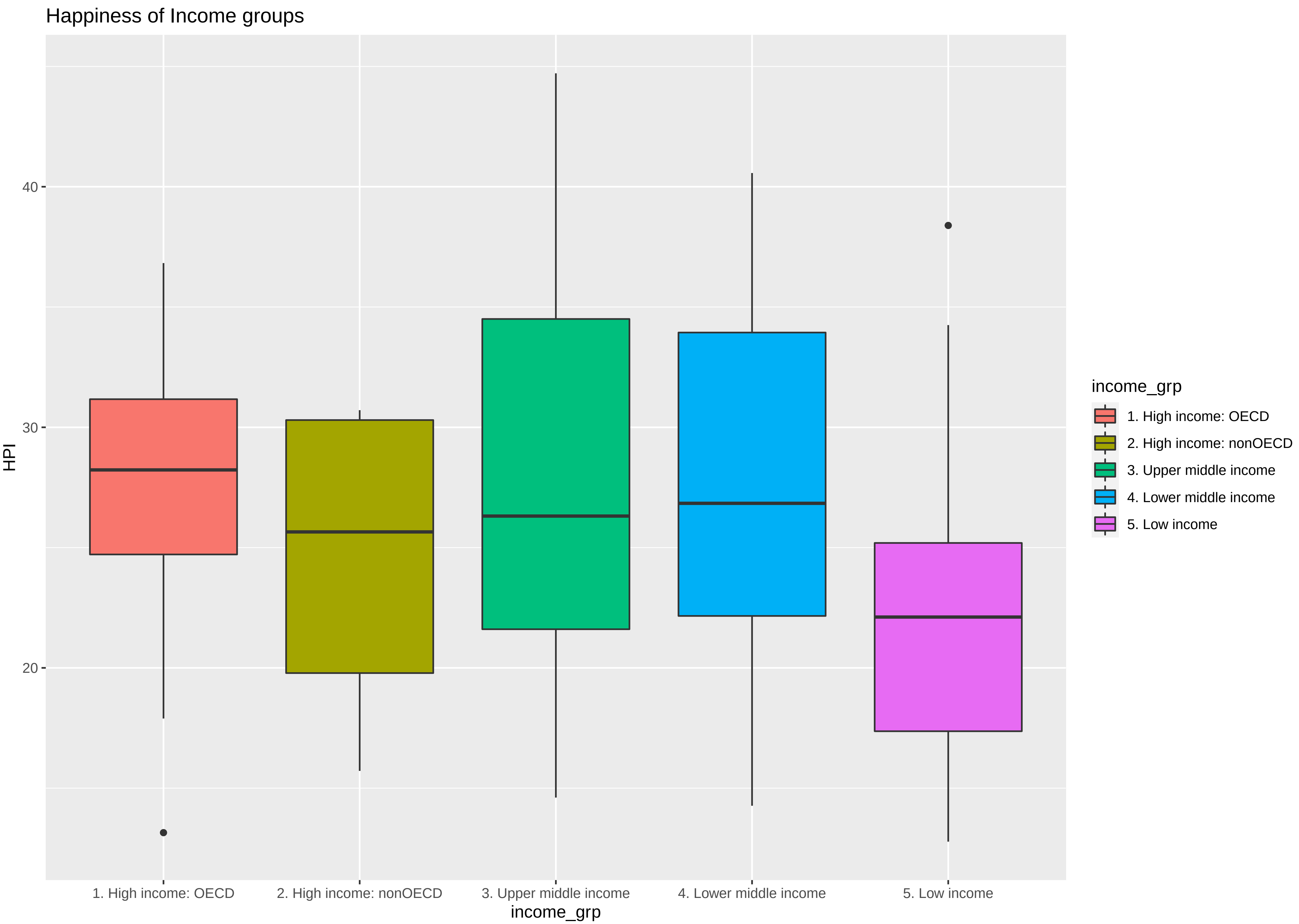

Viewer Zoom
HPI
10 to 15
15 to 20
20 to 25
25 to 30
30 to 35
35 to 40
40 to 45
Missing
Leaflet | Tiles © Esri — Esri, DeLorme, NAVTEQ
economy
1. Developed region: G7
2. Developed region: nonG7
3. Emerging region: BRIC
4. Emerging region: MIKT
5. Emerging region: G20
6. Developing region
7. Least developed region
Leaflet | Tiles © Esri — Esri, DeLorme, NAVTEQ
11:56
2022-10-12


Viewer Zoom
+
–
life_exp
45 to 50
50 to 55
55 to 60
60 to 65
65 to 70
70 to 75
75 to 80
80 to 85
Missing


economy
1. Developed region: G7
2. Developed region: nonG7
3. Emerging region: BRIC
4. Emerging region: MIKT
5. Emerging region: G20
6. Developing region
7. Least developed region

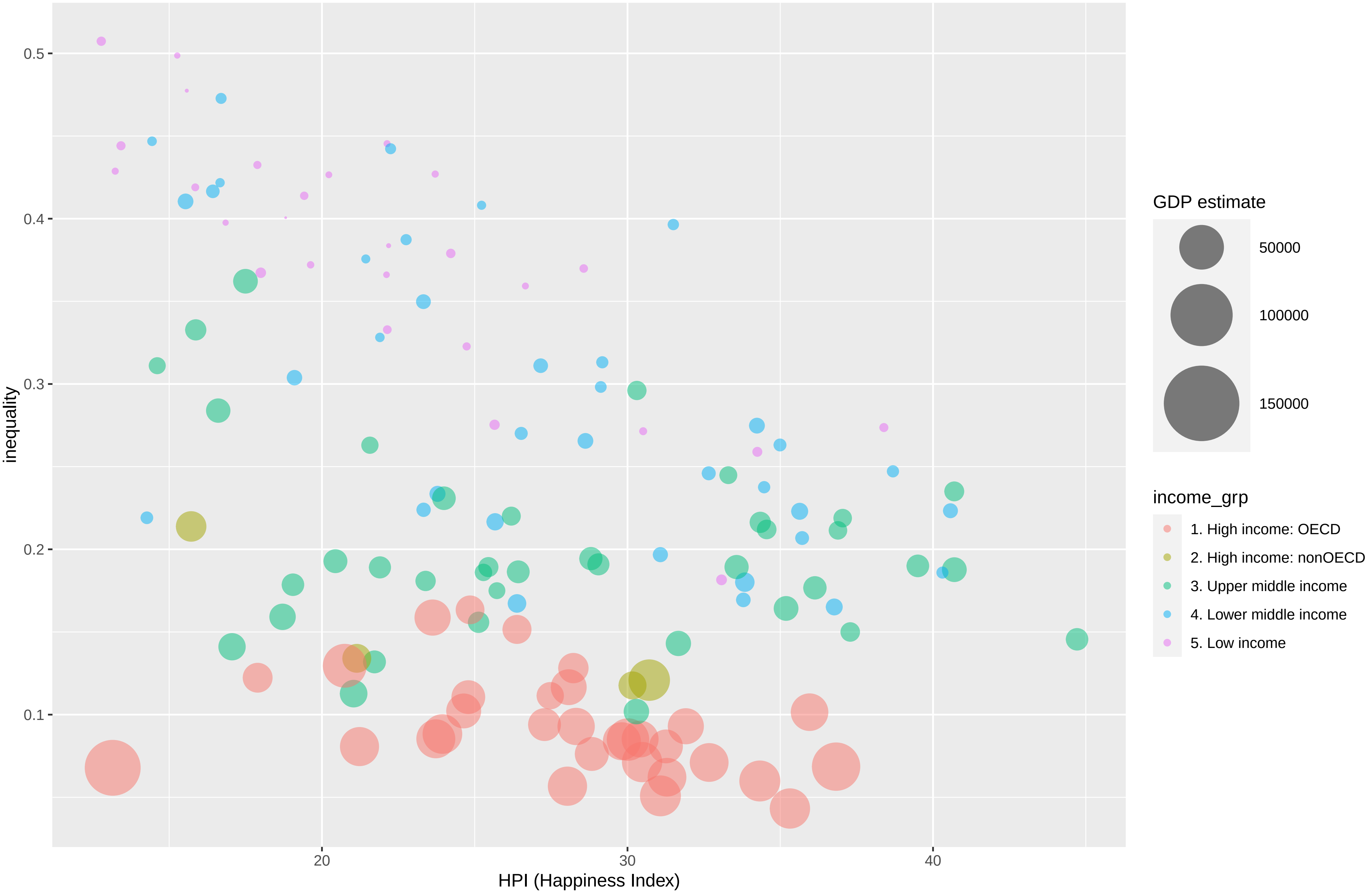
Happiness of Economic groups




Happiness of Income groups
HPI
40
30
20
1. High income: OECD
2. High income: nonOECD
3. Upper middle income
4. Lower middle income
5. Low income
income_grp
income_grp
1. High income: OECD
2. High income: nonOECD
3. Upper middle income
4. Lower middle income
5. Low income

