STAT 545 Homework 5

Anita

October 17, 2018

Part 1: Factor management

With the data set of your choice, after ensuring the variable(s) you're exploring are indeed factors, you are expected to:

Drop factor / levels; Reorder levels based on knowledge from data. We've elaborated on these steps for the gapminder and singer data sets below.

Be sure to also characterize the (derived) data before and after your factor re-leveling:

Explore the effects of arrange(). Does merely arranging the data have any effect on, say, a figure? Explore the effects of reordering a factor and factor reordering coupled with arrange(). Especially, what effect does this have on a figure? These explorations should involve the data, the factor levels, and some figures.

Elaboration for the gapminder data set Drop Oceania. Filter the Gapminder data to remove observations associated with the continent of Oceania. Additionally, remove unused factor levels. Provide concrete information on the data before and after removing these rows and Oceania; address the number of rows and the levels of the affected factors.

```
library(gapminder)
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------
- tidyverse 1.2.1 --
```

```
## v ggplot2 3.0.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------------------------------------ tidy
verse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(knitr)
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

Let's explore the gapminder dataset, the continent variable

```
is.factor(gapminder$continent)
```

```
## [1] TRUE
```

```
head(gapminder)
```

```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
```

```
levels(gapminder$continent)
```

```
## [1] "Africa"   "Americas" "Asia"     "Europe"   "Oceania"
```

```
gapminder%>%
  group_by(continent)%>%
  summarize(num=n())
```

```
## # A tibble: 5 x 2
##   continent   num
##   <fct>     <int>
## 1 Africa      624
## 2 Americas    300
## 3 Asia        396
## 4 Europe      360
## 5 Oceania      24
```

Continent is a factor with five levels, and a total of

```
624+300+396+360+24
```

```
## [1] 1704
```

rows

Drop Oceania

First, we will look at the data with Oceania

```
gapminder %>%
  summarize(
    nrow = nrow(gapminder),
    nlevels_continent = nlevels(gapminder$continent),
    nlevels_country = nlevels(gapminder$country)) %>%
  knitr::kable(col.names = c("Total rows in gapminder", "Levels of continent", "Levels of countr
y"))
```

Total rows in gapminder	Levels of continent	Levels of country
1704	5	142

Now, let’s see how the rows cgange if Oceania gets dropped.

```
gapminder_without_oceania <- gapminder %>%
  filter(continent != "Oceania")
gapminder_without_oceania %>%
  summarize(
    nrow = nrow(gapminder_without_oceania),
    nlevels_continent = nlevels(gapminder_without_oceania$continent),
    nlevels_country = nlevels(gapminder_without_oceania$country)) %>%
  knitr::kable(col.names = c("Total rows in gapminder without Oceania", "Levels of continent wit
hout Oceania", "Levels of country without Oceania"))
```

Total rows in gapminder without Oceania	Levels of continent without Oceania	Levels of country without Oceania
1680	5	142

Let's look how many rows each continent has

```
gapminder_without_oceania%>%
  group_by(continent)%>%
  summarize(num=n())
```

```
## # A tibble: 4 x 2
##   continent   num
##   <fct>     <int>
## 1 Africa      624
## 2 Americas    300
## 3 Asia        396
## 4 Europe      360
```

Continent is now a factor with four levels, and a total of

```
624+300+396+360
```

```
## [1] 1680
```

rows

Reorder the levels of country or continent. Use the forcats package to change the order of the factor levels, based on a principled summary of one of the quantitative variables. Consider experimenting with a summary statistic beyond the most basic choice of the median.

First let's look at the standard deviation of countries

```
library(forcats)

gapminder_original_order <- gapminder %>%
  filter(continent == "Africa") %>%
  group_by(country) %>%
  mutate(sd_life = sd(lifeExp)) %>%
  select(country, sd_life) %>%
  unique() # have to delete rows that repeat or I get an errr message
knitr::kable(gapminder_original_order)
```

country	sd_life
Algeria	10.340069
Angola	4.005276
Benin	6.128681
Botswana	5.929476
Burkina Faso	6.845792
Burundi	3.174882
Cameroon	5.467960

country	sd_life
Central African Republic	4.720690
Chad	4.887978
Comoros	8.132353
Congo, Dem. Rep.	2.869210
Congo, Rep.	4.878987
Cote d'Ivoire	4.421421
Djibouti	6.710003
Egypt	10.062500
Equatorial Guinea	5.600456
Eritrea	6.903925
Ethiopia	5.627192
Gabon	8.933194
Gambia	10.545929
Ghana	5.846972
Guinea	7.743160
Guinea-Bissau	4.937368
Kenya	5.596199
Lesotho	5.914277
Liberia	2.419094
Libya	11.372181
Madagascar	7.297844
Malawi	4.607323
Mali	6.808537
Mauritania	8.057280
Mauritius	6.497274
Morocco	9.806162
Mozambique	4.599184
Namibia	6.303906
Niger	6.509444
Nigeria	4.021207

country	sd_life
Reunion	8.434938
Rwanda	6.307415
Sao Tome and Principe	6.283923
Senegal	9.141934
Sierra Leone	3.937828
Somalia	4.503828
South Africa	5.455502
Sudan	6.927843
Swaziland	6.562668
Tanzania	3.602435
Togo	7.247043
Tunisia	10.701244
Uganda	3.747267
Zambia	4.453246
Zimbabwe	7.071816

Now, let’s rearrange from highest to lowest standard deviation

