TASK 4 - Data Visualisation

LAURA PAAABY

2022-09-27

4:W35: Visualize data (not only) with ggplot

Global development since 1957

learning how to create animations with gganimate package! :)

The data == a subset of data on global development from 1952 - 2007 in increments of 5 years - to capture the period between the Second World War and the Global Financial Crisis.

The task == Explore the data and visualise it in both static and animated ways, providing answers and solutions to 7 questions/tasks below.**

Get the necessary packages

First, start with installing the relevant packages 'tidyverse', 'gganimate', and 'gapminder'.

```
## Loading required package: ggplot2
```

```
## Warning: package 'gifski' was built under R version 4.0.5
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
```

```
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.2.1      ✔ stringr 1.4.0
## ✔ readr   2.1.2      ✔ forcats 0.5.2
## ✔ purrr   0.3.4
```

```
## Warning: package 'readr' was built under R version 4.0.5
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

Look at the data and tackle the tasks

First, see which specific years are actually represented in the dataset and what variables are being recorded for each country. Note that when you run the cell below, Rmarkdown will give you two results - one for each line - that you can flip between.

```
str(gapminder)
```

```
## tibble [1,704 × 6] (S3: tbl_df/tbl/data.frame)
##  $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ year     : int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ lifeExp  : num [1:1704] 28.8 30.3 32 34 36.1 ...
##  $ pop      : int [1:1704] 8425333 9240934 10267083 11537966 13079460 14880372 128
81816 13867957 16317921 22227415 ...
##  $ gdpPercap: num [1:1704] 779 821 853 836 740 ...
```

```
unique(gapminder$year)
```

```
##  [1] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 2002 2007
```

```
head(gapminder)
```

```
## # A tibble: 6 × 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.
```

```
gapminder$lifeExp
```

```
##   [1] 28.80100 30.33200 31.99700 34.02000 36.08800 38.43800 39.85400 40.82200
##   [9] 41.67400 41.76300 42.12900 43.82800 55.23000 59.28000 64.82000 66.22000
##  [17] 67.69000 68.93000 70.42000 72.00000 71.58100 72.95000 75.65100 76.42300
##  [25] 43.07700 45.68500 48.30300 51.40700 54.51800 58.01400 61.36800 65.79900
##  [33] 67.74400 69.15200 70.99400 72.30100 30.01500 31.99900 34.00000 35.98500
##  [41] 37.92800 39.48300 39.94200 39.90600 40.64700 40.96300 41.00300 42.73100
##  [49] 62.48500 64.39900 65.14200 65.63400 67.06500 68.48100 69.94200 70.77400
##  [57] 71.86800 73.27500 74.34000 75.32000 69.12000 70.33000 70.93000 71.10000
##  [65] 71.93000 73.49000 74.74000 76.32000 77.56000 78.83000 80.37000 81.23500
##  [73] 66.80000 67.48000 69.54000 70.14000 70.63000 72.17000 73.18000 74.94000
##  [81] 76.04000 77.51000 78.98000 79.82900 50.93900 53.83200 56.92300 59.92300
##  [89] 63.30000 65.59300 69.05200 70.75000 72.60100 73.92500 74.79500 75.63500
##  [97] 37.48400 39.34800 41.21600 43.45300 45.25200 46.92300 50.00900 52.81900
## [105] 56.01800 59.41200 62.01300 64.06200 68.00000 69.24000 70.25000 70.94000
## [113] 71.44000 72.80000 73.93000 75.35000 76.46000 77.53000 78.32000 79.44100
## [121] 38.22300 40.35800 42.61800 44.88500 47.01400 49.19000 50.90400 52.33700
## [129] 53.91900 54.77700 54.40600 56.72800 40.41400 41.89000 43.42800 45.03200
## [137] 46.71400 50.02300 53.85900 57.25100 59.95700 62.05000 63.88300 65.55400
## [145] 53.82000 58.45000 61.93000 64.79000 67.45000 69.86000 70.69000 71.14000
## [153] 72.17800 73.24400 74.09000 74.85200 47.62200 49.61800 51.52000 53.29800
## [161] 56.02400 59.31900 61.48400 63.62200 62.74500 52.55600 46.63400 50.72800
## [169] 50.91700 53.28500 55.66500 57.63200 59.50400 61.48900 63.33600 65.20500
## [177] 67.05700 69.38800 71.00600 72.39000 59.60000 66.61000 69.51000 70.42000
## [185] 70.90000 70.81000 71.08000 71.34000 71.19000 70.32000 72.14000 73.00500
## [193] 31.97500 34.90600 37.81400 40.69700 43.59100 46.13700 48.12200 49.55700
## [201] 50.26000 50.32400 50.65000 52.29500 39.03100 40.53300 42.04500 43.54800
## [209] 44.05700 45.91000 47.47100 48.21100 44.73600 45.32600 47.36000 49.58000
## [217] 39.41700 41.36600 43.41500 45.41500 40.31700 31.22000 50.95700 53.91400
## [225] 55.80300 56.53400 56.75200 59.72300 38.52300 40.42800 42.64300 44.79900
## [233] 47.04900 49.35500 52.96100 54.98500 54.31400 52.19900 49.85600 50.43000
## [241] 68.75000 69.96000 71.30000 72.13000 72.88000 74.21000 75.76000 76.86000
## [249] 77.95000 78.61000 79.77000 80.65300 35.46300 37.46400 39.47500 41.47800
## [257] 43.45700 46.77500 48.29500 50.48500 49.39600 46.06600 43.30800 44.74100
## [265] 38.09200 39.88100 41.71600 43.60100 45.56900 47.38300 49.51700 51.05100
## [273] 51.72400 51.57300 50.52500 50.65100 54.74500 56.07400 57.92400 60.52300
## [281] 63.44100 67.05200 70.56500 72.49200 74.12600 75.81600 77.86000 78.55300
## [289] 44.00000 50.54896 44.50136 58.38112 63.11888 63.96736 65.52500 67.27400
## [297] 68.69000 70.42600 72.02800 72.96100 50.64300 55.11800 57.86300 59.96300
## [305] 61.62300 63.83700 66.65300 67.76800 68.42100 70.31300 71.68200 72.88900
## [313] 40.71500 42.46000 44.46700 46.47200 48.94400 50.93900 52.93300 54.92600
## [321] 57.93900 60.66000 62.97400 65.15200 39.14300 40.65200 42.12200 44.05600
## [329] 45.98900 47.80400 47.78400 47.41200 45.54800 42.58700 44.96600 46.46200
## [337] 42.11100 45.05300 48.43500 52.04000 54.90700 55.62500 56.69500 57.47000
## [345] 56.43300 52.96200 52.97000 55.32200 57.20600 60.02600 62.84200 65.42400
## [353] 67.84900 70.75000 73.45000 74.75200 75.71300 77.26000 78.12300 78.78200
## [361] 40.47700 42.46900 44.93000 47.35000 49.80100 52.37400 53.98300 54.65500
## [369] 52.04400 47.99100 46.83200 48.32800 61.21000 64.77000 67.13000 68.50000
## [377] 69.61000 70.64000 70.46000 71.52000 72.52700 73.68000 74.87600 75.74800
## [385] 59.42100 62.32500 65.24600 68.29000 70.72300 72.64900 73.71700 74.17400
## [393] 74.41400 76.15100 77.15800 78.27300 66.87000 69.03000 69.90000 70.38000
## [401] 70.29000 70.71000 70.96000 71.58000 72.40000 74.01000 75.51000 76.48600
## [409] 70.78000 71.81000 72.35000 72.96000 73.47000 74.69000 74.63000 74.80000
## [417] 75.33000 76.11000 77.18000 78.33200 34.81200 37.32800 39.69300 42.07400
## [425] 44.36600 46.51900 48.81200 50.04000 51.60400 53.15700 53.37300 54.79100
## [433] 45.92800 49.82800 53.45900 56.75100 59.63100 61.78800 63.72700 66.04600
```