Inequality Vs HPI

indexed by GDP and Income Group

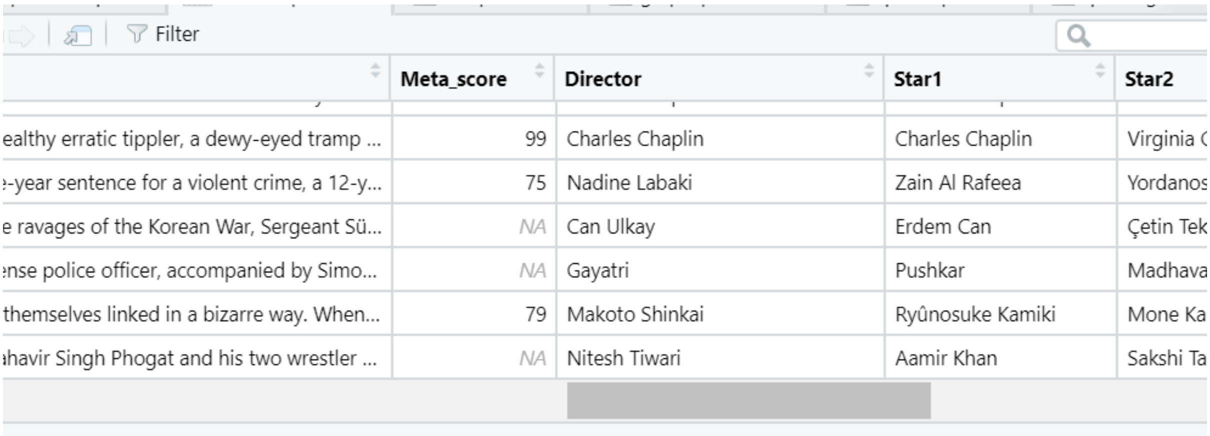


World Demographics

4. Data Visualization – VG question:

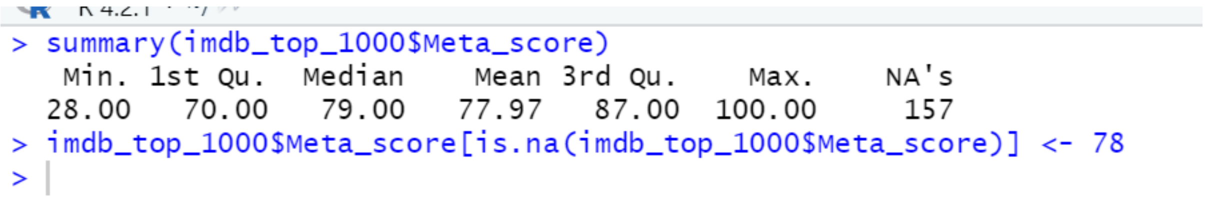
I spent so long trying to decide upon a dataset and the graphs to go with it for question 3 that I had less time than I would have liked for question 4. After university I went to film school and have written screenplays and advertisements on and off since then. It seemed natural for me to search for a dataset within the field. I went with the top 1000 films on imdb, mainly because the length was sufficient for the brief, though I have not exactly been wowed by the data on offer. Given my self-enforced time restrictions I have had to make do.

The dataset was not 'plot' ready, I found a number of rows with 'NA's'.



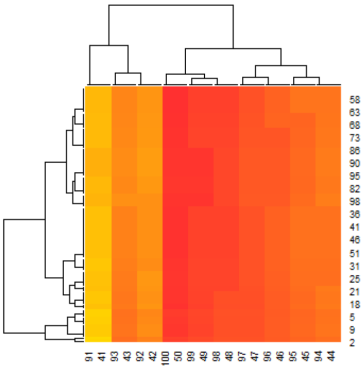
	Meta_score	Director	Star1	Star2
ealthy erratic tippler, a dewy-eyed tramp ...	99	Charles Chaplin	Charles Chaplin	Virginia C
-year sentence for a violent crime, a 12-y...	75	Nadine Labaki	Zain Al Rafeea	Yordanos
e ravages of the Korean War, Sergeant Sü...	NA	Can Ulkay	Erdem Can	Çetin Tek
ense police officer, accompanied by Simo...	NA	Gayatri	Pushkar	Madhava
themselves linked in a bizarre way. When...	79	Makoto Shinkai	Ryûnosuke Kamiki	Mone Ka
ahavir Singh Phogat and his two wrestler ...	NA	Nitesh Tiwari	Aamir Khan	Sakshi Ta

I used the same approach as in question 2, I got a mean for the row and replaced the missing values.



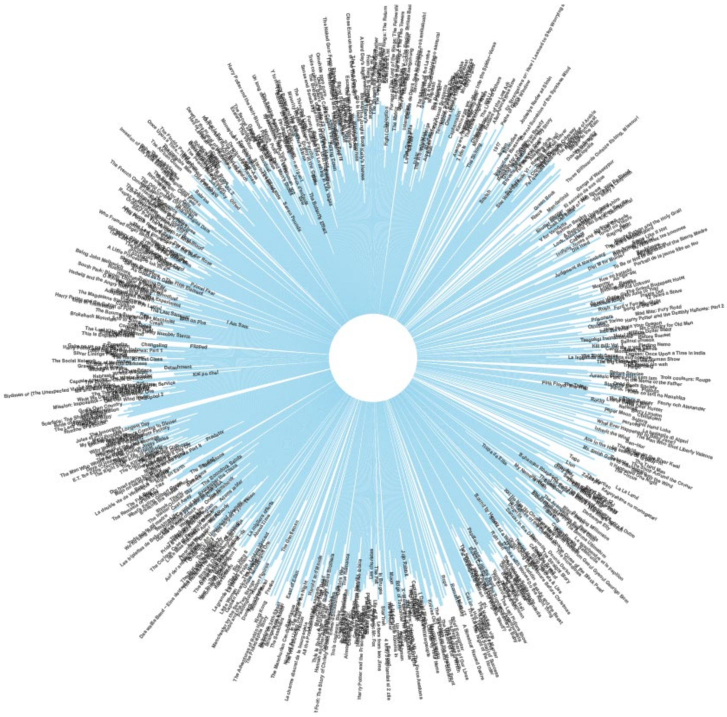
```
> summary(imdb_top_1000$Meta_score)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  28.00   70.00   79.00   77.97   87.00  100.00     157 
> imdb_top_1000$Meta_score[is.na(imdb_top_1000$Meta_score)] <- 78
> 
```

With a thousand rows of data I considered it wise to start with a landscape view, in order to get an idea of what I was dealing with. Here are a couple of valiant but failed attempts to discover insights from a mess of mass:



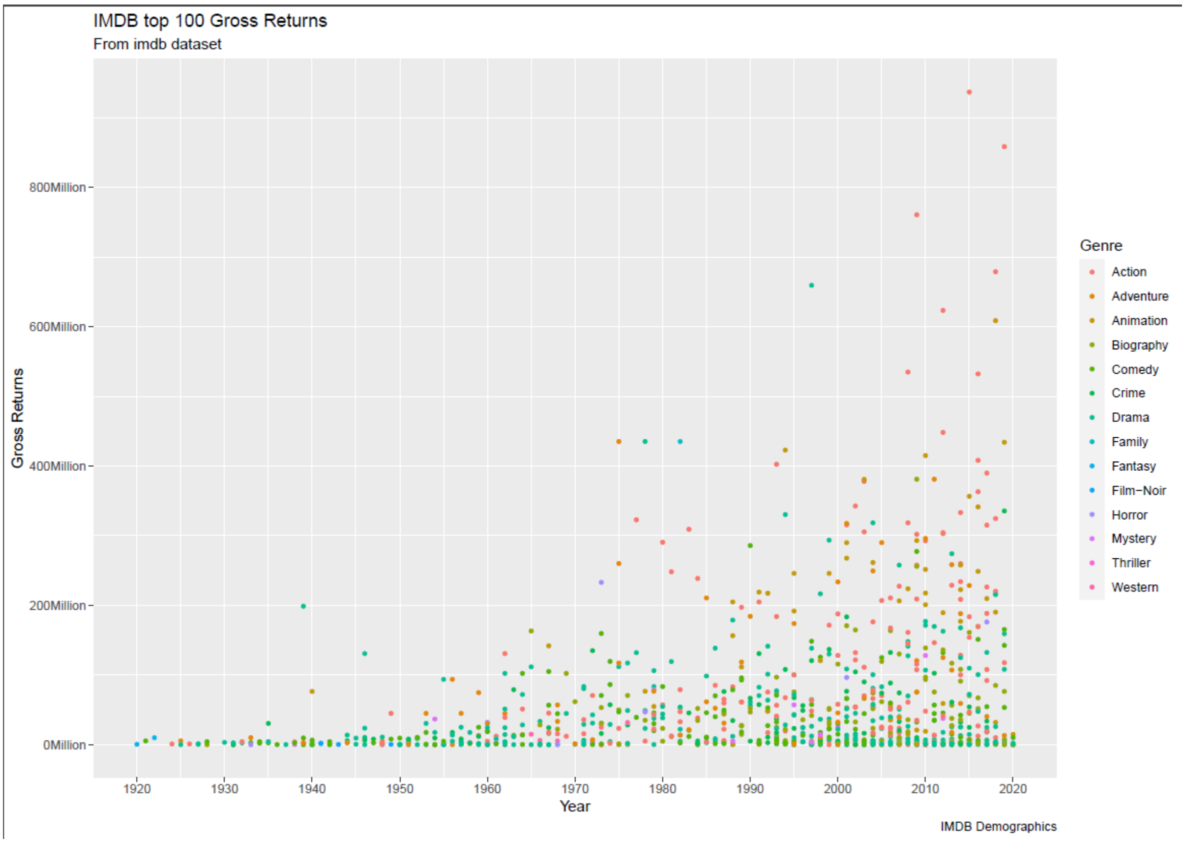
I started with a heatmap, but even with a thousand elements in the dataset it doesn't equate to much range when you consider the tens of thousands of rated films in the database.