```
gapminder_new_order <- gapminder_original_order %>%
  arrange(desc(sd_life))

knitr::kable(gapminder_new_order)
```

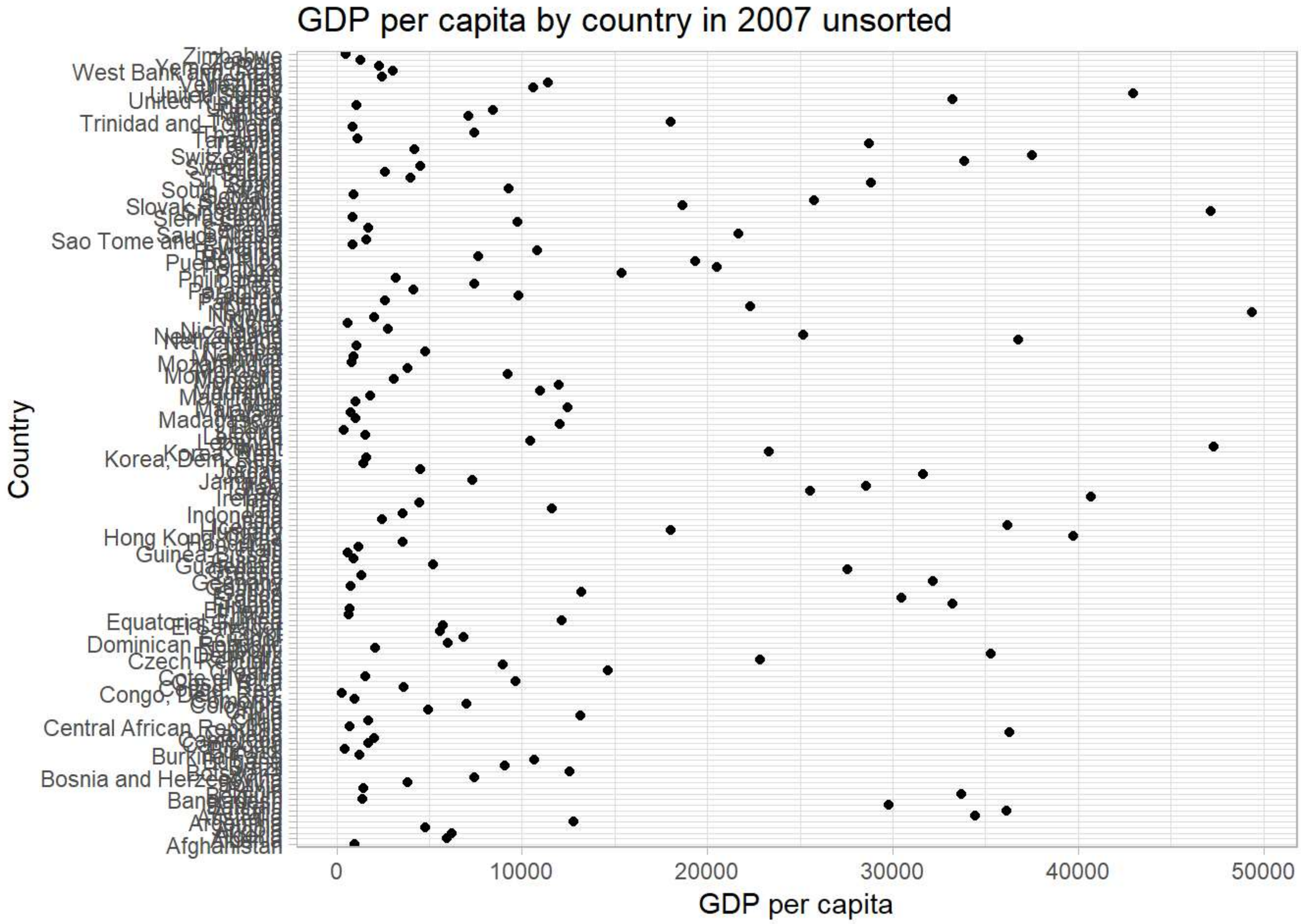
country	sd_life
Libya	11.372181
Tunisia	10.701244
Gambia	10.545929
Algeria	10.340069
Egypt	10.062500
Morocco	9.806162
Senegal	9.141934
Gabon	8.933194
Reunion	8.434938

country	sd_life
Comoros	8.132353
Mauritania	8.057280
Guinea	7.743160
Madagascar	7.297844
Togo	7.247043
Zimbabwe	7.071816
Sudan	6.927843
Eritrea	6.903925
Burkina Faso	6.845792
Mali	6.808537
Djibouti	6.710003
Swaziland	6.562668
Niger	6.509444
Mauritius	6.497274
Rwanda	6.307415
Namibia	6.303906
Sao Tome and Principe	6.283923
Benin	6.128681
Botswana	5.929476
Lesotho	5.914277
Ghana	5.846972
Ethiopia	5.627192
Equatorial Guinea	5.600456
Kenya	5.596199
Cameroon	5.467960
South Africa	5.455502