```
## [441] 68.45700 69.95700 70.84700 72.23500 48.35700 51.35600 54.64000 56.67800
## [449] 58.79600 61.31000 64.34200 67.23100 69.61300 72.31200 74.17300 74.99400
## [457] 41.89300 44.44400 46.99200 49.29300 51.13700 53.31900 56.00600 59.79700
## [465] 63.67400 67.21700 69.80600 71.33800 45.26200 48.57000 52.30700 55.85500
## [473] 58.20700 56.69600 56.60400 63.15400 66.79800 69.53500 70.73400 71.87800
## [481] 34.48200 35.98300 37.48500 38.98700 40.51600 42.02400 43.66200 45.66400
## [489] 47.54500 48.24500 49.34800 51.57900 35.92800 38.04700 40.15800 42.18900
## [497] 44.14200 44.53500 43.89000 46.45300 49.99100 53.37800 55.24000 58.04000
## [505] 34.07800 36.66700 40.05900 42.11500 43.51500 44.51000 44.91600 46.68400
## [513] 48.09100 49.40200 50.72500 52.94700 66.55000 67.49000 68.75000 69.83000
## [521] 70.87000 72.52000 74.55000 74.83000 75.70000 77.13000 78.37000 79.31300
## [529] 67.41000 68.93000 70.51000 71.55000 72.38000 73.83000 74.89000 76.34000
## [537] 77.46000 78.64000 79.59000 80.65700 37.00300 38.99900 40.48900 44.59800
## [545] 48.69000 52.79000 56.56400 60.19000 61.36600 60.46100 56.76100 56.73500
## [553] 30.00000 32.06500 33.89600 35.85700 38.30800 41.84200 45.58000 49.26500
## [561] 52.64400 55.86100 58.04100 59.44800 67.50000 69.10000 70.30000 70.80000
## [569] 71.00000 72.50000 73.80000 74.84700 76.07000 77.34000 78.67000 79.40600
## [577] 43.14900 44.77900 46.45200 48.07200 49.87500 51.75600 53.74400 55.72900
## [585] 57.50100 58.55600 58.45300 60.02200 65.86000 67.86000 69.51000 71.00000
## [593] 72.34000 73.68000 75.24000 76.67000 77.03000 77.86900 78.25600 79.48300
## [601] 42.02300 44.14200 46.95400 50.01600 53.73800 56.02900 58.13700 60.78200
## [609] 63.37300 66.32200 68.97800 70.25900 33.60900 34.55800 35.75300 37.19700
## [617] 38.84200 40.76200 42.89100 45.55200 48.57600 51.45500 53.67600 56.00700
## [625] 32.50000 33.48900 34.48800 35.49200 36.48600 37.46500 39.32700 41.24500
## [633] 43.26600 44.87300 45.50400 46.38800 37.57900 40.69600 43.59000 46.24300
## [641] 48.04200 49.92300 51.46100 53.63600 55.08900 56.67100 58.13700 60.91600
## [649] 41.91200 44.66500 48.04100 50.92400 53.88400 57.40200 60.90900 64.49200
## [657] 66.39900 67.65900 68.56500 70.19800 60.96000 64.75000 67.65000 70.00000
## [665] 72.00000 73.60000 75.45000 76.20000 77.60100 80.00000 81.49500 82.20800
## [673] 64.03000 66.41000 67.96000 69.50000 69.76000 69.95000 69.39000 69.58000
## [681] 69.17000 71.04000 72.59000 73.33800 72.49000 73.47000 73.68000 73.73000
## [689] 74.46000 76.11000 76.99000 77.23000 78.77000 78.95000 80.50000 81.75700
## [697] 37.37300 40.24900 43.60500 47.19300 50.65100 54.20800 56.59600 58.55300
## [705] 60.22300 61.76500 62.87900 64.69800 37.46800 39.91800 42.51800 45.96400
## [713] 49.20300 52.70200 56.15900 60.13700 62.68100 66.04100 68.58800 70.65000
## [721] 44.86900 47.18100 49.32500 52.46900 55.23400 57.70200 59.62000 63.04000
## [729] 65.74200 68.04200 69.45100 70.96400 45.32000 48.43700 51.45700 54.45900
## [737] 56.95000 60.41300 62.03800 65.04400 59.46100 58.81100 57.04600 59.54500
## [745] 66.91000 68.90000 70.29000 71.08000 71.28000 72.03000 73.10000 74.36000
## [753] 75.46700 76.12200 77.78300 78.88500 65.39000 67.84000 69.39000 70.75000
## [761] 71.63000 73.06000 74.45000 75.60000 76.93000 78.26900 79.69600 80.74500
## [769] 65.94000 67.81000 69.24000 71.06000 72.19000 73.48000 74.98000 76.42000
## [777] 77.44000 78.82000 80.24000 80.54600 58.53000 62.61000 65.61000 67.51000
## [785] 69.00000 70.11000 71.21000 71.77000 71.76600 72.26200 72.04700 72.56700
## [793] 63.03000 65.50000 68.73000 71.43000 73.42000 75.38000 77.11000 78.67000
## [801] 79.36000 80.69000 82.00000 82.60300 43.15800 45.66900 48.12600 51.62900
## [809] 56.52800 61.13400 63.73900 65.86900 68.01500 69.77200 71.26300 72.53500
## [817] 42.27000 44.68600 47.94900 50.65400 53.55900 56.15500 58.76600 59.33900
## [825] 59.28500 54.40700 50.99200 54.11000 50.05600 54.08100 56.65600 59.94200
## [833] 63.98300 67.15900 69.10000 70.64700 69.97800 67.72700 66.66200 67.29700
## [841] 47.45300 52.68100 55.29200 57.71600 62.61200 64.76600 67.12300 69.81000
## [849] 72.24400 74.64700 77.04500 78.62300 55.56500 58.03300 60.47000 64.62400
## [857] 67.71200 69.34300 71.30900 74.17400 75.19000 76.15600 76.90400 77.58800
## [865] 55.92800 59.48900 62.09400 63.87000 65.42100 66.09900 66.98300 67.92600
## [873] 69.29200 70.26500 71.02800 71.99300 42.13800 45.04700 47.74700 48.49200
## [881] 49.76700 52.20800 55.07800 57.18000 59.68500 55.55800 44.59300 42.59200
```

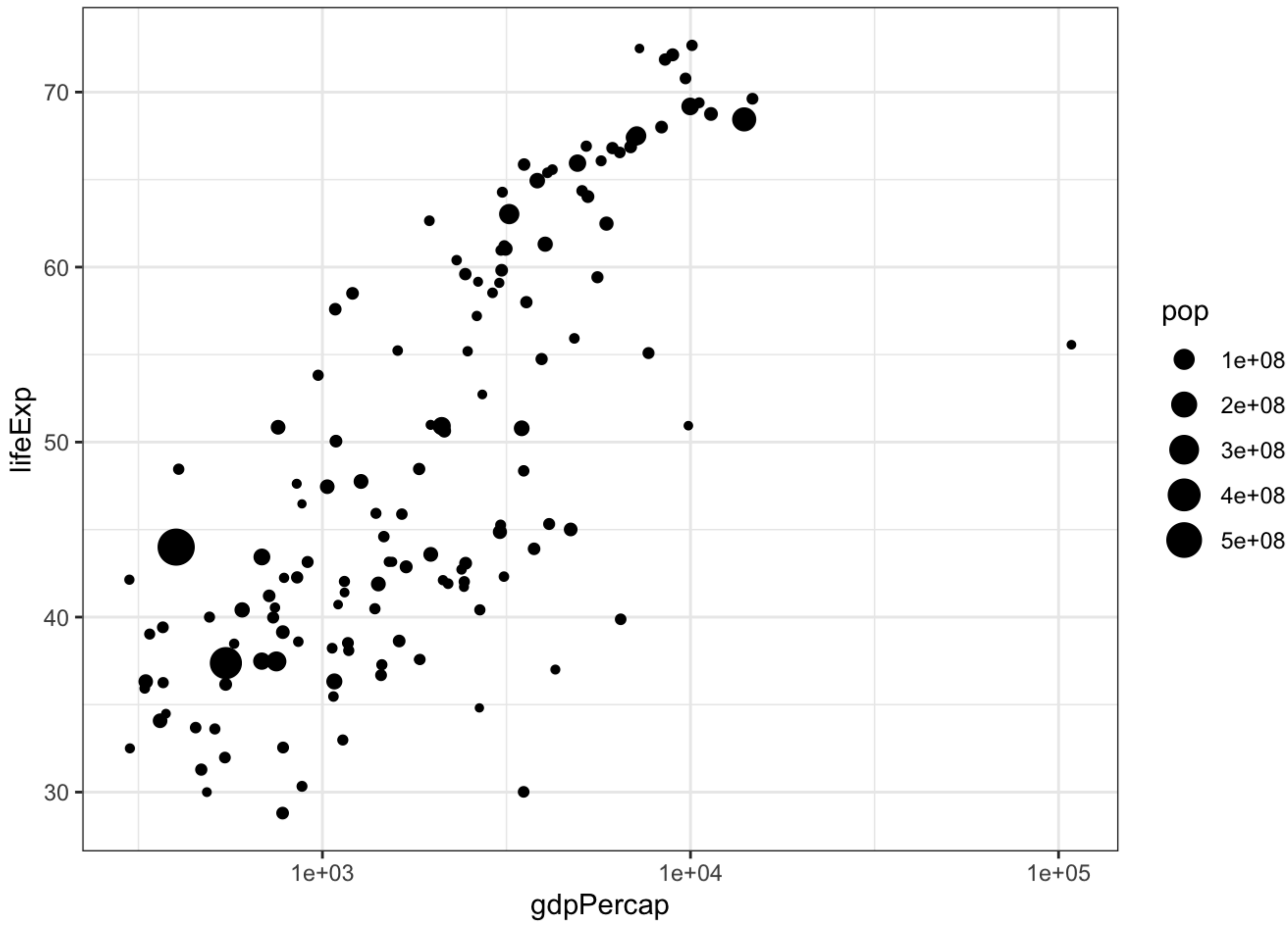
```
##  [889] 38.48000 39.48600 40.50200 41.53600 42.61400 43.76400 44.85200 46.02700
##  [897] 40.80200 42.22100 43.75300 45.67800 42.72300 45.28900 47.80800 50.22700
##  [905] 52.77300 57.44200 62.15500 66.23400 68.75500 71.55500 72.73700 73.95200
##  [913] 36.68100 38.86500 40.84800 42.88100 44.85100 46.88100 48.96900 49.35000
##  [921] 52.21400 54.97800 57.28600 59.44300 36.25600 37.20700 38.41000 39.48700
##  [929] 41.76600 43.76700 45.64200 47.45700 49.42000 47.49500 45.00900 48.30300
##  [937] 48.46300 52.10200 55.73700 59.37100 63.01000 65.25600 68.00000 69.50000
##  [945] 70.69300 71.93800 73.04400 74.24100 33.68500 35.30700 36.93600 38.48700
##  [953] 39.97700 41.71400 43.91600 46.36400 48.38800 49.90300 51.81800 54.46700
##  [961] 40.54300 42.33800 44.24800 46.28900 48.43700 50.85200 53.59900 56.14500
##  [969] 58.33300 60.43000 62.24700 64.16400 50.98600 58.08900 60.24600 61.55700
##  [977] 62.94400 64.93000 66.71100 68.74000 69.74500 70.73600 71.95400 72.80100
##  [985] 50.78900 55.19000 58.29900 60.11000 62.36100 65.03200 67.40500 69.49800
##  [993] 71.45500 73.67000 74.90200 76.19500 42.24400 45.24800 48.25100 51.25300
## [1001] 53.75400 55.49100 57.48900 60.22200 61.27100 63.62500 65.03300 66.80300
## [1009] 59.16400 61.44800 63.72800 67.17800 70.63600 73.06600 74.10100 74.86500
## [1017] 75.43500 75.44500 73.98100 74.54300 42.87300 45.42300 47.92400 50.33500
## [1025] 52.86200 55.73000 59.65000 62.67700 65.39300 67.66000 69.61500 71.16400
## [1033] 31.28600 33.77900 36.16100 38.11300 40.32800 42.49500 42.79500 42.86100
## [1041] 44.28400 46.34400 44.02600 42.08200 36.31900 41.90500 45.10800 49.37900
## [1049] 53.07000 56.05900 58.05600 58.33900 59.32000 60.32800 59.90800 62.06900
## [1057] 41.72500 45.22600 48.38600 51.15900 53.86700 56.43700 58.96800 60.83500
## [1065] 61.99900 58.90900 51.47900 52.90600 36.15700 37.68600 39.39300 41.47200
## [1073] 43.97100 46.74800 49.59400 52.53700 55.72700 59.42600 61.34000 63.78500
## [1081] 72.13000 72.99000 73.23000 73.82000 73.75000 75.24000 76.05000 76.83000
## [1089] 77.42000 78.03000 78.53000 79.76200 69.39000 70.26000 71.24000 71.52000
## [1097] 71.89000 72.22000 73.84000 74.32000 76.33000 77.55000 79.11000 80.20400
## [1105] 42.31400 45.43200 48.63200 51.88400 55.15100 57.47000 59.29800 62.00800
## [1113] 65.84300 68.42600 70.83600 72.89900 37.44400 38.59800 39.48700 40.11800
## [1121] 40.54600 41.29100 42.59800 44.55500 47.39100 51.31300 54.49600 56.86700
## [1129] 36.32400 37.80200 39.36000 41.04000 42.82100 44.51400 45.82600 46.88600
## [1137] 47.47200 47.46400 46.60800 46.85900 72.67000 73.44000 73.47000 74.08000
## [1145] 74.34000 75.37000 75.97000 75.89000 77.32000 78.32000 79.05000 80.19600
## [1153] 37.57800 40.08000 43.16500 46.98800 52.14300 57.36700 62.72800 67.73400
## [1161] 71.19700 72.49900 74.19300 75.64000 43.43600 45.55700 47.67000 49.80000
## [1169] 51.92900 54.04300 56.15800 58.24500 60.83800 61.81800 63.61000 65.48300
## [1177] 55.19100 59.20100 61.81700 64.07100 66.21600 68.68100 70.47200 71.52300
## [1185] 72.46200 73.73800 74.71200 75.53700 62.64900 63.19600 64.36100 64.95100
## [1193] 65.81500 66.35300 66.87400 67.37800 68.22500 69.40000 70.75500 71.75200
## [1201] 43.90200 46.26300 49.09600 51.44500 55.44800 58.44700 61.40600 64.13400
## [1209] 66.45800 68.38600 69.90600 71.42100 47.75200 51.33400 54.75700 56.39300
## [1217] 58.06500 60.06000 62.08200 64.15100 66.45800 68.56400 70.30300 71.68800
## [1225] 61.31000 65.77000 67.64000 69.61000 70.85000 70.67000 71.32000 70.98000
## [1233] 70.99000 72.75000 74.67000 75.56300 59.82000 61.51000 64.39000 66.60000
## [1241] 69.26000 70.41000 72.77000 74.06000 74.86000 75.97000 77.29000 78.09800
## [1249] 64.28000 68.54000 69.62000 71.10000 72.16000 73.44000 73.75000 74.63000
## [1257] 73.91100 74.91700 77.77800 78.74600 52.72400 55.09000 57.66600 60.54200
## [1265] 64.27400 67.06400 69.88500 71.91300 73.61500 74.77200 75.74400 76.44200
## [1273] 61.05000 64.10000 66.80000 66.80000 69.21000 69.46000 69.66000 69.53000
## [1281] 69.36000 69.72000 71.32200 72.47600 40.00000 41.50000 43.00000 44.10000
## [1289] 44.60000 45.00000 46.21800 44.02000 23.59900 36.08700 43.41300 46.24200
## [1297] 46.47100 48.94500 51.89300 54.42500 56.48000 58.55000 60.35100 61.72800
## [1305] 62.74200 63.30600 64.33700 65.52800 39.87500 42.86800 45.91400 49.90100
## [1313] 53.88600 58.69000 63.01200 66.29500 68.76800 70.53300 71.62600 72.77700
## [1321] 37.27800 39.32900 41.45400 43.56300 45.81500 48.87900 52.37900 55.76900
## [1329] 58.19600 60.18700 61.60000 63.06200 57.99600 61.68500 64.53100 66.91400
```

