Exploratory Data Analysis with R

Creating Animated Graphics Using `magick` and `gganimate`

Xuemao Zhang
East Stroudsburg University

November 2, 2022

Outline

- The `magick` package
- Image Vectors
- Animation using `magick`
- Animation using `gganimate`
- `You can disable/enable mouse click advance by pressing key 'k', when viewing the presentation.`

Image Vectors

The `magick` package

- The magick package is an ambitious effort to modernize and simplify high-quality image processing in R.

- The package `magick` is maintained by Jeroen Ooms.

- The magick package can do transformations of graphs, layers and animations.

- Another animation R package is gganimate. It extends the grammar of graphics as implemented by `ggplot2`(https://cran.r-project.org/web/packages/gganimate/vignettes/gganimate.html). `gganimate` vignettes: https://cran.r-project.org/web/packages/gganimate/vignettes/gganimate.html.

- `magick` supports the pipe `%>%` operator used in `ggplot2`.

The `magick` package

- It wraps the ImageMagick STL which is perhaps the most comprehensive open-source image processing library available today.

- Overwhelming amount of functions: convert, resize, flip, mirror, rotate, distort, transform image, adjust color, draw text, draw shapes, special effects, ...

- This subsection is based on R:vignette The magick package: Advanced Image-Processing in R.

- Lots of image processing functions are descried in R:vignette The magick package: Advanced Image-Processing in R. We focus on animated graphics.

Image IO: read and write

- `magick` automatically converts and renders all common image formats.

- Images can be read directly from a file path, URL, or raw vector with image data with the function `image_read()`.

```
library(magick);
frink=image_read('https://jeroen.github.io/images/frink.png');
print(frink);
```

```
##   format width height colorspace matte filesize density
## 1    PNG   220    445       sRGB  TRUE    73494   72x72
```

Image IO: converting formats

- We use `image_write()` to export an image in any format to a file on disk, or in memory if `path = NULL`.

```
image_write(frink, path = "frink.png", format = "png");
```

- `magick` keeps the image in memory in it’s original format. Specify the `format` parameter in `image_convert()` to convert to another format.

Image IO: output

```
frink_gif=image_convert(frink,  format = "gif");
# convert from png to gif
```

- IDE’s with a built-in web browser (as you have seen in RStudio - html output) automatically display magick images in the viewer. This results in a neat interactive image editing environment.

```
print(frink);
```

```
##   format width height colorspace matte filesize density
## 1    PNG   220    445       sRGB  TRUE    73494   72x72
```

Transformations: Resize

- resize proportionally to width

```
image_scale(frink, "300"); #width: 300px
```

Transformations: Resize

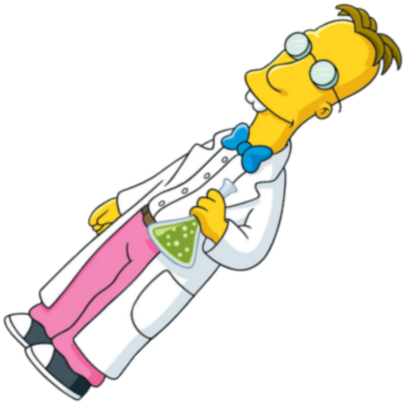
- resize proportionally to height

```
image_scale(frink, "x300"); # height: 300px
```