Guinea-Bissau	4.937368
Chad	4.887978
Congo, Rep.	4.878987
Central African Republic	4.720690

country	sd_life
Malawi	4.607323
Mozambique	4.599184
Somalia	4.503828
Zambia	4.453246
Cote d'Ivoire	4.421421
Nigeria	4.021207
Angola	4.005276
Sierra Leone	3.937828
Uganda	3.747267
Tanzania	3.602435
Burundi	3.174882
Congo, Dem. Rep.	2.869210
Liberia	2.419094

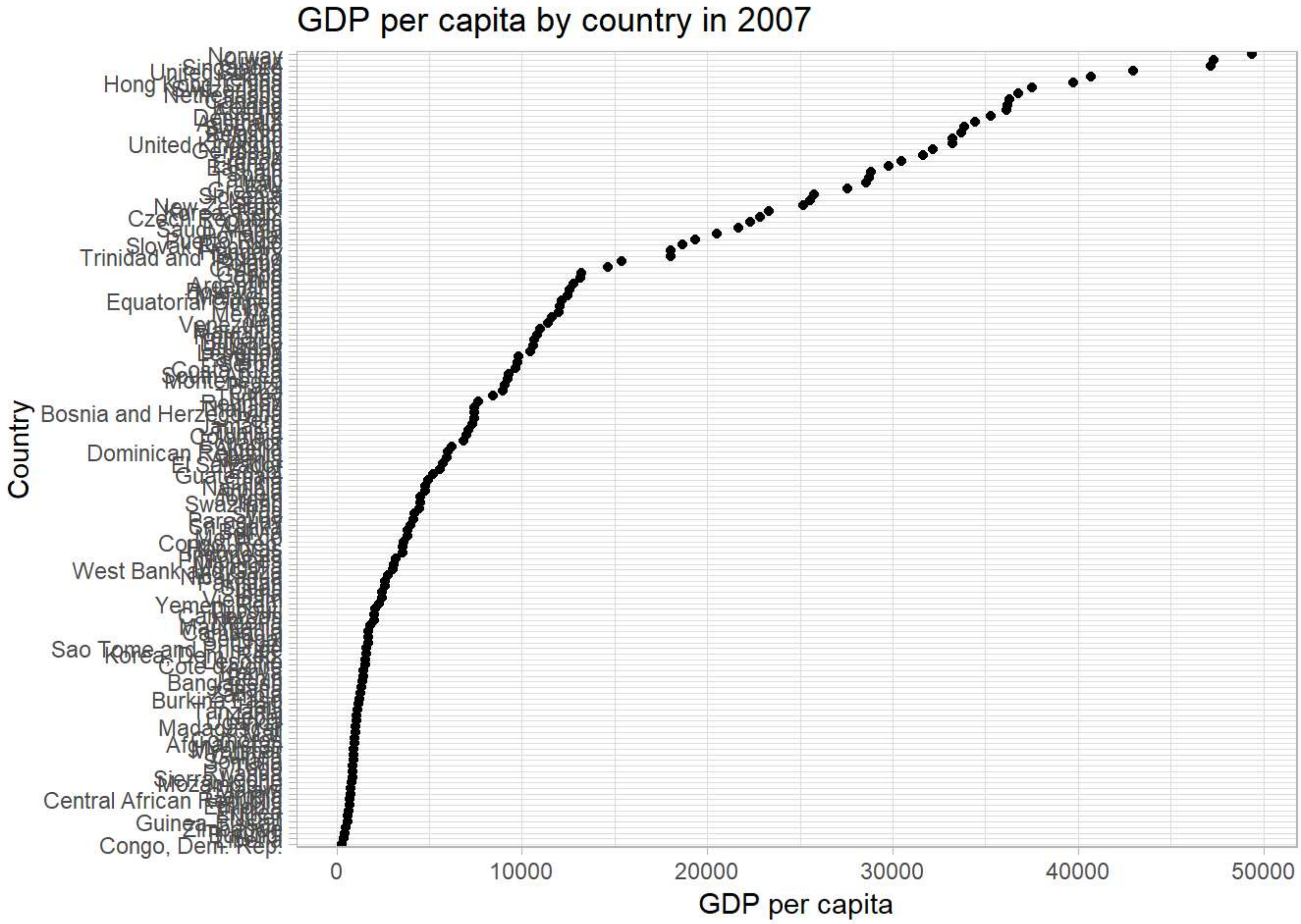
Now lets look at a figure. Here we look at GDP per capita per country in 2007.

```
gap_2007 <- gapminder %>%
  filter(year == 2007)
ggplot(gap_2007, aes(gdpPercap, country)) + geom_point()+
  xlab( "GDP per capita") +
  ylab( "Country" ) +
  ggtitle( "GDP per capita by country in 2007 unsorted" ) +
  theme_light()
```



Unfortunately, the data is unsorted and so it’s not so easy to look at it. Now let’s arrange it by GDP.