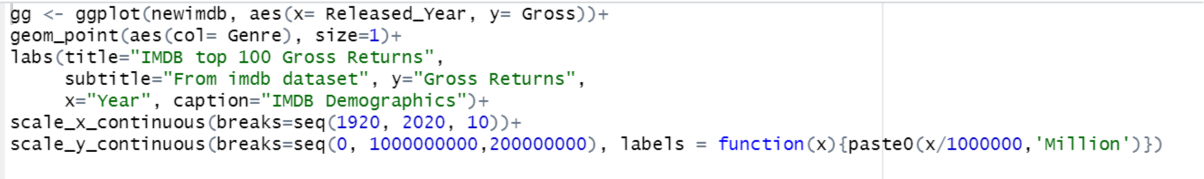
Attempted a circular plot that came out eye strainingly mental:



I dare you to search for movie titles. Unfortunately, in the confusion of trying to figure out what I had created and the need to create something that made sense I deleted the code that formed this aberration.

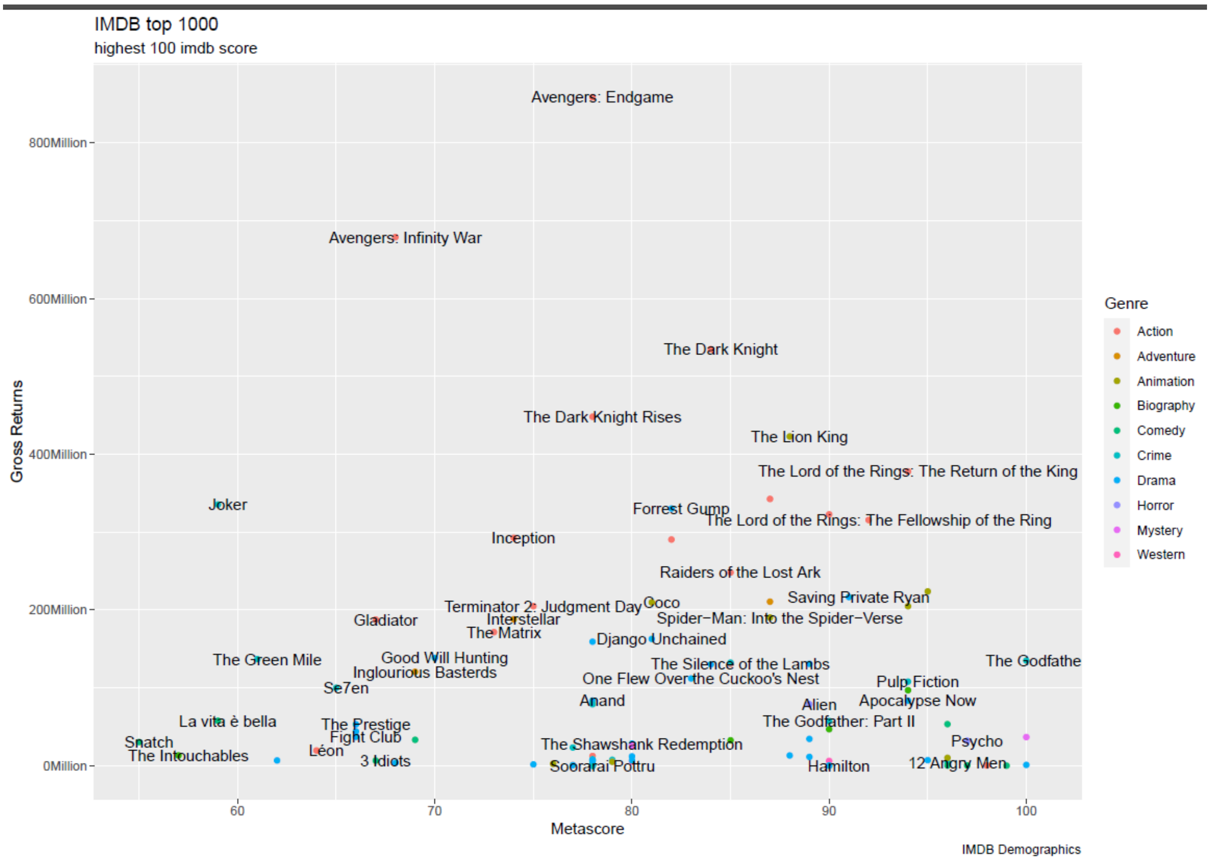
So I turned to the tried and tested, and created a scatterplot when compared ‘Gross profit’ to ‘Release Year’ and using genre as the key. The idea of using actual film names for this graph was a non-starter. The problem with the result is that the profits of older films have not been adjusted according to inflation, so the results are skewed heavily in favour of films released in the last twenty years or so.



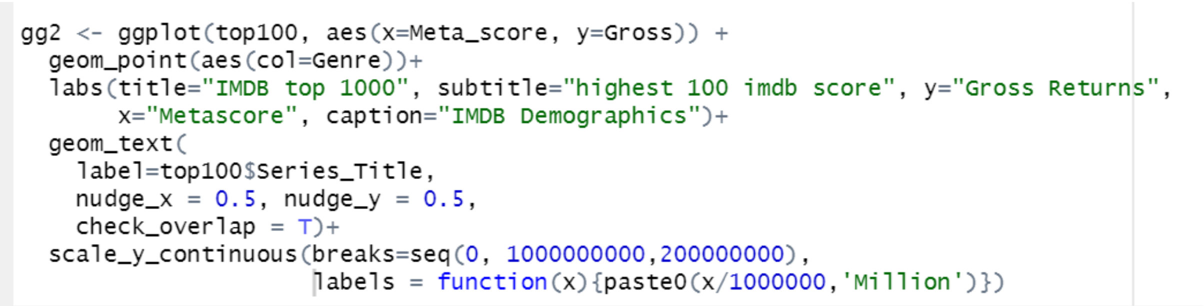


```
gg <- ggplot(newimdb, aes(x= Released_Year, y= Gross))+
geom_point(aes(col= Genre), size=1)+
labs(title="IMDB top 100 Gross Returns",
     subtitle="From imdb dataset", y="Gross Returns",
     x="Year", caption="IMDB Demographics")+
scale_x_continuous(breaks=seq(1920, 2020, 10))+
scale_y_continuous(breaks=seq(0, 1000000000,200000000), labels = function(x){paste0(x/1000000,'Million')})
```

I did take the top hundred grossing films and plotted their gross profit to their ‘Metascores’ (a much more accurate measure of their critical worth than the ‘IMDB rating’, in my opinion). In this graph I could include the names of the films and again used ‘genre’ as a legend. I see it as the most natural and significant way to divide large groups of movies.

