Visualizing Data in R using **ggplot2**

Lauren Nichols
Center for Data and Visualization Sciences – Duke Libraries

Lmn7@duke.edu

Jan 30, 2026

https://duke.is/ggplot2

Data Visualization Specialists

askdata@duke.edu

Center for Data and Visualization Sciences

Lauren Nichols

McCall Pitcher

Drew Keener

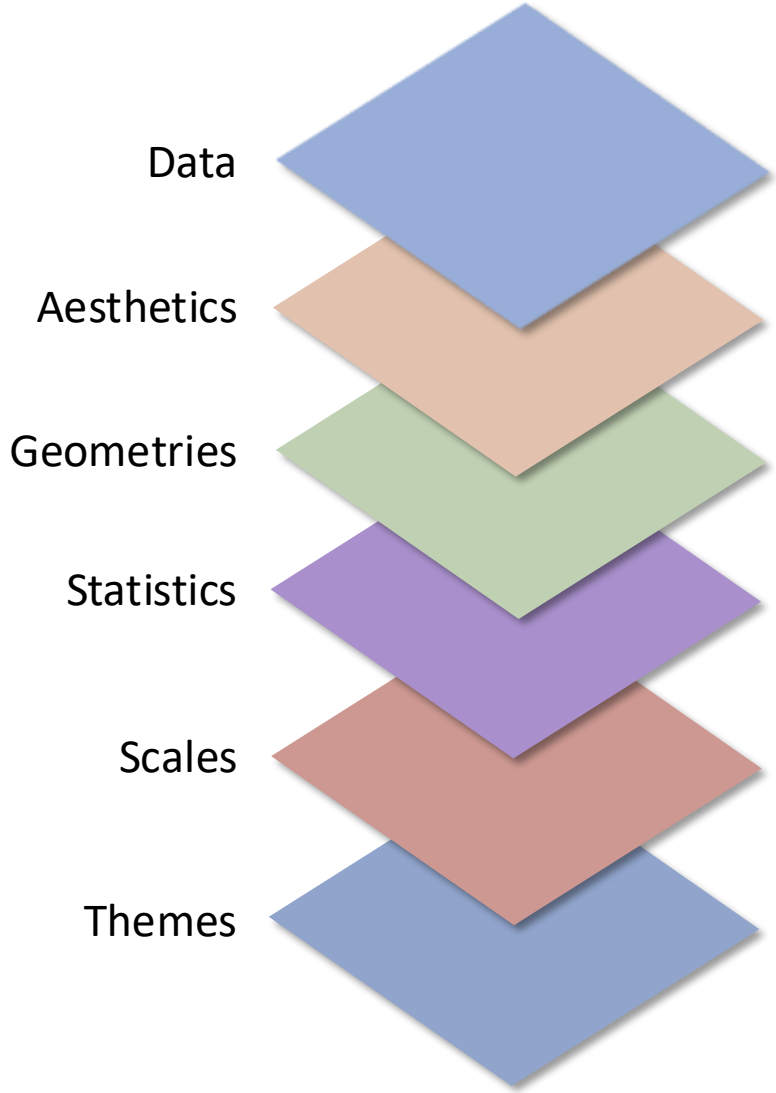
mapping

Eric Monson

“gg” in ggplot2 stands for
“Grammar of Graphics”

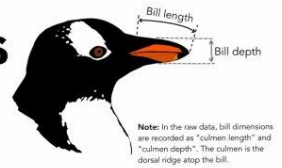
"gg" in ggplot2 stands for
"Grammar of Graphics"

Graphics are built by layering elements

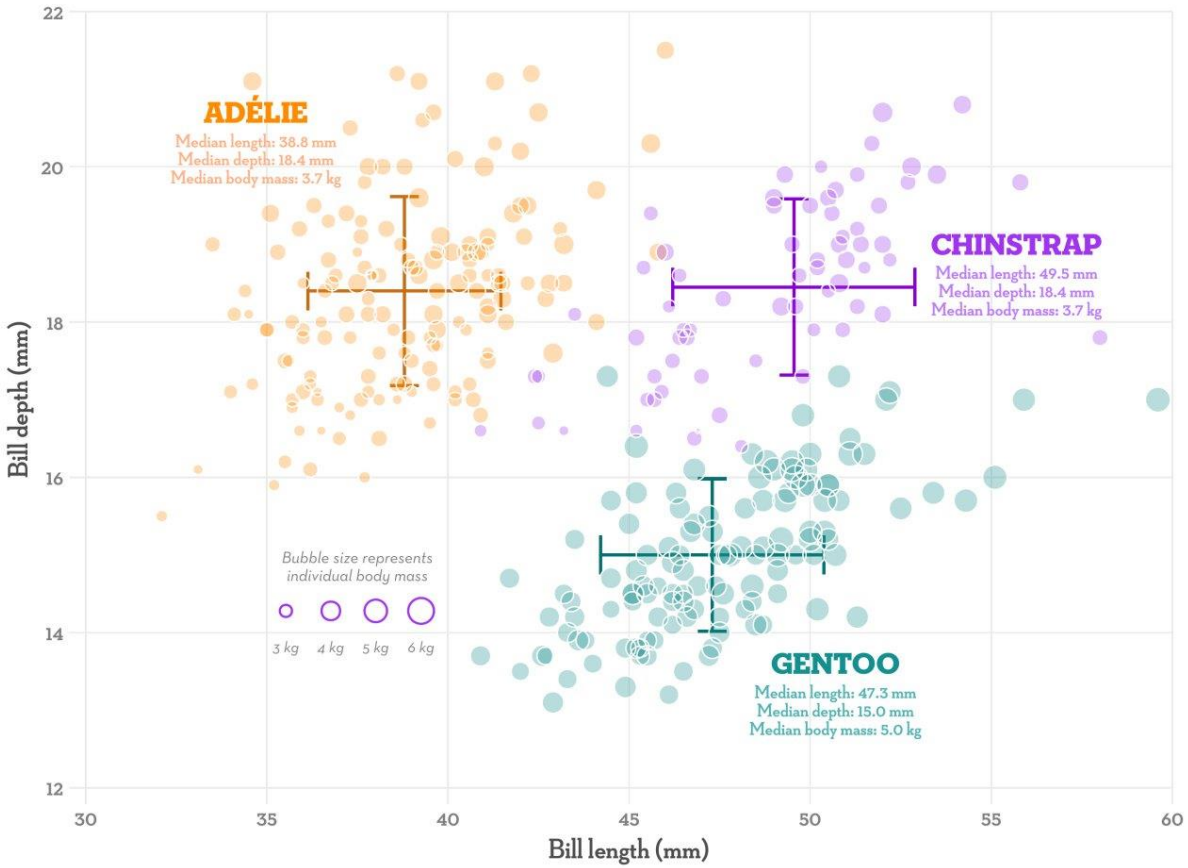


BILL DIMENSIONS OF BRUSH-TAILED PENGUINS

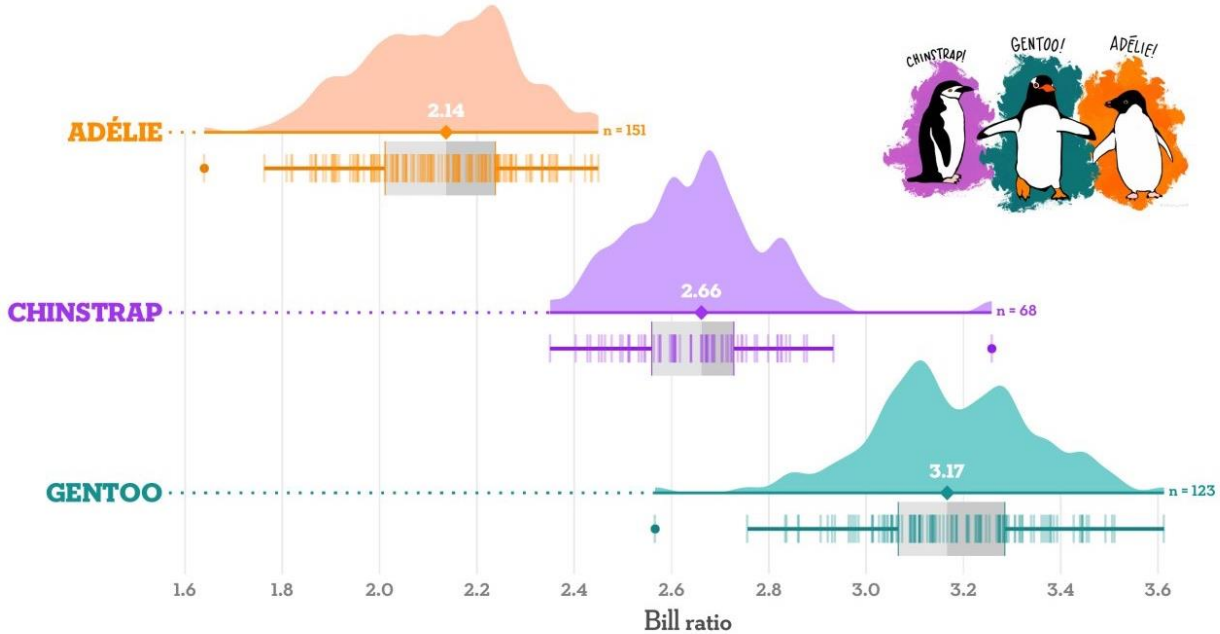
Pygoscelis adéliae (Adélie penguin) • P. antarctica (Chinstrap penguin) • P. papua (Gentoo penguin)



A. Scatterplot of bill length versus bill depth (error bars show median +/- sd)



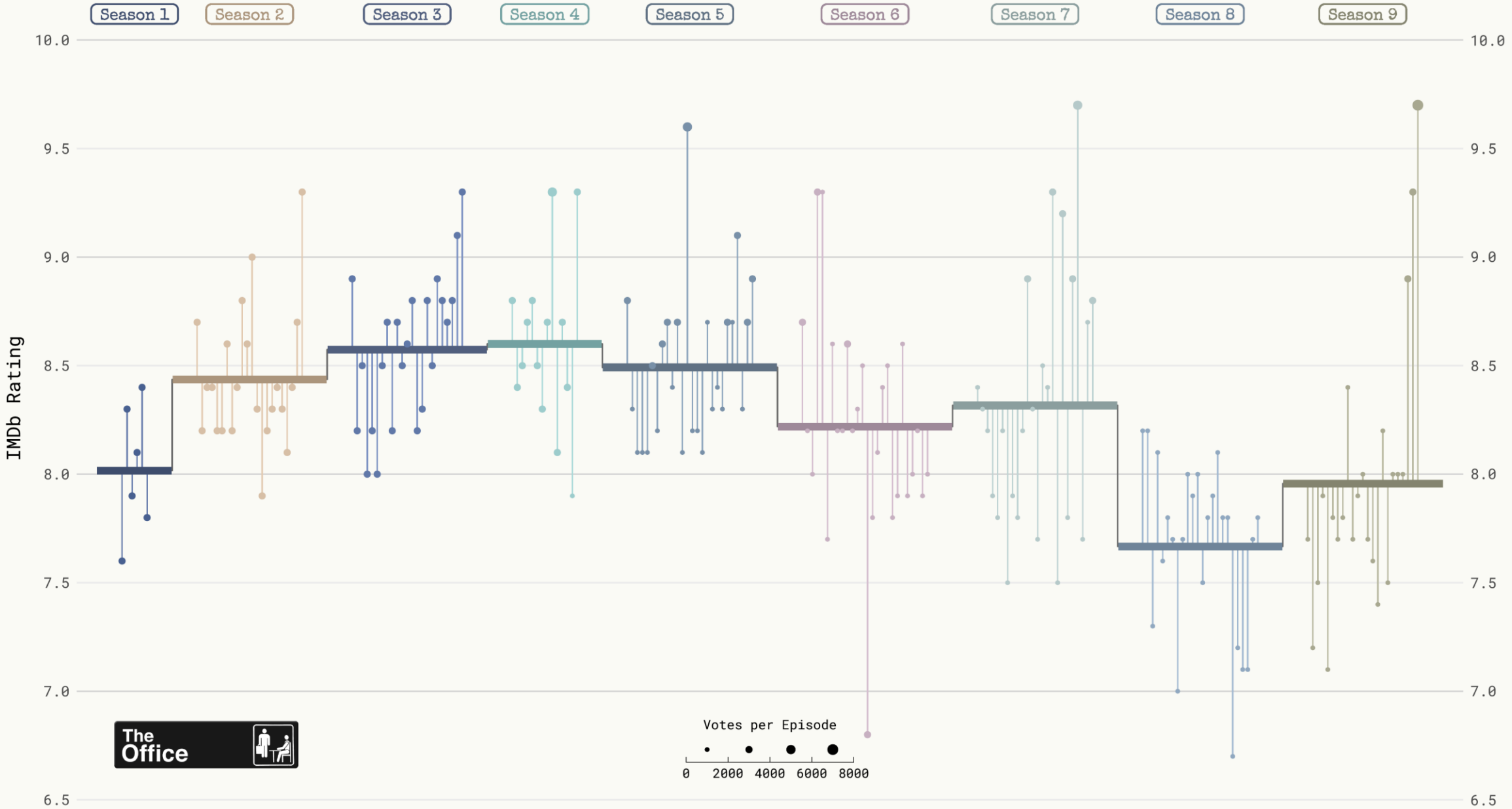
B. Distribution of the bill ratio, estimated as bill length divided by bill depth



Note: In the original data, bill dimensions are recorded as "culmen length" and "culmen depth". The culmen is the dorsal (upper) ridge of a bird's bill.

Visualization: Cédric Scherer • Data: Gorman, Williams & Fraser (2014) DOI: 10.1371/journal.pone.0090081 • Illustrations: Allison Horst


Where surfers travel.
data-to-viz.com | NASA.gov | 10,000 #surf tweets recovered



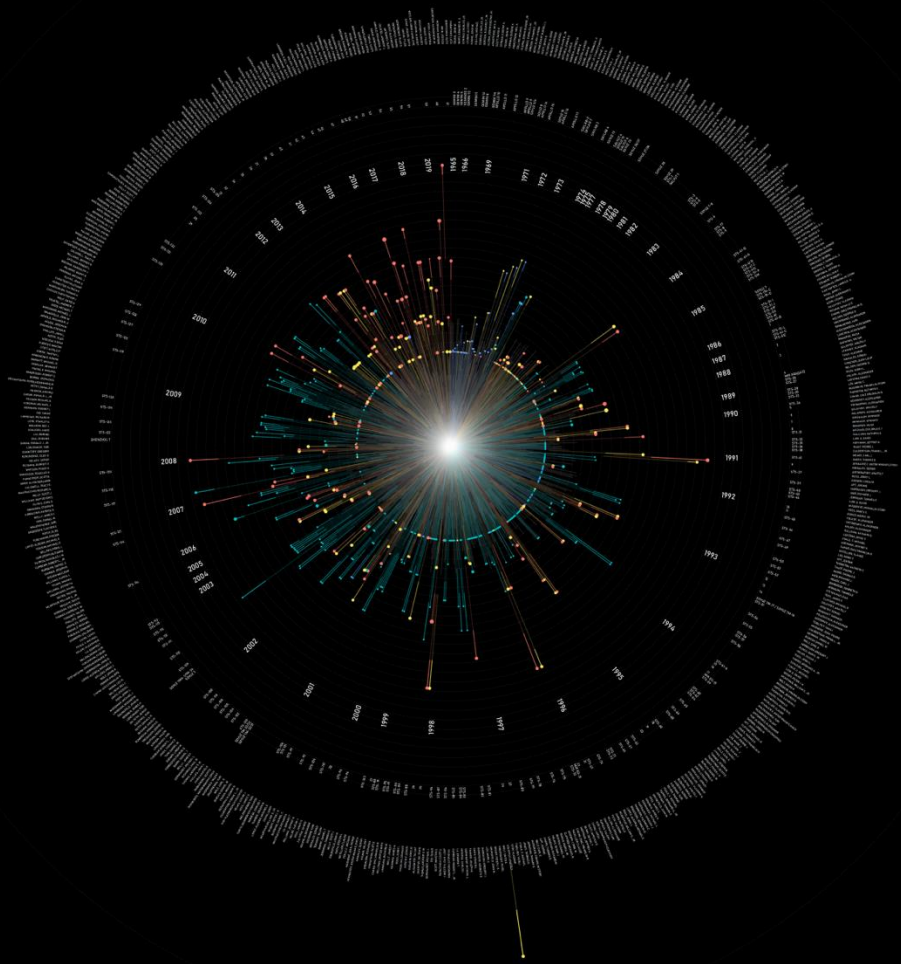
Visualization by Cédric Scherer • Data by IMDb via data.world • Fanart Logo by ArieS

SPACE.

The Final Frontier

Duration of astronauts' extravehicular activities during each space mission

Visualization by Joseph Shaw / @JosephShaw_ | Data - Astronaut Database - Mariya Stavnichuk & Tatsuya Corlett



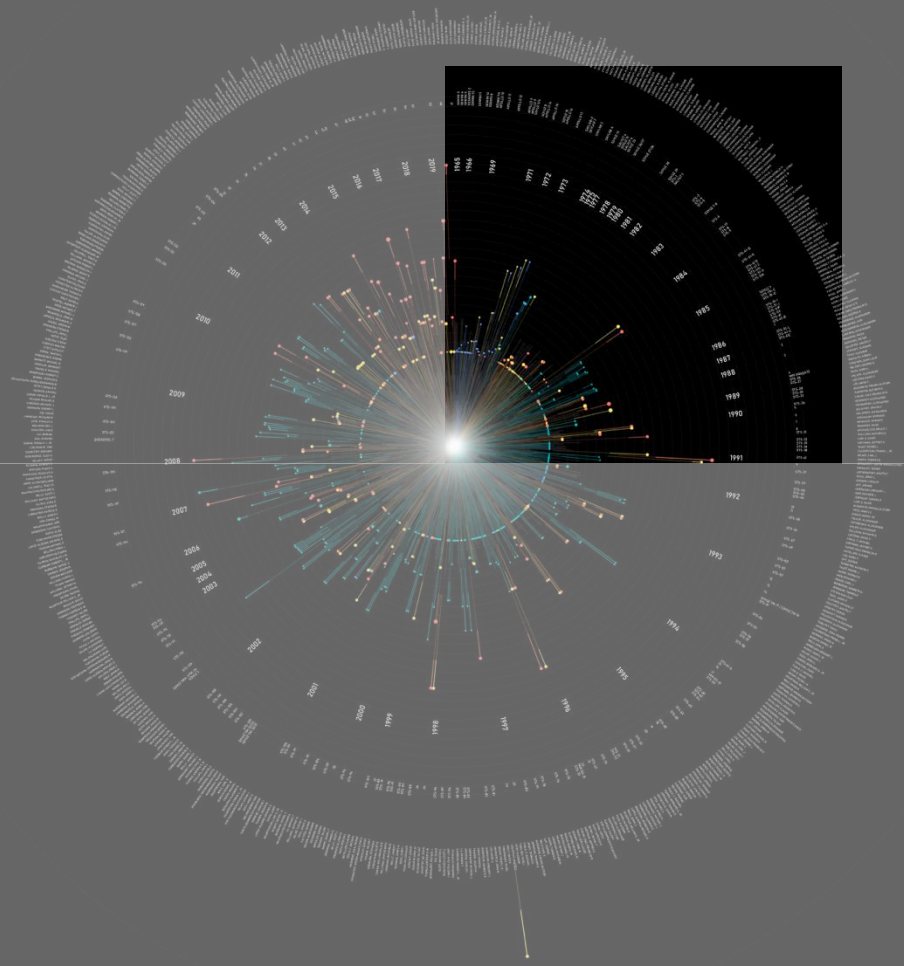
github.com/joseph-shaw/tidytuesday/

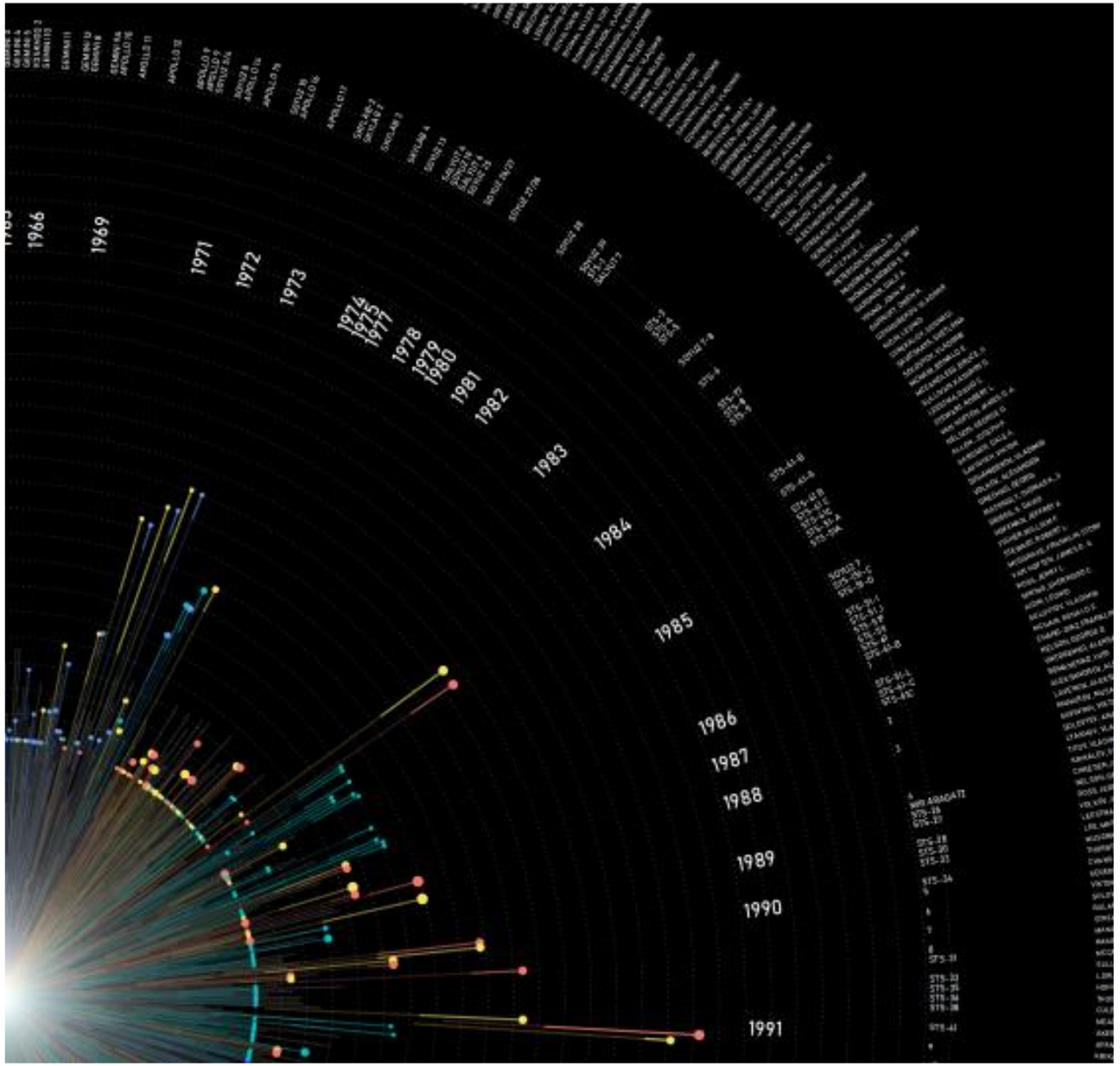
SPACE.