```
## [1337] 68.70000 70.30000 70.16200 71.21800 71.65900 72.23200 73.21300 74.00200
## [1345] 30.33100 31.57000 32.76700 34.11300 35.40000 36.78800 38.44500 40.00600
## [1353] 38.33300 39.89700 41.01200 42.56800 60.39600 63.17900 65.79800 67.94600
## [1361] 69.52100 70.79500 71.76000 73.56000 75.78800 77.15800 78.77000 79.97200
## [1369] 64.36000 67.45000 70.33000 70.98000 70.35000 70.45000 70.80000 71.08000
## [1377] 71.38000 72.71000 73.80000 74.66300 65.57000 67.85000 69.15000 69.18000
## [1385] 69.82000 70.97000 71.06300 72.25000 73.64000 75.13000 76.66000 77.92600
## [1393] 32.97800 34.97700 36.98100 38.97700 40.97300 41.97400 42.95500 44.50100
## [1401] 39.65800 43.79500 45.93600 48.15900 45.00900 47.98500 49.95100 51.92700
## [1409] 53.69600 55.52700 58.16100 60.83400 61.88800 60.23600 53.36500 49.33900
## [1417] 64.94000 66.66000 69.69000 71.44000 73.06000 74.39000 76.30000 76.90000
## [1425] 77.57000 78.77000 79.78000 80.94100 57.59300 61.45600 62.19200 64.26600
## [1433] 65.04200 65.94900 68.75700 69.01100 70.37900 70.45700 70.81500 72.39600
## [1441] 38.63500 39.62400 40.87000 42.85800 45.08300 47.80000 50.33800 51.74400
## [1449] 53.55600 55.37300 56.36900 58.55600 41.40700 43.42400 44.99200 46.63300
## [1457] 49.55200 52.53700 55.56100 57.67800 58.47400 54.28900 43.86900 39.61300
## [1465] 71.86000 72.49000 73.37000 74.16000 74.72000 75.44000 76.42000 77.19000
## [1473] 78.16000 79.39000 80.04000 80.88400 69.62000 70.56000 71.32000 72.77000
## [1481] 73.78000 75.39000 76.21000 77.41000 78.03000 79.37000 80.62000 81.70100
## [1489] 45.88300 48.28400 50.30500 53.65500 57.29600 61.19500 64.59000 66.97400
## [1497] 69.24900 71.52700 73.05300 74.14300 58.50000 62.40000 65.20000 67.50000
## [1505] 69.39000 70.59000 72.16000 73.40000 74.26000 75.25000 76.99000 78.40000
## [1513] 41.21500 42.97400 44.24600 45.75700 47.62000 49.91900 50.60800 51.53500
## [1521] 50.44000 48.46600 49.65100 52.51700 50.84800 53.63000 56.06100 58.28500
## [1529] 60.40500 62.49400 64.59700 66.08400 67.29800 67.52100 68.56400 70.61600
## [1537] 38.59600 41.20800 43.92200 46.76900 49.75900 52.88700 55.47100 56.94100
## [1545] 58.06100 58.39000 57.56100 58.42000 59.10000 61.80000 64.90000 65.40000
## [1553] 65.90000 68.30000 68.83200 69.58200 69.86200 69.46500 68.97600 69.81900
## [1561] 44.60000 47.10000 49.57900 52.05300 55.60200 59.83700 64.04800 66.89400
## [1569] 70.00100 71.97300 73.04200 73.92300 43.58500 48.07900 52.09800 54.33600
## [1577] 57.00500 59.50700 61.03600 63.10800 66.14600 68.83500 70.84500 71.77700
## [1585] 39.97800 42.57100 45.34400 48.05100 51.01600 50.35000 49.84900 51.50900
## [1593] 48.82500 44.57800 47.81300 51.54200 69.18000 70.42000 70.76000 71.36000
## [1601] 72.01000 72.76000 74.04000 75.00700 76.42000 77.21800 78.47100 79.42500
## [1609] 68.44000 69.49000 70.21000 70.76000 71.34000 73.38000 74.65000 75.02000
## [1617] 76.09000 76.81000 77.31000 78.24200 66.07100 67.04400 68.25300 68.46800
## [1625] 68.67300 69.48100 70.80500 71.91800 72.75200 74.22300 75.30700 76.38400
## [1633] 55.08800 57.90700 60.77000 63.47900 65.71200 67.45600 68.55700 70.19000
## [1641] 71.15000 72.14600 72.76600 73.74700 40.41200 42.88700 45.36300 47.83800
## [1649] 50.25400 55.76400 58.81600 62.82000 67.66200 70.67200 73.01700 74.24900
## [1657] 43.16000 45.67100 48.12700 51.63100 56.53200 60.76500 64.40600 67.04600
## [1665] 69.71800 71.09600 72.37000 73.42200 32.54800 33.97000 35.18000 36.98400
## [1673] 39.84800 44.17500 49.11300 52.92200 55.59900 58.02000 60.30800 62.69800
## [1681] 42.03800 44.07700 46.02300 47.76800 50.10700 51.38600 51.82100 50.82100
## [1689] 46.10000 40.23800 39.19300 42.38400 48.45100 50.46900 52.35800 53.99500
## [1697] 55.63500 57.67400 60.36300 62.35100 60.37700 46.80900 39.98900 43.48700
```

The dataset contains information on each country in the sampled year, its continent, life expectancy, population, and GDP per capita. *(pr indbygger)*

Let's plot all the countries in 1952.

```
theme_set(theme_bw())  # set theme to white background for better visibility
ggplot(subset(gapminder, year == 1952), aes(gdpPercap, lifeExp, size = pop)) +
  geom_point() +
  scale_x_log10()
```



The plot here visualises how the *GDP per capita* seems to model the *life expectancy* of the citisents - the greater the GDP the longer the life expectancy. However we clearly see an outlier to the right, not following the regular spread of the data…

lets find out:

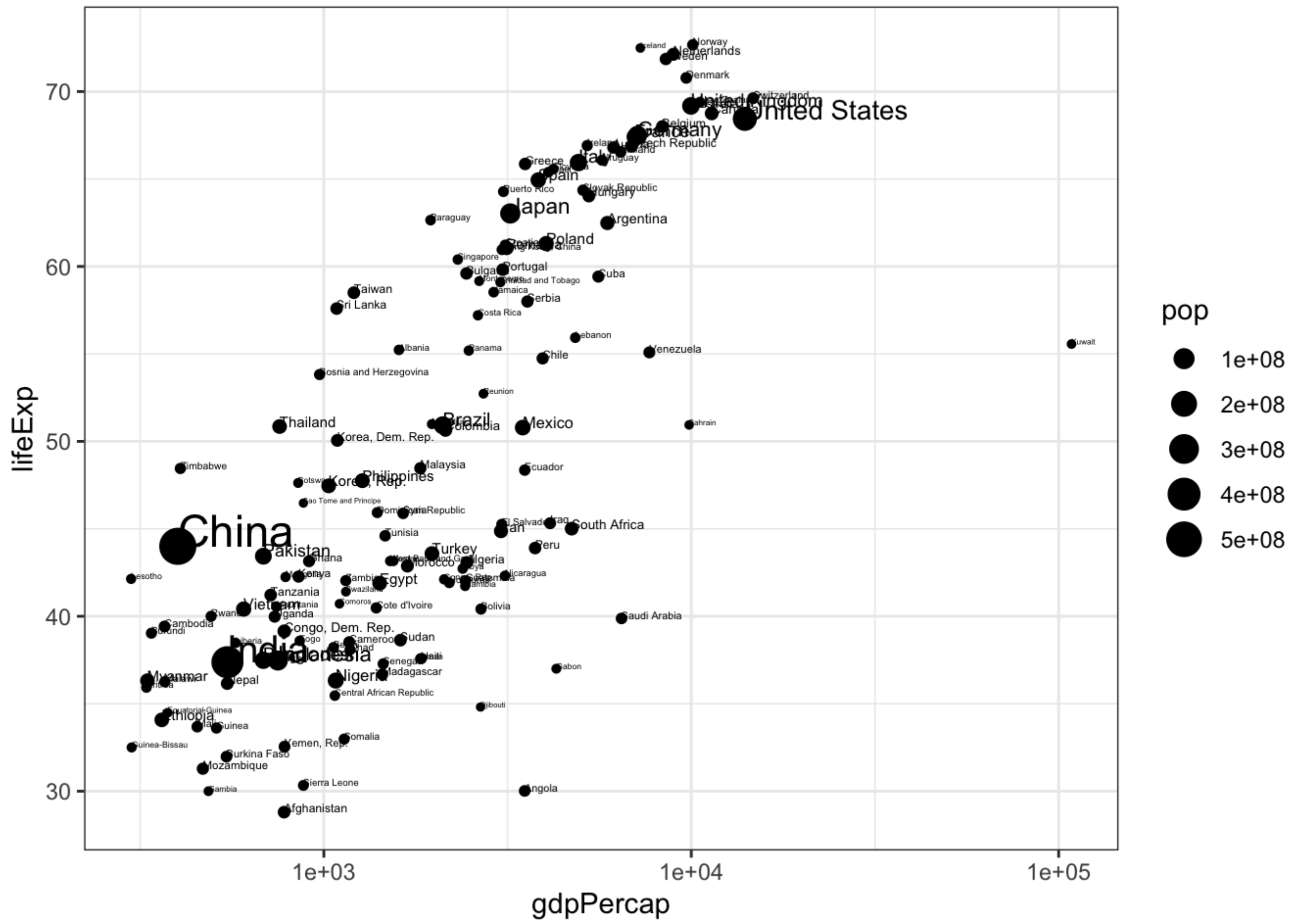
1. *Why does it make sense to have a log10 scale on x axis?*

A logarithmic scale shows exponential growth on a graph. It's a nonlinear scale that's frequently used for analyzing a large range of quantities compactly. When applied here, it is useful to visualise the economic growth of the countries as economic growth are exponential, due to its inflationary nature. This is usually done when the x axis concerns some kind of time measure, which is not the case atm, but when we animate the plot and time is a parameter of interest, the log scale becomes useful to visualise the growth as somewhat linear with time - despite its exponential nature.

something with the outlier

2. *Who is the outlier (the richest country in 1952 - far right on x axis)?* This is first figured out visually:

```
theme_set(theme_bw())  # set theme to white background for better visibility
ggplot(subset(gapminder, year == 1952), aes(gdpPercap, lifeExp, size = pop, label = c
ountry)) +
  geom_point() +
  geom_text(hjust=0, vjust=0) +
  scale_x_log10()
```



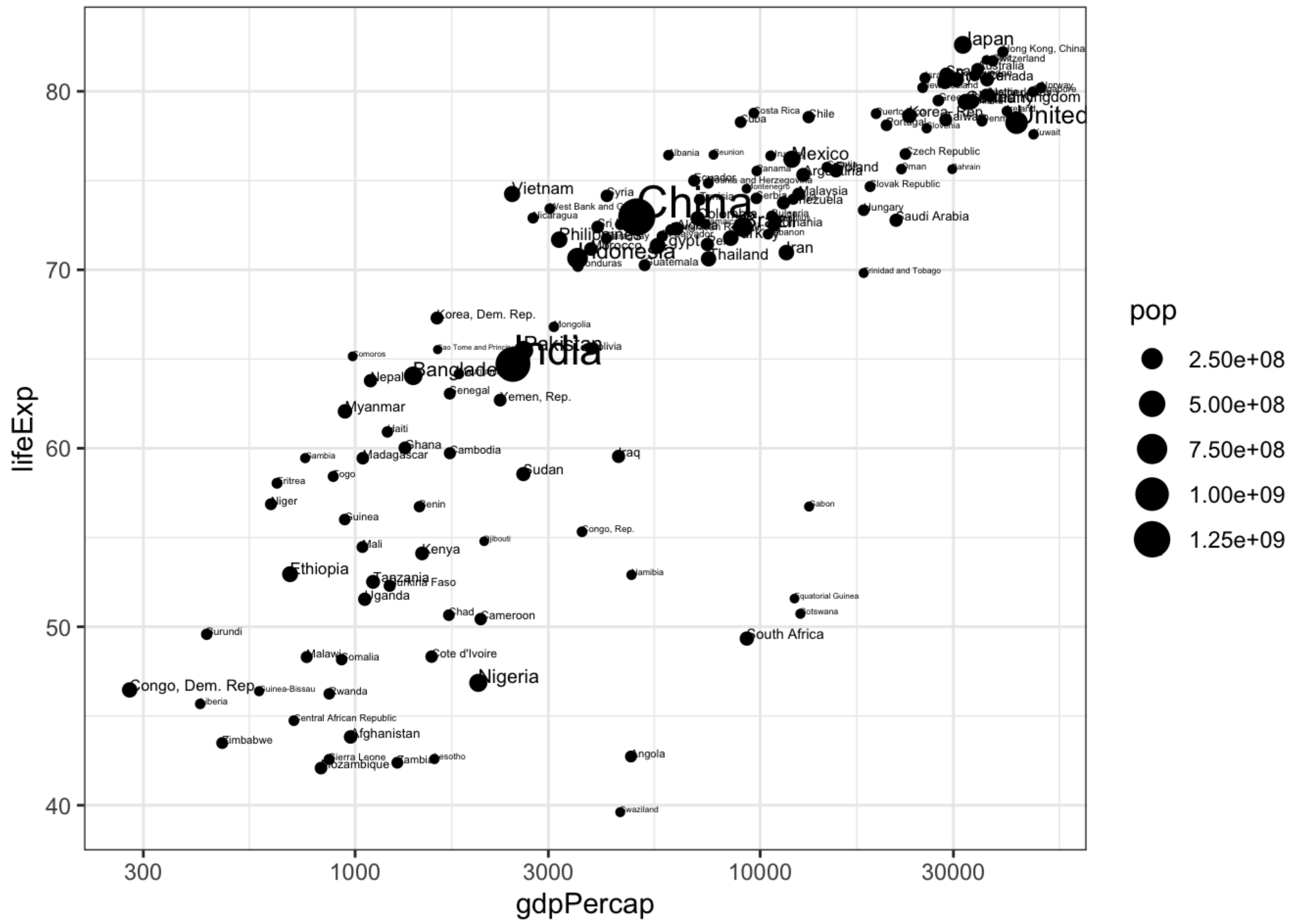
After having added the name of the country to each point, the outlier seems to be **Kuwait**. Another way this could be detected was by looking at the raw data, and find the country with the highest GDP in 1952:

```
dat1952 <- gapminder %>%
  filter(year == "1952") %>%
  mutate(country = as.character(country))

target_outlier <- dat1952$country[(which.max(dat1952$gdpPercap))]
target_outlier
```

```
## [1] "Kuwait"
```

Next, you can generate a similar plot for 2007 and compare the differences *Even though it is a bit messy I like to have the names on the dots, so they stay :)*

```
ggplot(subset(gapminder, year == 2007), aes(gdpPercap, lifeExp, size = pop, label = c
ountry)) +
  geom_point() +
  geom_text(hjust=0, vjust=0) +
  scale_x_log10()
```



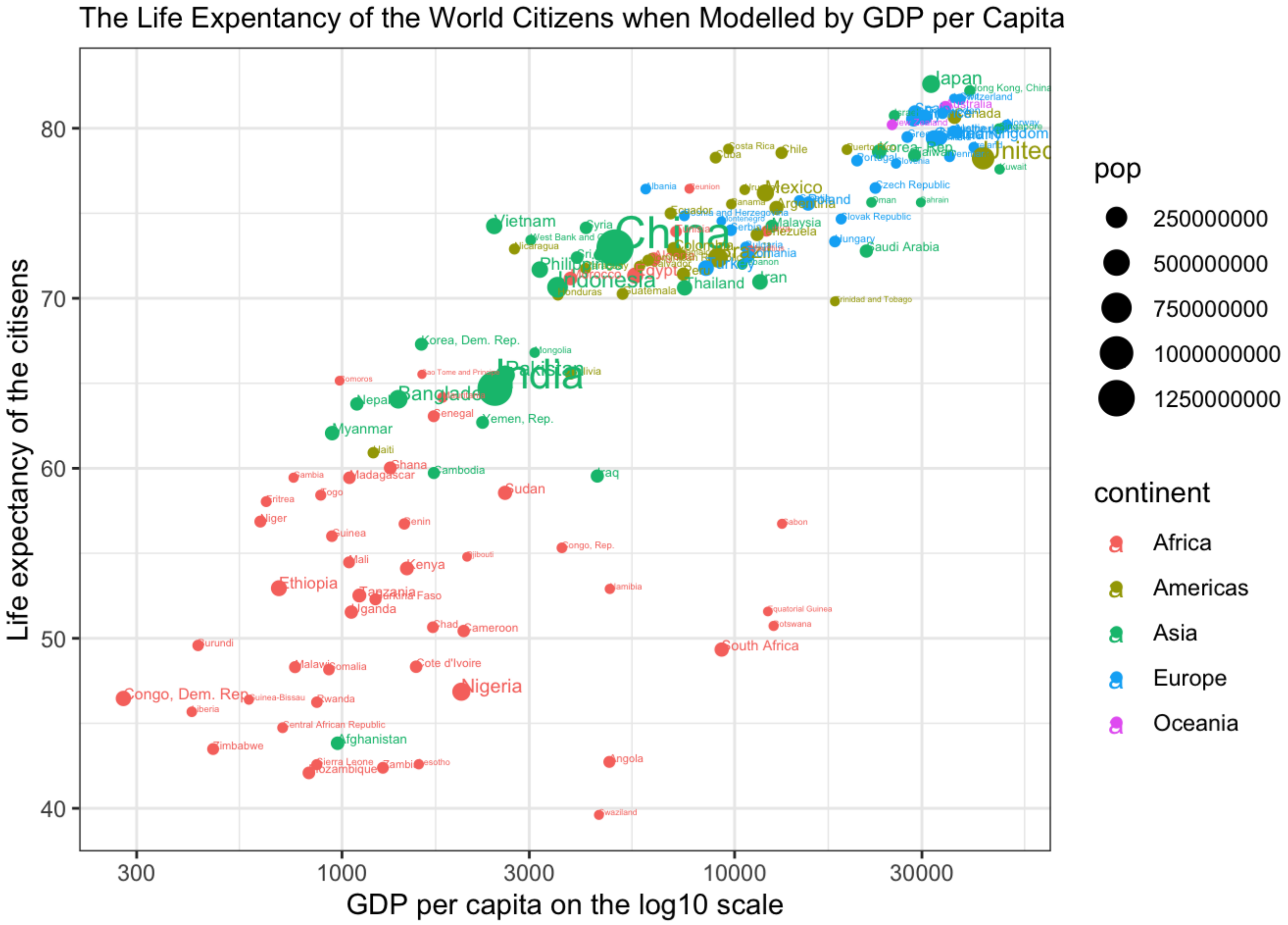
...

The black bubbles are a bit hard to read, the comparison would be easier with a bit more visual differentiation.

Tasks:

3. *Differentiate the **continents** by color, and fix the axis labels and units to be more legible

```
options(scipen=10000) ### to make it non-scientific notation and thus more legible

ggplot(subset(gapminder, year == 2007), aes(gdpPercap, lifeExp, size = pop, label = c
ountry, colour = continent)) +
  geom_point() +
  geom_text(hjust=0, vjust=0)+
  scale_x_log10()+
  labs(subtitle = "The Life Expentancy of the World Citizens when Modelled by GDP per
Capita ", x = "GDP per capita on the log10 scale", y = "Life expectancy of the citise
ns")
```



4. *What are the five richest countries in the world in 2007?* lets find out by the similar method as before:

```
#### first lets find the three highest values
datGDP <- gapminder %>%
  filter(year == "2007") %>%
  select(gdpPercap)

top_three <- datGDP%>%
  top_n(3)
```

```
## Selecting by gdpPercap
```

```
#### lets then find out which country holds these values:
#### first I filter the data so we are working with all variables, but only from 2007
dat2007 <- gapminder %>%
  filter(year == "2007") %>%
   mutate(country = as.character(country))

#### then lets find the three richest:
richest <- dat2007$country[which(grepl(top_three[1,1], dat2007$gdpPercap))]
second_richest <- dat2007$country[which(grepl(top_three[2,1], dat2007$gdpPercap))]
third_richest <- dat2007$country[which(grepl(top_three[3,1], dat2007$gdpPercap))]

### so the three richest countries is:
richest
```

```
## [1] "Kuwait"
```

```
second_richest
```

```
## [1] "Norway"
```

```
third_richest
```

```
## [1] "Singapore"
```

Kuwait, Norway and Singapore are thus the richest three countries as measured by GDP per capita in 2007, which by visual inspection appears to match the graph.

Make it move!

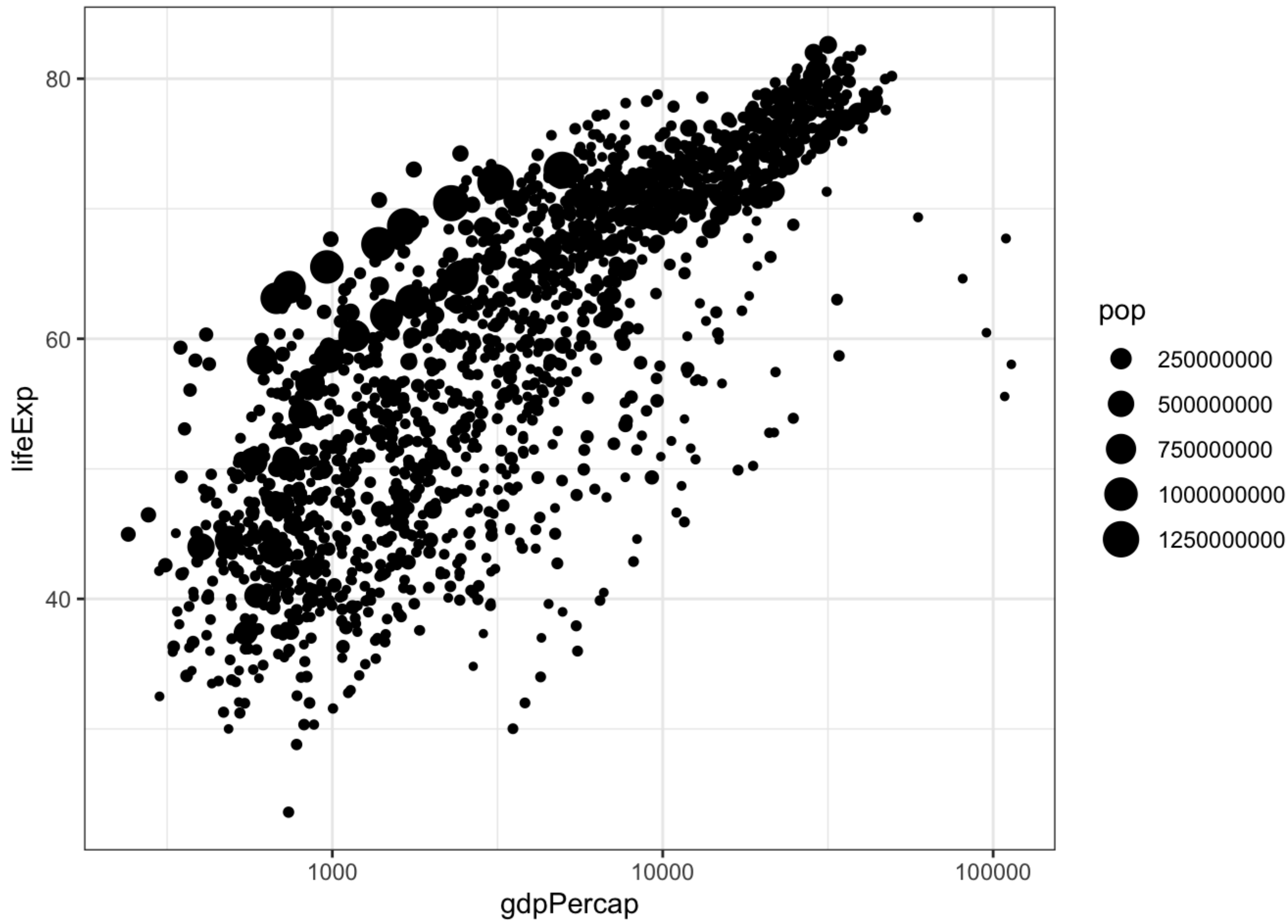
The comparison would be easier if we had the two graphs together, animated. We have a lovely tool in R to do this: the `gganimate` package. Beware that there may be other packages your operating system needs in order to glue interim images into an animation or video. Read the messages when installing the package.

Also, there are *two* ways of animating the gapminder ggplot.

Option 1: Animate using transition_states()

The first step is to create the object-to-be-animated

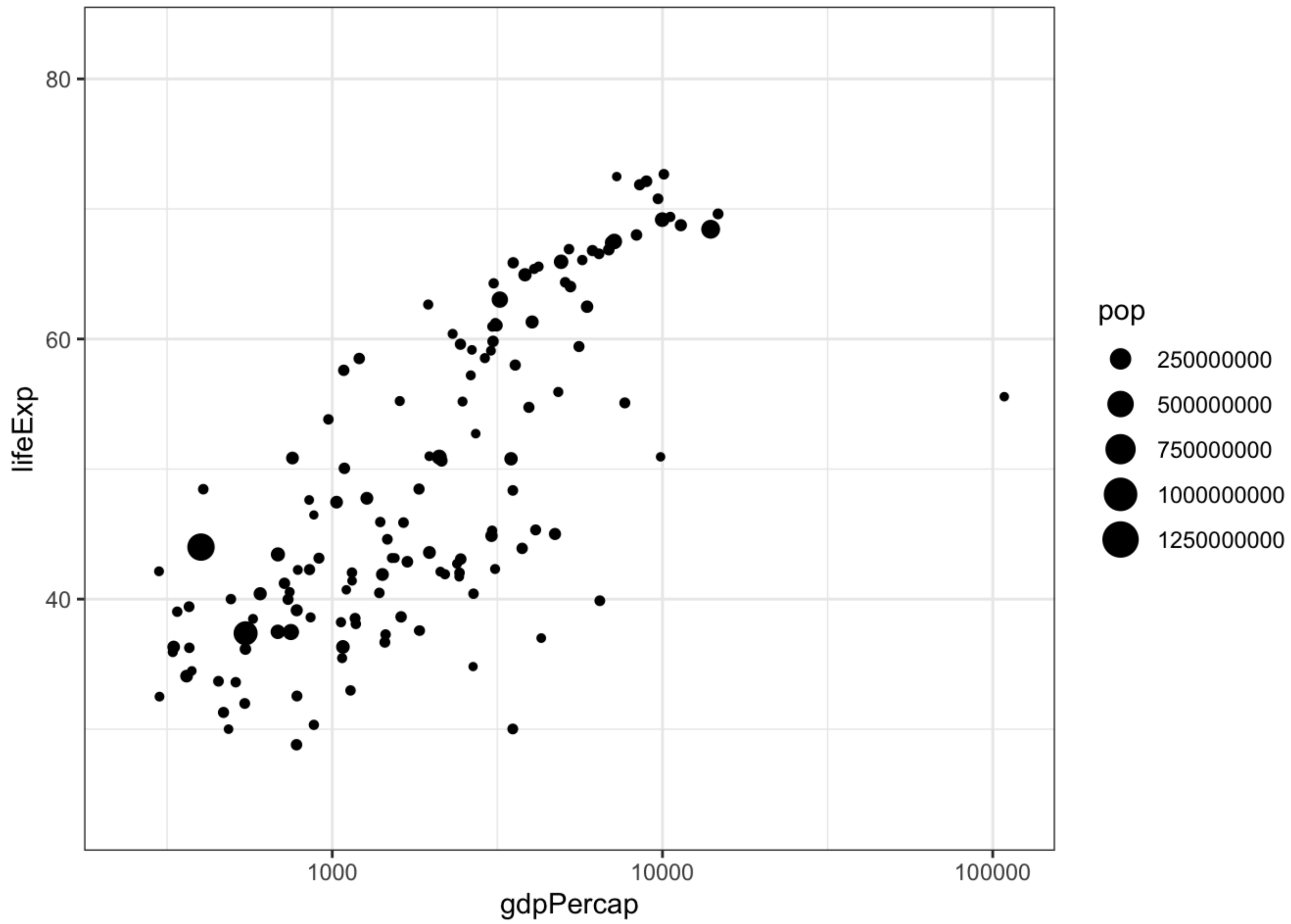
```
options(scipen=10000)