```
#Let's use `fct_reorder()` to reorder the countries by gdp per capita, and produce the same plo
t:
gap_2007 %>%
  mutate(country = fct_reorder(country, gdpPercap)) %>%
ggplot(aes(gdpPercap, country)) + geom_point()+
  xlab( "GDP per capita") +
  ylab( "Country" ) +
  ggtitle( "GDP per capita by country in 2007" ) +
  theme_light()
```



Part 2: File I/O Experiment with one or more of write_csv()/read_csv() (and/or TSV friends), saveRDS()/readRDS(), dput()/dget(). Create something new, probably by filtering or grouped-summarization of Singer or Gapminder. I highly recommend you fiddle with the factor levels, i.e. make them non-alphabetical (see previous section). Explore whether this survives the round trip of writing to file then reading back in.

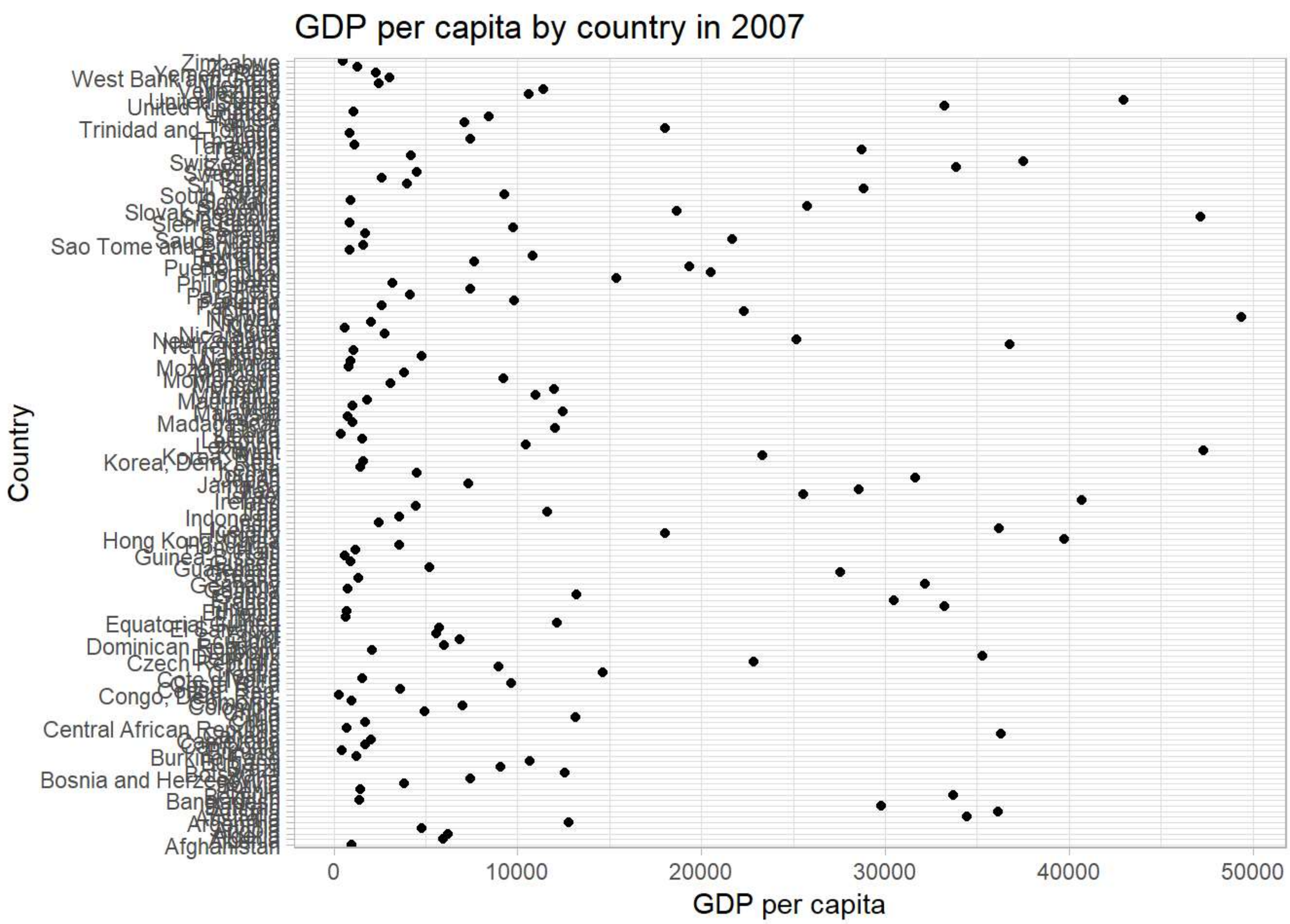
First I export the dataset I created above to a csv file.