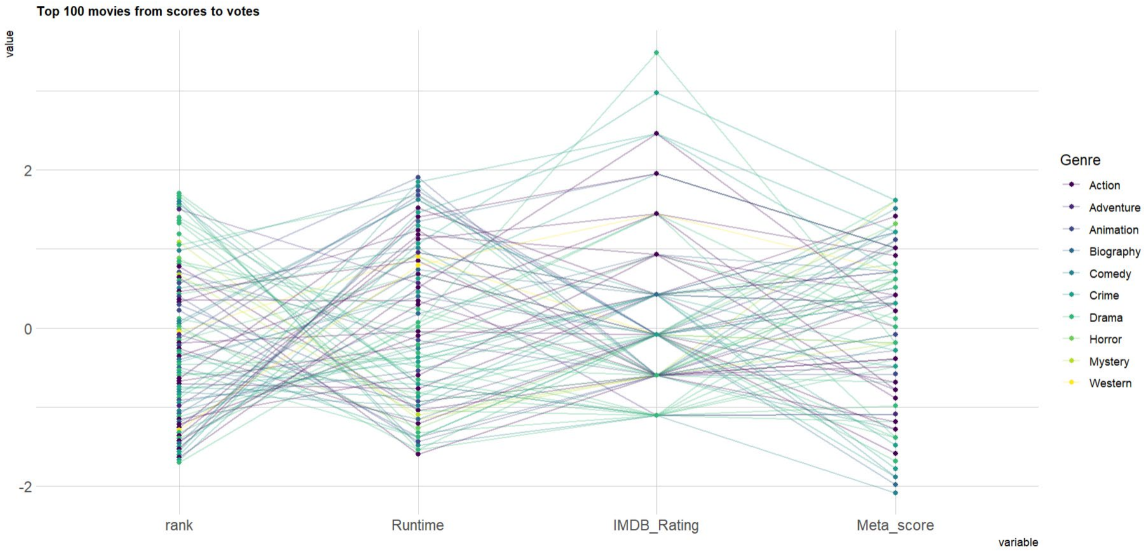


The graph clearly shows the dominance of action movies at the top of the box office, and within the genre how indomitable the comic book has been over the past decades.

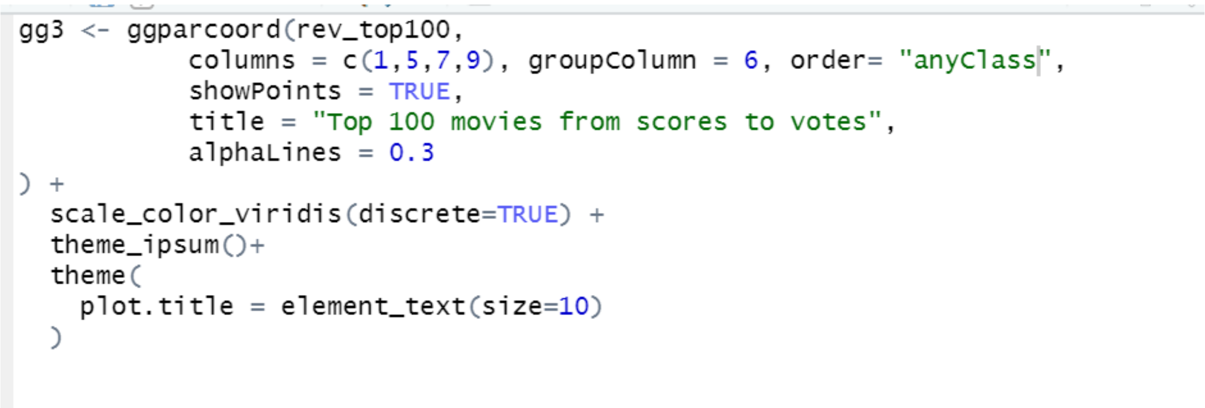


```
gg2 <- ggplot(top100, aes(x=Meta_score, y=Gross)) +
  geom_point(aes(col=Genre))+
  labs(title="IMDB top 1000", subtitle="highest 100 imdb score", y="Gross Returns",
       x="Metascore", caption="IMDB Demographics")+
  geom_text(
    label=top100$Series_Title,
    nudge_x = 0.5, nudge_y = 0.5,
    check_overlap = T)+
  scale_y_continuous(breaks=seq(0, 1000000000,200000000),
                     labels = function(x){paste0(x/1000000,'Million')})
```

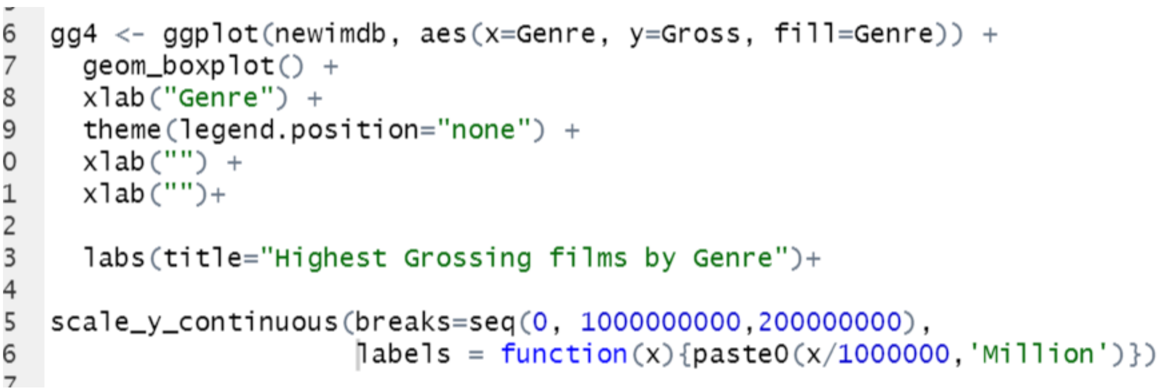
I wanted to find a way to show how people voting compared to critic’s votes, which is the essential difference with the IMDB and metascore rating system. This was the graph that I tried out to show the difference. I also included a row for film length to see if that reflected any opinions on the films themselves. Genre again indexes, but the results are far from clear. What I think can be seen quite clearly is how critics tend to judge films more harshly than the average cinema goer.

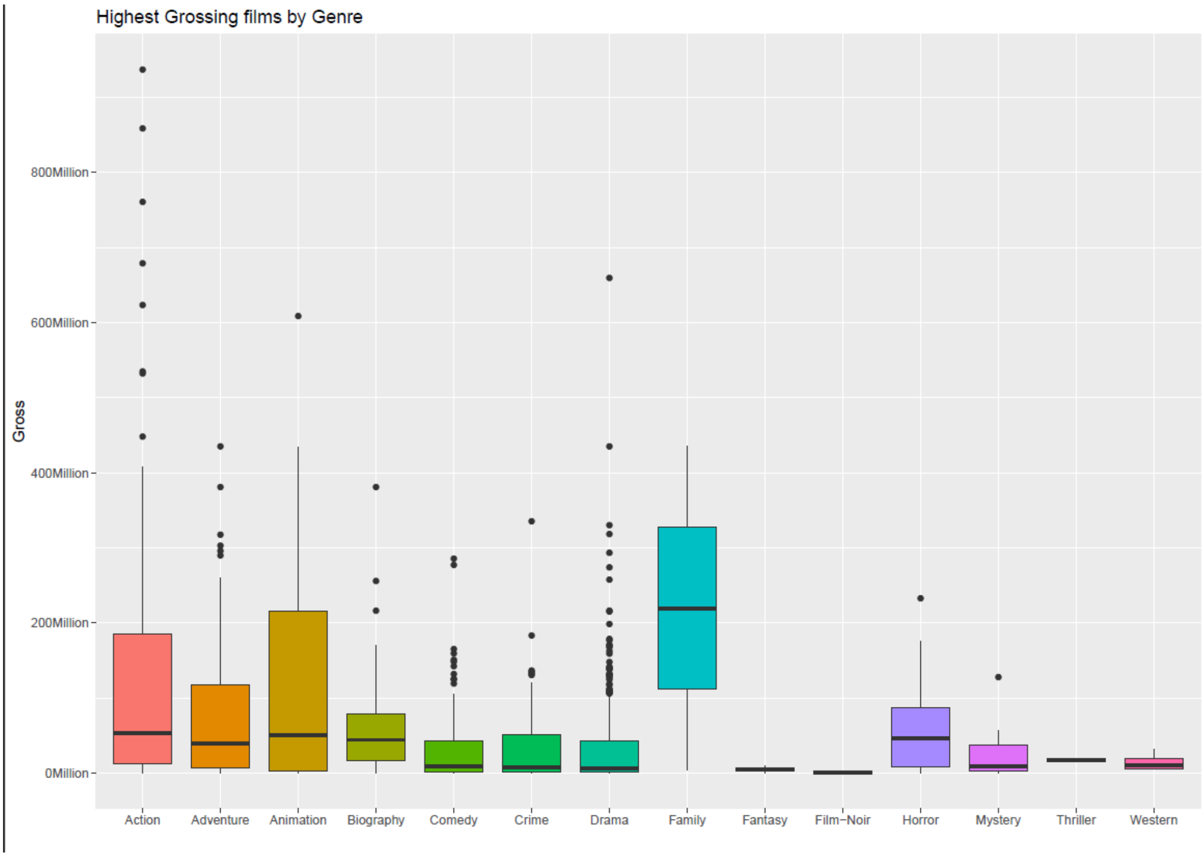


The third row is the public voting, the last row the critic’s.


```
gg3 <- ggparcoord(rev_top100,
           columns = c(1,5,7,9), groupColumn = 6, order= "anyClass",
           showPoints = TRUE,
           title = "Top 100 movies from scores to votes",
           alphaLines = 0.3
) +
  scale_color_viridis(discrete=TRUE) +
  theme_ipsum()+
  theme(
    plot.title = element_text(size=10)
  )
```

‘I will end my graphing of the thousand highest rated movies on IMDB with a boxplot, which shows much more clearly than the scatterplot how successful different genres have been over the decades. What stands out here is that although actions movies have outliers that are more successful on orders of magnitude from most other films, it is the family film which is the safest bet from an investment perspective, as they’re most likely to break even or make a profit.