Transformations: Rotate

- Rotate a graph

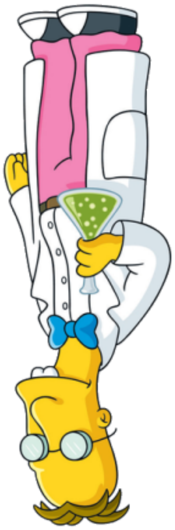
```
image_rotate(frink, 45);
```



Transformations: Mirror

- Mirror a graph

```
image_flip(frink);
```



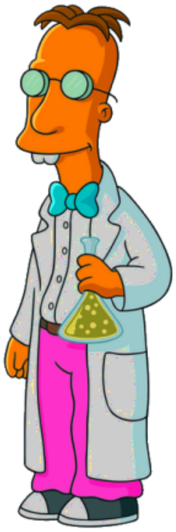
Transformations: Mirror

- Mirror a graph

```
image_flop(frink);
```

Transformations: Brightness, Saturation, Hue

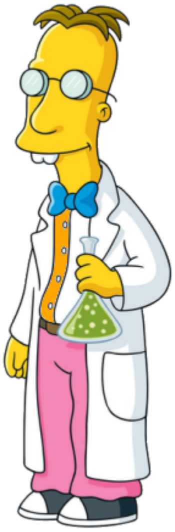
```
image_modulate(frink, brightness = 80, saturation = 120, hue = 90);
```



Transformations: image fill

- With image_fill we can flood fill starting at pixel point. The fuzz parameter allows for the fill to cross for adjacent pixels with similarish colors.

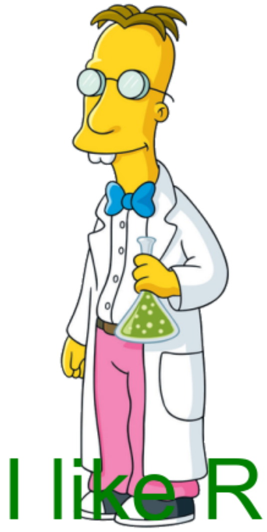
```
image_fill(frink, "orange", point = "+100+200", fuzz = 20); ## Paint the
        shirt orange
```



Text annotation

- It can be useful to print some text on top of images. See https://cran.r-project.org/web/packages/magick/vignettes/intro.html#cut_and_edit for more information.

```
# Add some text
image_annotate(frink, "I like R!", size = 70,
               gravity = "southwest", color = "green");
```



The pipe %>% operator

- magick supports the pipe %>% operator used in ggplot2.

```
frink1 = image_read('https://jeroen.github.io/images/frink.png')%>%
  image_rotate(270) %>% image_scale("300") %>%
  image_background("hotpink") %>%
  image_border("#000080", "30x20") %>%
  image_annotate("ESU Data", color = "red",
                 location ="+25+12", size = 40);
print(frink1);
```

```
##   format width height colorspace matte filesize density
## 1    PNG   360    188       sRGB  TRUE        0   72x72
```

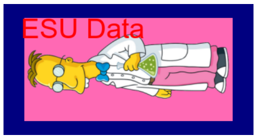


Image Vectors

- **Animation** is the illusion of movement created by showing a series of still pictures in rapid succession.

- For example, an animated GIF image is comprised of multiple images. Each of these images must be the same size, in terms of width and height. The image should not exceed 256 colors.

- The following example shows the frames information of an animated GIF image.

```r
library(magick);
# Download earth gif and make it a bit smaller
earth = image_read("https://jeroen.github.io/images/earth.gif")%>%
  image_scale("400x"); # resize proportionally to width: 300px
earth;
```

Image Vectors

```
length(earth);
```

```
## [1] 44
```

```
head(image_info(earth));
```

```
##   format width height colorspace matte filesize density
## 1    GIF   400    400       sRGB FALSE        0   72x72
## 2    GIF   400    400       sRGB  TRUE        0   72x72
## 3    GIF   400    400       sRGB  TRUE        0   72x72
## 4    GIF   400    400       sRGB  TRUE        0   72x72
## 5    GIF   400    400       sRGB  TRUE        0   72x72
## 6    GIF   400    400       sRGB  TRUE        0   72x72
```

Image Vectors

- All functions in `magick` have been **vectorized** to support working with layers, compositions or animation. Each image can be a component of a vector.

- The standard base vector methods `[` `[[`, `$`, `c()` and `length()` are used to manipulate sets of images which can then be treated as **layers** or **frames**.