```
write_csv(gap_2007, "gap_2007.csv")
```

Now, let’s see if this new data file when we try to create the same plot as above is ordered by GDP per capita

```
read_csv("gap_2007.csv") %>%  #import .csv
  ggplot(aes(gdpPercap, country)) + geom_point()+
  xlab( "GDP per capita") +
  ylab( "Country" ) +
  ggtitle( "GDP per capita by country in 2007" ) +
  theme_light()
```

```
## Parsed with column specification:
## cols(
##   country = col_character(),
##   continent = col_character(),
##   year = col_integer(),
##   lifeExp = col_double(),
##   pop = col_integer(),
##   gdpPercap = col_double()
## )
```



As we can see, it isn't.

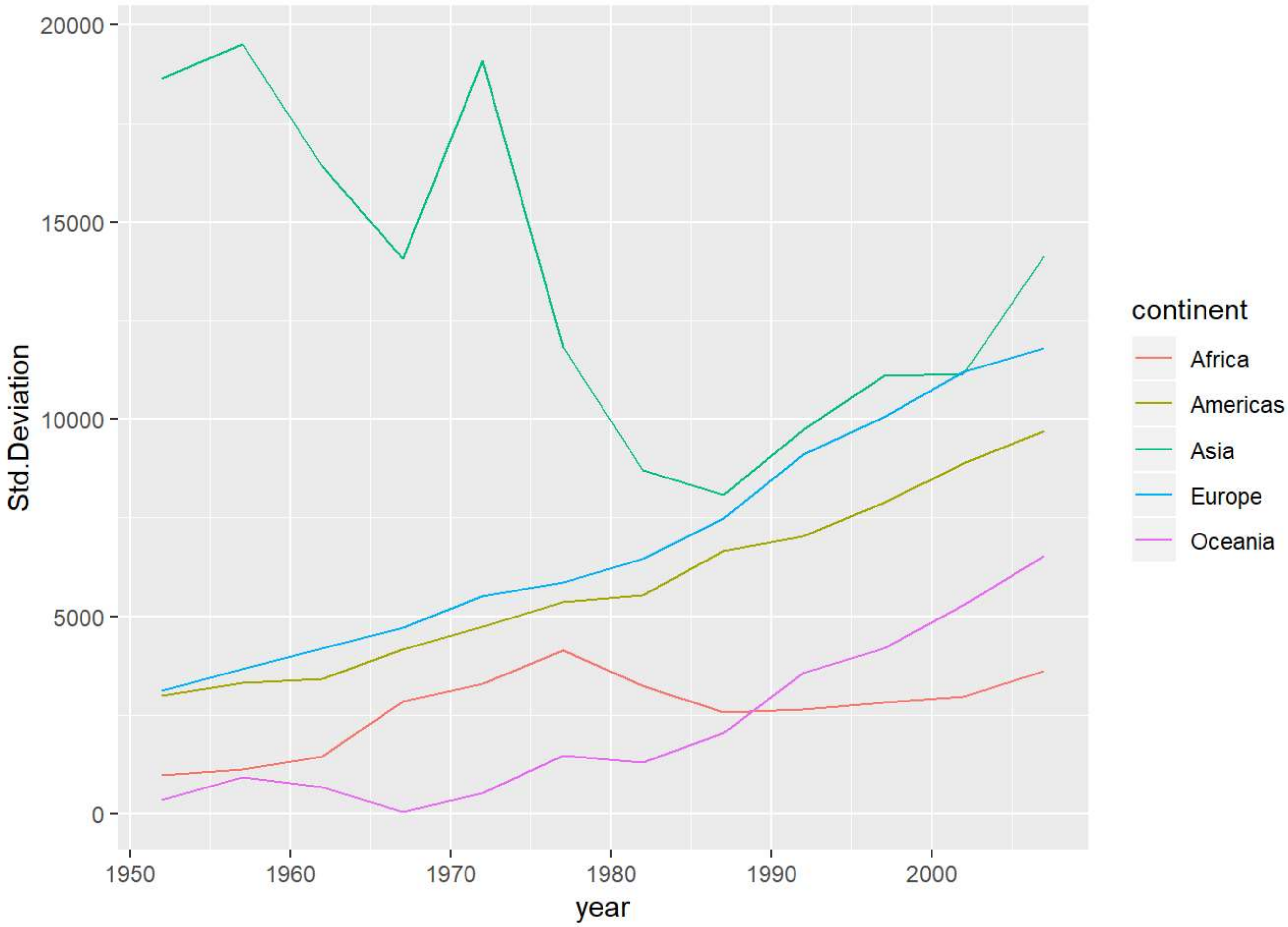