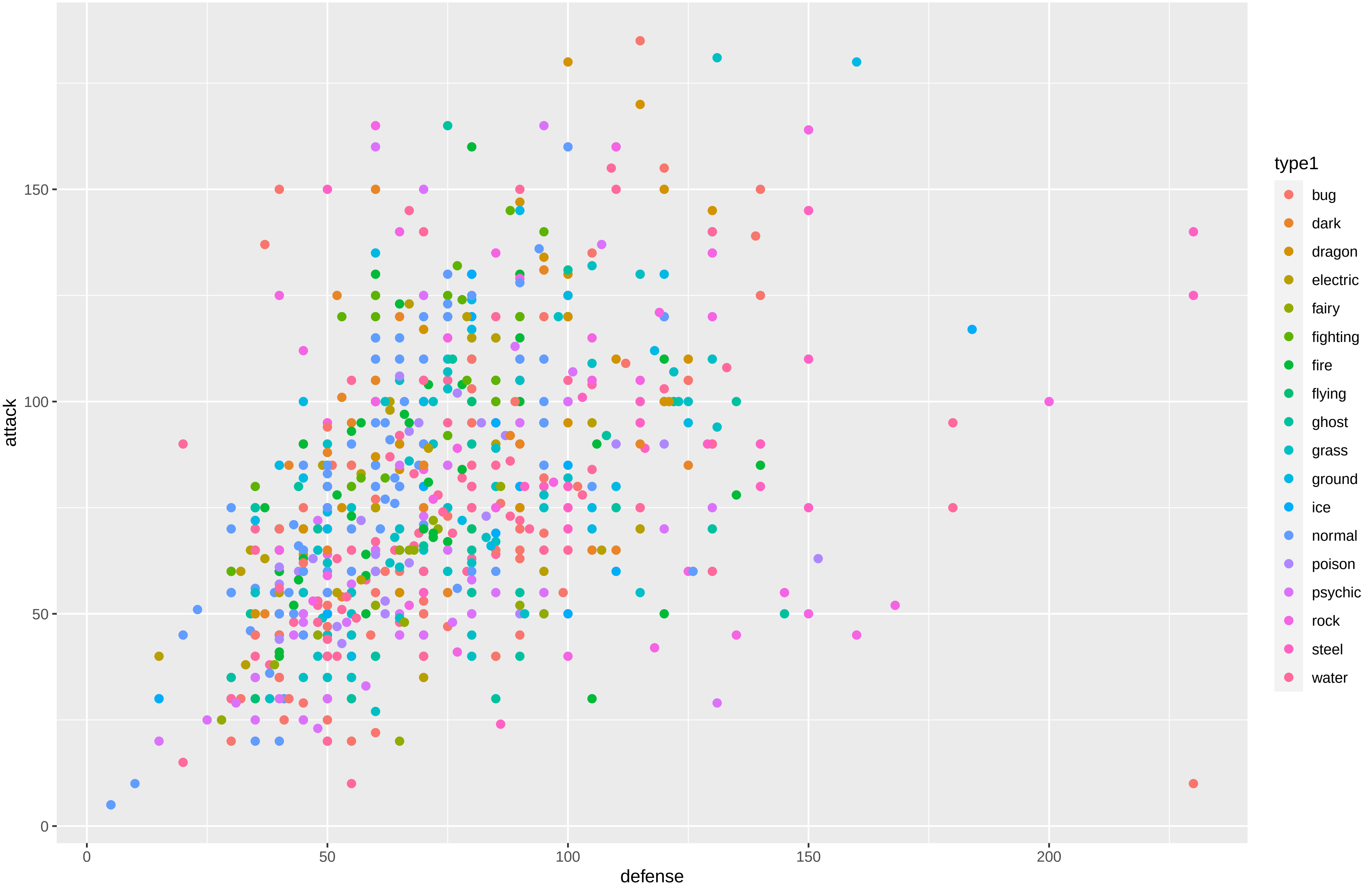

```
gg4 <- ggplot(newimdb, aes(x=Genre, y=Gross, fill=Genre)) +
  geom_boxplot() +
  xlab("Genre") +
  theme(legend.position="none") +
  xlab("") +
  xlab("")+

  labs(title="Highest Grossing films by Genre")+

scale_y_continuous(breaks=seq(0, 1000000000,200000000),
                   labels = function(x){paste0(x/1000000,'Million')})
```



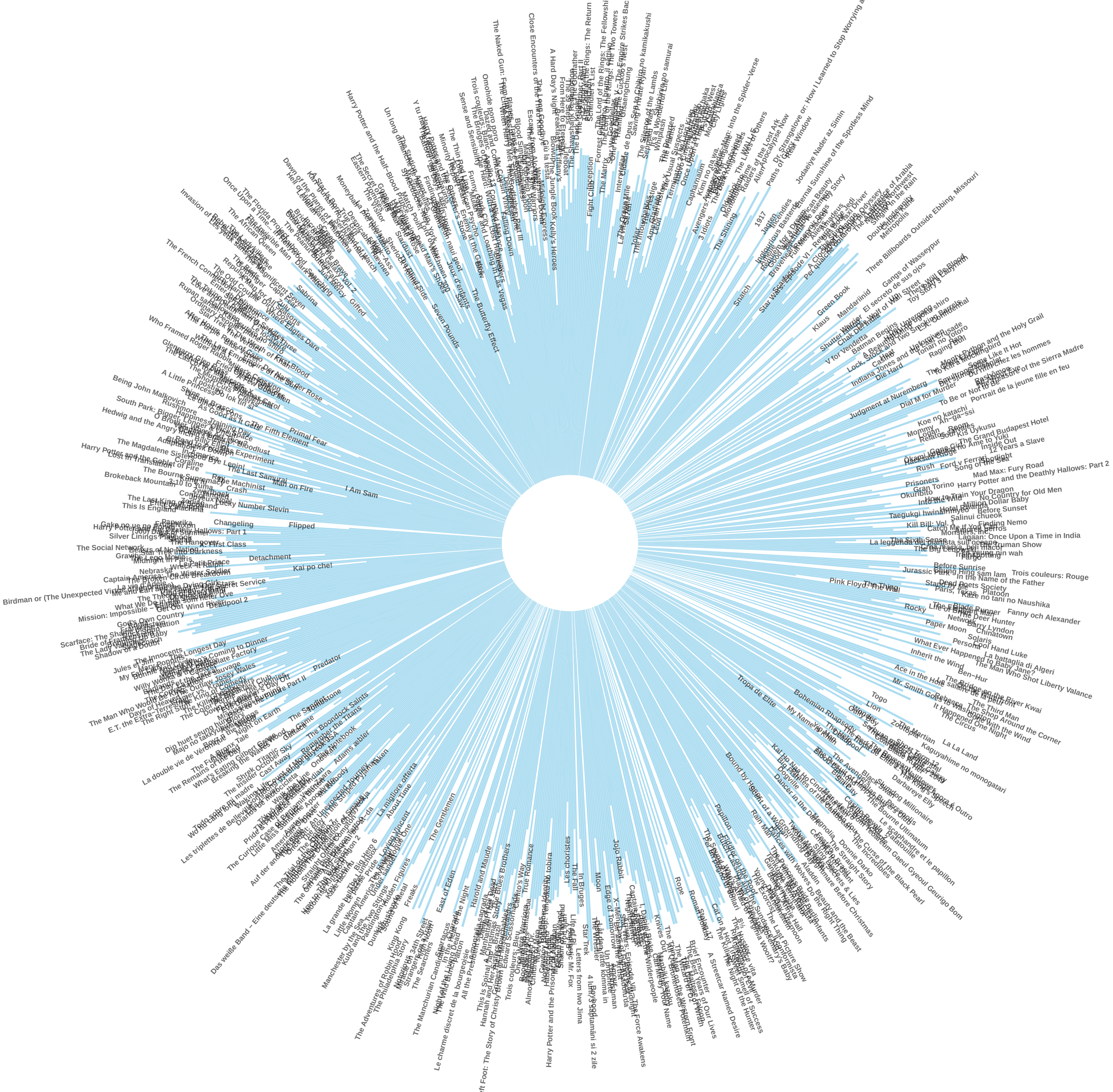
I hope this has been an enlightening look into how successful different genres of film have been throughout the world. It was incredibly interesting trying out different techniques and plotting styles. I just wish I'd had more time and was able to experiment more. This was a fun start though.