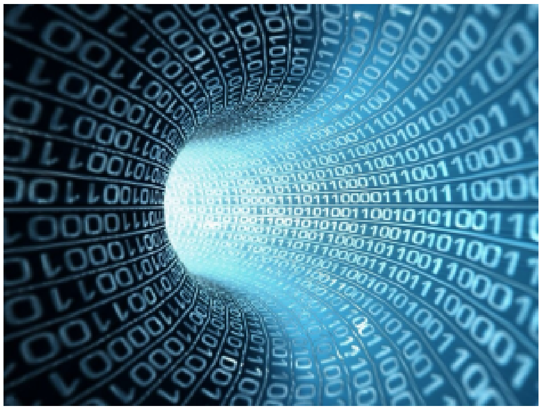
- Animation methods

  - *composing layers*
  - *displaying pages: When reading a PDF document using `pdftools::image_read_pdf`, each page becomes an element of the vector.*
  - *displaying frames.*

Animation using Layers

- We can **stack layers** on top of each other as we would in Photoshop.

```
bigdata = image_read('https://jeroen.github.io/images/bigdata.jpg');
frink = image_read("https://jeroen.github.io/images/frink.png");
img = c(bigdata, frink) %>% image_scale("300x300");
img;
```



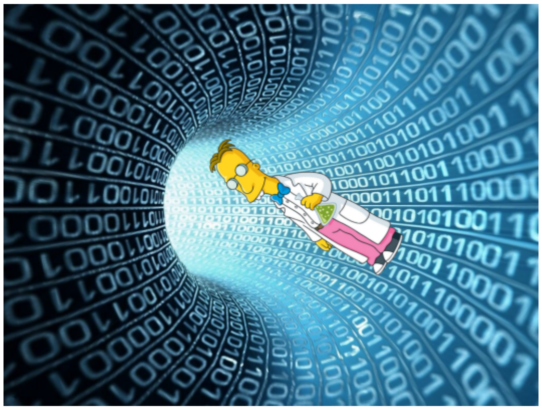
```
image_info(img);
```

```
##   format width height colorspace matte filesize density
## 1   JPEG   300    225       sRGB FALSE        0   72x72
## 2    PNG   148    300       sRGB  TRUE        0   72x72
```

Animation using Layers

- Composing allows for combining two images on a specific position

```
frink2=frink%>%image_background("none")%>%
  image_rotate(300)%>% image_scale("x200");
image_composite(image_scale(bigdata, "x400"),
                frink2, offset = "+180+100");
```

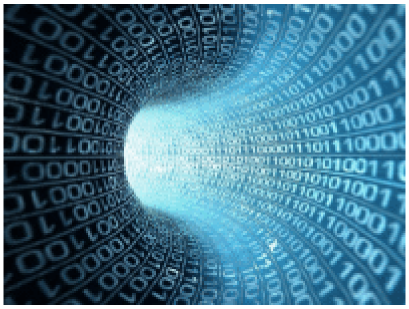


```
image_write(frink2, path = "frink2.gif", format = "gif");
```

Animation using Frames

- We can also make images frames in an animation!

```
image_animate(image_scale(img, "200x200"), fps = 1);
```



```
# fps means frames per second;
```

Animation using `magick`

R graphics device

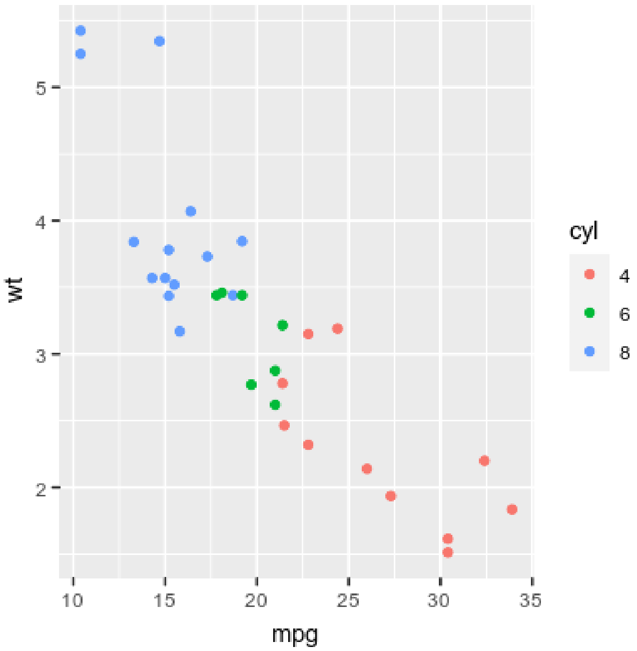
- The `magick::image_graph()` function opens a new graphics. It returns an image object to which the plot(s) will be written. Each “page” in the plotting device will become a **frame** in the image object.

```
library(ggplot2); library(dplyr);
# Produce image using graphics device
fig = image_graph(width = 400, height = 400, res = 96);
mtcars%>%mutate(cyl=factor(cyl))%>%ggplot(aes(mpg, wt, color=cyl))+
  geom_point();
dev.off();  # dev.off() shuts down the current device
```

```
## png
##   2
```

```
print(fig);
```

```
## # A tibble: 1 x 7
##   format width height colorspace matte filesize density
##   <chr>  <int>  <int> <chr>      <lgl>    <int> <chr>
## 1 PNG      400    400 sRGB       TRUE         0 96x96
```