The Final Frontier

Duration of astronauts' extravehicular activities during each space mission

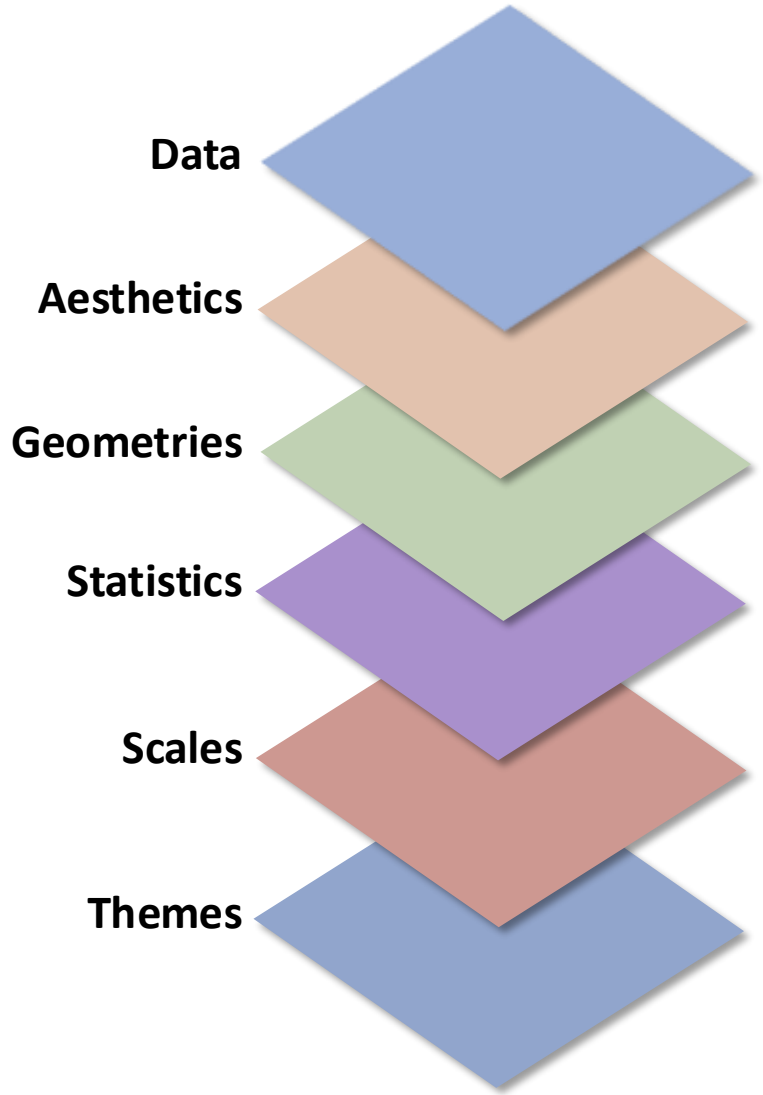
Visualization by Joseph Shaw / @JosephShaw_ | Data - Astronaut Database - Mariya Stavnichuk & Tatsuya Corlett



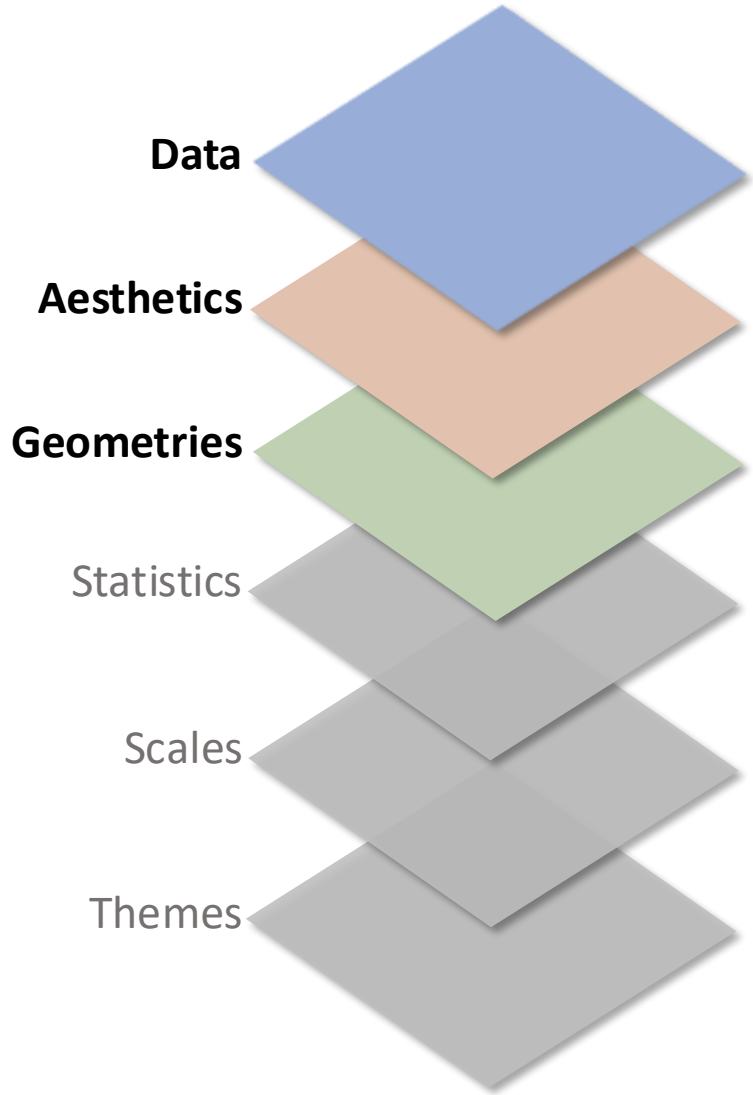


github.com/joseph-shaw/tidytuesday/

```
ggplot( data = yourdata ,
        mapping = aes(x axis, y axis) ) +
        geometry function +
        statistics function +
        scales function +
        themes function
```



```
ggplot( data = yourdata ,
        mapping = aes(x axis, y axis) ) +
        geometry function +
        statistics function +
        scales function +
        themes function
```



```
ggplot( data = yourdata ,
        mapping = aes(x axis, y axis) ) +
        geometry function
```

Data Format

ggplot expects your data to be in a “tidy” format

Wide data

animal	female	male
dog	3	4
cat	3	1
rat	3	5

Long data (aka “Tidy” data)

animal	sex	number
dog	female	3
cat	female	3
rat	female	3
dog	male	4
cat	male	1
rat	male	5

Data Format

ggplot expects your data to be in a “tidy” format

Wide data

animal	female	male
dog	3	4
cat	3	1
rat	3	5

Long data (aka “Tidy” data)

animal	sex	number
dog	female	3
cat	female	3
rat	female	3
dog	male	4
cat	male	1
rat	male	5

Data Format

Wide data

animal	female	male
dog	3	4
cat	3	1
rat	3	5

Long data

animal	sex	number
dog	female	3
cat	female	3
rat	female	3
dog	male	4
cat	male	1
rat	male	5

```
Long_data <- Wide_data |>
                    pivot_longer(names_to = sex,
                values_to = number)
```

```
happy <- read_csv("Happiness.csv")
```

```
happy
```

A tibble: 569 × 9

country_name <chr>	continent <chr>	year <dbl>	happiness <dbl>	GDP <dbl>
Afghanistan	Asia	2019	2.375	7.640
Afghanistan	Asia	2017	2.662	7.648
Central Africa…	Africa	2016	2.693	6.707
Afghanistan	Asia	2018	2.694	7.631
Zimbabwe	Africa	2019	2.694	7.698
South Sudan	Africa	2017	2.817	*NA*
South Sudan	Africa	2016	2.888	*NA*
Tanzania	Africa	2016	2.903	7.775
Yemen	Asia	2018	3.058	7.444
Rwanda	Africa	2017	3.108	7.568

1–10 of 569 rows | 1–5 of 9 columns Previous 1 2 3 4 5 6 … 57 Next

```
happy <- read_csv("Happiness.csv")

happy
```

A tibble: 569 × 9

country_name <chr>	continent <chr>	year <dbl>	happiness <dbl>	GDP <dbl>
Afghanistan	Asia	2019	2.375	7.640
Afghanistan	Asia	2017	2.662	7.648
Central Africa…	Africa	2016	2.693	6.707
Afghanistan	Asia	2018	2.694	7.631
Zimbabwe	Africa	2019	2.694	7.698
South Sudan	Africa	2017	2.817	*NA*
South Sudan	Africa	2016	2.888	*NA*
Tanzania	Africa	2016	2.903	7.775
Yemen	Asia	2018	3.058	7.444
Rwanda	Africa	2017	3.108	7.568

1–10 of 569 rows | 1–5 of 9 columns Previous 1 2 3 4 5 6 … 57 Next

```
ggplot( data = yourdata ,
        mapping = aes(x axis, y axis) ) +
        geometry function
```

```
ggplot( data = happy ,
        mapping = aes(x axis, y axis) ) +
        geometry function
```

```
ggplot( data = happy     ,
        mapping = aes(x axis, y axis) ) +
        geometry function
```

happy

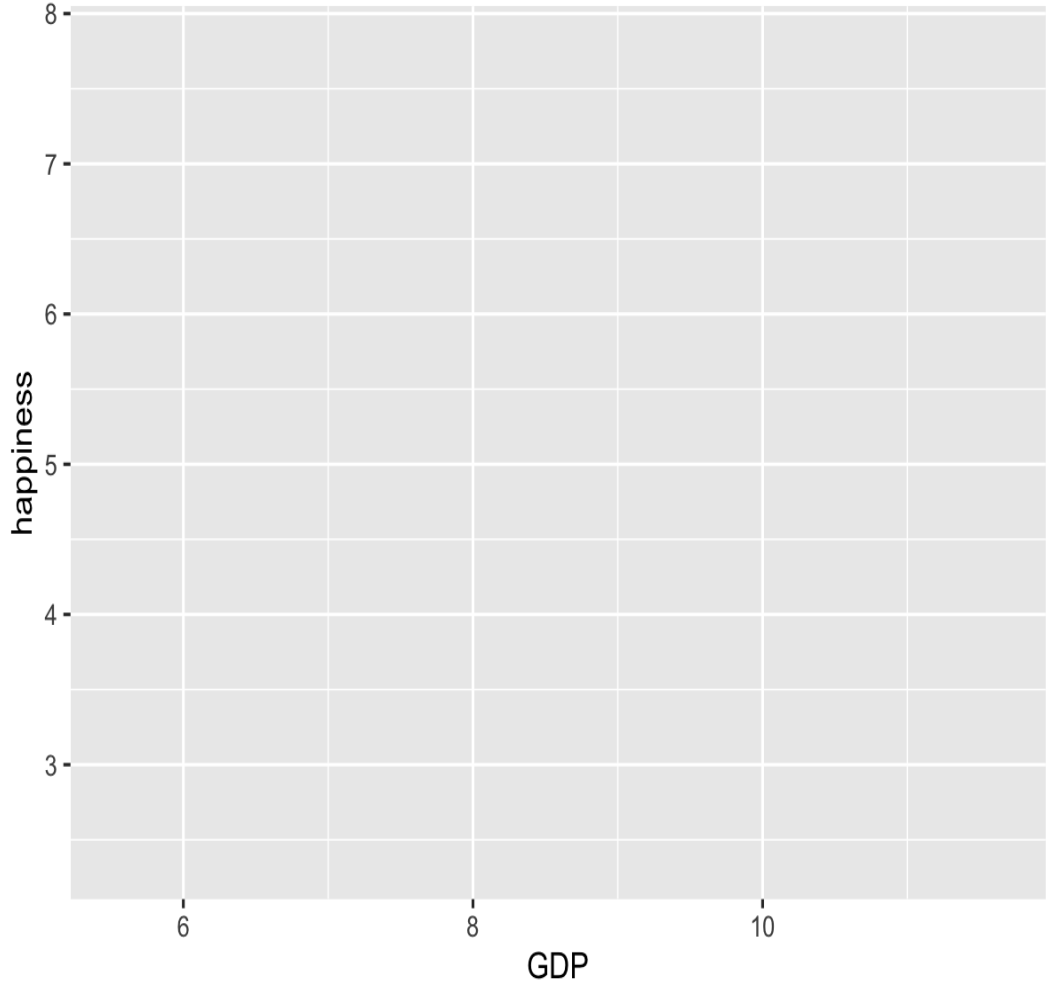
A tibble: 569 × 9

country_name <chr>	continent <chr>	year <dbl>	happiness <dbl>	GDP <dbl>
Afghanistan	Asia	2019	2.375	7.640
Afghanistan	Asia	2017	2.662	7.648
Central Africa…	Africa	2016	2.693	6.707
Afghanistan	Asia	2018	2.694	7.631
Zimbabwe	Africa	2019	2.694	7.698
South Sudan	Africa	2017	2.817	*NA*
South Sudan	Africa	2016	2.888	*NA*
Tanzania	Africa	2016	2.903	7.775
Yemen	Asia	2018	3.058	7.444
Rwanda	Africa	2017	3.108	7.568

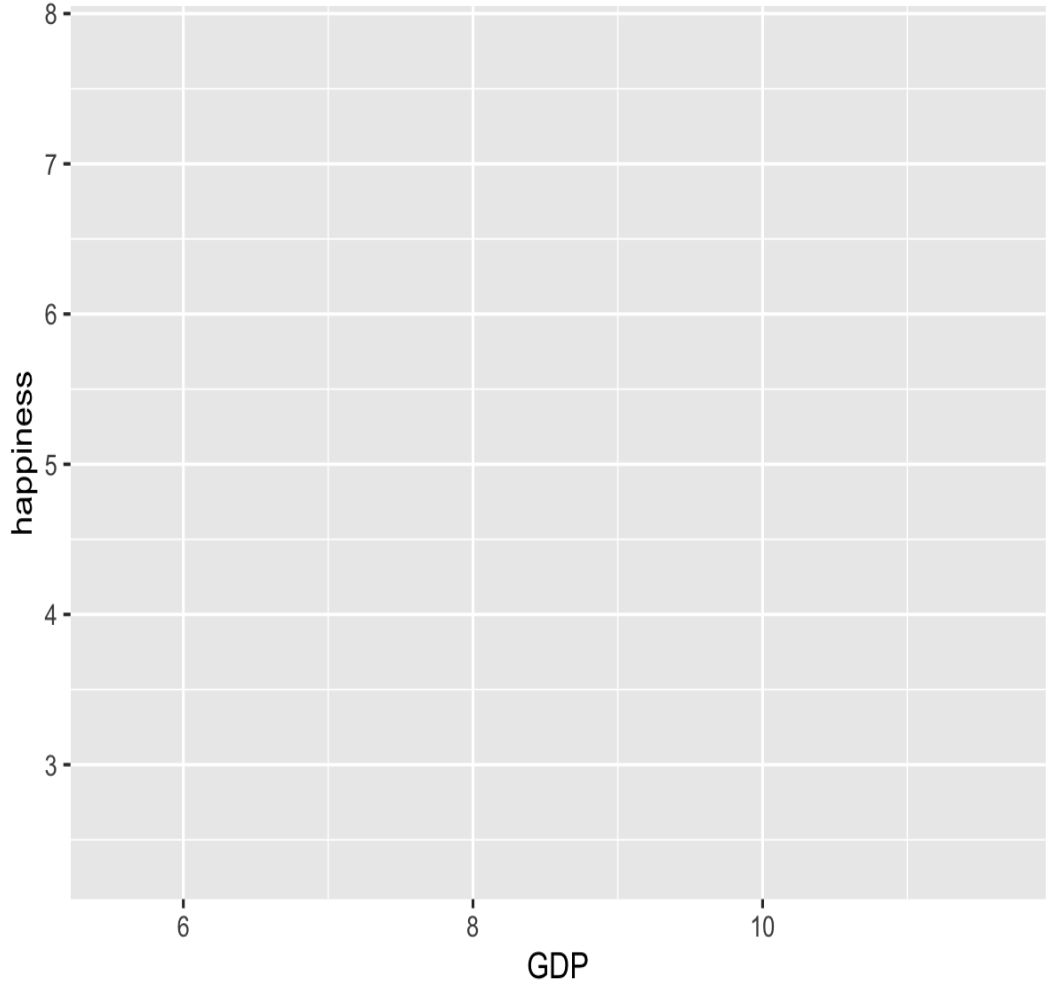
1–10 of 569 rows | 1–5 of 9 columns Previous 1 2 3 4 5 6 … 57 Next

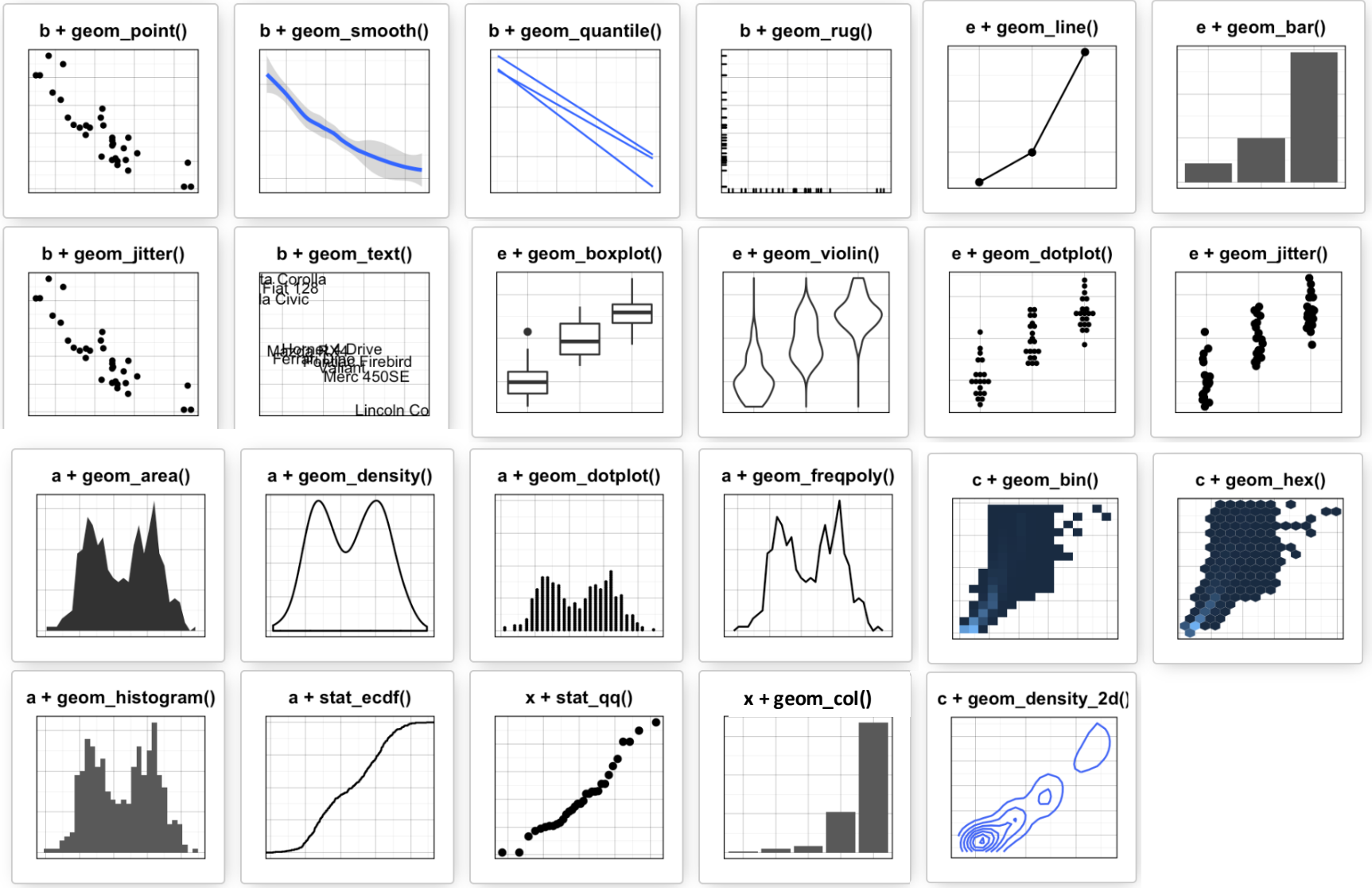
```
ggplot( data = happy     ,
        mapping = aes(GDP, happiness) ) +
        geometry function
```