anim <- ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop)) +
  geom_point() +
  scale_x_log10()  # convert x to log scale
anim
```



...

This plot collates all the points across time. The next step is to split it into years and animate it. This may take some time, depending on the processing power of your computer (and other things you are asking it to do). Beware that the animation might appear in the bottom right 'Viewer' pane, not in this rmd preview. You need to `knit` the document to get the visual inside an html file.

```
anim + transition_states(year,
                         transition_length = 1,
                         state_length = 1)
```



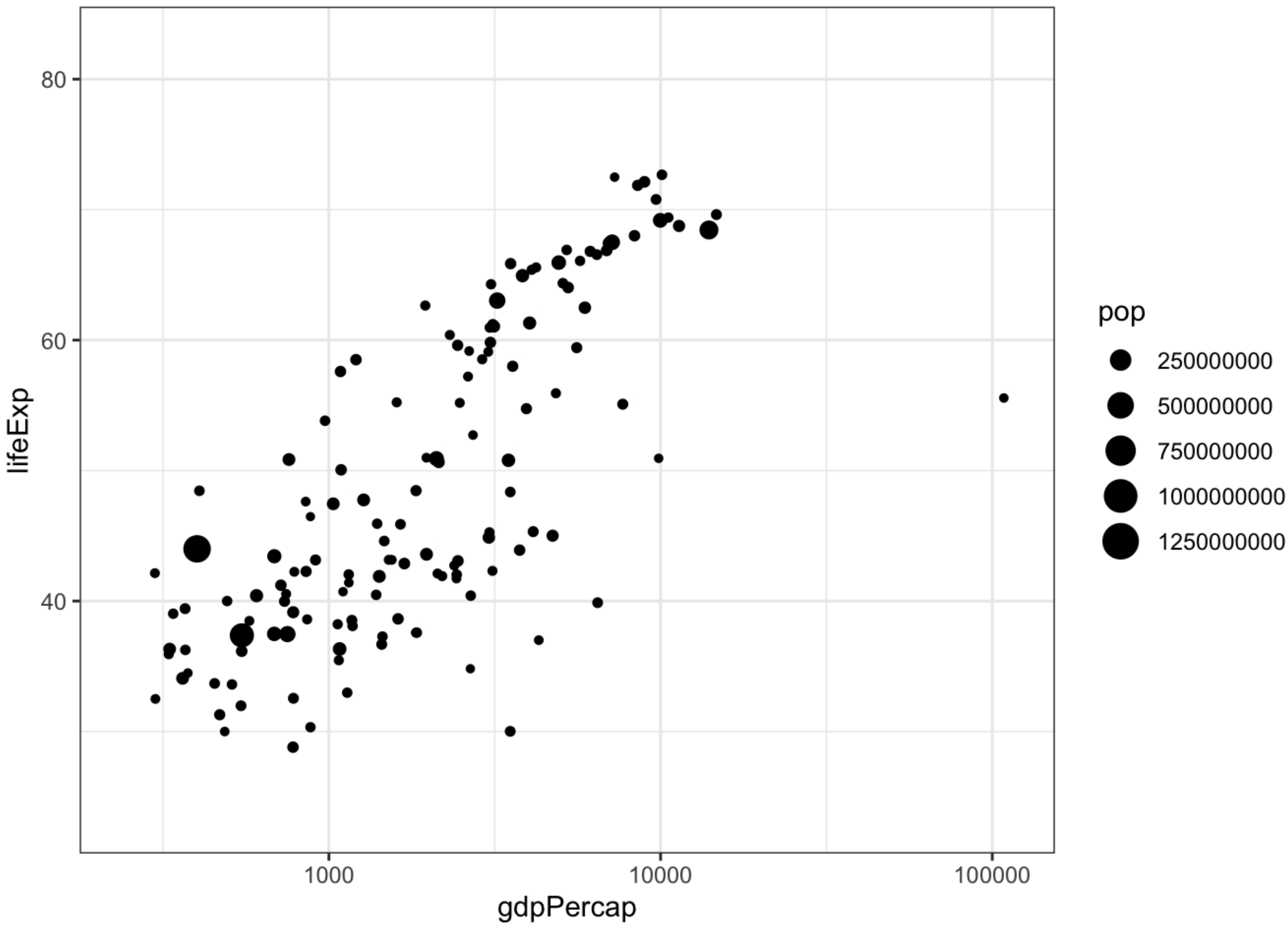
…

Notice how the animation moves jerkily, ‘jumping’ from one year to the next 12 times in total. This is a bit clunky, which is why it’s good we have another option.

Option 2 Animate using transition_time()

This option smoothes the transition between different ‘frames’, because it interpolates and adds transitional years where there are gaps in the timeseries data.

```
anim2 <- ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop)) +
  geom_point() +
  scale_x_log10() + # convert x to log scale
  transition_time(year)
anim2
```



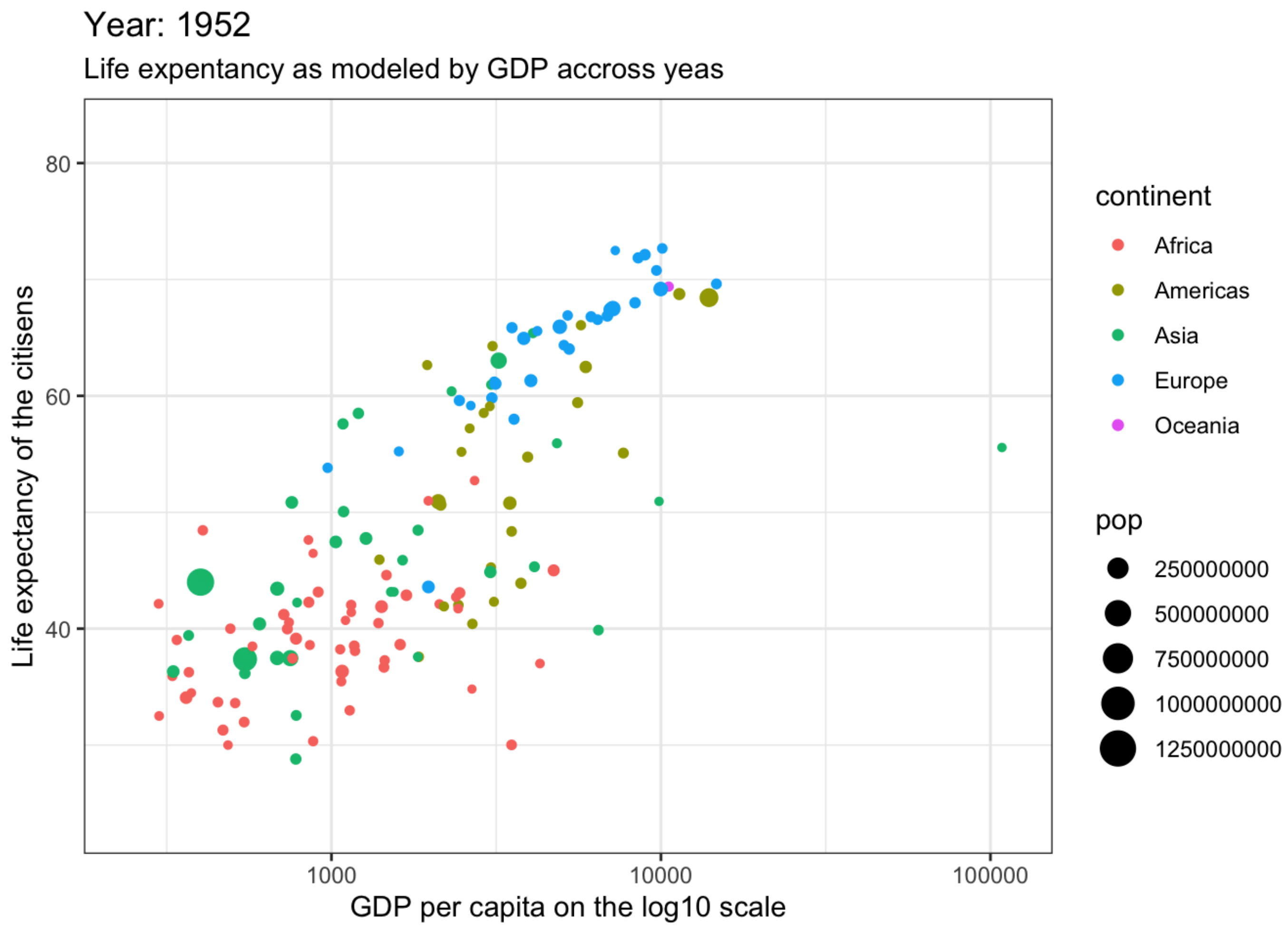
The much smoother movement in Option 2 will be much more noticeable if you add a title to the chart, that will page through the years corresponding to each frame.

Now, choose one of the animation options and get it to work. You may need to troubleshoot your installation of `gganimate` and other packages

5. *Can you add a title to one or both of the animations above that will change in sync with the animation?*

i choose to do it in the latter animation_

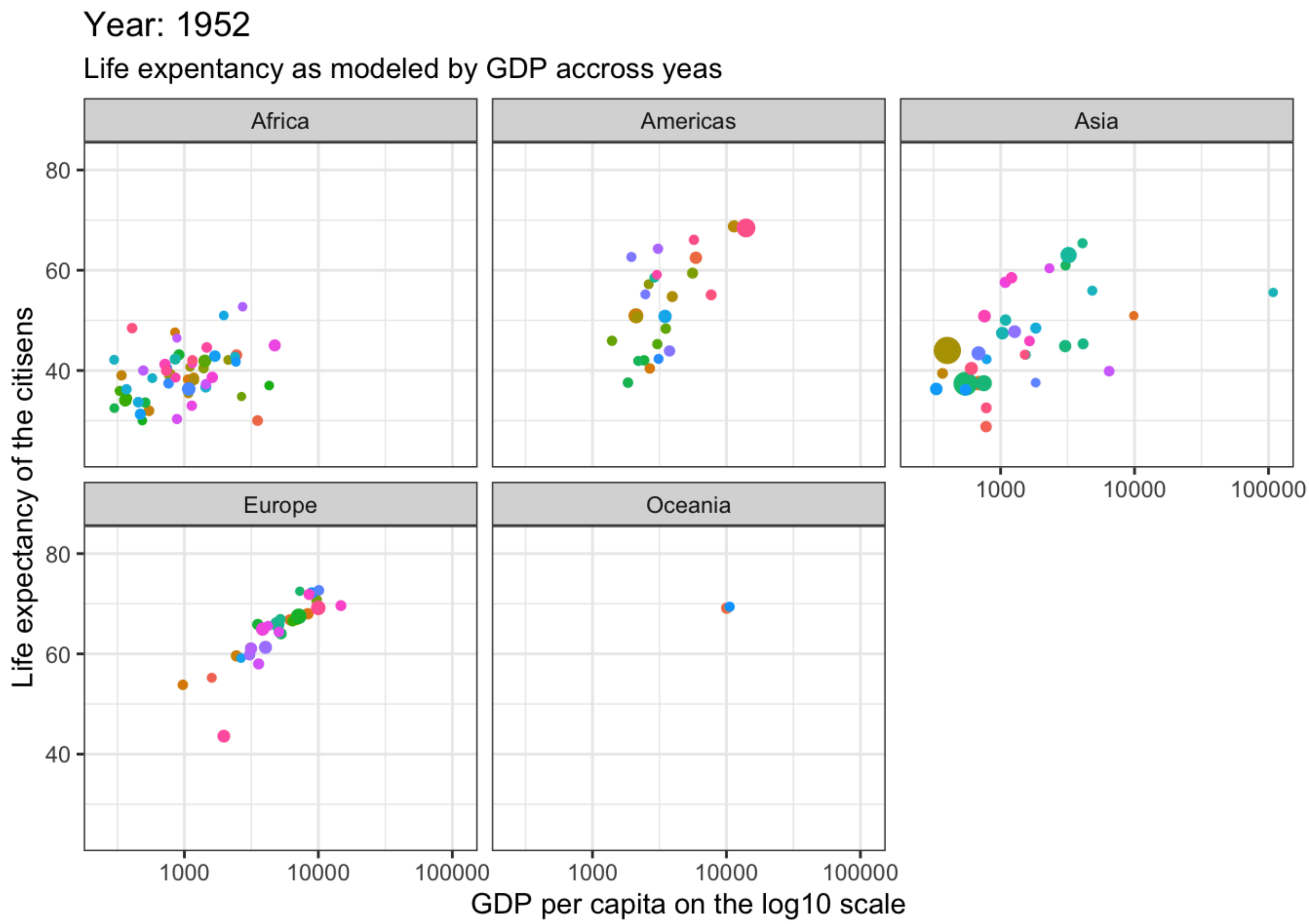
```
options(scipen=10000)