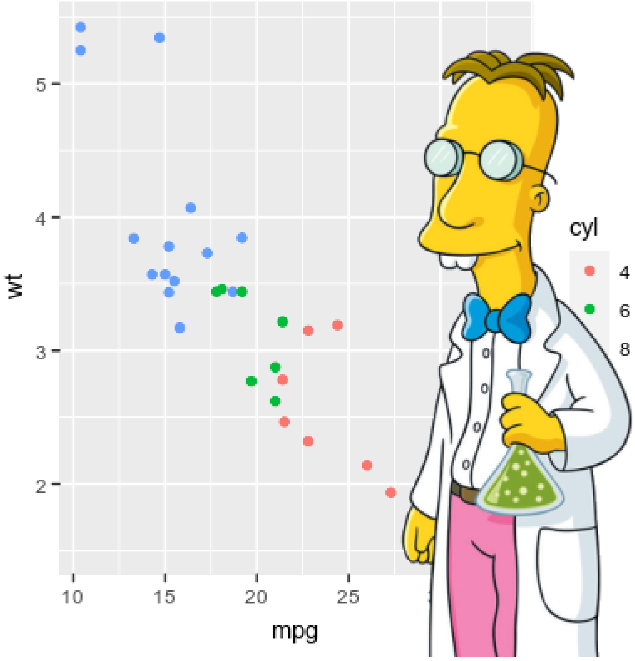
Animation using magick

R graphics device

- Then we can easily post-process the figure using regular image operations.

```
out =image_composite(fig, frink, offset = "+200+30");
print(out);
```

```
## # A tibble: 1 x 7
##   format width height colorspace matte filesize density
##   <chr>  <int>  <int> <chr>      <lgl>    <int> <chr>
## 1 PNG      400    400 sRGB       TRUE         0 96x96
```



Animation using `magick`

An example

- The graphics device supports **multiple frames** which makes it easy to create animated graphics.

- A small demonstration data sampled every five years.

```
library(gapminder);
summary(gapminder);
```

```
##         country        continent        year         lifeExp
##  Afghanistan:  12   Africa  :624   Min.   :1952   Min.   :23.60
##  Albania    :  12   Americas:300   1st Qu.:1966   1st Qu.:48.20
##  Algeria    :  12   Asia    :396   Median :1980   Median :60.71
##  Angola     :  12   Europe  :360   Mean   :1980   Mean   :59.47
##  Argentina  :  12   Oceania : 24   3rd Qu.:1993   3rd Qu.:70.85
##  Australia  :  12                  Max.   :2007   Max.   :82.60
##  (Other)    :1632
##       pop              gdpPercap
##  Min.   :6.001e+04   Min.   :   241.2
##  1st Qu.:2.794e+06   1st Qu.:  1202.1
##  Median :7.024e+06   Median :  3531.8
##  Mean   :2.960e+07   Mean   :  7215.3
##  3rd Qu.:1.959e+07   3rd Qu.:  9325.5
##  Max.   :1.319e+09   Max.   :113523.1
##
```

- Gapminder Foundation Wiki is a non-profit venture registered in Stockholm, Sweden, that promotes sustainable global development and achievement of the United Nations Millennium Development Goals by increased use and understanding of statistics and other information about social, economic and environmental development at local, national and global levels.