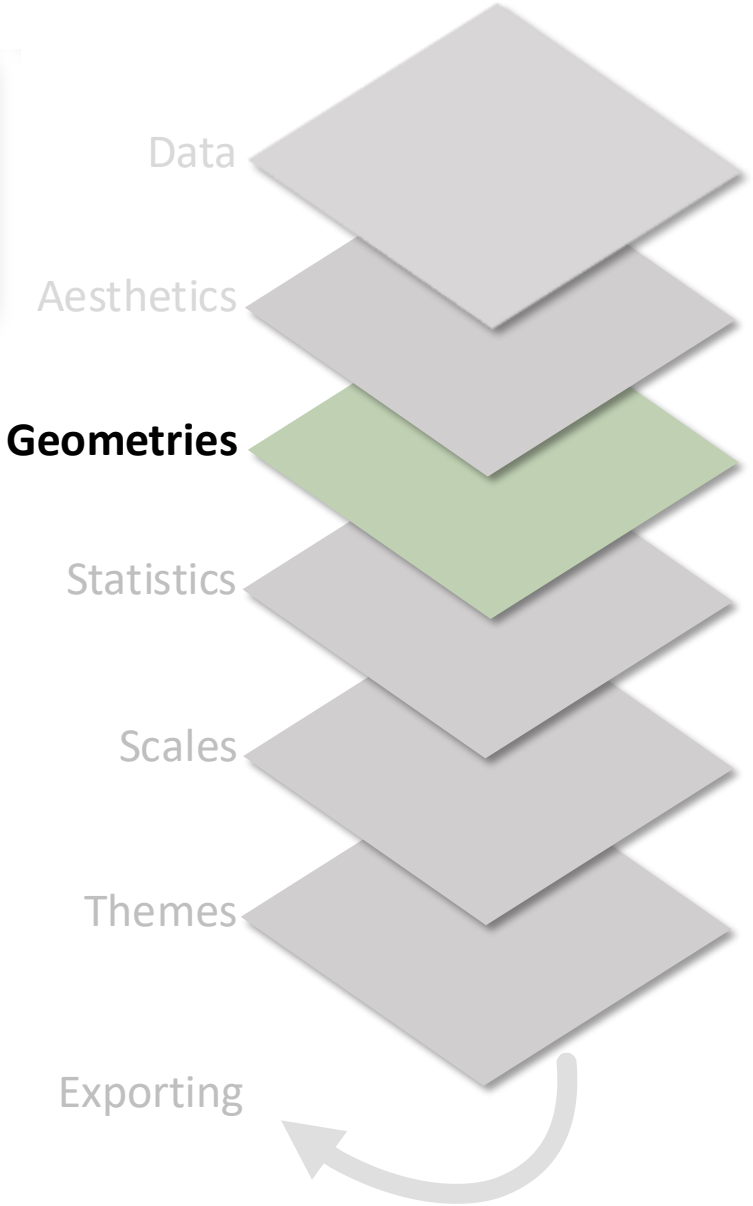
```
ggplot( data = happy ,
        mapping = aes(GDP, happiness) ) +
        geometry function
```



```
ggplot( data = happy ,
        mapping = aes(GDP, happiness) ) +
        geometry function
```

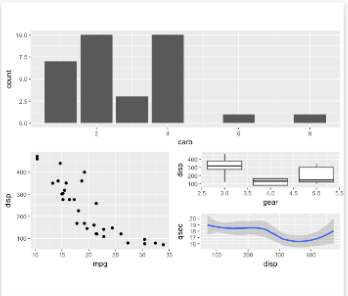



b + geom_point()
b + geom_smooth()
b + geom_quantile()
b + geom_rug()
e + geom_line()
e + geom_bar()
b + geom_jitter()
b + geom_text()
Fiat 128
Merc 450SE
Lincoln Co
e + geom_boxplot()
e + geom_violin()
e + geom_dotplot()
e + geom_jitter()
a + geom_area()
a + geom_density()
a + geom_dotplot()
a + geom_freqpoly()
c + geom_bin()
c + geom_hex()
a + geom_histogram()
a + stat_ecdf()
x + stat_qq()
x + geom_col()
c + geom_density_2d()


Data
Aesthetics
Geometries
Statistics
Scales
Themes
Exporting

Over 120 ggplot2 extension libraries

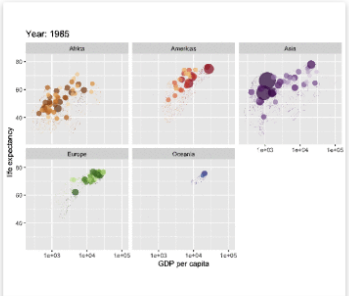
Showing 107 of 127



patchwork Star 2269

Easy composition of ggplot plots using arithmetic operators

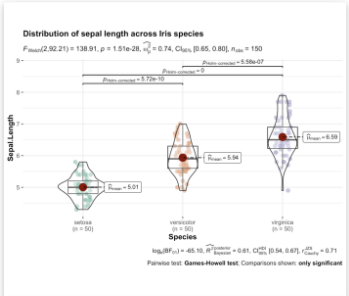
- author: thomasp85
- tags: visualization, composition
- js libraries:



gganimate Star 1885

A Grammar of Animated Graphics.

- author: thomasp85
- tags: visualization, general
- js libraries:



ggstatsplot Star 1767

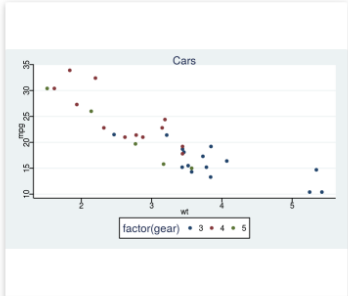
'ggstatsplot' provides a collection of functions to enhance 'ggplot2' plots with results from statistical tests.

- author: IndrajeetPatil
- tags: visualization, statistics
- js libraries:

esquisse Star 1674

Explore and Visualize Your Data Interactively with `ggplot2`

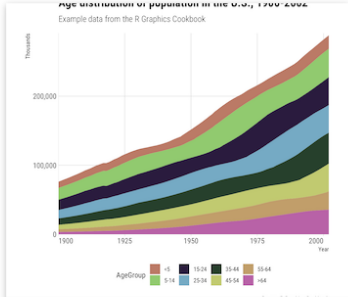
- author: dreamrs
- tags: visualization, interface
- js libraries:



ggthemes Star 1253

Some extra geoms, scales, and themes for ggplot.

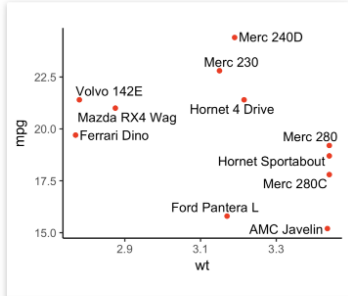
- author: jrnold
- tags: visualization, general, themes
- js libraries:



hrbrthemes Star 1148

A compilation of extra {ggplot2} themes, scales and utilities, including a spell check function for plot label fields and an overall emphasis on typography.

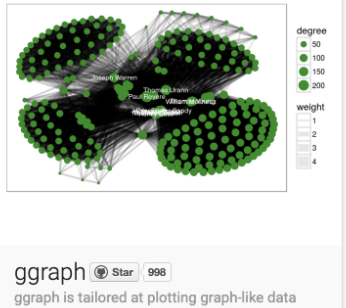
- author: hrbrmstr
- tags: theme, typography
- js libraries:



ggrepel Star 1127

Repel overlapping text labels away from each other.

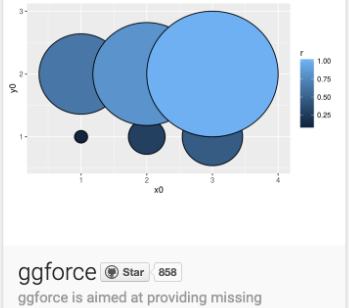
- author: slowkow
- tags: visualization, general
- js libraries:



ggraph Star 998

ggraph is tailored at plotting graph-like data structures (graphs, networks, trees, hierarchies...).

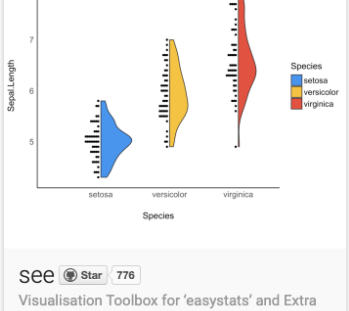
- author: thomasp85
- tags: visualization, general
- js libraries:



ggforce Star 858

ggforce is aimed at providing missing functionality to ggplot2 through the extension system introduced with ggplot2 v2.0.0.

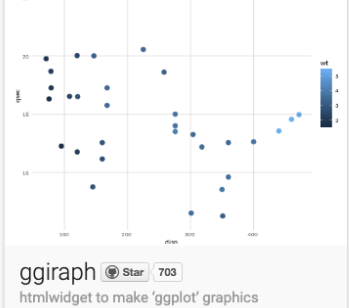
- author: thomasp85
- tags: visualization, general
- js libraries:



see Star 776

Visualisation Toolbox for 'easystats' and Extra Geoms, Themes and Color Palettes for 'ggplot2'

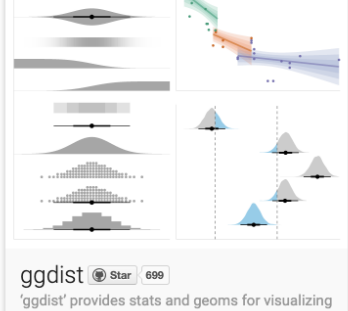
- author: easystats
- tags: visualizations, statistics
- js libraries:



ggiraph Star 703

htmlwidget to make 'ggplot' graphics interactive.

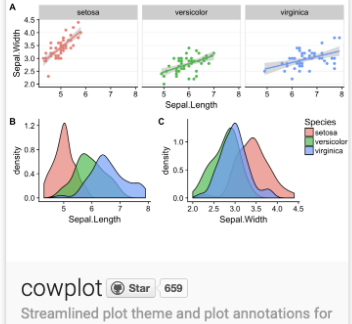
- author: davidgohel
- tags: visualization, general
- js libraries:



ggdist Star 699

'ggdist' provides stats and geoms for visualizing distributions and uncertainty.

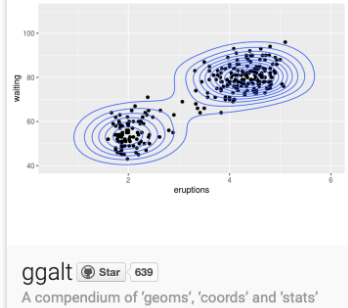
- author: mjskay
- tags: visualization, uncertainty, confidence, probability
- js libraries:



cowplot Star 659

Streamlined plot theme and plot annotations for 'ggplot2'

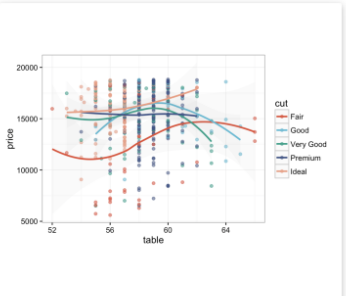
- author: clauswilke
- tags: visualization, general, themes
- js libraries:



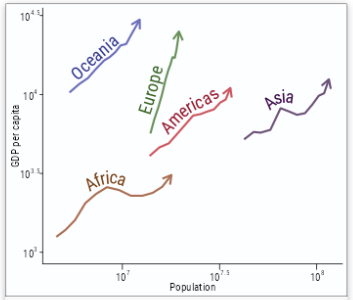
ggalt Star 639

A compendium of 'geoms', 'coords' and 'stats' for 'ggplot2'.

- author: hrbrmstr
- tags: visualization, general
- js libraries:

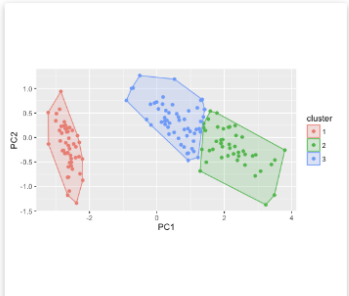


ggeei Star 504

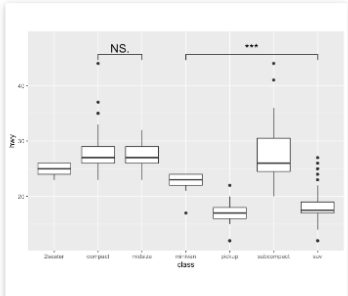


geomtextpath Star 500

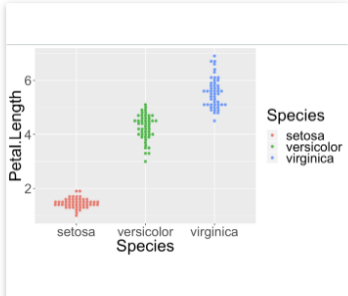
CCally Star 458



ggfortify Star 548

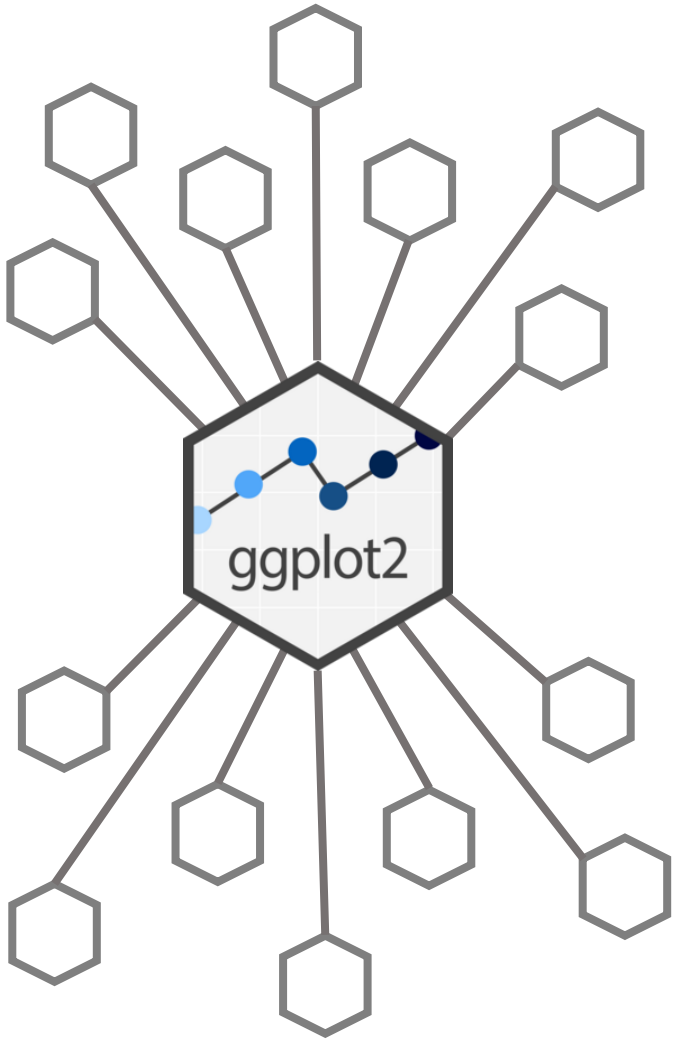


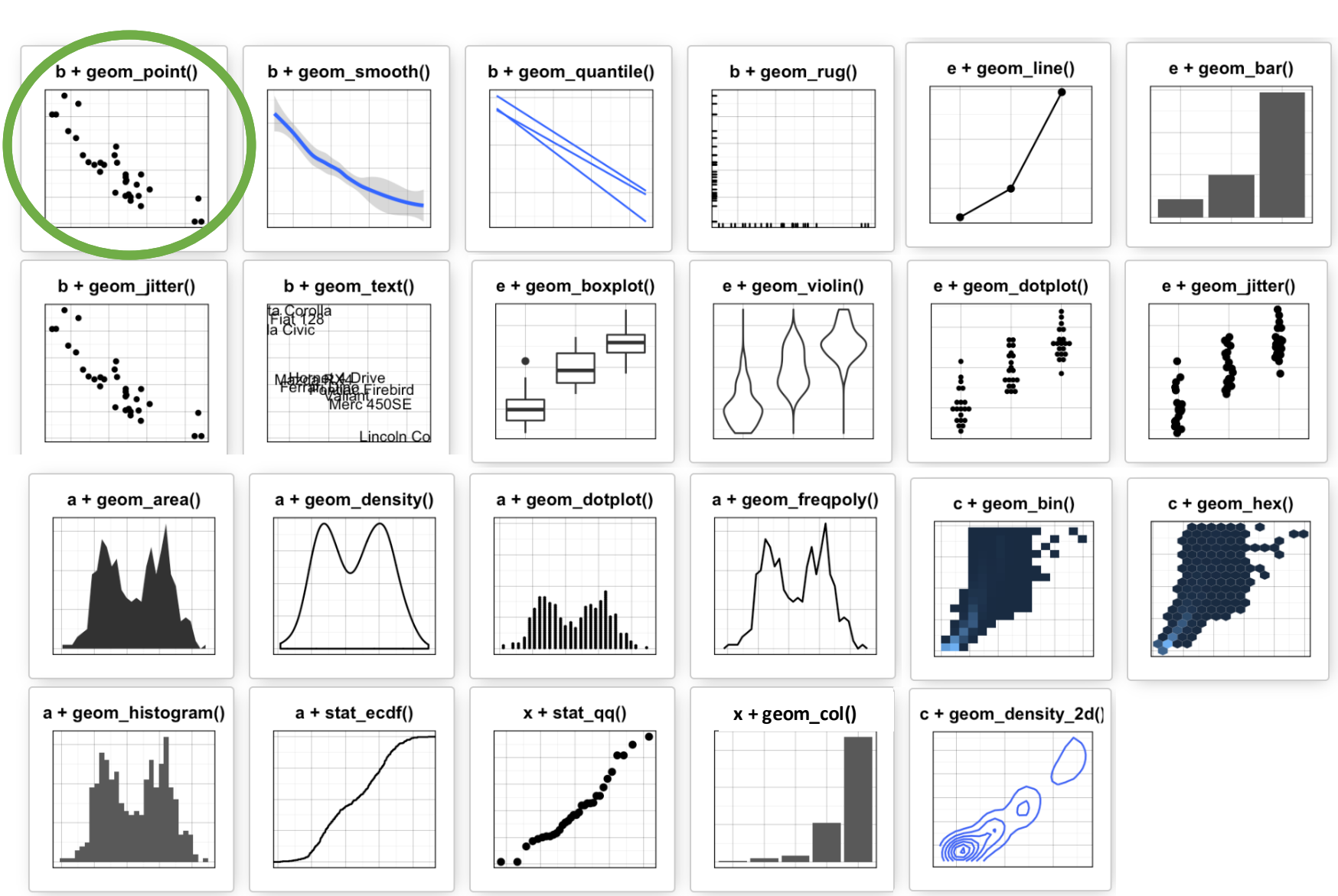
ggsignif Star 547

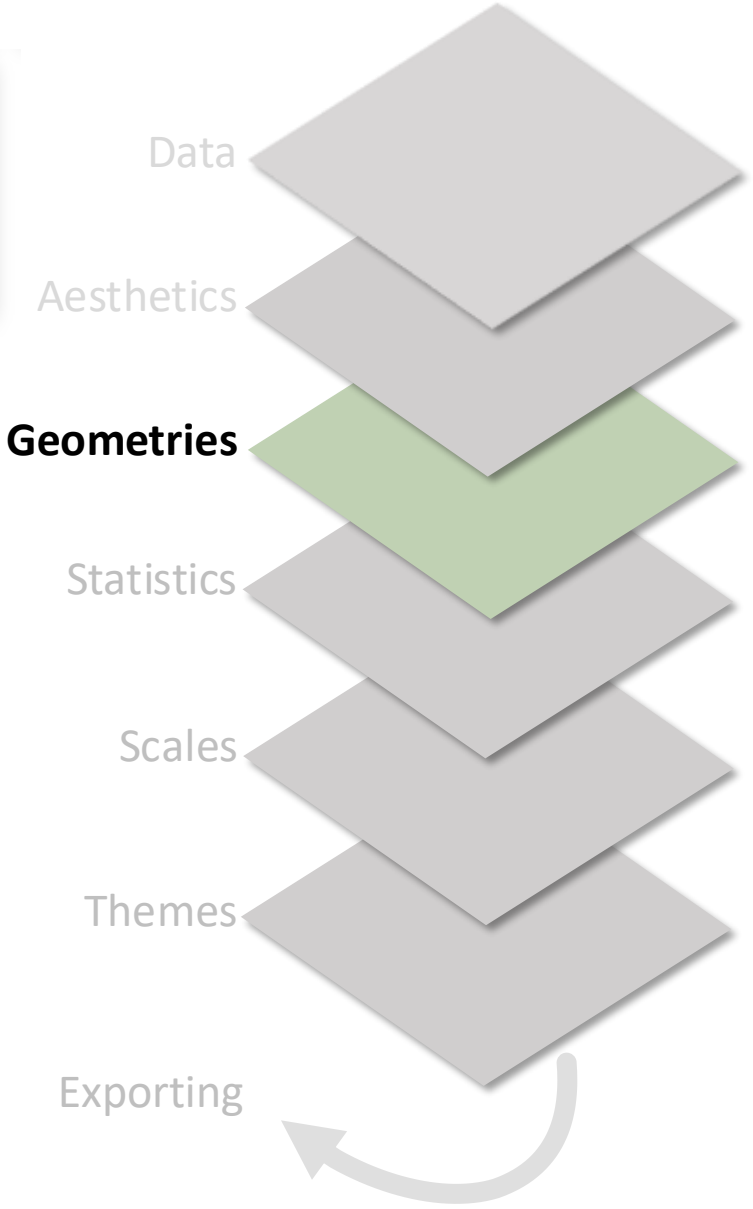


ggbeeswarm Star 507

ggbump Star 477

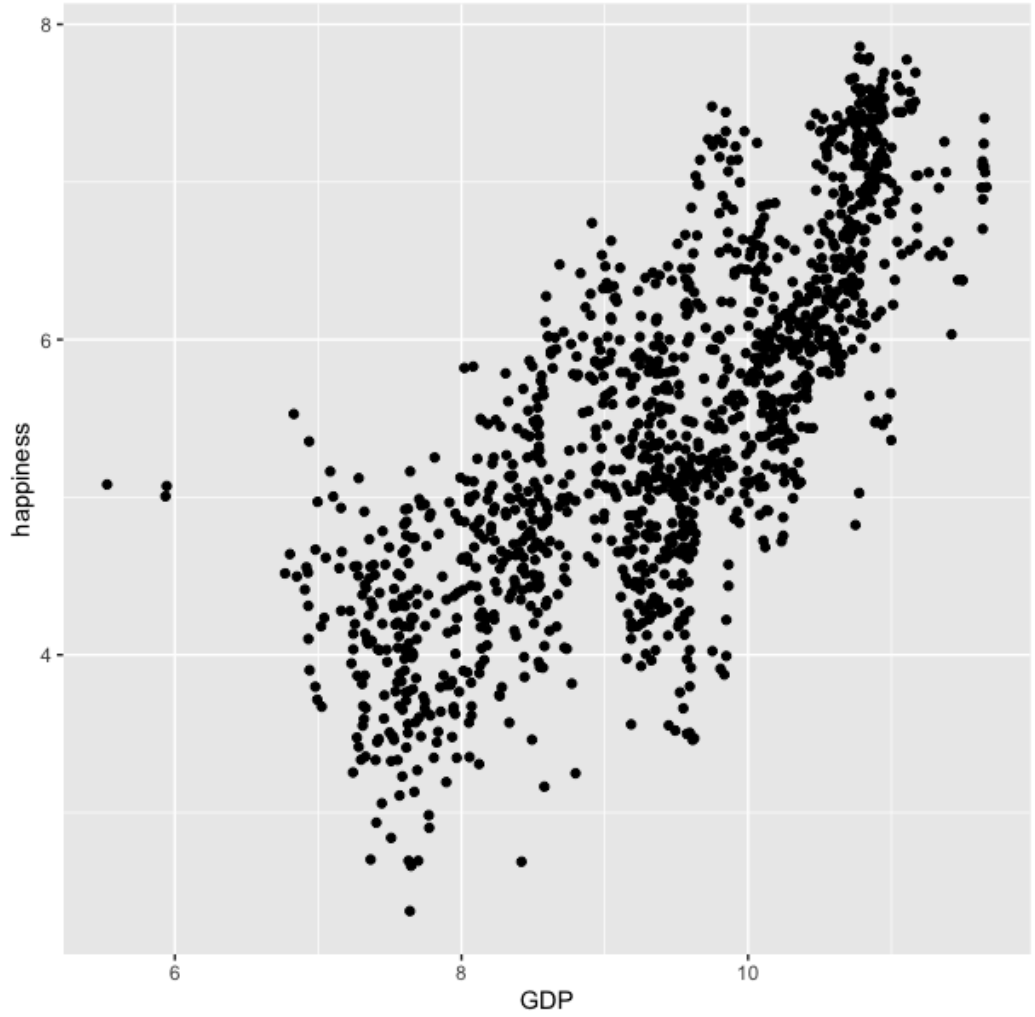

ggplot2


b + geom_point()
b + geom_smooth()
b + geom_quantile()
b + geom_rug()
e + geom_line()
e + geom_bar()
b + geom_jitter()
b + geom_text()
e + geom_boxplot()
e + geom_violin()
e + geom_dotplot()
e + geom_jitter()
a + geom_area()
a + geom_density()
a + geom_dotplot()
a + geom_freqpoly()
c + geom_bin()
c + geom_hex()
a + geom_histogram()
a + stat_ecdf()
x + stat_qq()
x + geom_col()
c + geom_density_2d()


Data
Aesthetics
Geometries
Statistics
Scales
Themes
Exporting

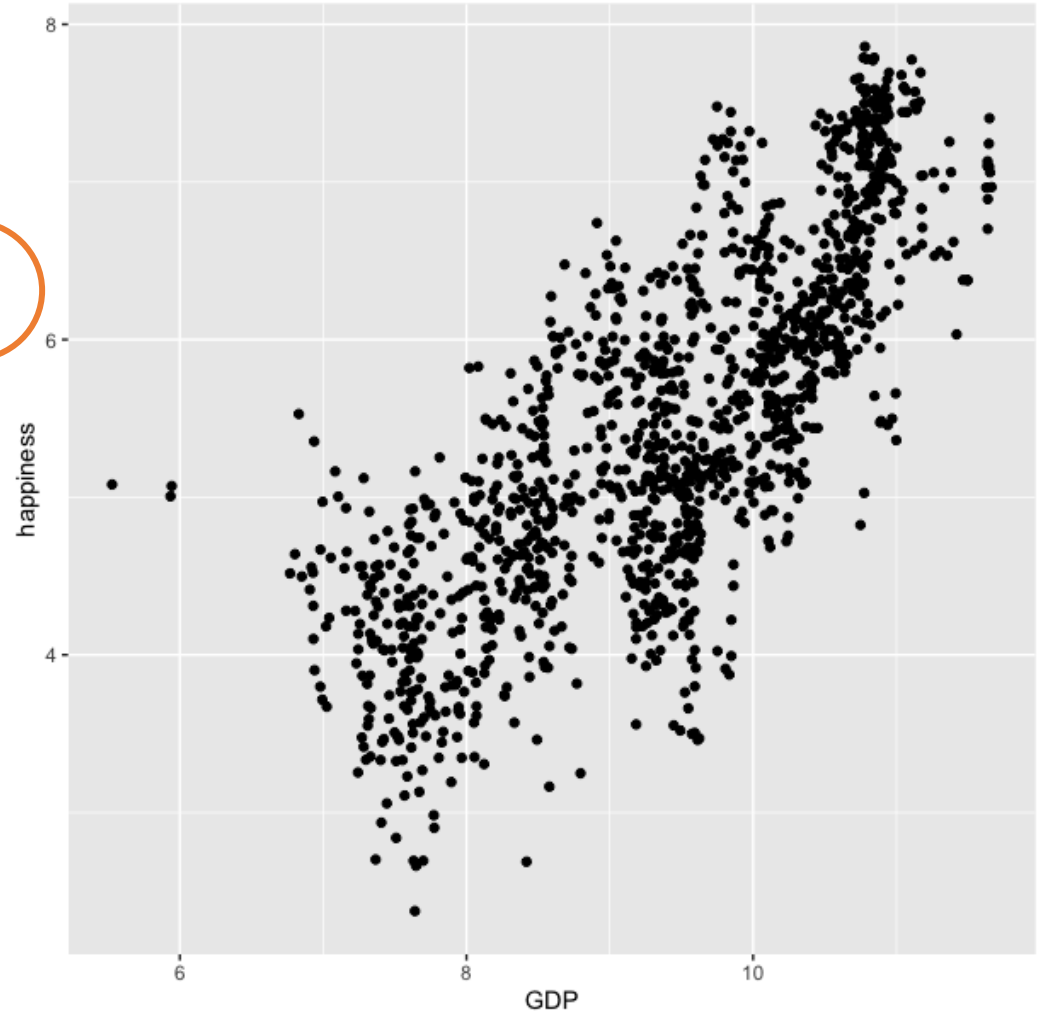
```
ggplot( data = happy ,
        mapping = aes(GDP, happiness) ) +
        geom_point()
```

```
ggplot( data = happy ,
        mapping = aes(GDP, happiness) ) +
        geom_point()
```



Chaining method

```
ggplot( data = happy ,
        mapping = aes(GDP, happiness) ) +
        geom_point()
```



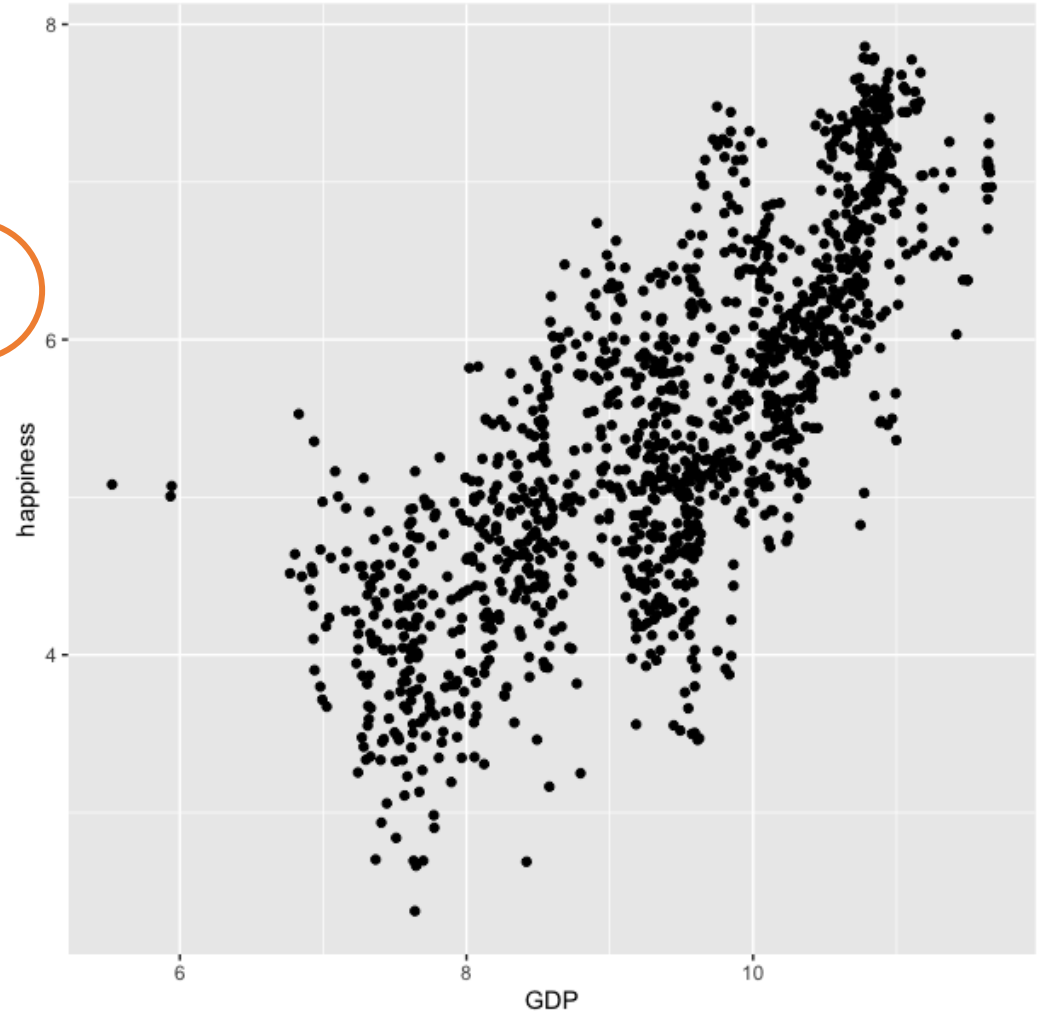
"and then…"

dplyr %>% or |>

ggplot +

Chaining method

```
ggplot( data = happy ,
        mapping = aes(GDP, happiness) ) +
        geom_point()
```



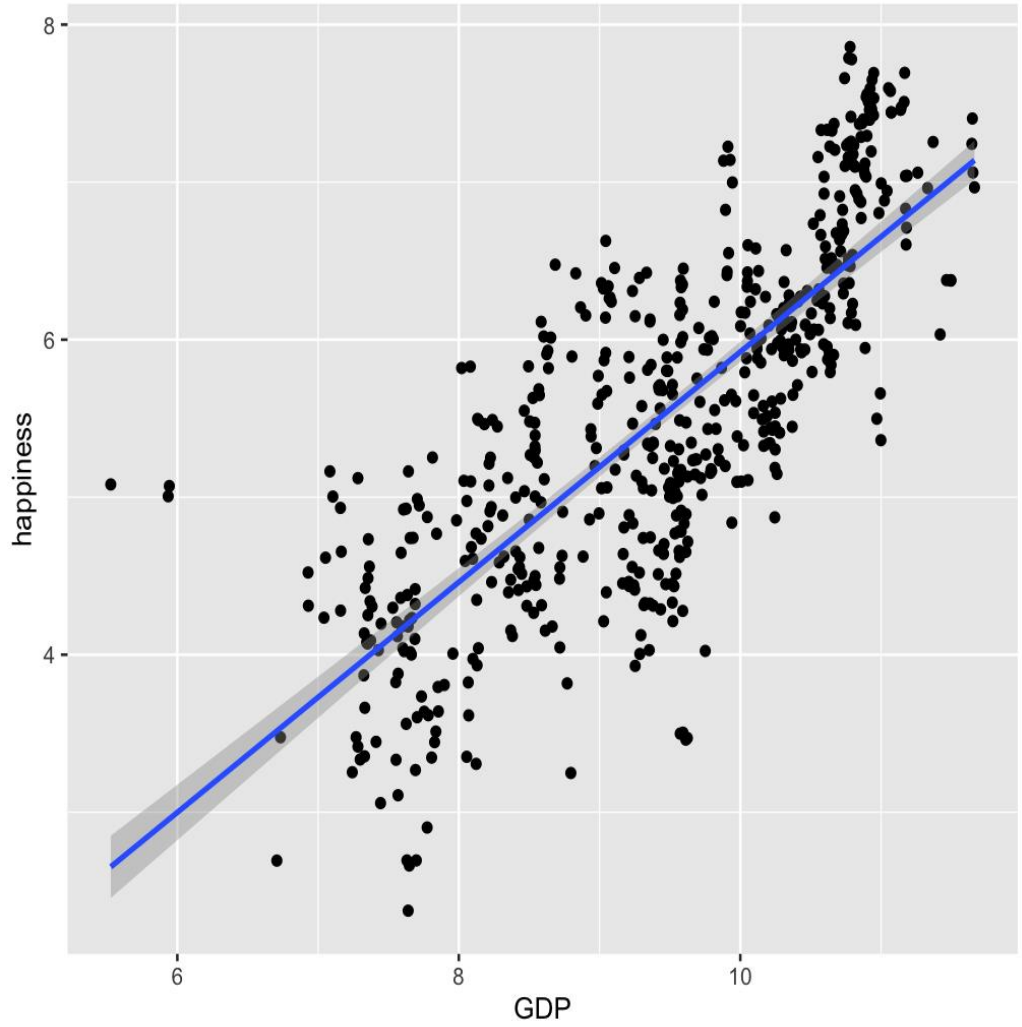
"and then…"

dplyr %>% or |>

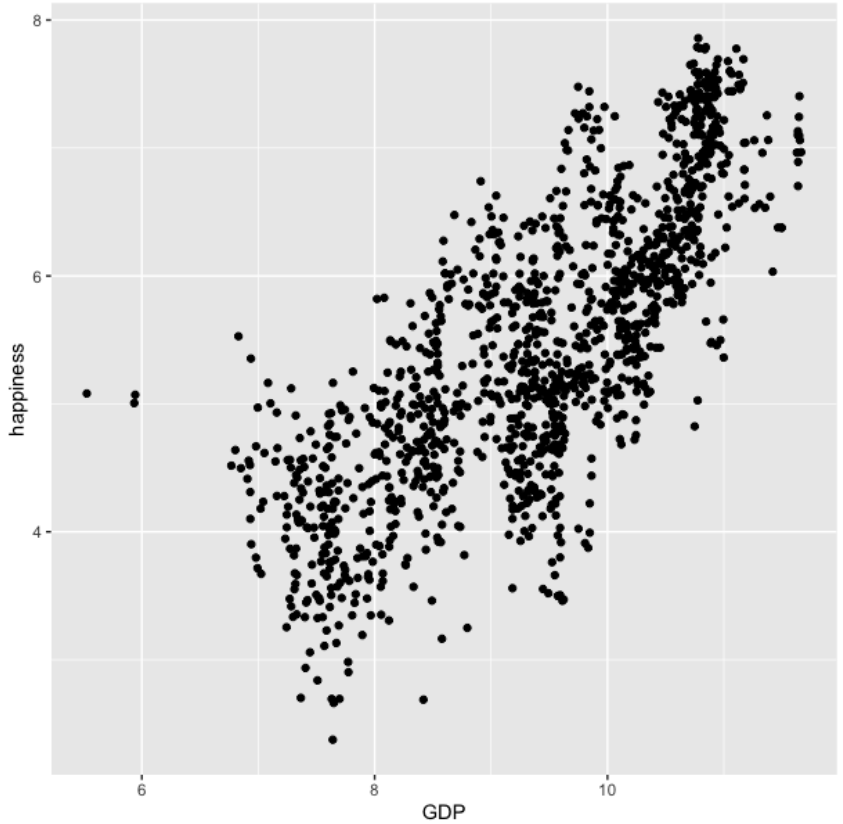
ggplot +

Chaining method

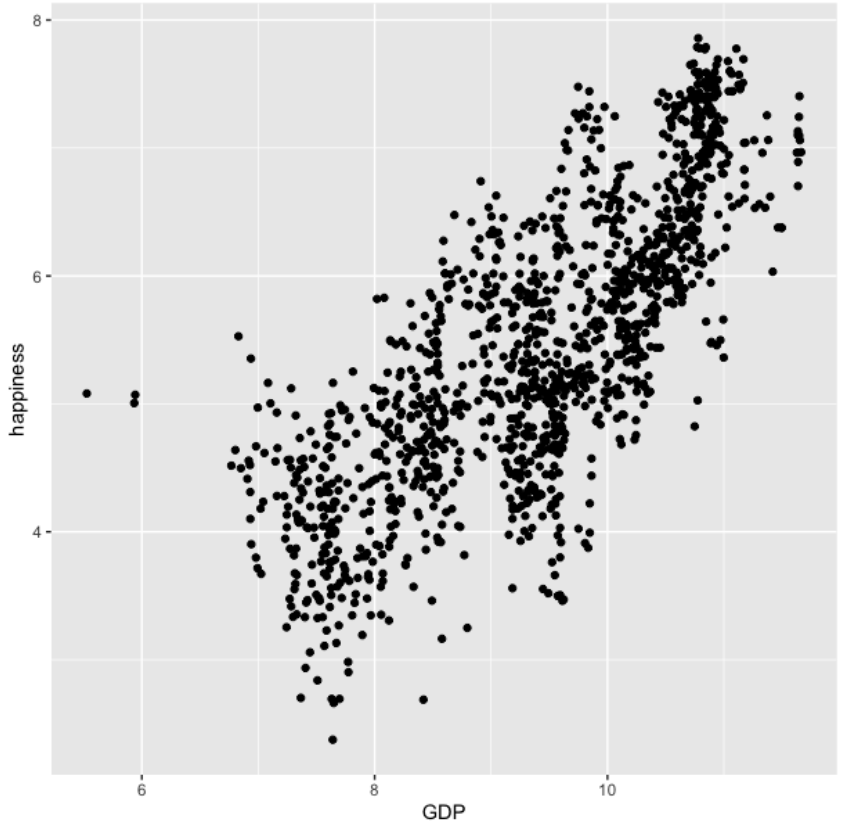
```
ggplot( data = happy ,
        mapping = aes(GDP, happiness) ) +
        geom_point() +
        geom_smooth(method="lm")
```



```
ggplot(  data = happy  ,  aes(x=GDP, y=happiness)  ) +
  geom_point()
```

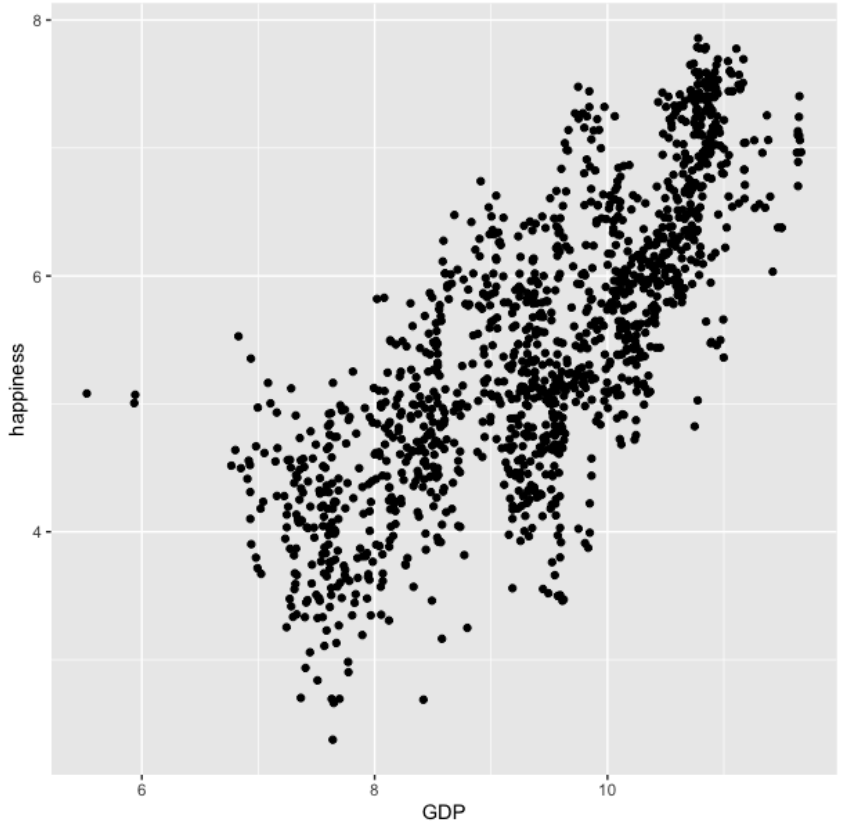


```
ggplot( data = happy , aes(x=GDP, y=happiness) ) +
  geom_point()
```



```
ggplot( data = happy , aes(x=GDP, y=happiness) ) +
  geom_point()
```