Part 3: Visualization design Remake at least one figure or create a new one, in light of something you learned in the recent class meetings about visualization design and color. Maybe juxtapose your first attempt and what you obtained after some time spent working on it. Reflect on the differences. If using Gapminder, you can use the country or continent color scheme that ships with Gapminder. Consult the dimensions listed in All the Graph Things.

Then, make a new graph by converting this visual (or another, if you'd like) to a plotly graph. What are some things that plotly makes possible, that are not possible with a regular ggplot2 graph?

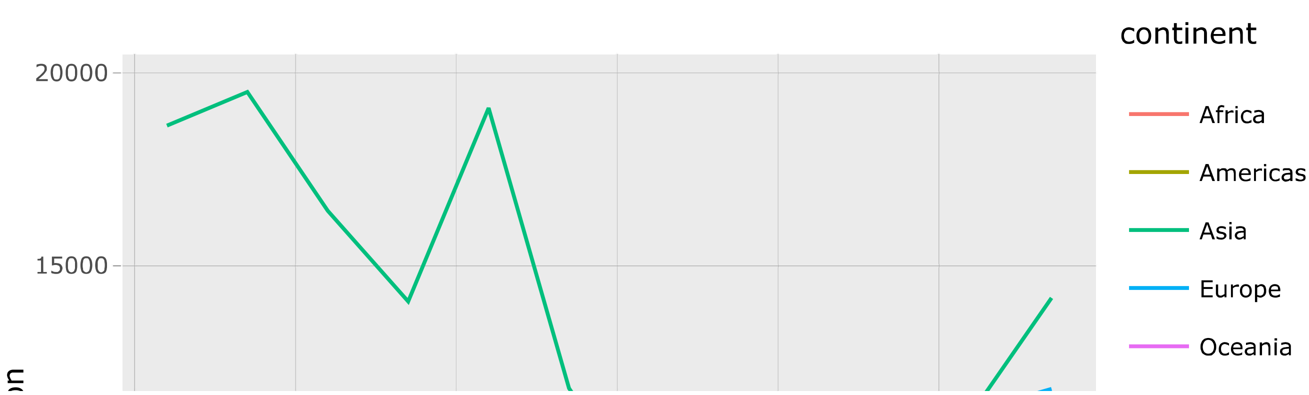
Spread of GDP per cap by year by continent

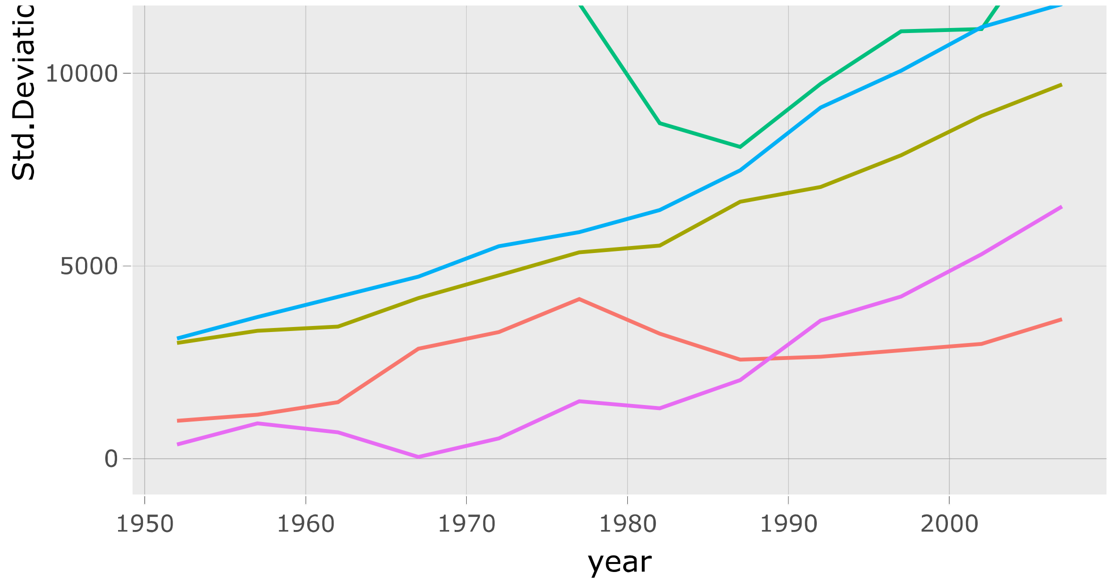
Now let's look at a graph that I made for a previous homework asignment and compare ggplot with plotly