Pokemon

Attack vs Defence

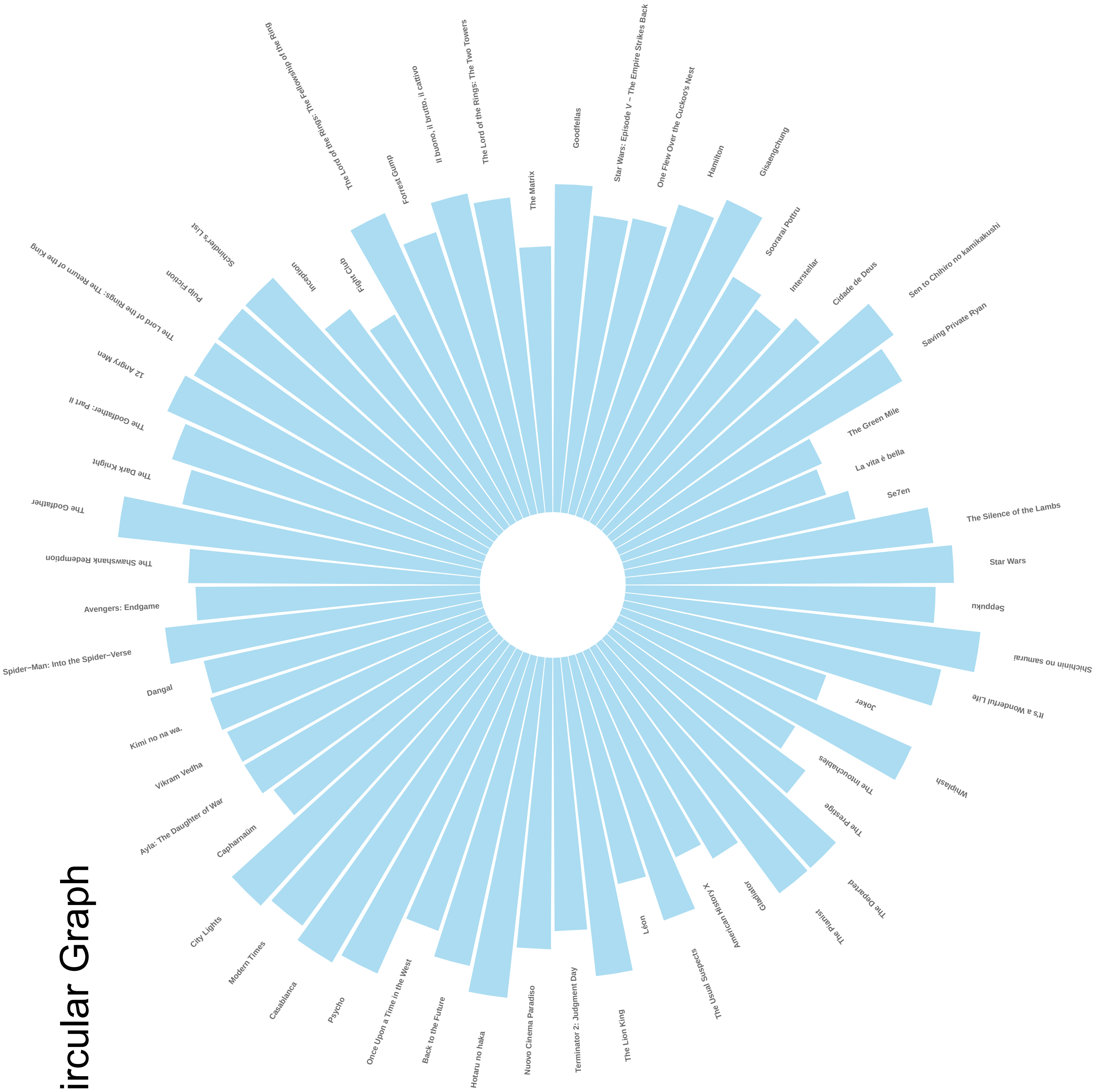


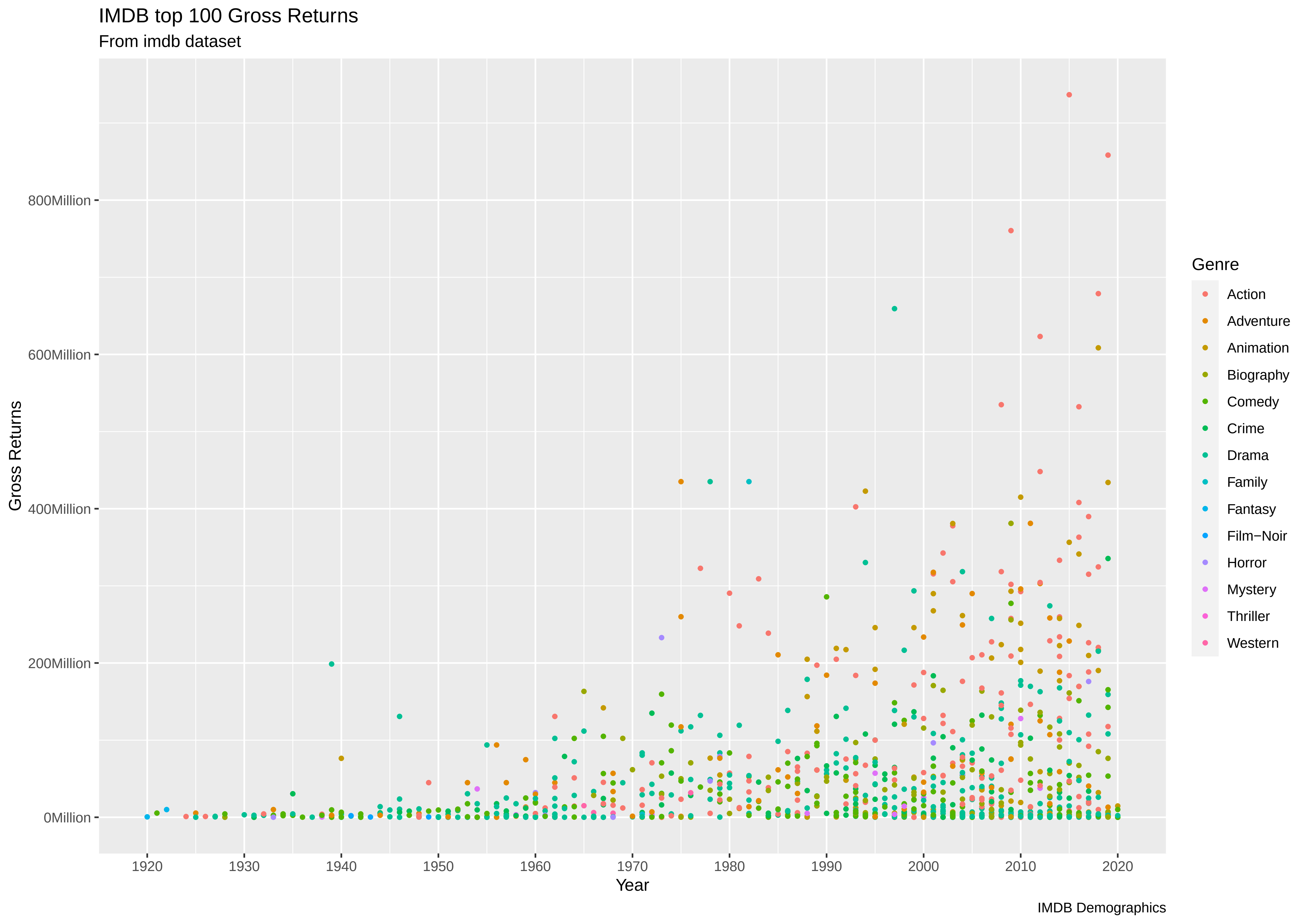
Pokemon Demographics



The Naked Gun: From the Files of Police Squad!
Close Encounters of the Third Kind
A Hard Day's Night
Harry Potter and the Half-Blood Prince
Sense and Sensibility
Fight Club
Interstellar
Inception
Forrest Gump
The Matrix
The Empire Strikes Back
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb
Eternal Sunshine of the Spotless Mind
Jodaeiye Nader az Simin
Three Billboards Outside Ebbing, Missouri
Paths of Glory
Rear Window
Raiders of the Lost Ark
Alien
The Lives of Others
Spider-Man: Into the Spider-Verse
Avengers: Infinity War
Capharnaüm
Terminator 2: Judgment Day
The Usual Suspects
The Prestige
The Pianist
The Shining
3 Idiots
Once Upon a Time in the West
Toy Story
Lawrence of Arabia
Metropolis
Double Indemnity
Snatch
Star Wars: Episode VI - Return of the Jedi
Gangs of Wasseypur
El secreto de sus ojos
Toy Story 3
Green Book
Klaus
Mandariinid
Shutter Island
V for Vendetta
Chak De! India
Batman Begins
Lock, Stock and Two Smoking Barrels
Casino
Die Hard
Indiana Jones and the Last Crusade
Raging Bull
Unforgiven
Monty Python and the Holy Grail
To Kill a Mockingbird
Some Like It Hot
Les quatre cents coups
The Treasure of the Sierra Madre
Portrait de la jeune fille en feu
Dial M for Murder
Judgment at Nuremberg
To Be or Not to Be
Koe no katachi
Mommy
Room
Soul
Kis Uykusu
The Grand Budapest Hotel
Inside Out
12 Years a Slave
Hacksaw Ridge
Rush
Ford v Ferrari
Song of the Sea
Mad Max: Fury Road
Harry Potter and the Deathly Hallows: Part 2
Prisoners
Gran Torino
Okuribito
How to Train Your Dragon
No Country for Old Men
Million Dollar Baby
Into the Wild
Before Sunset
Finding Nemo
Kill Bill: Vol. 1
Catch Me If You Can
Monsters, Inc.
Amores perros
Lagaan: Once Upon a Time in India
The Sixth Sense
The Truman Show
The Big Lebowski
Fargo
Before Sunrise
Jurassic Park
Trois couleurs: Rouge
In the Name of the Father
Dead Poets Society
Platoon
Pink Floyd: The Wall
Stand by Me
Kaze no tani no Naushika
Fanny och Alexander
Rocky
Blade Runner
The Deer Hunter
Network
Barry Lyndon
Chinatown
Paper Moon
Solaris
Persona
Cool Hand Luke
La battaglia di Algeri
What Ever Happened to Baby Jane?
The Man Who Shot Liberty Valance
Inherit the Wind
Ben-Hur
Ace in the Hole
The Bridge on the River Kwai
Mr. Smith Goes to Washington
The Third Man
The Shop Around the Corner
Gone with the Wind
It Happened One Night
The Circus
Tropa de Elite
Bohemian Rhapsody
Togo
Lion
La La Land
The Martian
Zootopia
Kaguyahime no monogatari
My Name Is Khan
Slumdog Millionaire
Bound by Honor
The Bourne Ultimatum
Pirates of the Caribbean: The Curse of the Black Pearl