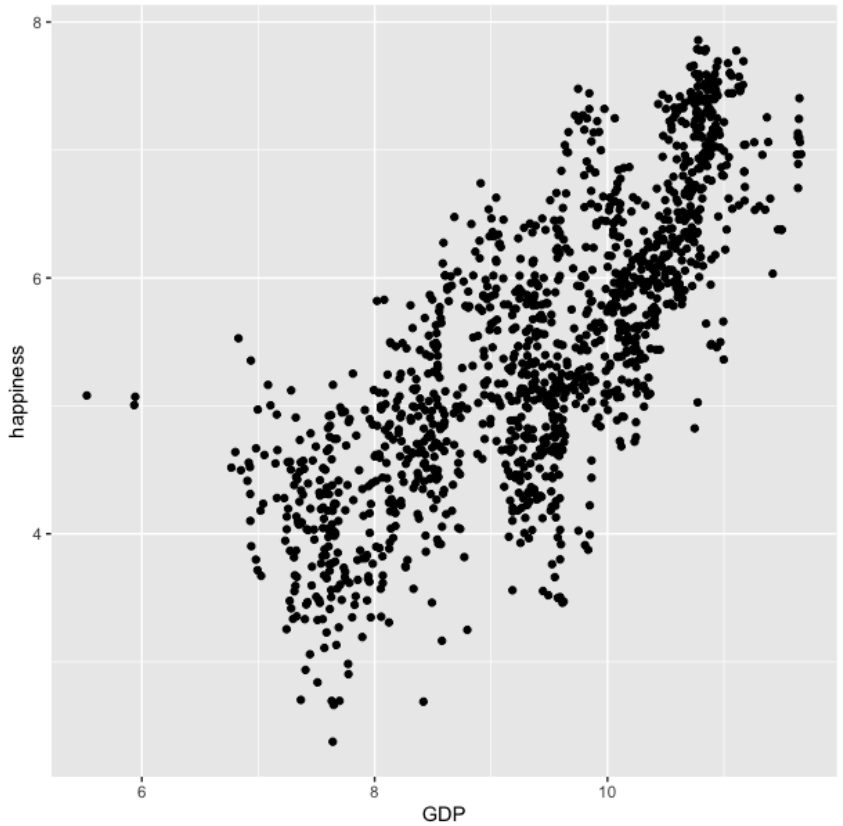
```
ggplot( happy , aes(GDP, happiness) ) +
  geom_point()
```



```
ggplot(  data = happy  ,
         aes(x=GDP, y=happiness)  ) +
         geom_point()
```

```
ggplot(  happy  ,  aes(GDP, happiness)  ) +
         geom_point()
```

```
ggplot(  ) +
         geom_point(  happy  ,  aes(GDP, happiness)  )
```

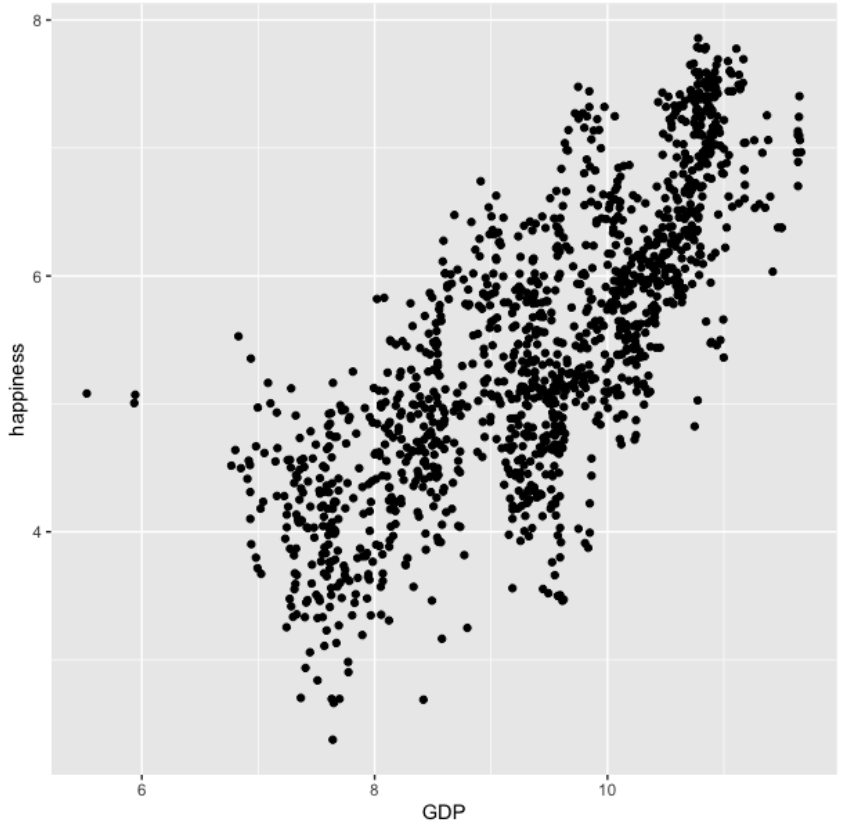


```
ggplot(  data = happy  ,
         aes(x=GDP, y=happiness)  ) +
         geom_point()
```

```
ggplot(  happy  ,  aes(GDP, happiness)  ) +
         geom_point()
```

```
ggplot(  ) +
    geom_point(  happy  ,  aes(GDP, happiness)  )
```

```
happy  %>%
ggplot(  aes(GDP, happiness)  ) +
         geom_point()
```

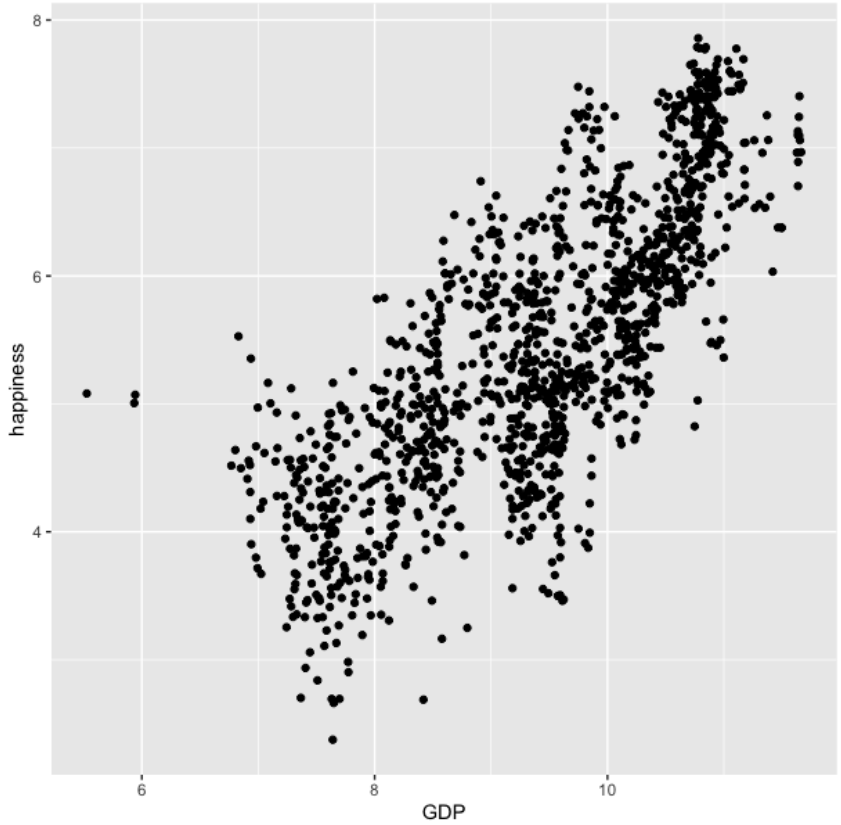


```
ggplot(  data = happy  ,
         aes(x=GDP, y=happiness)  ) +
         geom_point()
```

```
ggplot(  happy  ,  aes(GDP, happiness)  ) +
         geom_point()
```

```
ggplot(  ) +
    geom_point(  happy  ,  aes(GDP, happiness)  )
```

```
happy  %>%
ggplot(  aes(GDP, happiness)  ) +
         geom_point()
```



These all produce the same figure!

Aesthetics define any data that gets mapped to geometry (such as color, size, shape, etc.)

```
ggplot(data = yourData,
       aes(x = xaxis,y = yaxis))+
       geom_point()
```

Aesthetics define any data that gets mapped to geometry (such as color, size, shape, etc.)

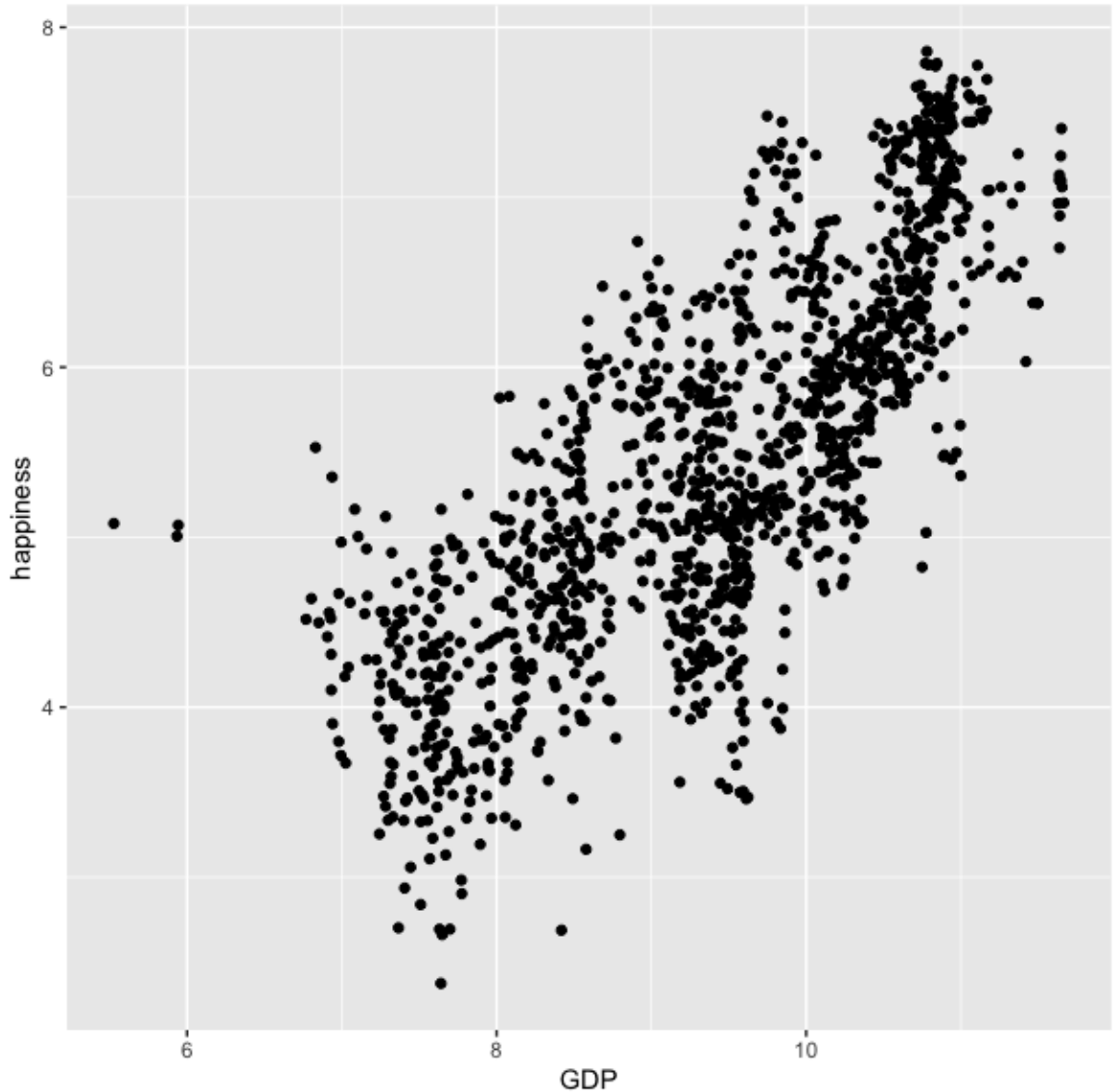
```
ggplot(data = happy,
       aes(x = happiness,y = GDP,
           shape=
           size=
           col=
       geom_point()
```

A tibble: 569 × 9

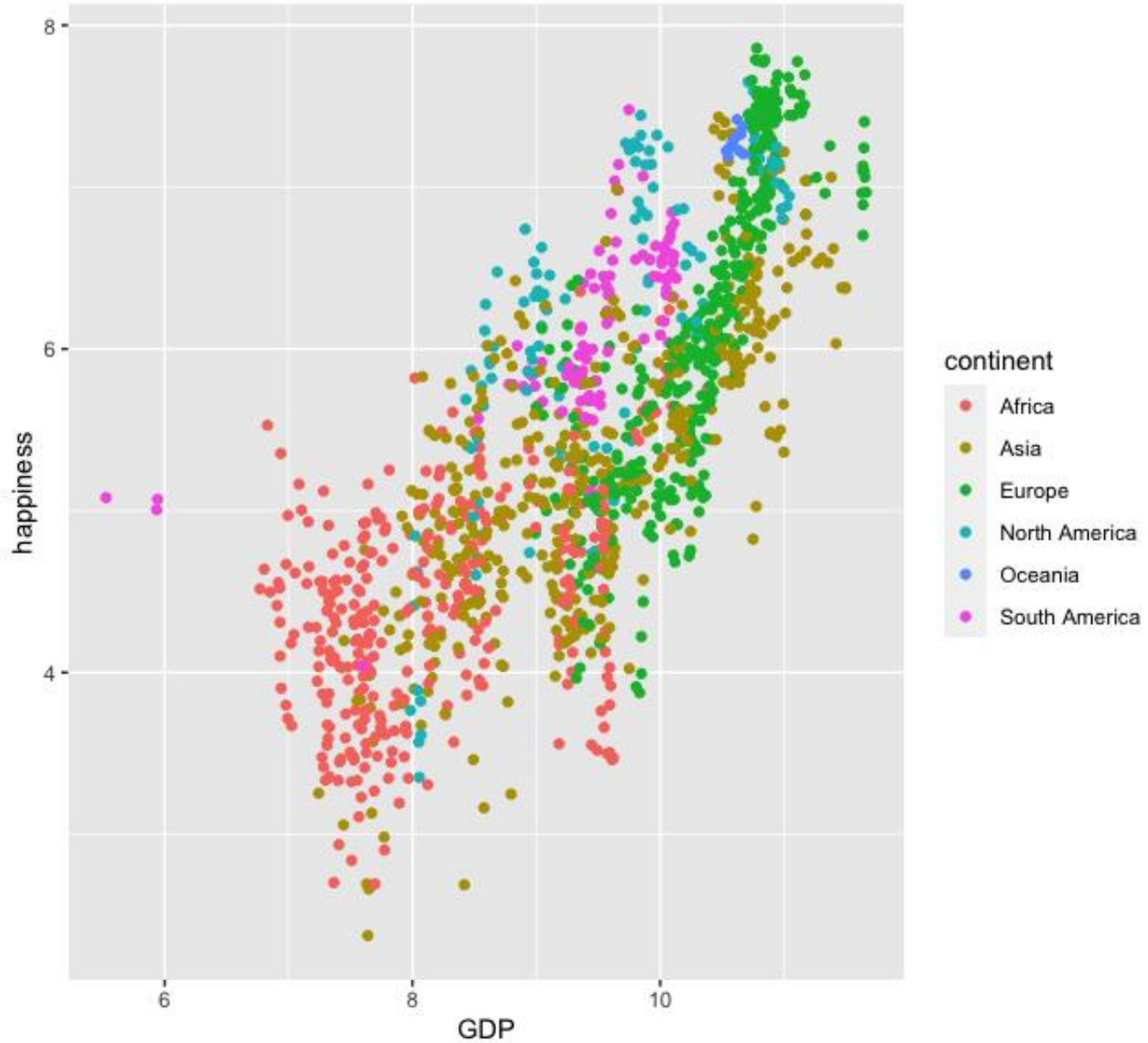
country_name <chr>	continent <chr>	year <dbl>	happiness <dbl>	GDP <dbl>
Afghanistan	Asia	2019	2.375	7.640
Afghanistan	Asia	2017	2.662	7.648
Central Africa…	Africa	2016	2.693	6.707
Afghanistan	Asia	2018	2.694	7.631
Zimbabwe	Africa	2019	2.694	7.698
South Sudan	Africa	2017	2.817	*NA*
South Sudan	Africa	2016	2.888	*NA*
Tanzania	Africa	2016	2.903	7.775
Yemen	Asia	2018	3.058	7.444
Rwanda	Africa	2017	3.108	7.568

1–10 of 569 rows | 1–5 of 9 columns

```
ggplot(happy, aes(GDP, happiness)) +
  geom_point()
```

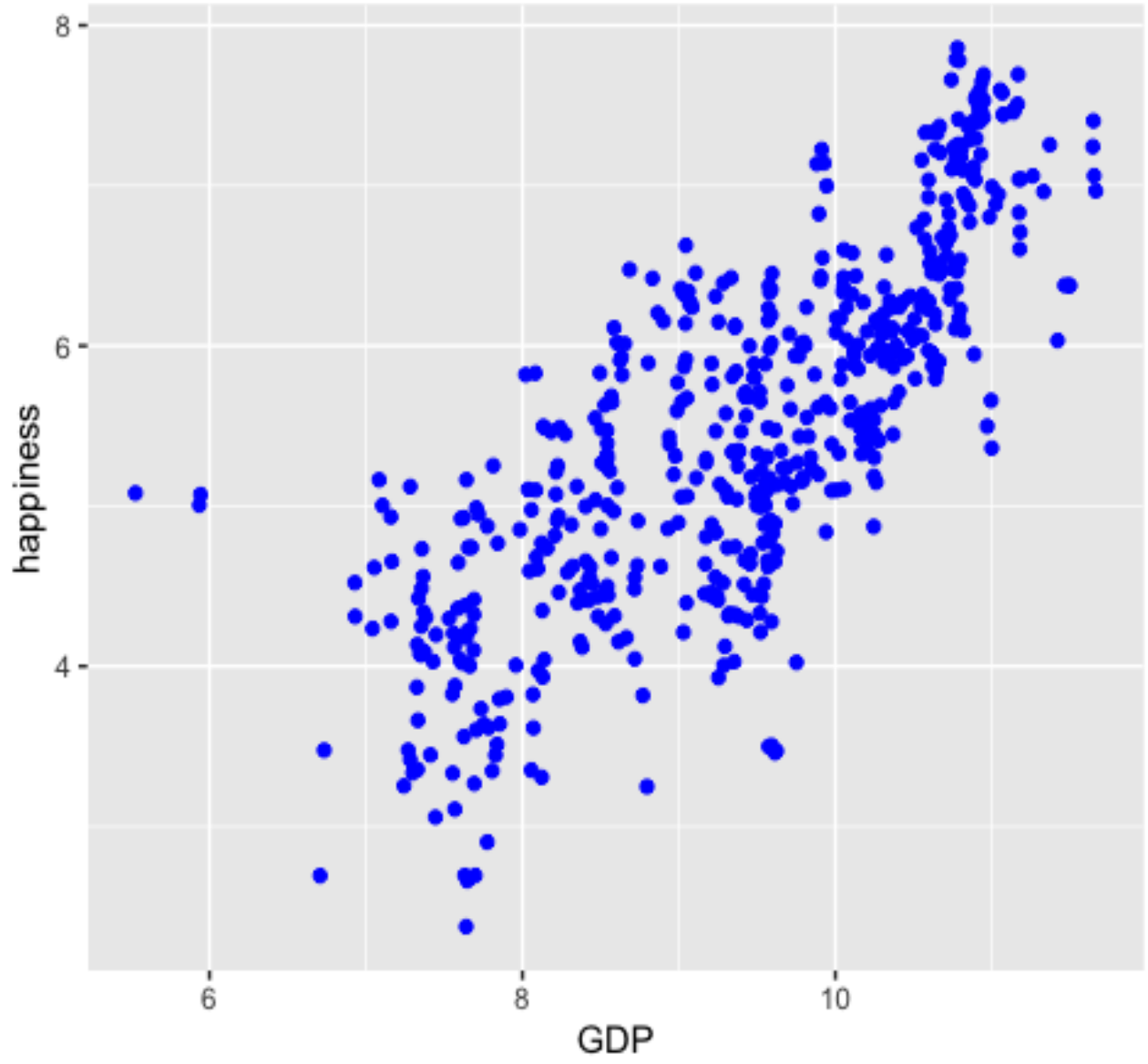


```
ggplot(happy, aes(GDP, happiness)) +
        geom_point(aes(col=continent))
```



```
ggplot(happy, aes(GDP, happiness)) +
        geom_point(aes(col=continent))
```

```
ggplot(happy, aes(GDP, happiness)) +
        geom_point(col='blue')
```



```
ggplot(happy, aes(GDP, happiness)) +
        geom_point(aes(col=continent))
```

```
ggplot(happy, aes(GDP, happiness)) +
        geom_point(col='blue')
```