```
gdp.2 <-  gapminder %>%
  group_by(continent, year) %>%
  summarize(Std.Deviation = sd(gdpPercap),
            Variance = var(gdpPercap))

ggplot(gdp.2, aes(year)) +
  geom_line(aes(y=Std.Deviation, color=continent)) +
  scale_size_area()
```




```
gdp_spread <- ggplot(gdp.2, aes(year)) +
  geom_line(aes(y=Std.Deviation, color=continent)) +
  scale_size_area()

ggplotly(gdp_spread)
```





We can alos look at other functions that can make use of ggplot (such as visreg) and see if they can likewise be converted into plotly.

In this example, I'm looking at an interaction between population and GDP per capita in predicting life expectancy for the year 2007. (Note: This makes little sense theoretically and as we can see the interaction term is not significant, but it serves to illustrate the possibilities of plotly.)

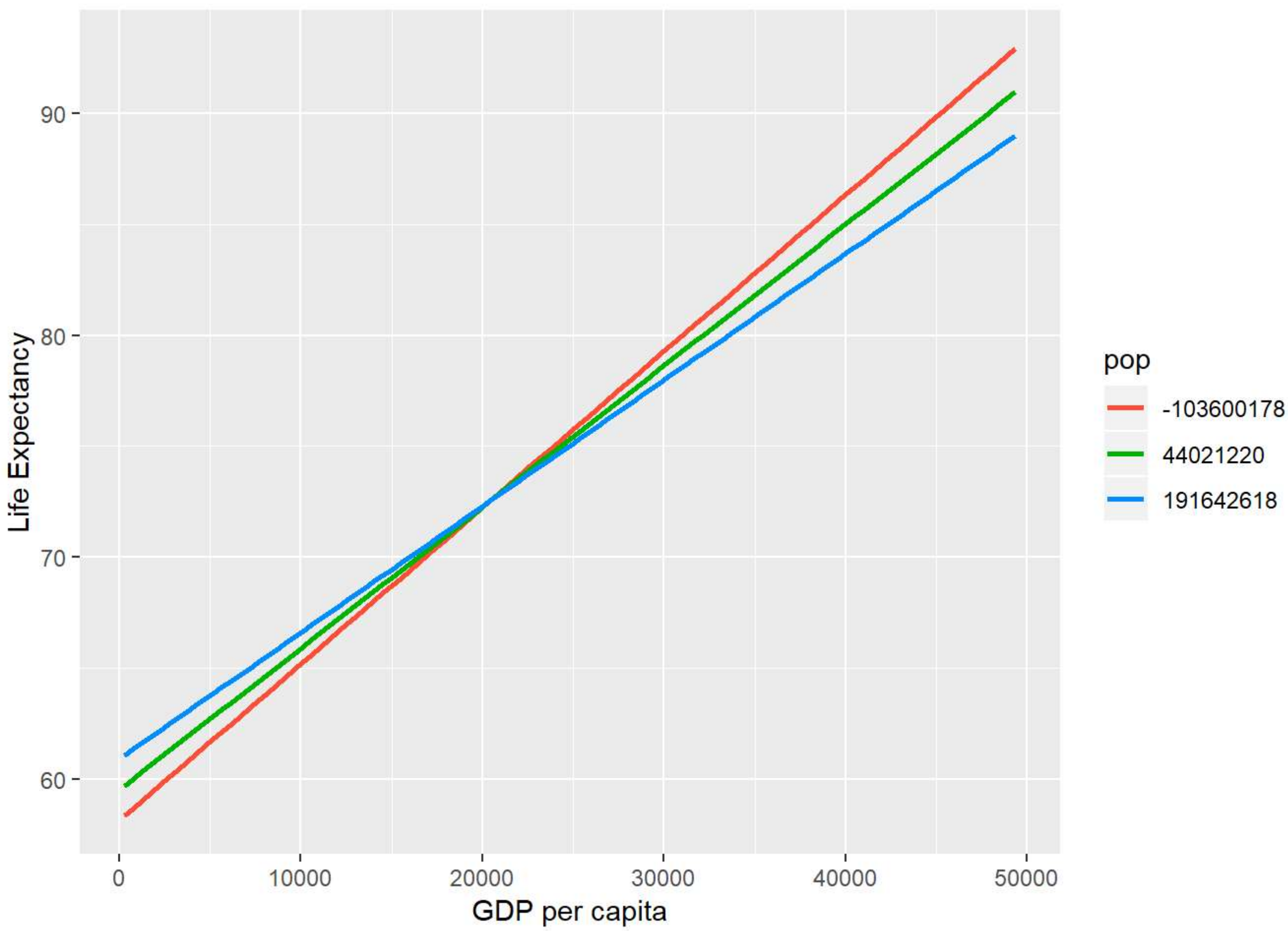
```
gap_2007 <- gapminder %>%
  filter(year == 2007)