anim_w_title <- ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop, colour = contin
ent)) +
  geom_point() +
  scale_x_log10() + # convert x to log scale
  labs(title = "Year: {closest_state}", subtitle = "Life expentancy as modeled by GDP
accross yeas", x = "GDP per capita on the log10 scale", y = "Life expectancy of the c
itisens") + # the {} adds the year transition as the title
  transition_states(year, transition_length = 3, state_length = 1) +
  enter_fade() +
  exit_fade()

anim_w_title
```



In this plot the title changes with the transition of the points in the graph, and reveals in which year the data is from. Additionally the labels of the axes explains what the information x and y hold. The colour of the points reveals which continent the country is from, while the size of the point the population size. Both the axes and the population size has been made into whole numbers to avoid scientific notions, and hereby make the plot more readable.

Another way this can be visualised is by facetwrapping it for continents:

```
anim_for_facet <- ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop, colour = coun
try)) +
  geom_point(show.legend = FALSE) +
  scale_x_log10() + # convert x to log scale
  labs(title = "Year: {closest_state}", subtitle = "Life expentancy as modeled by GDP
accross yeas", x = "GDP per capita on the log10 scale", y = "Life expectancy of the c
itisens") + # the {} adds the year transition as the title
  transition_states(year, transition_length = 3, state_length = 1) +
  enter_fade() +
  exit_fade()


anim_for_facet + facet_wrap(~continent) +
  transition_time(year) +
  labs(title = "Year: {frame_time}")
```



6. *Can you made the axes' labels and units more readable? Consider expanding the abreviated lables as well as the scientific notation in the legend and x axis to whole numbers.*

This has all been done above, by the use of the code snippet: **options(scipen=100000)**. However one should remember that the values on the x axes are still on the log10 scale**.**

7. *Come up with a question you want to answer using the gapminder data and write it down. Then, create a data visualisation that answers the question and explain how your visualization answers the question.*

My Questions:

1) Which country had the shortest life expectancy in the earliest year (1952) versus the latest year (2007)?

2) Which 3 countries have on average had the lowest life expectancy across all years?

Lets start with the latter, and find the lowest life expectancy across all years:

```
# making a df only holding the mean of life expentancy and country
mean_life_dat <- gapminder %>%
  group_by(country) %>%
  summarise_at(vars(lifeExp), list(Mean_life_exp = mean)) %>%
  mutate(country = as.character(country))


minimum <- sort(mean_life_dat$Mean_life_exp,decreasing=F)[1:3]


#### indexing using the three smallest values
first_shortest <- mean_life_dat$country[which(grepl(minimum[1], mean_life_dat$Mean_li
fe_exp))]
second_shortest <- mean_life_dat$country[which(grepl(minimum[2], mean_life_dat$Mean_l
ife_exp))]
third_shortest <- mean_life_dat$country[which(grepl(minimum[3], mean_life_dat$Mean_li
fe_exp))]


#### THE THREE COUNTRIES THAT ACROSS ALL TIMES HAVE THE LOWEST LIFE EXPECTANCY
first_shortest
```

```
## [1] "Sierra Leone"
```

```
second_shortest
```

```
## [1] "Afghanistan"
```

```
third_shortest
```

```
## [1] "Angola"
```

I can thus conclude that across all time, **Sierra Leone, Afghanistan and Zimbabwe** have the shortest life expectancy, respectively.

Now to the other question: *Which country had the shortest and longest life expectancy in the earliest year (1952) versus the latest year (2007)?*

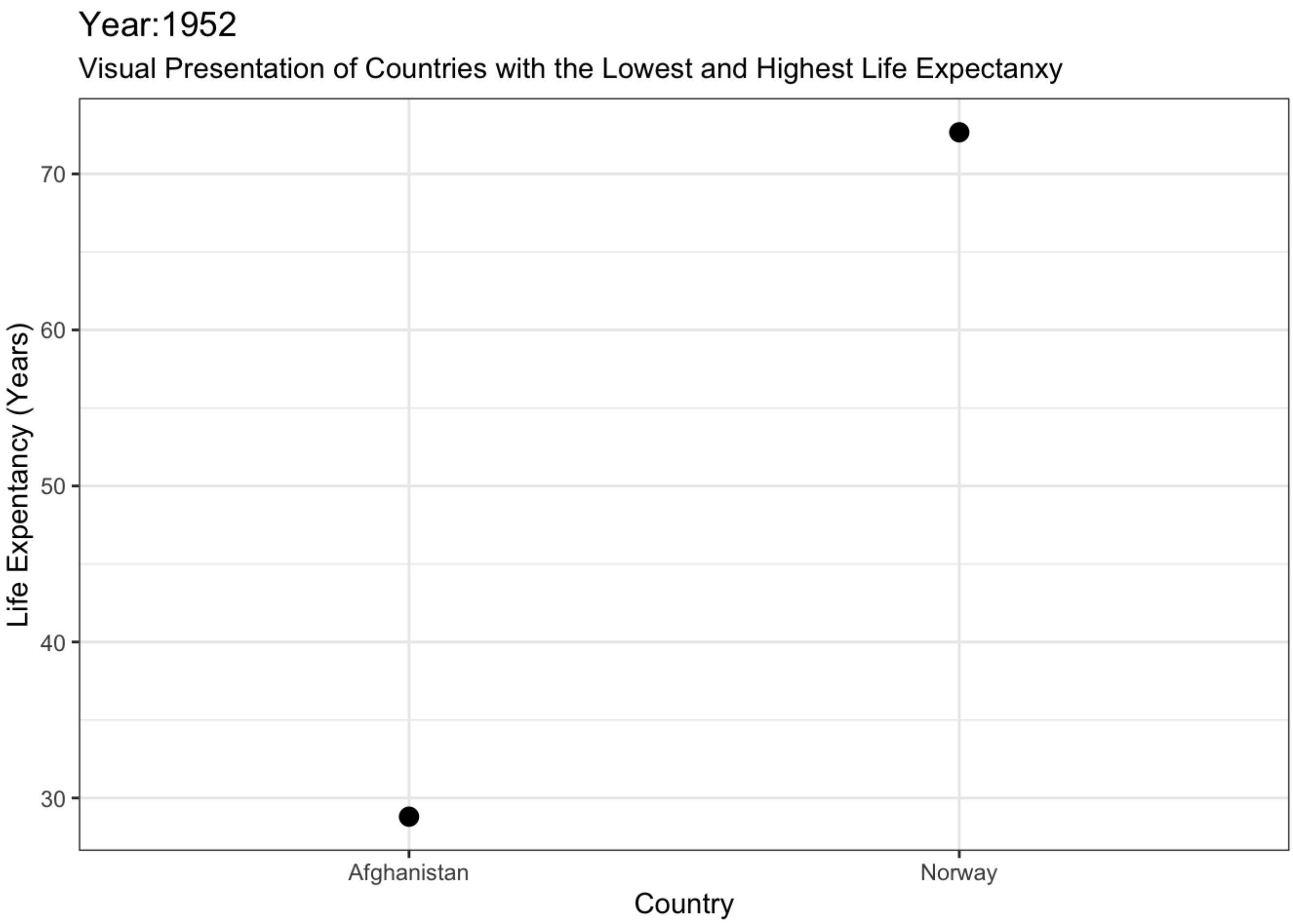
just to try, lets now make a function that can take in the year and find the shortest life expectancy and plot them up against one another:

```r
year_function <- function(year_arg) {
  ### fixing the given data
  dat <- gapminder %>%
    mutate(country = as.character(country)) %>%
    filter(year == year_arg)

  ### finding the country with shortest life expentancy
  short_country <- dat$country[(which.min(dat$lifeExp))]
  long_country <- dat$country[(which.max(dat$lifeExp))]

  ### finding the actual value
  min <- min(dat$lifeExp)
  max <- max(dat$lifeExp)

    # return the values
    nice_results <- tibble("Country" = c(short_country, long_country), "Life Expectan
cy in Years" = c(min, max))
    print(nice_results)

  # LETS now plot the values - to do so they must be combined in a df:
      country_list <- c(short_country, long_country)
      value_list <- c(min, max)
      df = data.frame(country_list, value_list)

  #### plotting both values in one plot
plot <- ggplot(df, aes(country_list, value_list)) +
    geom_point( size = 3) +
    labs(subtitle = "Visual Presentation of Countries with the Lowest and Highest Lif
e Expectanxy", x = "Country", y = "Life Expentancy (Years)") +
    ggtitle(paste0('Year:', year_arg)) +
    theme_bw()

print(plot)
}
```

using the function:

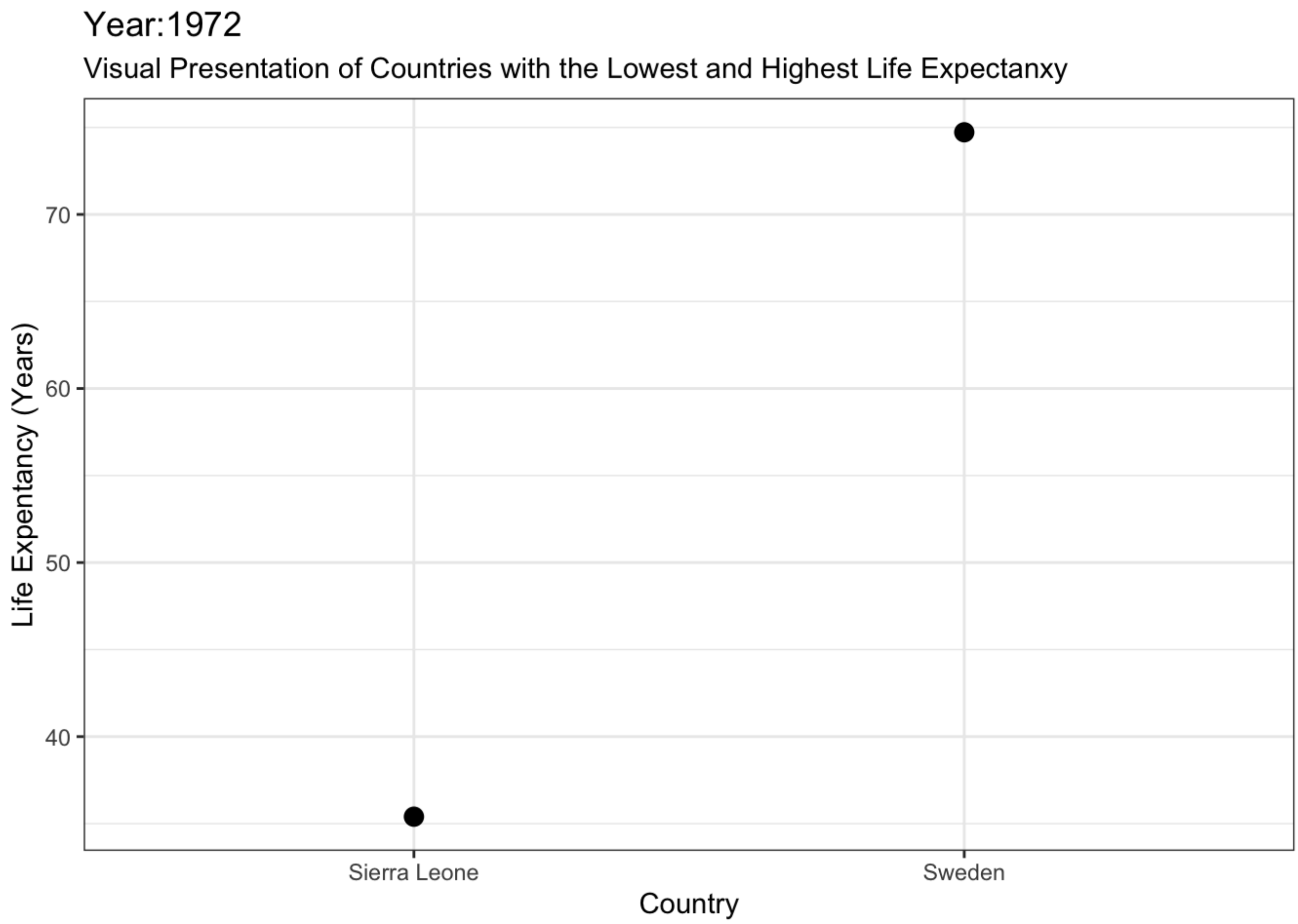
```r
year_function(1952)
```