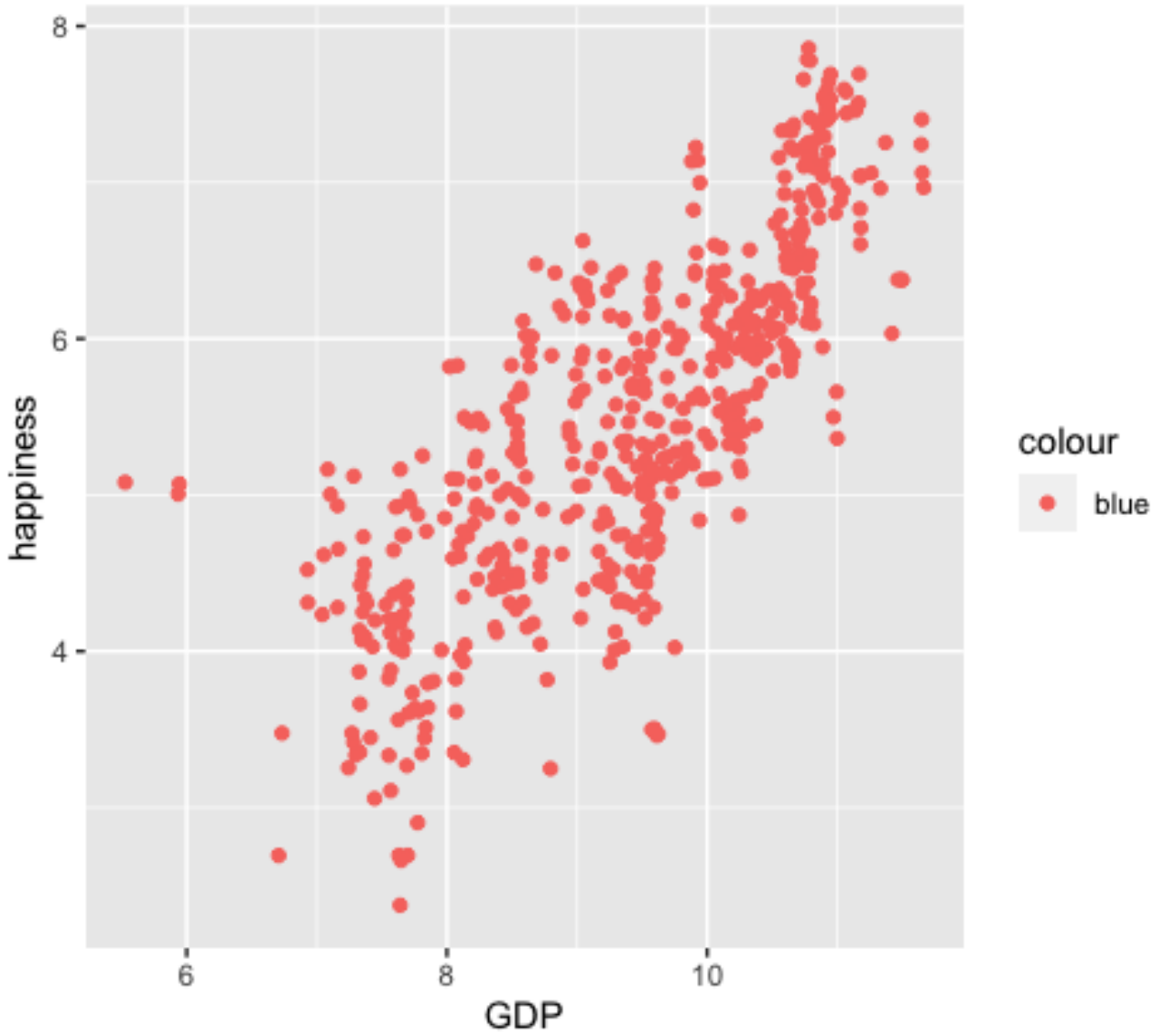
```
ggplot(happy, aes(GDP, happiness)) +
        geom_point(aes(col='blue'))
```

```
ggplot(happy, aes(GDP, happiness)) +
        geom_point(aes(col=continent))
```

```
ggplot(happy, aes(GDP, happiness)) +
        geom_point(col='blue')
```

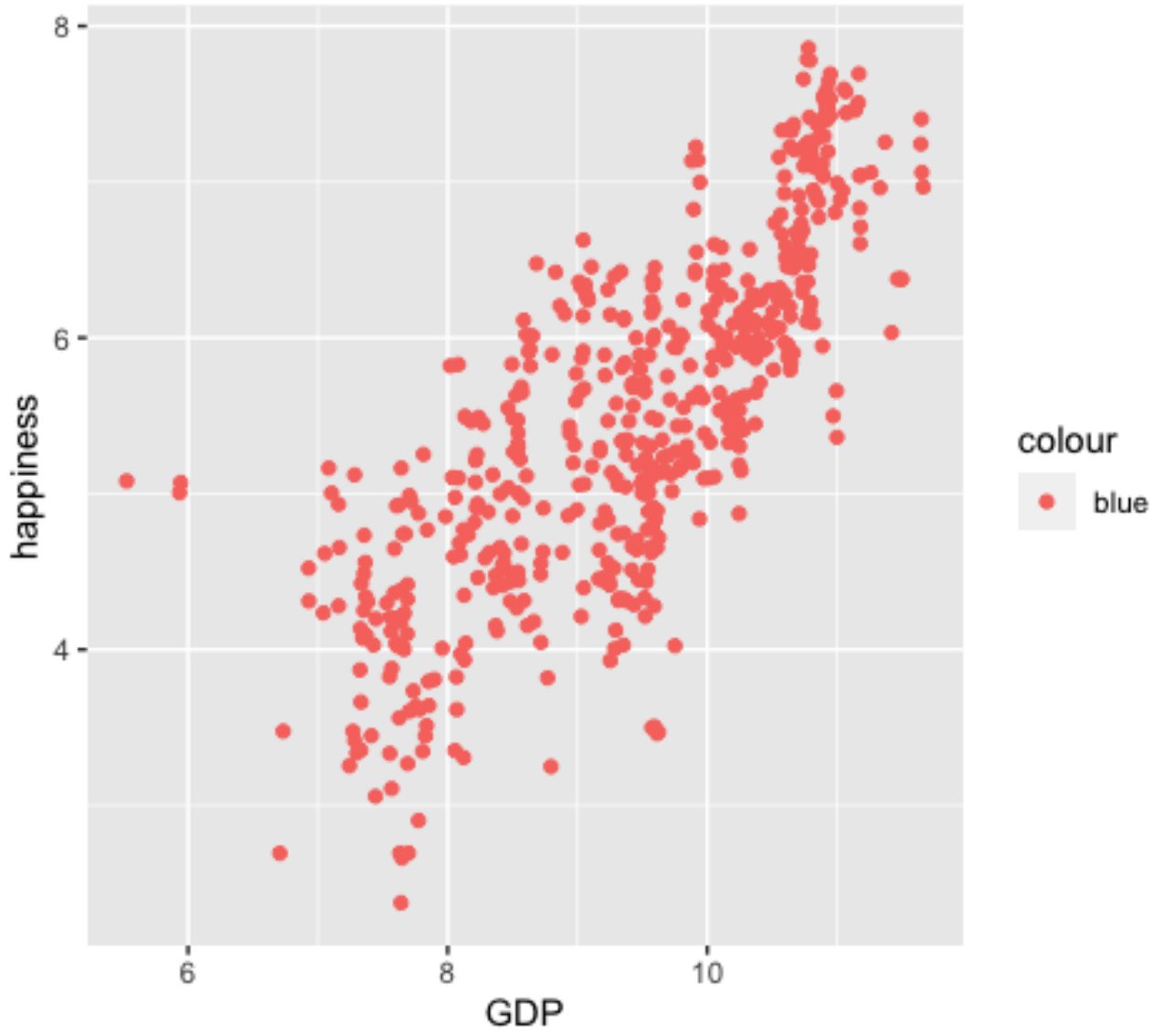
```
ggplot(happy, aes(GDP, happiness)) +
        geom_point(aes(col='blue'))
```

?!?



happiness	GDP	life_expect	colour
4.220	7.650	52.925	blue
2.662	7.648	53.250	blue
2.694	7.631	53.575	blue
2.375	7.640	53.900	blue
4.511	9.417	69.025	blue
4.640	9.455	69.050	blue

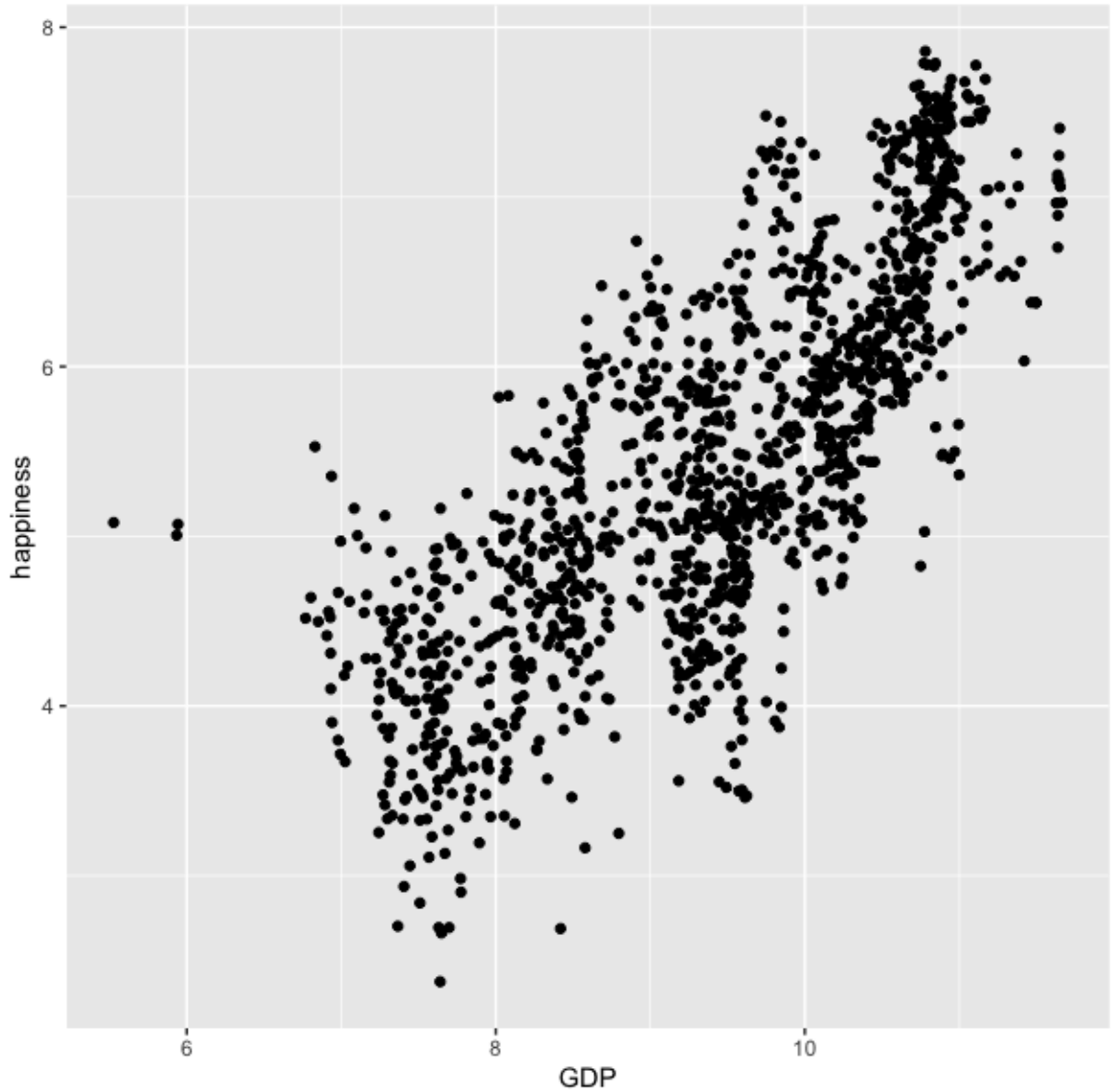
```
ggplot(happy, aes(GDP, happiness)) +
        geom_point(aes(col='blue'))
```



```
ggplot(happy, aes(GDP, happiness)) +
        geom_point()
```

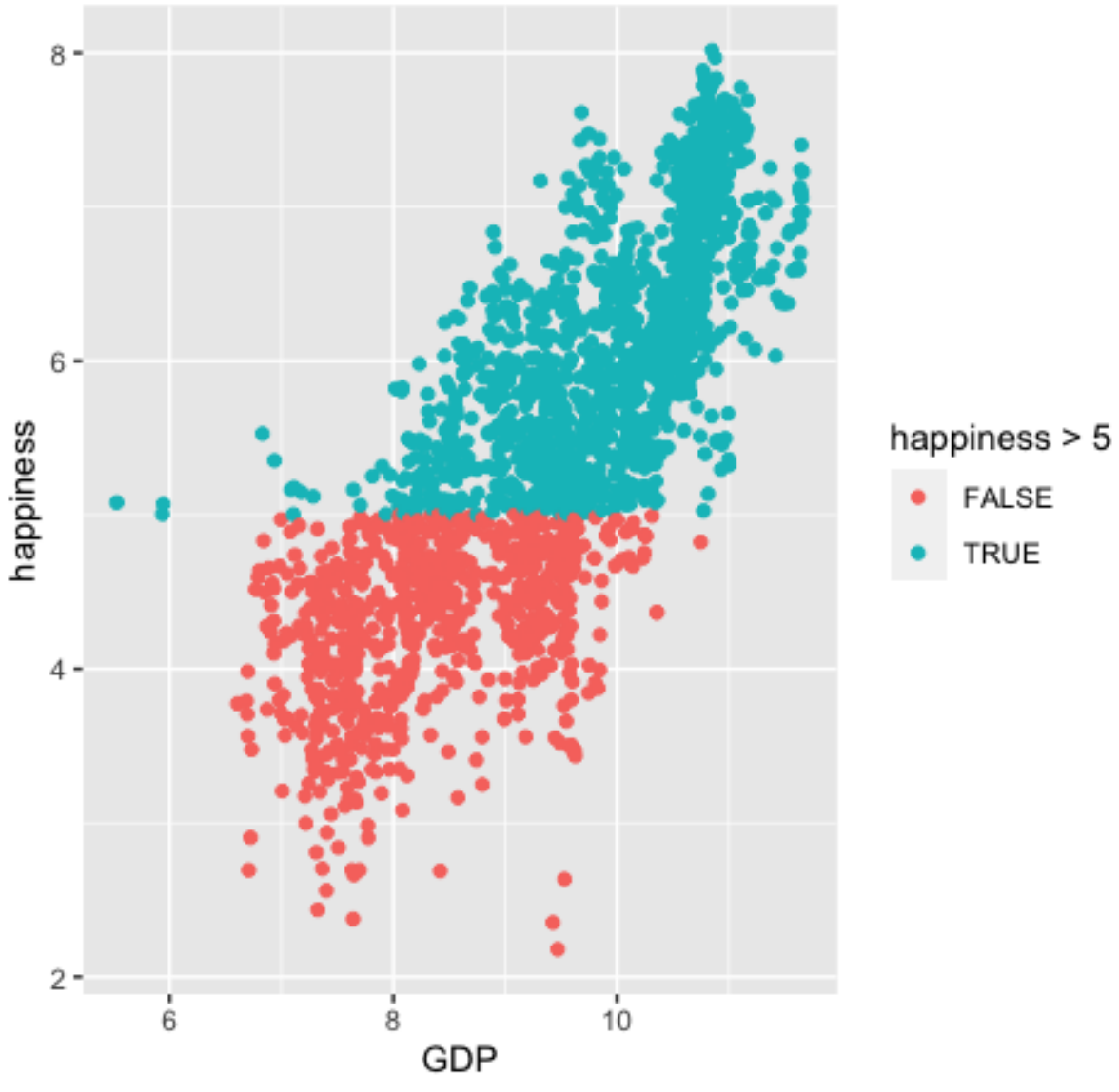
We can also set conditional formatting.

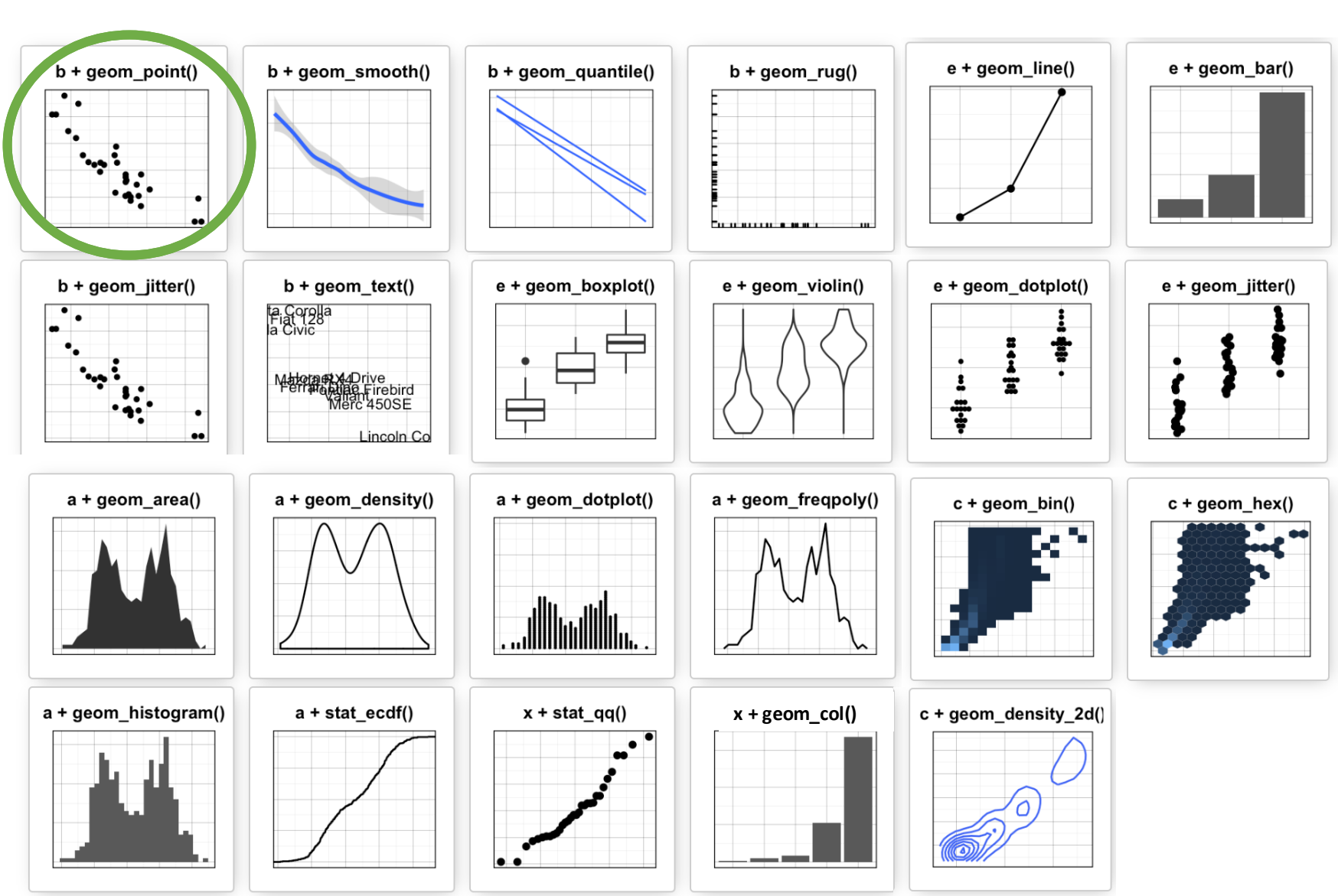
e.g. Use different colors for points where happiness is > 5