Le scaphandre et le papillon
Donnie Darko
Aladdin
The Straight Story
The Incredibles
Beauty and the Beast
The Nightmare Before Christmas
Papillon
Jojo Rabbit
Rope
A Streetcar Named Desire
Brief Encounter
The Best Years of Our Lives
The Night of the Hunter
Sweet Smell of Success
Who's Afraid of Virginia Woolf?
The Last Picture Show
Star Wars: Episode VII - The Force Awakens
Knives Out
Moon
In Bruges
The Fall
Letters from Iwo Jima
4 luni, 3 saptamâni si 2 zile
Star Trek
Fantastic Mr. Fox
Harry Potter and the Prisoner of Azkaban
Almost Famous
Trois couleurs: Bleu
Harold and Maude
East of Eden
Night of the Living Dead
Le charme discret de la bourgeoisie
All the President's Men
The Manchurian Candidate
King Kong
The Philadelphia Story
The Adventures of Robin Hood
Manchester by the Sea
Kubo and the Two Strings
Freaks
The Gentleman
About Time
La migliore offerta
Big Hero 6
Das weiße Band - Eine deutsche Kindergeschichte
Auf der anderen Seite
Les triplettes de Belleville
The Curious Case of Benjamin Button
Little Miss Sunshine
Pride & Prejudice
Todo sobre mi madre
Cast Away
Shrek
Titanic
The Insider
The Remains of the Day
La double vie de Véronique
What's Eating Gilbert Grape
Breaking the Waves
Dip huet seung hung
The Boondock Saints
Predator
E.T. the Extra-Terrestrial
The Man Who Would Be King
Days of Heaven
The Right Stuff
The Killing
The Color Purple
Jules et Jim
The Innocents
The Longest Day
Guess Who's Coming to Dinner
Charlie and the Chocolate Factory
Scarface: The Shame of the Nation
Bride of Frankenstein
The Lady Vanishes
Shadow of a Doubt
Mission: Impossible
Birdman or (The Unexpected Virtue of Ignorance)
Captain America: The Winter Soldier
The Social Network
Harry Potter and the Deathly Hallows: Part 1
Silver Linings Playbook
X-Men: First Class
Star Trek Into Darkness
Midnight in Paris
Nebraska
Gravity
Deadpool 2
Kingsman: The Secret Service
Kai po che!
Detachment
Flipped
Changeling
I Am Sam
Man on Fire
Lucky Number Slevin
This Is England
The Last King of Scotland
Ex Machina
Brokeback Mountain
3:10 to Yuma
The Bourne Supremacy
The Last Samurai
The Machinist
Crash
Lost in Translation
Coraline
Harry Potter and the Goblet of Fire
The Magdalene Sisters
Good Bye Lenin!
Hedwig and the Angry Inch
South Park: Bigger, Longer & Uncut
Being John Malkovich
Rushmore
Primal Fear
The Fifth Element
As Good as It Gets
A Little Princess
Who Framed Roger Rabbit
The French Connection
Invasion of the Body Snatchers
Once Upon a Time... in Hollywood
The Florida Project
Gifted
Sabrina
The Butterfly Effect
Seven Pounds
Fear and Loathing in Las Vegas
The Jungle Book
Kelly's Heroes
Un long dimanche de fiançailles
Harry Potter and the Order of the Phoenix