```
## # A tibble: 2 × 2
##   Country     `Life Expectancy in Years`
##   <chr>                            <dbl>
## 1 Afghanistan                       28.8
## 2 Norway                            72.7
```



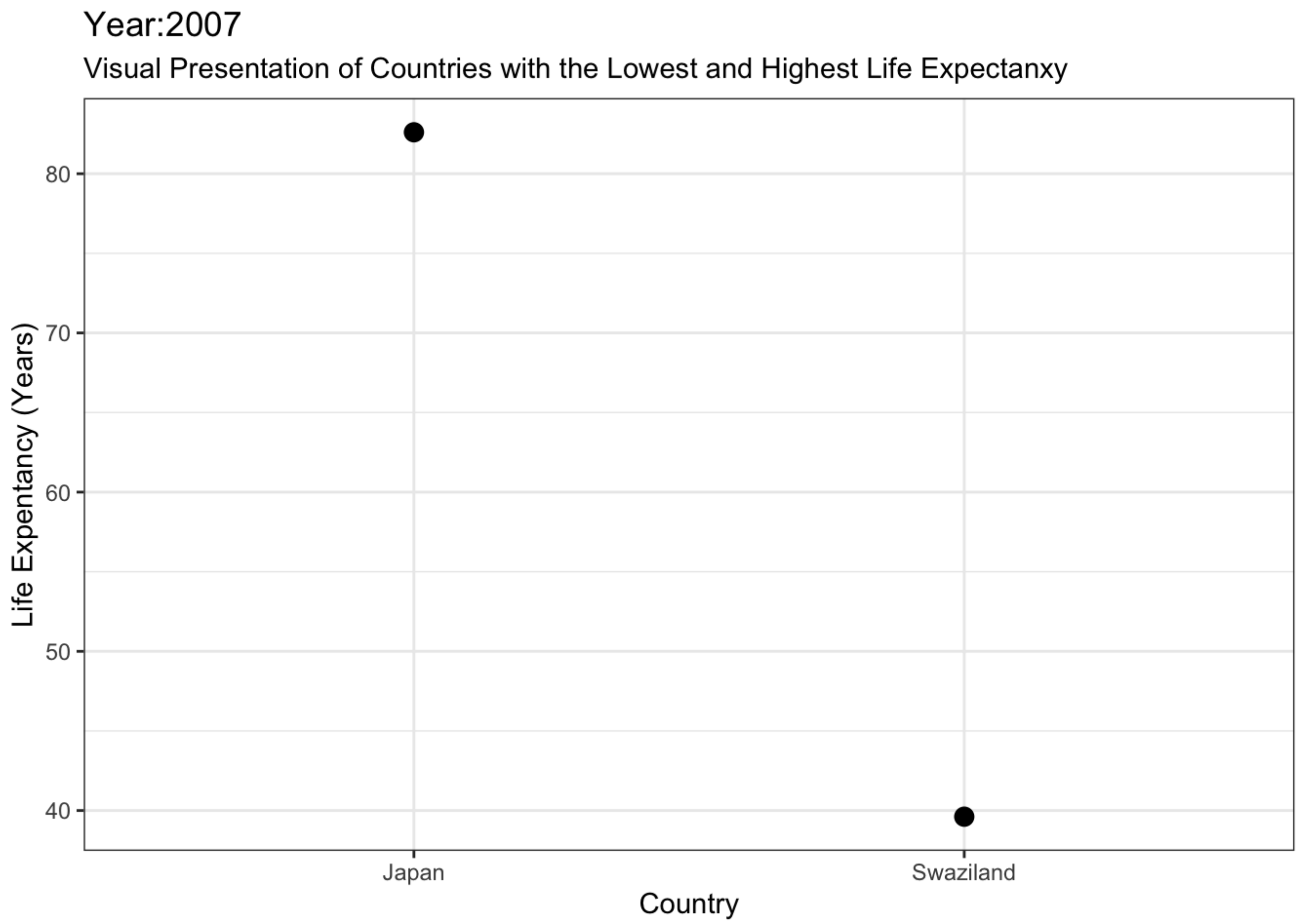

```
year_function(1972)
```

```
## # A tibble: 2 × 2
##   Country      `Life Expectancy in Years`
##   <chr>                             <dbl>
## 1 Sierra Leone                       35.4
## 2 Sweden                             74.7
```




```
year_function(2007)
```

```
## # A tibble: 2 × 2
##   Country   `Life Expectancy in Years`
##   <chr>                          <dbl>
## 1 Swaziland                       39.6
## 2 Japan                           82.6
```



this answers the question, by showing how the countries with the shortest and longest life expectancy in 1952 were Afghanistan (28,801) and Norway (72,670), whereas in 2007 Swaziland (39.613) has the shortest and Japan (82.603) the longest life expectancy. As shown above, the function can find these values to any given year in the df, not just the earliest and latest.

okay this is alright, but it isn't moving - lets instead try to make a animated ggplot, that show us the countries holding the highest and lowest life expectancy for each year.

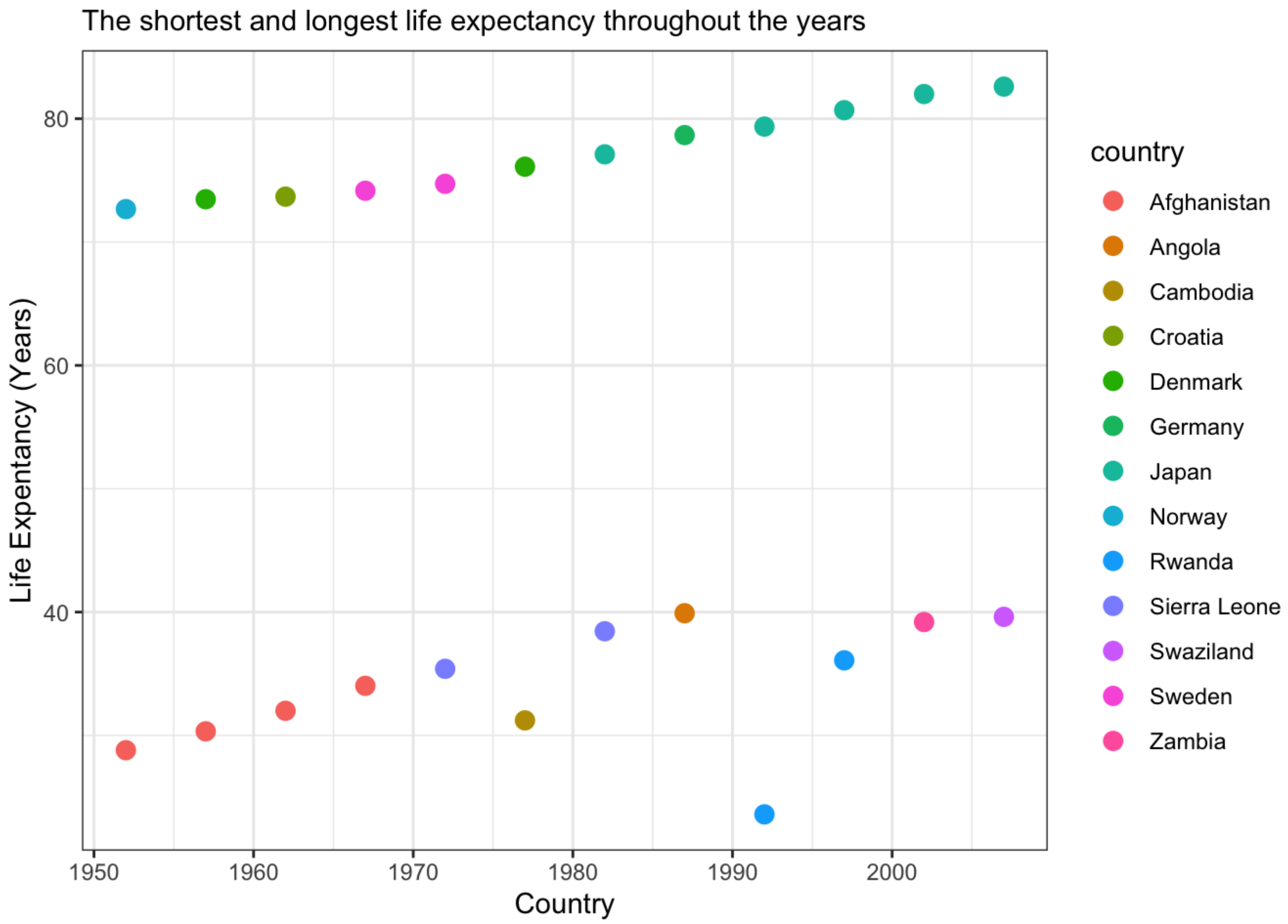
Animated mins and maxs.

```
#### lets first find the country with the shortest and longest life expectancy and th
ose values like above:
df_anim <- gapminder %>%
    group_by(year) %>%
    summarise(min_exp = min(lifeExp),
              max_exp = max(lifeExp))

gapminder$country <- as.character(gapminder$country)

for (i in 1:12) {
  df_anim$max_country[i] <- gapminder$country[which(gapminder$lifeExp == df_anim$max_
exp[i])]
  df_anim$min_country[i] <- gapminder$country[which(gapminder$lifeExp == df_anim$min_
exp[i])]
}


#### selecting the columns i wanna work with and saving it in df
df_country <- df_anim %>%
  select(year, max_country, min_country)

df_value <- df_anim %>%
  select(year, max_exp, min_exp)

##### now getting them into long format
values <- df_value %>%
  pivot_longer(cols = !year) %>%
  select(year, value) %>%
  rename(life_exp = value)

countries <- df_country %>%
  pivot_longer(cols = !year) %>%
  select(year, value) %>%
  rename(country = value,
         extra_year = year)


data_combined <- cbind(countries, values) %>%
  select(year, country, life_exp)

### now we we have data that can be worked with - could this have been done smarter,
 probably .. :D
## but as we can see we have the min and max life expectancy for each year, and the c
ountry it belongs to.
head(data_combined)
```