- Bubble chart - An extension of a scatterplot, a bubble chart is commonly used to visualize relationships between three or more numeric variables. Each bubble in a chart represents a single data point.

- *The best stats you've ever seen | Hans Rosling (2 minutes-6 minutes): https://www.youtube.com/watch?v=hVimVzgtD6w*
- *https://www.gapminder.org/fw/world-health-chart/*

Animation using magick

An example

```
library(ggplot2);
Years = unique(gapminder$year); #obtain all years
Img = image_graph(600, 340, res = 96); #open a new graphics device
for (k in 1:length(Years)){
data = gapminder[gapminder$year==Years[k], ]; #subset by year
p=ggplot(data, aes(gdpPercap, lifeExp,
                   size = pop, color = continent)) +
  scale_size("population", limits = range(data$pop)) +
  geom_point() + ylim(20, 90) +
  scale_x_log10(limits = range(data$gdpPercap)) + #log transformation
ggtitle(data$year) + labs(x ="gdpPercap", y = "lifeExp")+theme_classic();
print(p);
}
dev.off();
```

```
## png
##   2
```

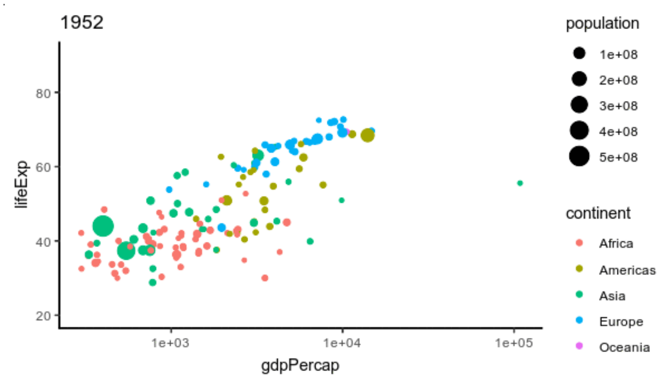
```
Img_Animation =image_animate(Img, fps = 1);
```

Animation using magick

An example

```
image_write(Img_Animation, path = "Img_Animation.gif");
# write it to a file
print(Img_Animation);
```

```
## # A tibble: 12 x 7
##    format width height colorspace matte filesize density
##    <chr>  <int>  <int> <chr>      <lgl>    <int> <chr>
##  1 gif      600    340 sRGB       TRUE         0 96x96
##  2 gif      600    340 sRGB       TRUE         0 96x96
##  3 gif      600    340 sRGB       TRUE         0 96x96
##  4 gif      600    340 sRGB       TRUE         0 96x96
##  5 gif      600    340 sRGB       TRUE         0 96x96
##  6 gif      600    340 sRGB       TRUE         0 96x96
##  7 gif      600    340 sRGB       TRUE         0 96x96
##  8 gif      600    340 sRGB       TRUE         0 96x96
##  9 gif      600    340 sRGB       TRUE         0 96x96
## 10 gif      600    340 sRGB       TRUE         0 96x96
## 11 gif      600    340 sRGB       TRUE         0 96x96
## 12 gif      600    340 sRGB       TRUE         0 96x96
```



Animation using gganimate

package gganimate

gganimate extends the grammar of graphics as implemented by ggplot2 to include the description of animation. It does this by providing a range of new grammar classes that can be added to the plot object in order to customise how it should change with time.

- `transition_*()` defines how the data should be spread out and how it relates to itself across time.
- `view_*()` defines how the positional scales should change along the animation.
- `shadow_*()` defines how data from other points in time should be presented in the given point in time.
- `enter_*()/exit_*()` defines how new data should appear and how old data should disappear during the course of the animation.
- `ease_aes()` defines how different aesthetics should be eased during transitions.

Animation using `gganimate`

package `gganimate`

How does gganimate work?