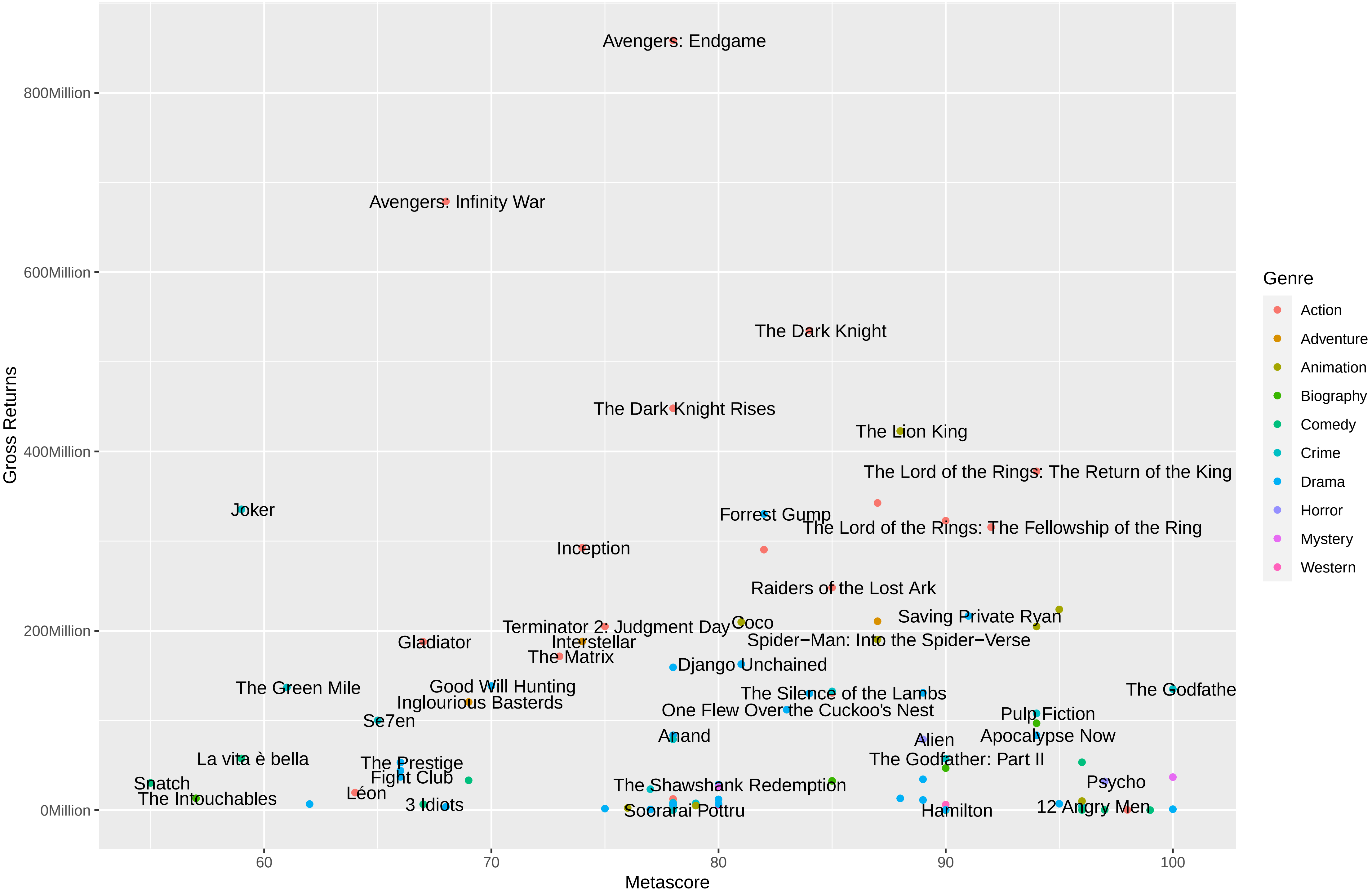
Better Circular Graph



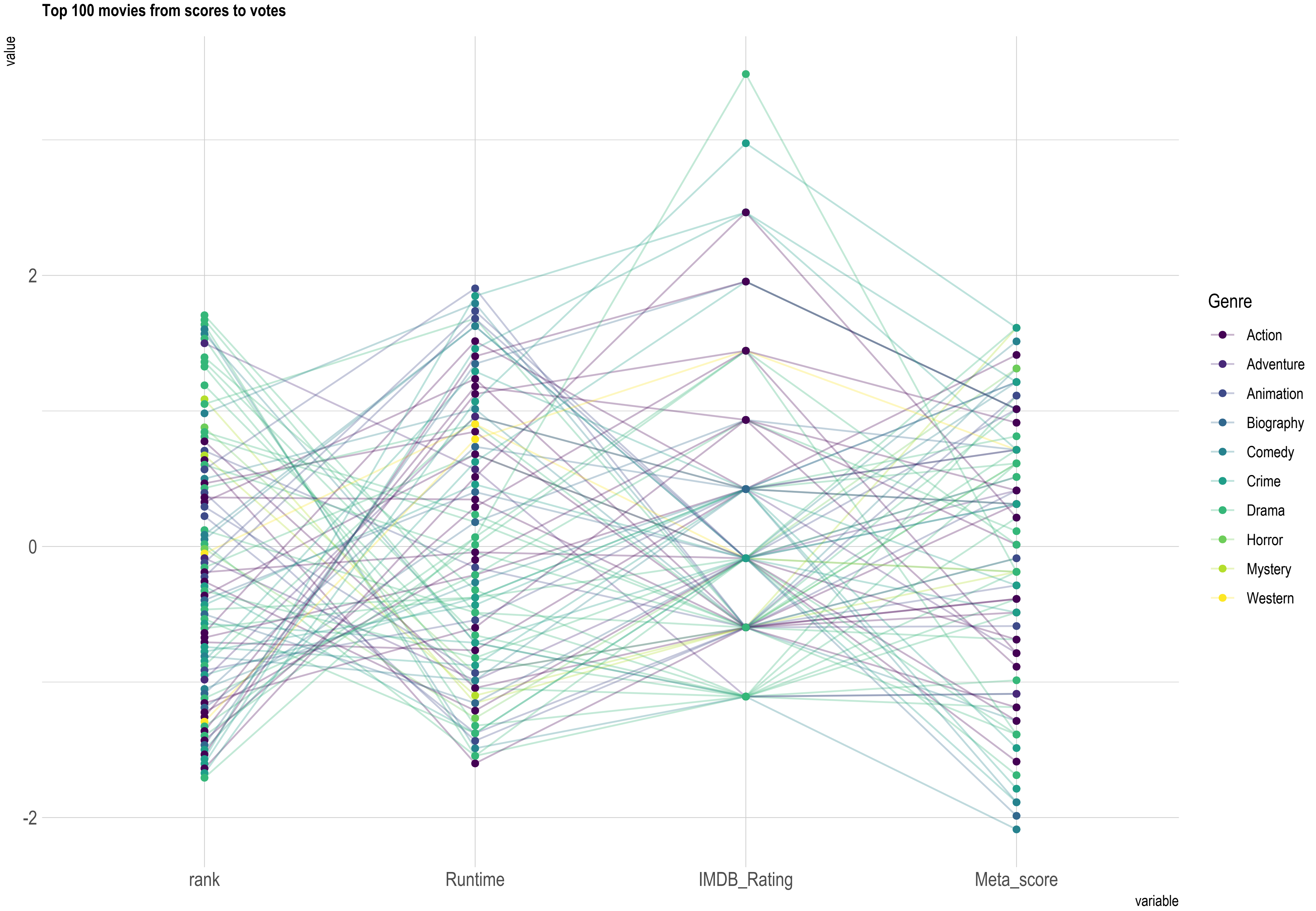

IMDB top 100 Gross Returns
From imdb dataset
Gross Returns
0Million
200Million
400Million
600Million
800Million
Year
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010
2020
Genre
Action
Adventure
Animation
Biography
Comedy
Crime
Drama
Family
Fantasy
Film−Noir
Horror
Mystery
Thriller
Western
IMDB Demographics

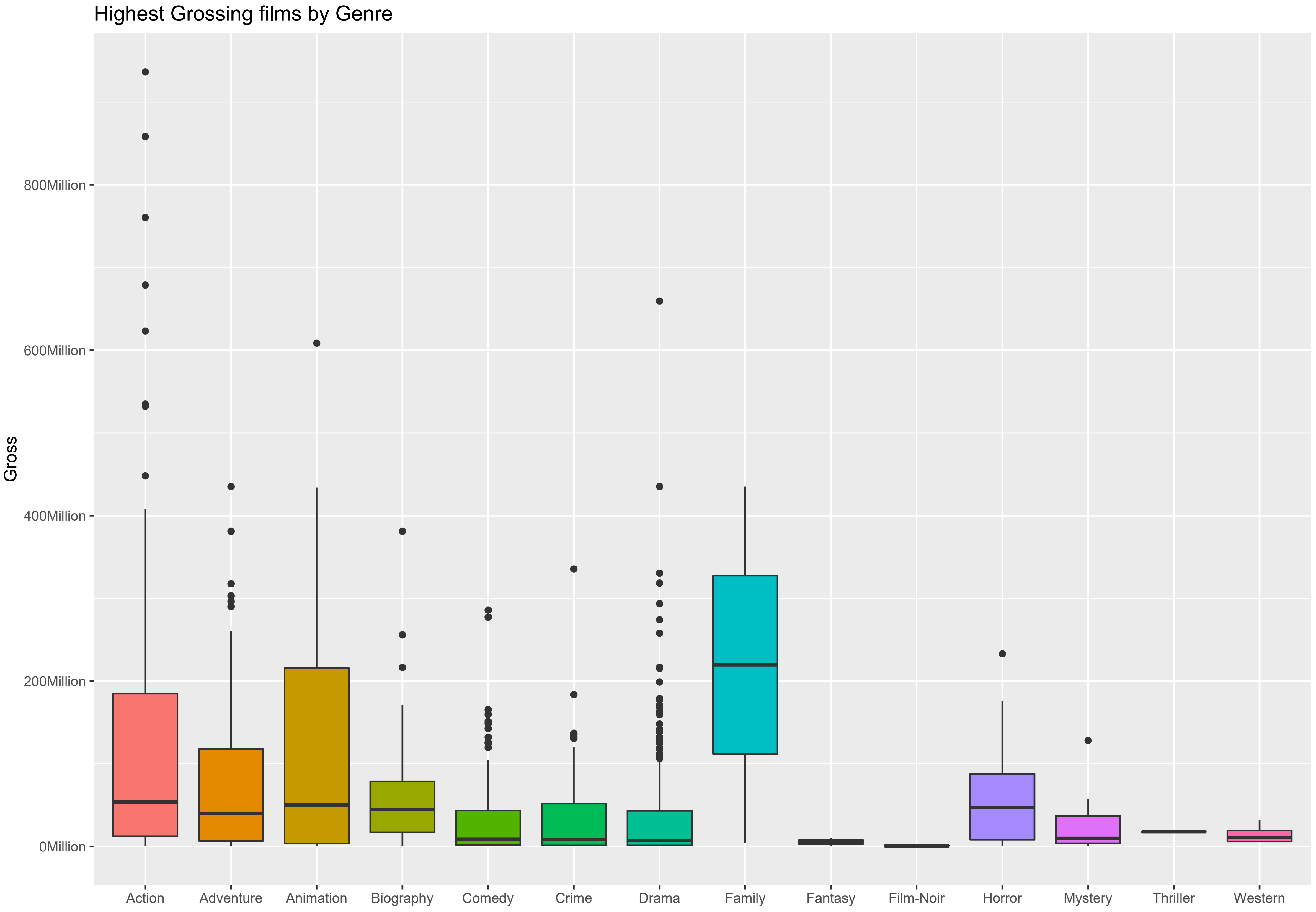
IMDB top 1000

highest 100 imdb score



IMDB Demographics


Top 100 movies from scores to votes
value
2
0
-2
rank
Runtime
IMDB_Rating
Meta_score
variable
Genre
Action
Adventure
Animation
Biography
Comedy
Crime
Drama
Horror
Mystery
Western


Highest Grossing films by Genre
Gross
800Million
600Million
400Million
200Million
0Million
Action
Adventure
Animation
Biography
Comedy
Crime
Drama
Family
Fantasy
Film-Noir
Horror
Mystery
Thriller
Western