```
##   year     country life_exp
## 1 1952      Norway   72.670
## 2 1952 Afghanistan   28.801
## 3 1957     Denmark   73.470
## 4 1957 Afghanistan   30.332
## 5 1962     Croatia   73.680
## 6 1962 Afghanistan   31.997
```

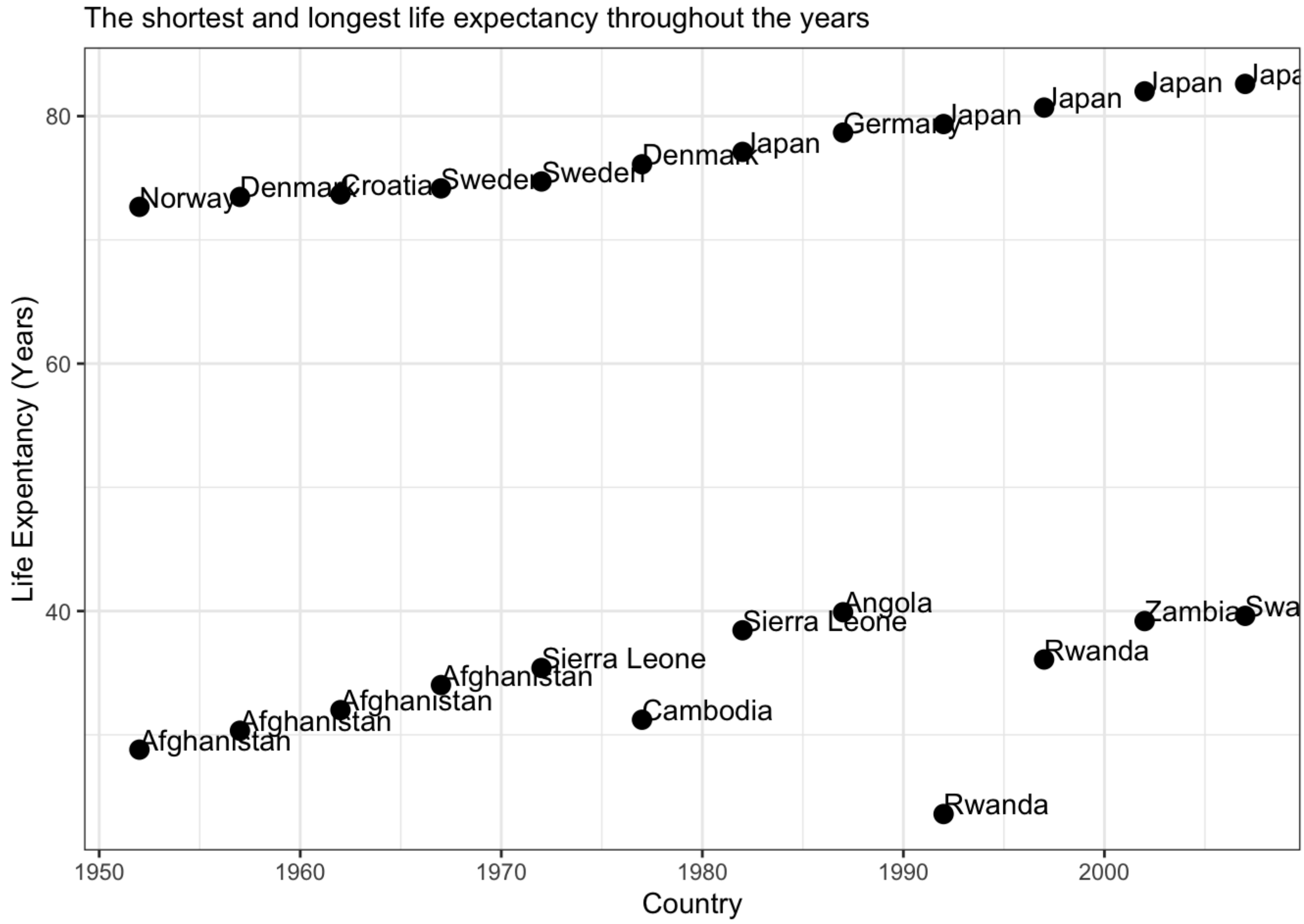
Plot Time

```
## lets first make a regular plot

ggplot(data_combined, aes(x = year, y = life_exp, col = country)) +
    geom_point( size = 3) +
    labs(subtitle = "The shortest and longest life expectancy throughout the years",
 x = "Country", y = "Life Expentancy (Years)") +
    #gtitle(paste0('Year:')) +
    theme_bw()
```



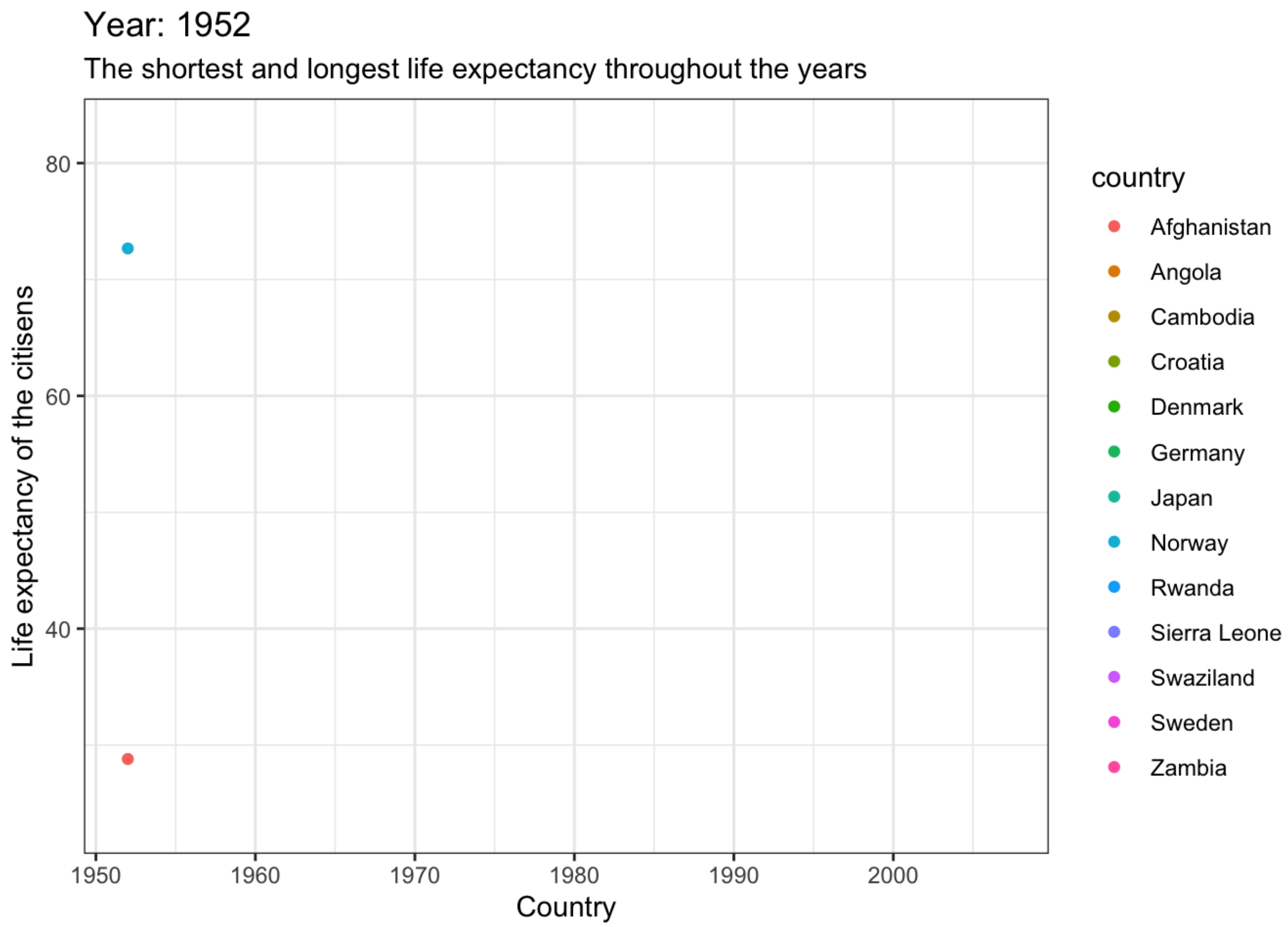
The colour coding kind of scams you, since Rwanda and Norway look very similiar. Lets try to label the points instead:

```
ggplot(data_combined, aes(x = year, y = life_exp, label = country)) +
    geom_point( size = 3) +
    geom_text(hjust=0, vjust=0) +
    labs(subtitle = "The shortest and longest life expectancy throughout the years",
 x = "Country", y = "Life Expentancy (Years)") +
    #gtitle(paste0('Year:')) +
    theme_bw()
```



The above plots are not to bad, but it could be fun if they moved.

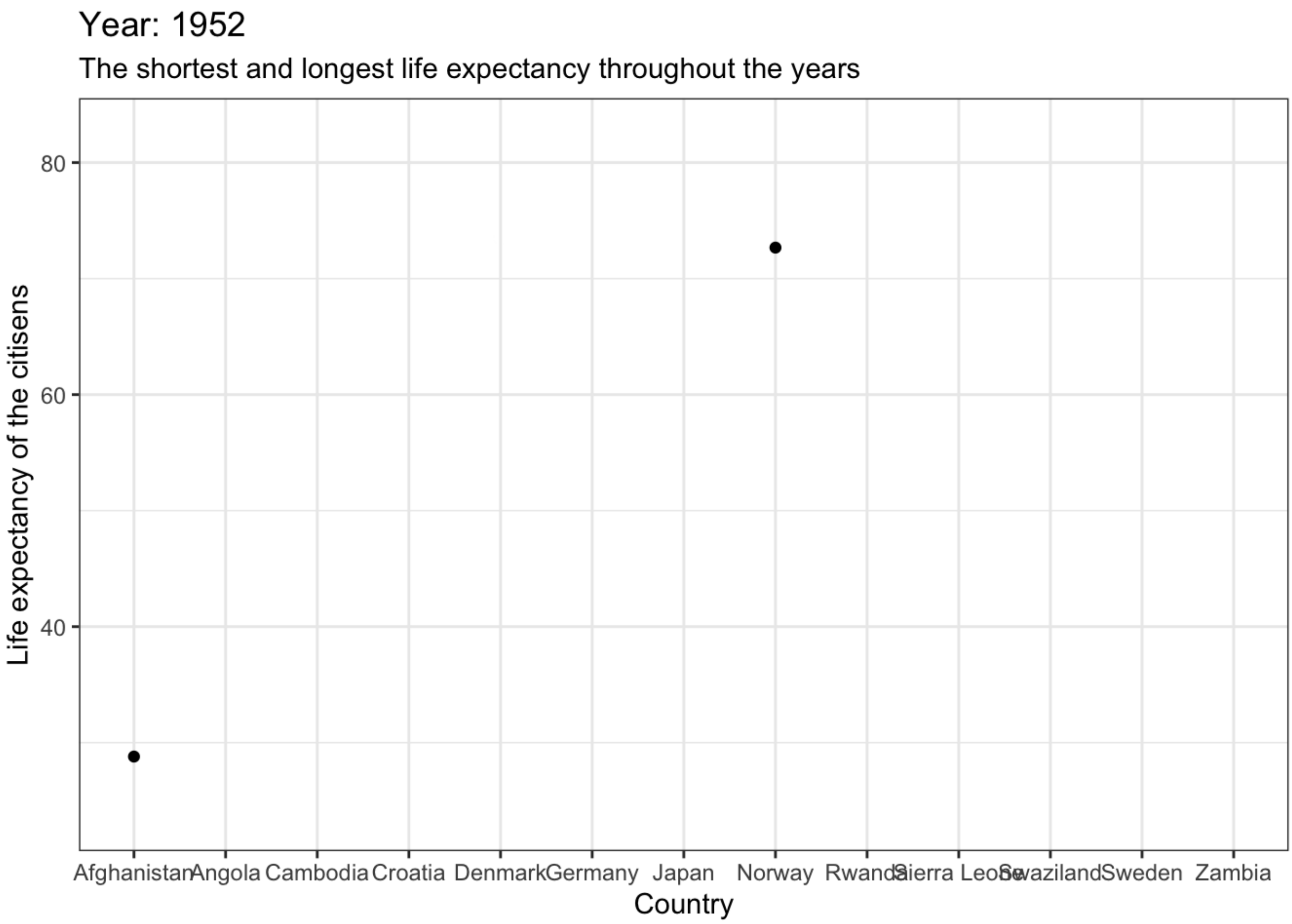
```
# lets make it move
ggplot(data_combined, aes(year, life_exp, colour = country)) +
  geom_point() +
  labs(title = "Year: {closest_state}", subtitle = "The shortest and longest life exp
ectancy throughout the years", x = "Country", y = "Life expectancy of the citisens")
 + # the {} adds the year transition as the title
  transition_states(year, transition_length = 3, state_length = 3) +
  enter_fade() +
  exit_fade()
```



Okay this is cool, but doesn't give us much we didn't already have in the not-moving plot above.

another way it could be visualised is by having countries on the x axis:

```
ggplot(data_combined, aes(country, life_exp)) +
  geom_point() +
  labs(title = "Year: {closest_state}", subtitle = "The shortest and longest life exp
ectancy throughout the years", x = "Country", y = "Life expectancy of the citisens")
 + # the {} adds the year transition as the title
  transition_states(year, transition_length = 3, state_length = 3) +
  enter_fade() +
  exit_fade()
```



not sure this is the best way to use the animation either, but it is a cool and different way to visualise which countries had the shortest and longest life expectancy at the different points in time.