- Start with a ggplot2 specification
- Add layers with graphical primitives (geoms)
- Add formatting specifications
- Add animation specifications

Animation using gganimate

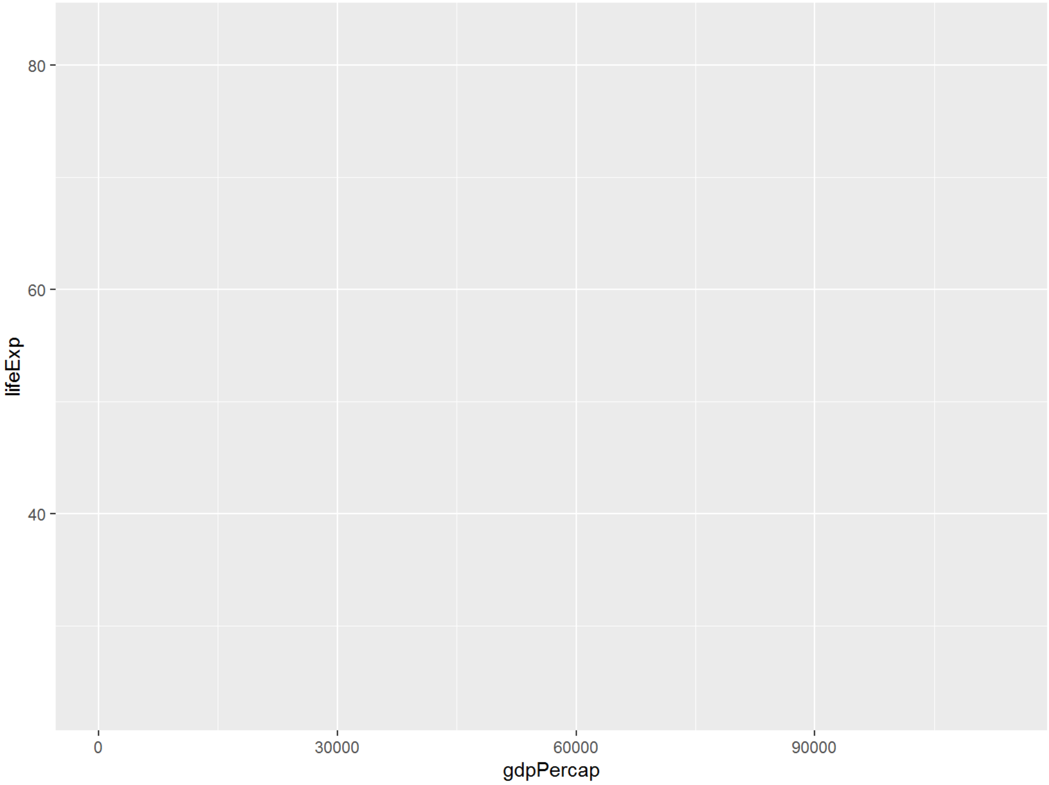
Example using gganimate

- Start by passing the data to ggplot

```
library(gganimate);
ggplot(data=gapminder)
```

- Add the mapping

```
ggplot(data=gapminder)+
   aes(gdpPercap, lifeExp,size = pop, color = continent)
```