m1 <- lm(lifeExp ~ gdpPercap*pop, data=gap_2007)
summary(m1)
```

```
##
## Call:
## lm(formula = lifeExp ~ gdpPercap * pop, data = gap_2007)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.466  -5.910   1.877   6.942  13.393
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.910e+01  1.056e+00  55.947   <2e-16 ***
## gdpPercap      6.575e-04  6.393e-05  10.284   <2e-16 ***
## pop            9.386e-09  6.428e-09   1.460    0.146
## gdpPercap:pop -4.595e-13  7.586e-13  -0.606    0.546
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.89 on 138 degrees of freedom
## Multiple R-squared:  0.4693, Adjusted R-squared:  0.4578
## F-statistic: 40.68 on 3 and 138 DF,  p-value: < 2.2e-16
```

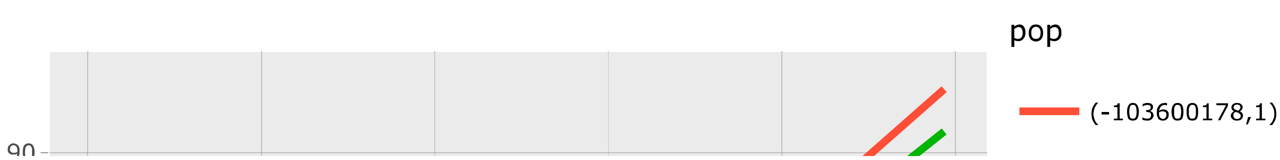
```
psych::describe(gap_2007$pop) #extract mean and sd to look at population as the moderator and de
fine three levels (mean and +/- 1sd)
```

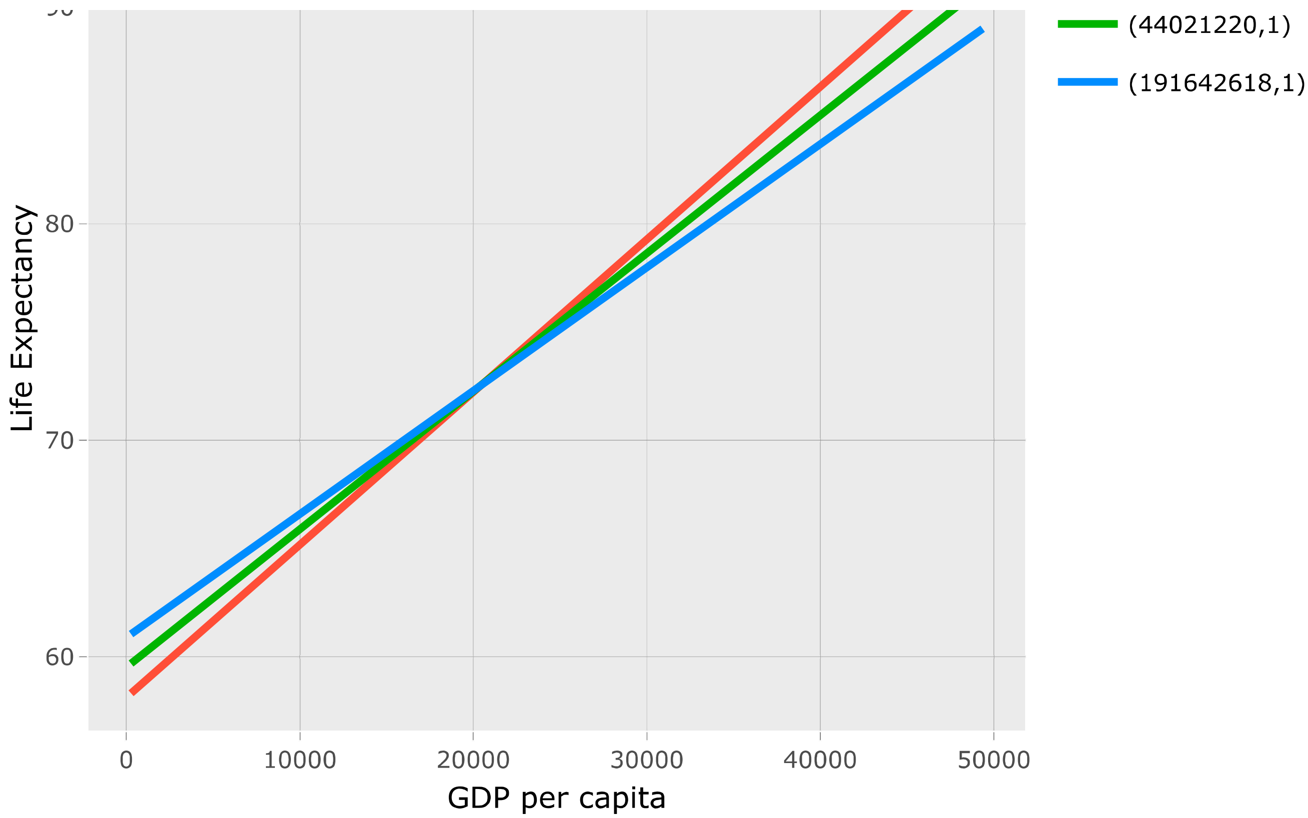
```
##    vars   n     mean        sd   median  trimmed      mad    min
## X1    1 142 44021220 147621398 10517531 18112754 12212490 199579
##           max      range skew kurtosis       se
## X1 1318683096 1318483517 7.25    55.42 12388113
```

```
library(visreg)
visreg(m1, "gdpPercap", by="pop", breaks=c(-103600178,44021220,191642618), overlay=TRUE,
       band=FALSE, ylab="Life Expectancy", xlab="GDP per capita",
       bty="n", partial=FALSE, rug=FALSE, gg=TRUE)
```




```
gap_interaction <-visreg(m1, "gdpPercap", by="pop", breaks=c(-103600178,44021220,191642618), ove
rlay=TRUE,
       band=FALSE, ylab="Life Expectancy", xlab="GDP per capita",
       bty="n", partial=FALSE, rug=FALSE, gg=TRUE)

ggplotly(gap_interaction)
```





Note; the plotly graphs won't render in this file, so you have to check them out seprately. Ytr hovering over them, they have many useful functions, such as zooming in and out.

Part 4: Writing figures to file Use ggsave() to explicitly save a plot to file. Then use to load and embed it in your report. You can play around with various options, such as:

Arguments of ggsave(), such as width, height, resolution or text scaling. Various graphics devices, e.g. a vector vs. raster format. Explicit provision of the plot object p via ggsave(…, plot = p). Show a situation in which this actually matters.

```
ggsave("gap_interaction.png", gap_interaction, width=40, height=40, units = "cm", device = 'png'
)
```

This graph got automatically saved in my Homework 5 folder on my laptop.