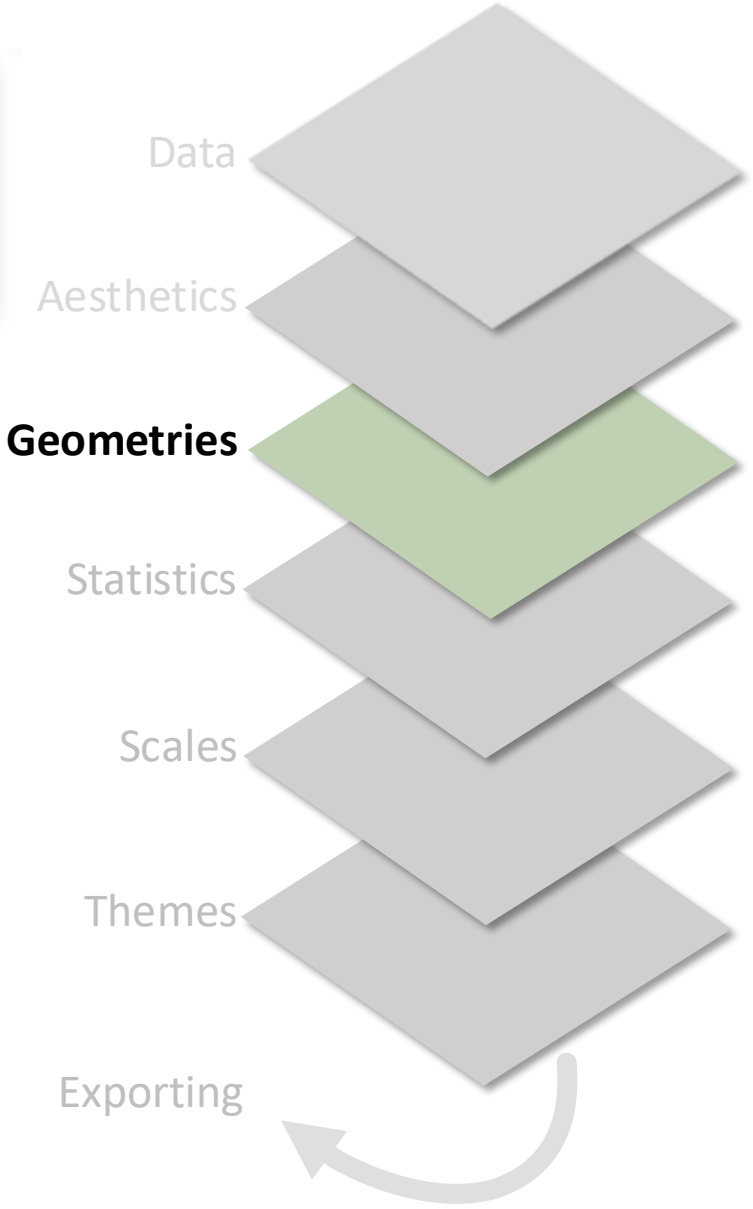


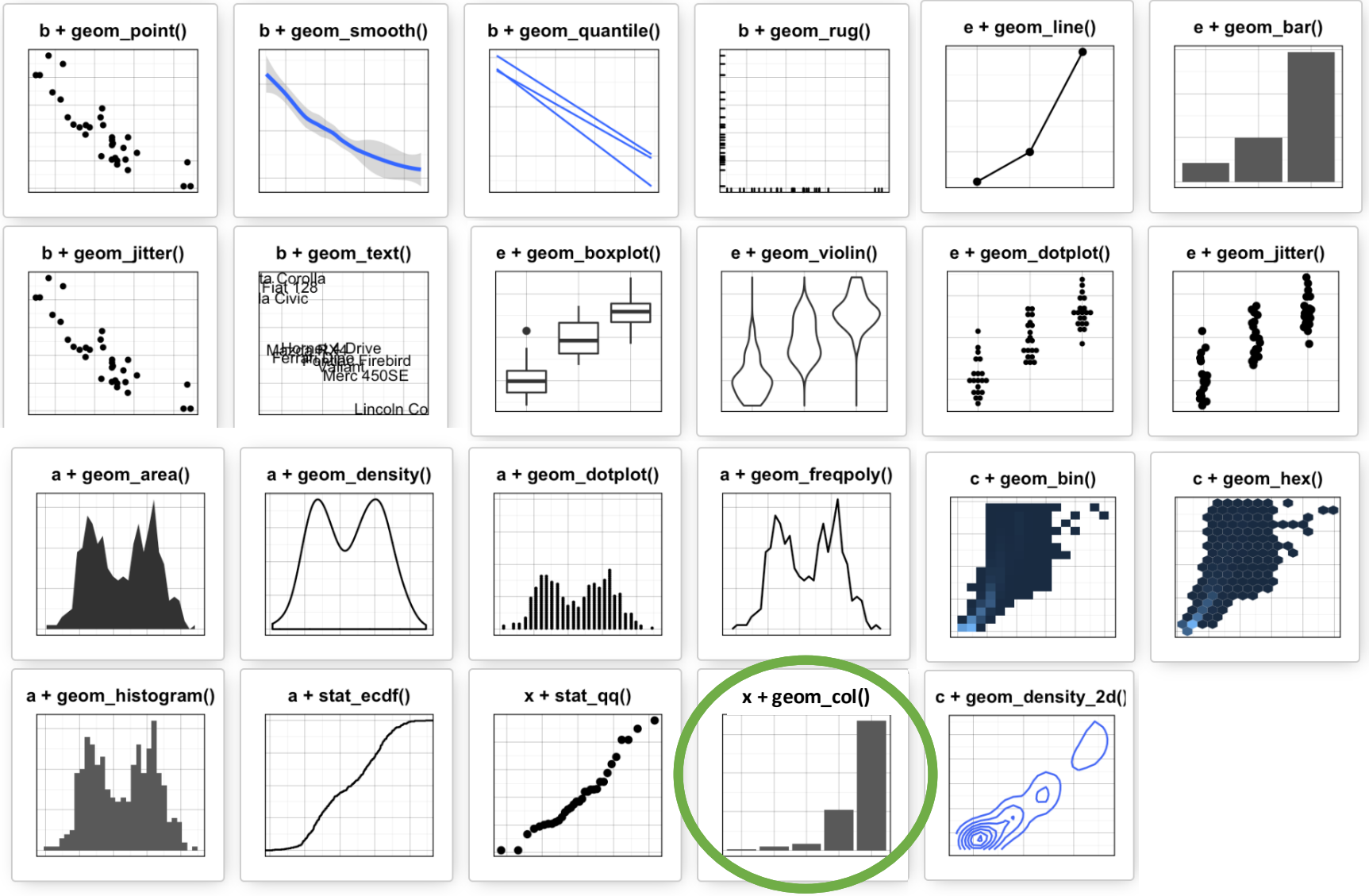
```
ggplot(happy, aes(GDP, happiness)) +
        geom_point()
```

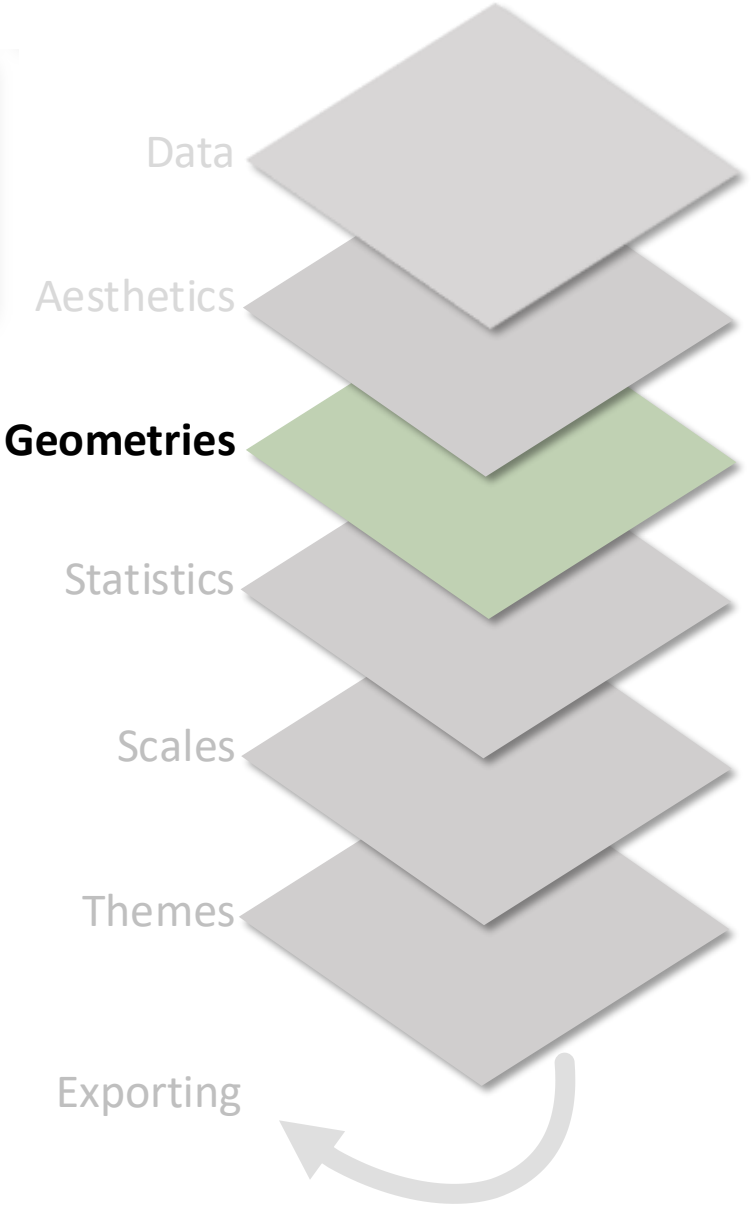
```
ggplot(happy, aes(GDP, happiness)) +
        geom_point(aes(col=happiness > 5)
```




b + geom_point()
b + geom_smooth()
b + geom_quantile()
b + geom_rug()
e + geom_line()
e + geom_bar()
b + geom_jitter()
b + geom_text()
e + geom_boxplot()
e + geom_violin()
e + geom_dotplot()
e + geom_jitter()
a + geom_area()
a + geom_density()
a + geom_dotplot()
a + geom_freqpoly()
c + geom_bin()
c + geom_hex()
a + geom_histogram()
a + stat_ecdf()
x + stat_qq()
x + geom_col()
c + geom_density_2d()


Data
Aesthetics
Geometries
Statistics
Scales
Themes
Exporting


b + geom_point()
b + geom_smooth()
b + geom_quantile()
b + geom_rug()
e + geom_line()
e + geom_bar()
b + geom_jitter()
b + geom_text()
e + geom_boxplot()
e + geom_violin()
e + geom_dotplot()
e + geom_jitter()
a + geom_area()
a + geom_density()
a + geom_dotplot()
a + geom_freqpoly()
c + geom_bin()
c + geom_hex()
a + geom_histogram()
a + stat_ecdf()
x + stat_qq()
x + geom_col()
c + geom_density_2d()


Data
Aesthetics
Geometries
Statistics
Scales
Themes
Exporting

Task: For the year **2019**, count the **number of observations per continent** in our dataset. Then plot this as a bar chart.

Task: For the year **2019**, count the **number of observations per continent** in our dataset. Then plot this as a bar chart.

```
happy
```

A tibble: 569 × 9

country_name <chr>	continent <chr>	year <dbl>	happiness <dbl>	GDP <dbl>
Afghanistan	Asia	2019	2.375	7.640
Afghanistan	Asia	2017	2.662	7.648
Central Africa…	Africa	2016	2.693	6.707
Afghanistan	Asia	2018	2.694	7.631
Zimbabwe	Africa	2019	2.694	7.698
South Sudan	Africa	2017	2.817	*NA*
South Sudan	Africa	2016	2.888	*NA*
Tanzania	Africa	2016	2.903	7.775
Yemen	Asia	2018	3.058	7.444
Rwanda	Africa	2017	3.108	7.568

Task: For the year **2019**, count the **number of observations per continent** in our dataset. Then plot this as a bar chart.

```
happy |>
  filter(year == 2019)
```

A tibble: 569 × 9

country_name <chr>	continent <chr>	year <dbl>	happiness <dbl>	GDP <dbl>
Afghanistan	Asia	2019	2.375	7.640
Afghanistan	Asia	2017	2.662	7.648
Central Africa…	Africa	2016	2.693	6.707
Afghanistan	Asia	2018	2.694	7.631
Zimbabwe	Africa	2019	2.694	7.698
South Sudan	Africa	2017	2.817	*NA*
South Sudan	Africa	2016	2.888	*NA*
Tanzania	Africa	2016	2.903	7.775
Yemen	Asia	2018	3.058	7.444
Rwanda	Africa	2017	3.108	7.568

Task: For the year **2019**, count the **number of observations per continent** in our dataset. Then plot this as a bar chart.

```
happy |>
  filter(year == 2019)
```

country_name <chr>	continent <chr>	year <dbl>	happiness <dbl>	GDP <dbl>
Afghanistan	Asia	2019	2.375	7.640
Zimbabwe	Africa	2019	2.694	7.698
India	Asia	2019	3.249	8.796
Rwanda	Africa	2019	3.268	7.692
Zambia	Africa	2019	3.307	8.123
Sierra Leone	Africa	2019	3.447	7.412
Botswana	Africa	2019	3.471	9.624
Lesotho	Africa	2019	3.512	7.837
Tanzania	Africa	2019	3.640	7.855
Malawi	Africa	2019	3.869	7.325

Task: For the year **2019**, count the **number of observations per continent** in our dataset. Then plot this as a bar chart.

```
happy |>
  filter(year == 2019) |>
  group_by(continent)
```

country_name <chr>	continent <chr>	year <dbl>	happiness <dbl>	GDP <dbl>
Afghanistan	Asia	2019	2.375	7.640
Zimbabwe	Africa	2019	2.694	7.698
India	Asia	2019	3.249	8.796
Rwanda	Africa	2019	3.268	7.692
Zambia	Africa	2019	3.307	8.123
Sierra Leone	Africa	2019	3.447	7.412
Botswana	Africa	2019	3.471	9.624
Lesotho	Africa	2019	3.512	7.837
Tanzania	Africa	2019	3.640	7.855
Malawi	Africa	2019	3.869	7.325

Task: For the year **2019**, count the **number of observations per continent** in our dataset. Then plot this as a bar chart.

```
happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n())
```

country_name <chr>	continent <chr>	year <dbl>	happiness <dbl>	GDP <dbl>
Afghanistan	Asia	2019	2.375	7.640
Zimbabwe	Africa	2019	2.694	7.698
India	Asia	2019	3.249	8.796
Rwanda	Africa	2019	3.268	7.692
Zambia	Africa	2019	3.307	8.123
Sierra Leone	Africa	2019	3.447	7.412
Botswana	Africa	2019	3.471	9.624
Lesotho	Africa	2019	3.512	7.837
Tanzania	Africa	2019	3.640	7.855
Malawi	Africa	2019	3.869	7.325

Task: For the year **2019**, count the **number of observations per continent** in our dataset. Then plot this as a bar chart.

```
happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n())
```

A tibble: 6 × 2

continent <chr>	n <int>
Africa	40
Asia	42
Europe	38
North America	11
Oceania	2
South America	10

6 rows

Task: For the year **2019**, count the **number of observations per continent** in our dataset. Then plot this as a bar chart.

```
happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot( )
```

A tibble: 6 × 2

continent <chr>	n <int>
Africa	40
Asia	42
Europe	38
North America	11
Oceania	2
South America	10

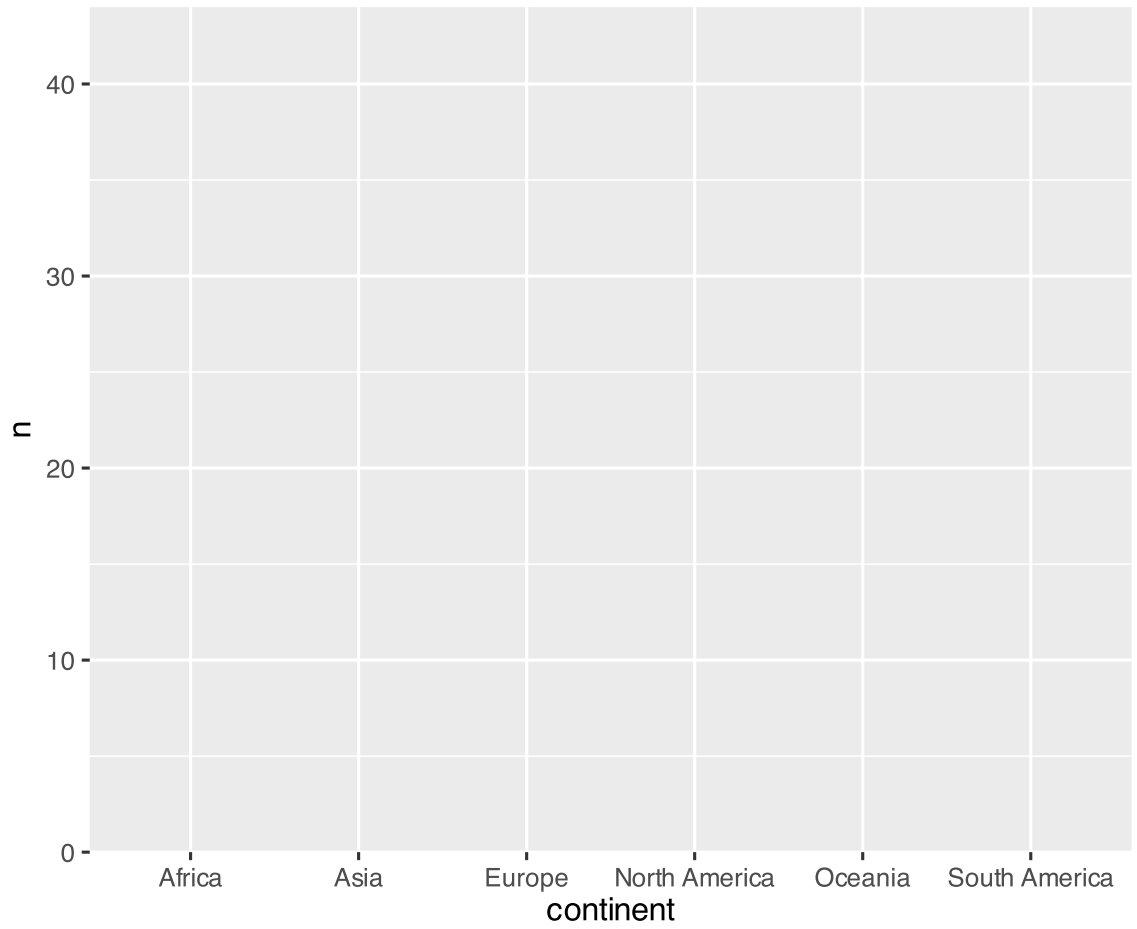
6 rows

Task: For the year **2019**, count the **number of observations per continent** in our dataset. Then plot this as a bar chart.

```
happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n)
```

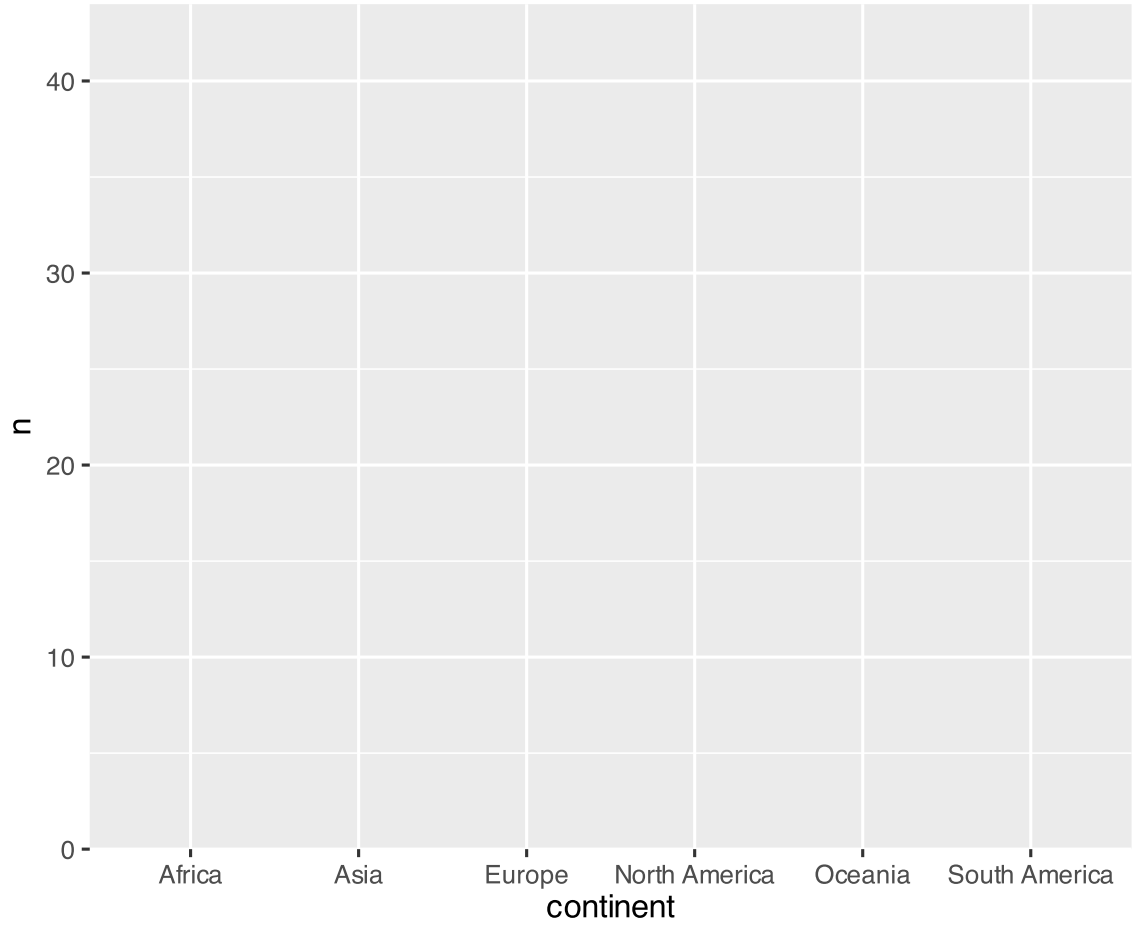
Task: For the year **2019**, count the **number of observations per continent** in our dataset. Then plot this as a bar chart.

```
happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n)
```



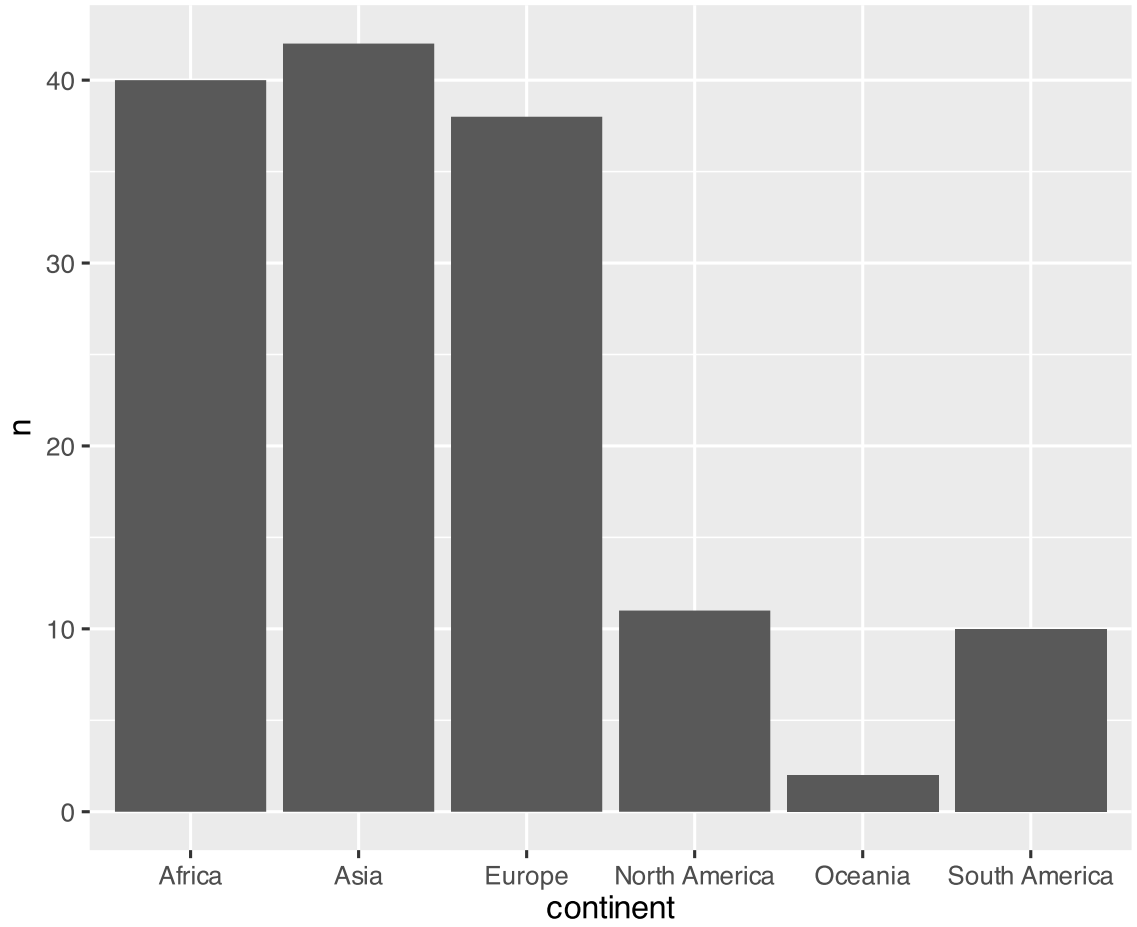
Task: For the year **2019**, count the **number of observations per continent** in our dataset. Then plot this as a bar chart.

```
happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col()
```