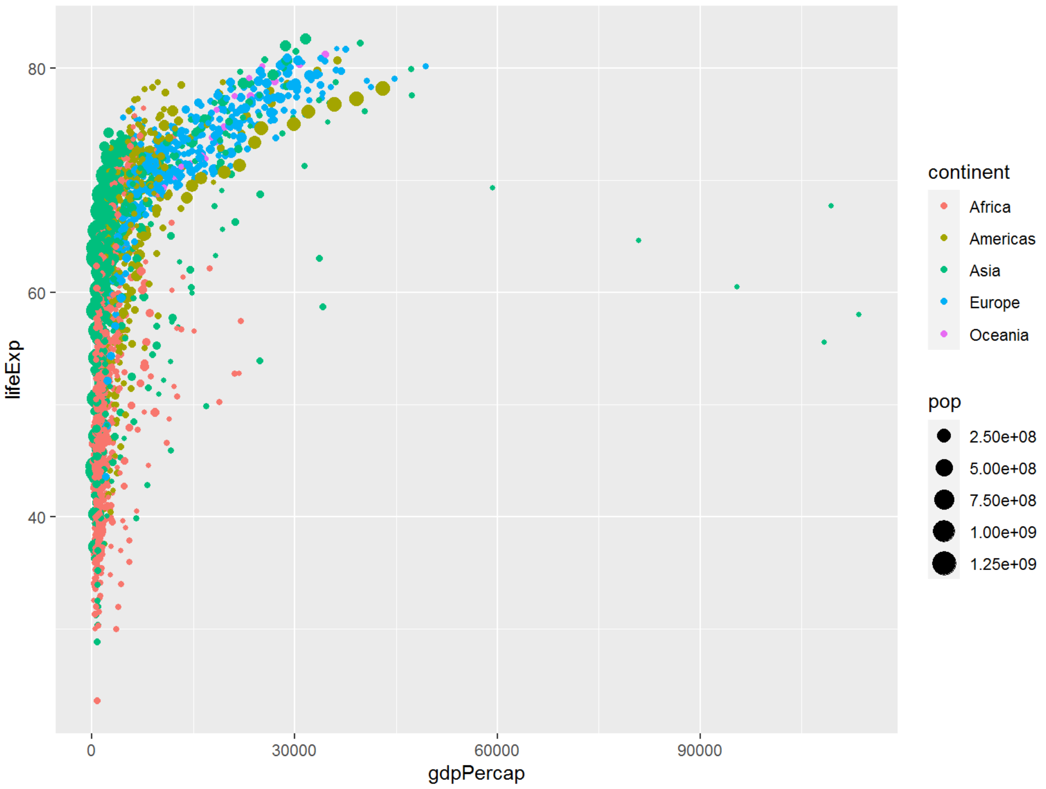


Animation using gganimate

Example using gganimate

- add a graphical primitive, let’s do points

```
ggplot(data=gapminder)+
   aes(gdpPercap, lifeExp,size = pop, color = continent)+
  geom_point()
```

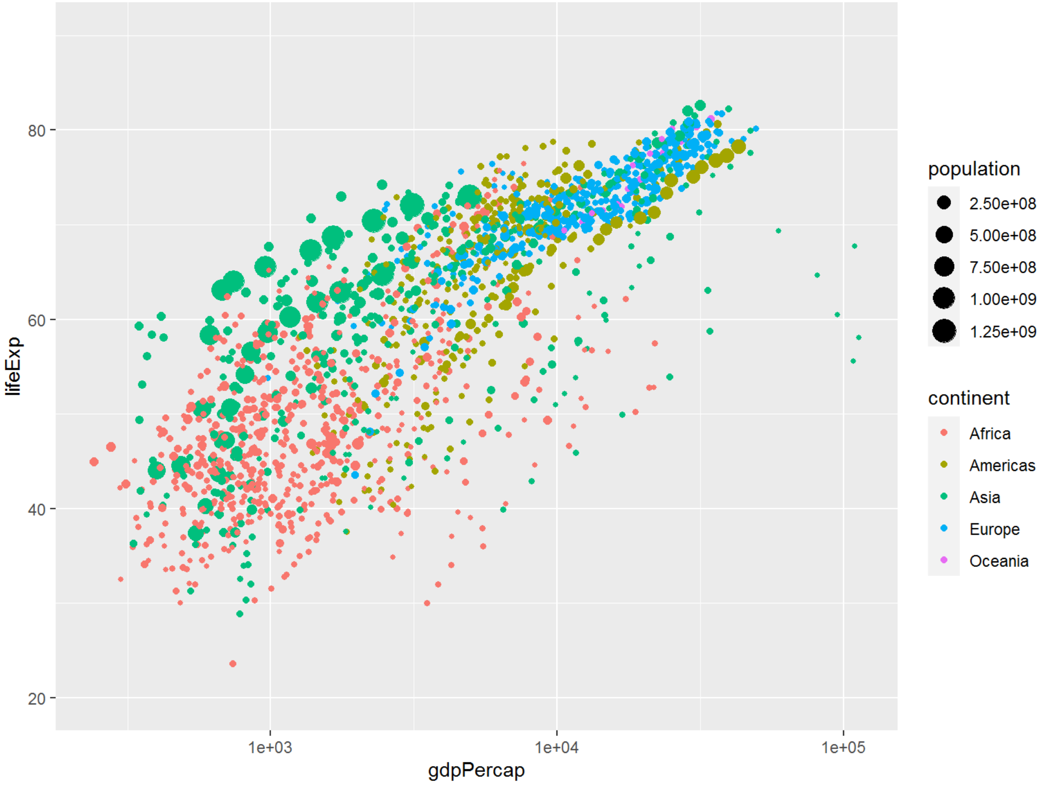


Animation using gganimate

Example using gganimate

- Add some other layers

```
ggplot(data=gapminder)+
   aes(gdpPercap, lifeExp,size = pop, color = continent)+
  geom_point()+
  scale_size("population", limits = range(gapminder$pop)) +
  ylim(20, 90) +
  scale_x_log10(limits = range(gapminder$gdpPercap)) + #log transformation
  labs(x ="gdpPercap", y = "lifeExp");
```



Animation using gganimate

Example using gganimate

- Just one extra line turns this into an animation!

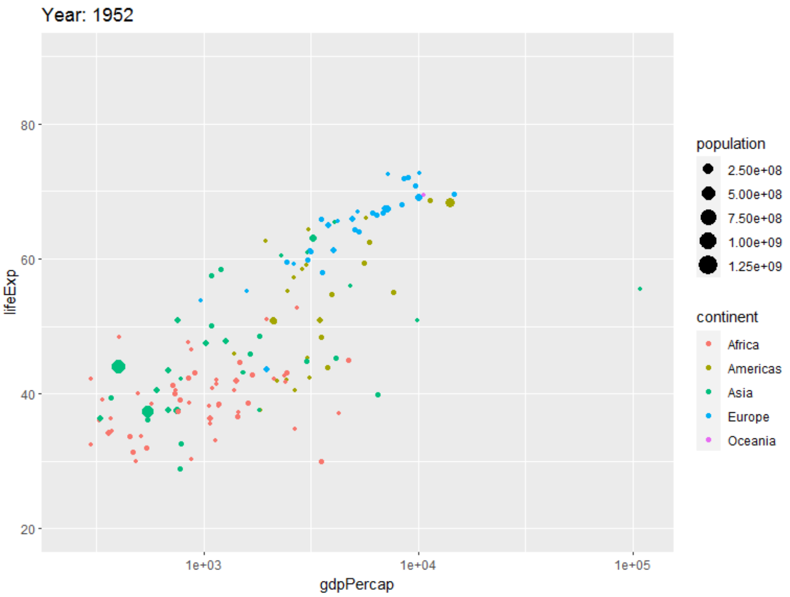
```
library(gganimate);
p=ggplot(data=gapminder, aes(gdpPercap, lifeExp,
                   size = pop, color = continent)) +
   geom_point()+
  scale_size("population", limits = range(gapminder$pop)) +
  ylim(20, 90) +
  scale_x_log10(limits = range(gapminder$gdpPercap)) + #log transformation
  labs(x ="gdpPercap", y = "lifeExp");
p2=p + transition_time(year)+
  labs(title =  'Year: {frame_time}');
#print(p2); #This will take some time to load
anim_save("gapminder_ani.gif", p2); #save the animation
```

Animation using gganimate

Example using gganimate

- Embed the animation to your presentation!

```
#print(p2); #This will take some time to load
knitr::include_graphics("gapminder_ani.gif");
```



Animation using `gganimate`

gganimate References

- Getting Started https://cran.r-project.org/web/packages/gganimate/vignettes/gganimate.html
- Developer's webpage https://github.com/thomasp85/gganimate
- Package manual https://cran.r-project.org/web/packages/gganimate/gganimate.pdf

License