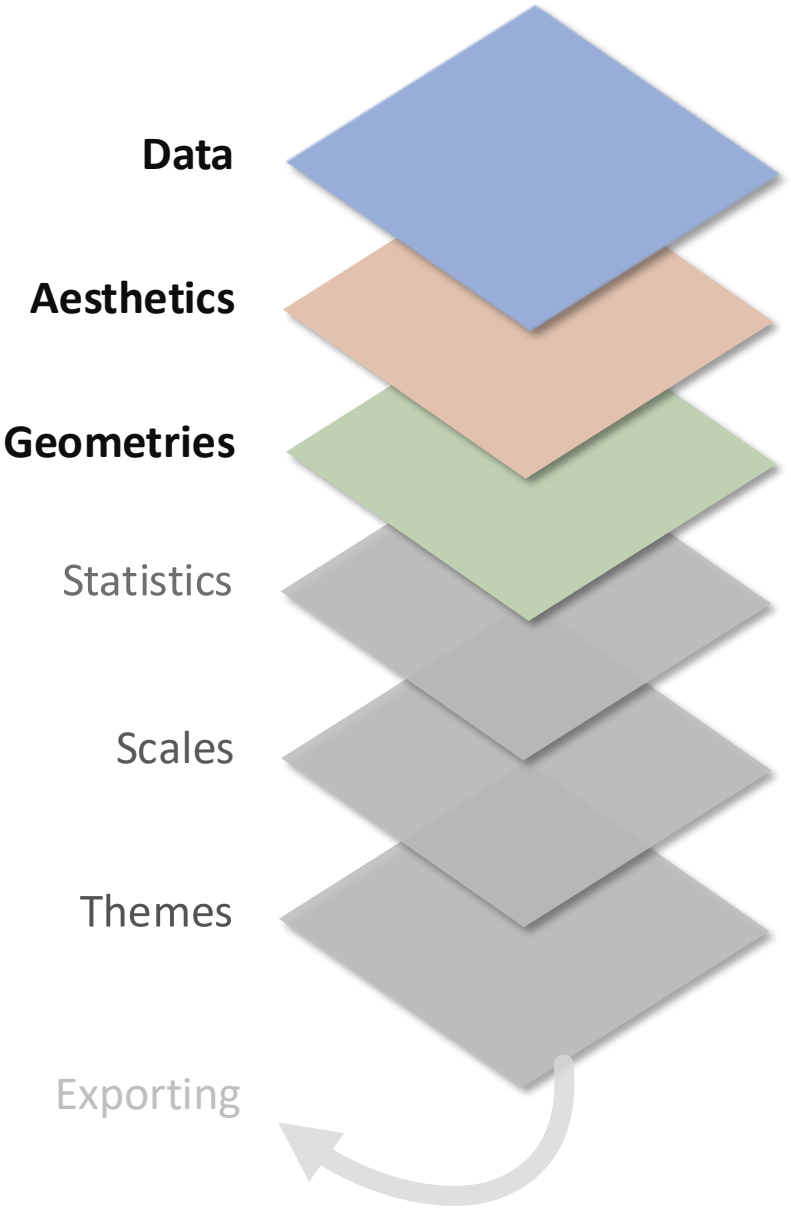


Task: For the year **2019**, count the **number of observations per continent** in our dataset. Then plot this as a bar chart.

```
happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col()
```

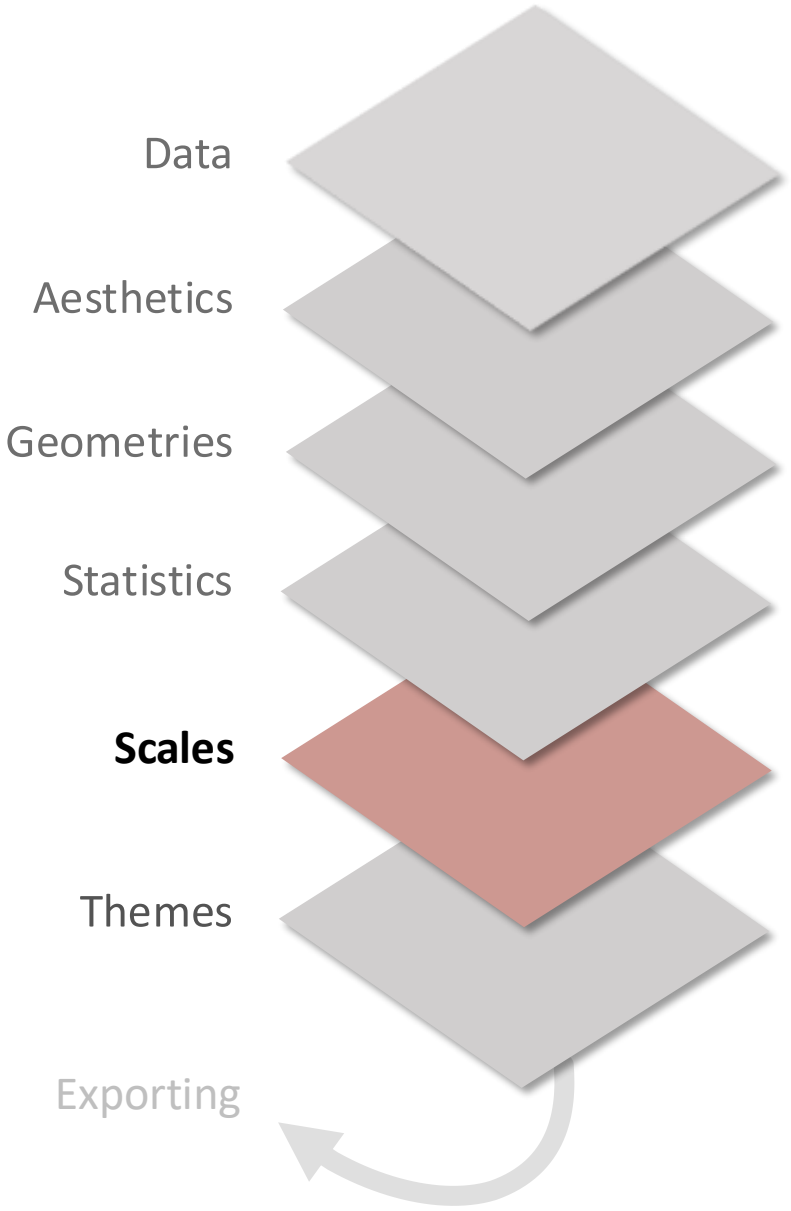


```
ggplot( data = yourdata ,
        mapping = aes(x axis, y axis) ) +
        geometry function +
        statistics function +
        scales function +
        themes function
```



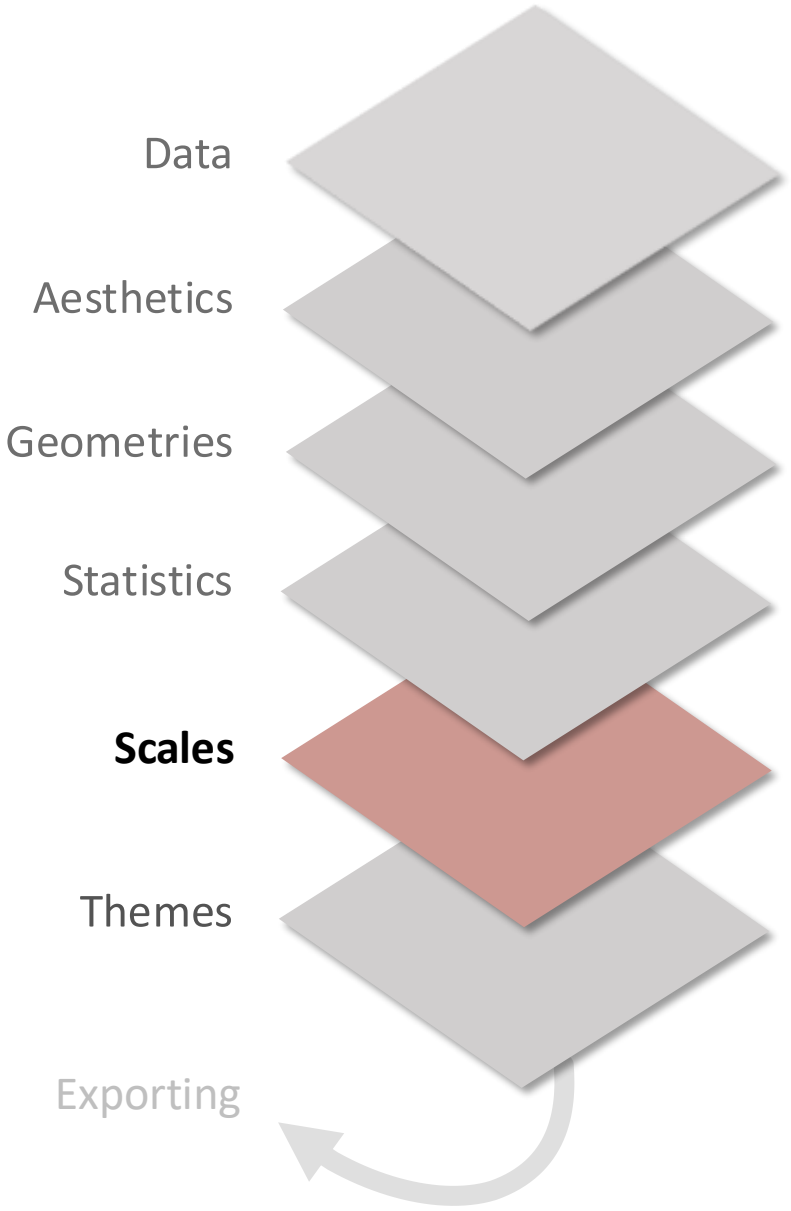
Scales

- Scales permeate all aspects of plotting in ggplot2
- Most defaults will be enough
- BUT learning how to manipulate them will give you much more control.



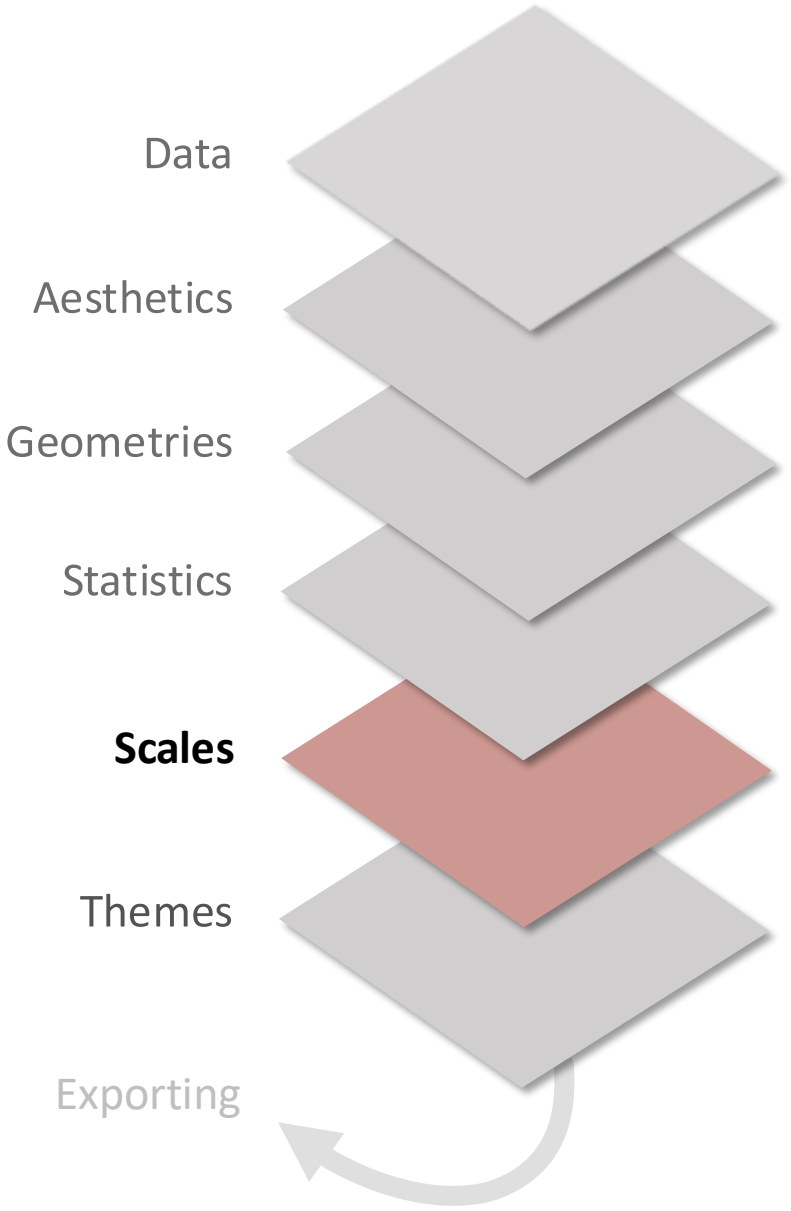
Scales

- Scales permeate all aspects of plotting in ggplot2
- Most defaults will be enough
- BUT learning how to manipulate them will give you much more control.



Scales

- Scales permeate all aspects of plotting in ggplot2
- Most defaults will be enough
- BUT learning how to manipulate them will give you much more control.



Three kinds of scales you are most likely to use:

- Axes Limits
- Bins
- Colors

Three kinds of scales you are most likely to use:

- Axes Limits
- Bins
- Colors

I'll cover these during the live demo portion

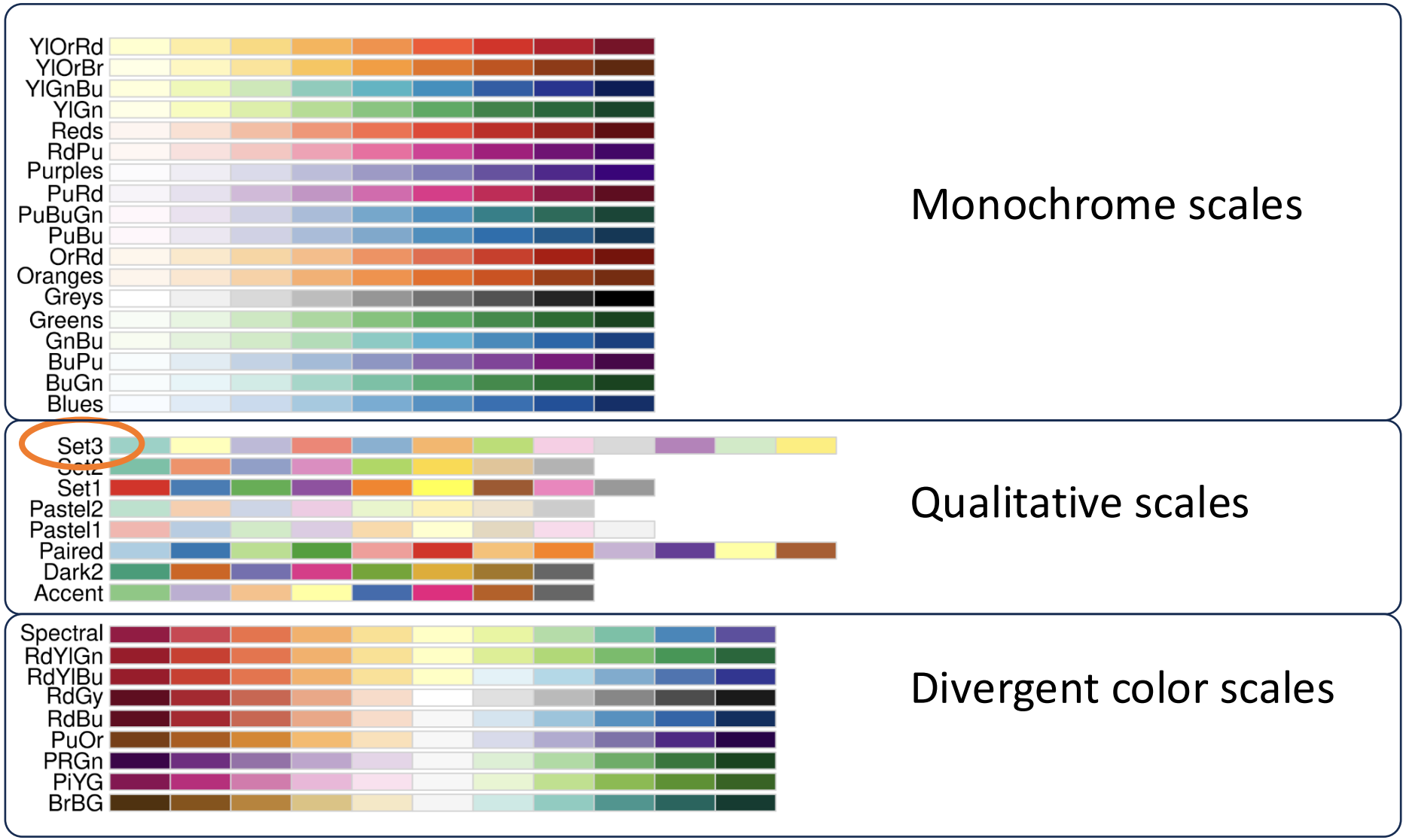
Three kinds of scales you are most likely to use:

- Axes Limits
- Bins
- Colors

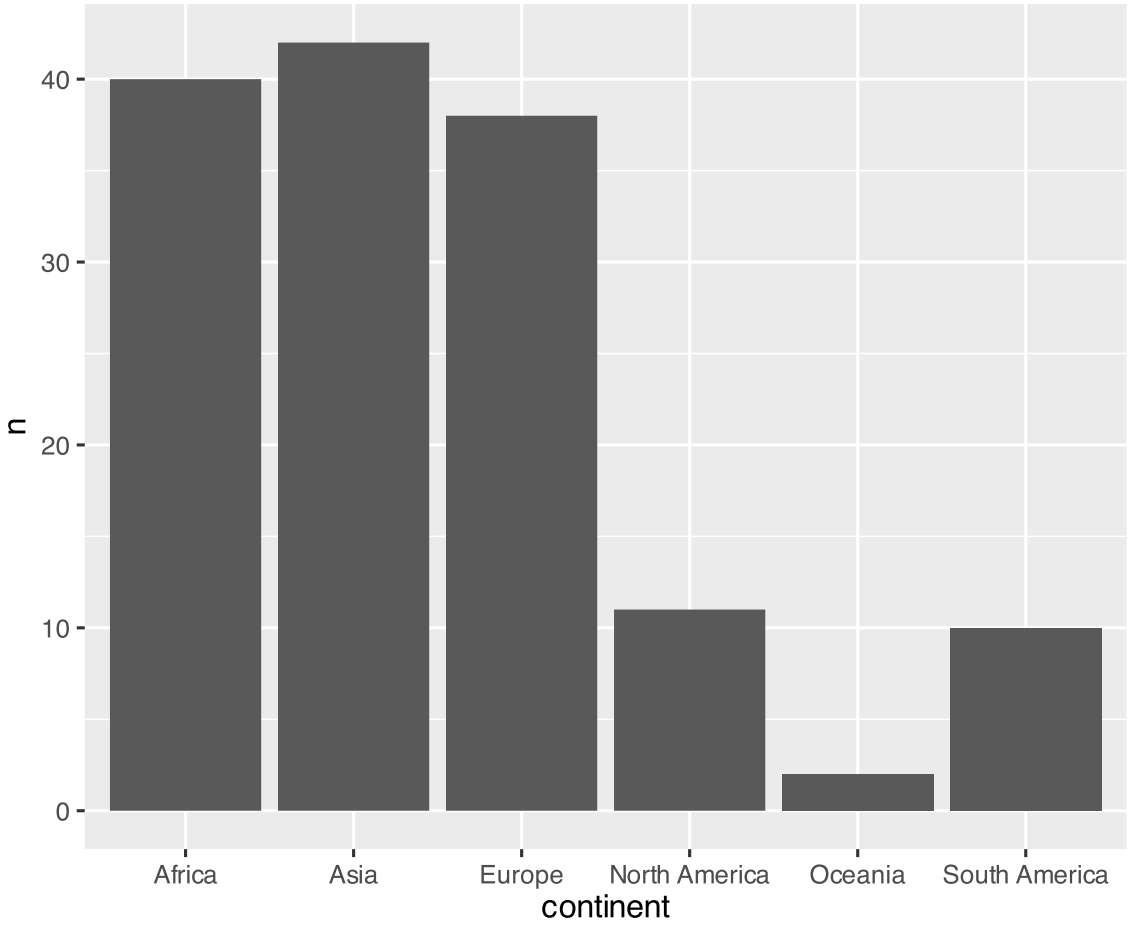
I want to spend some time talking about color scales

ggplot2 default colors for factors

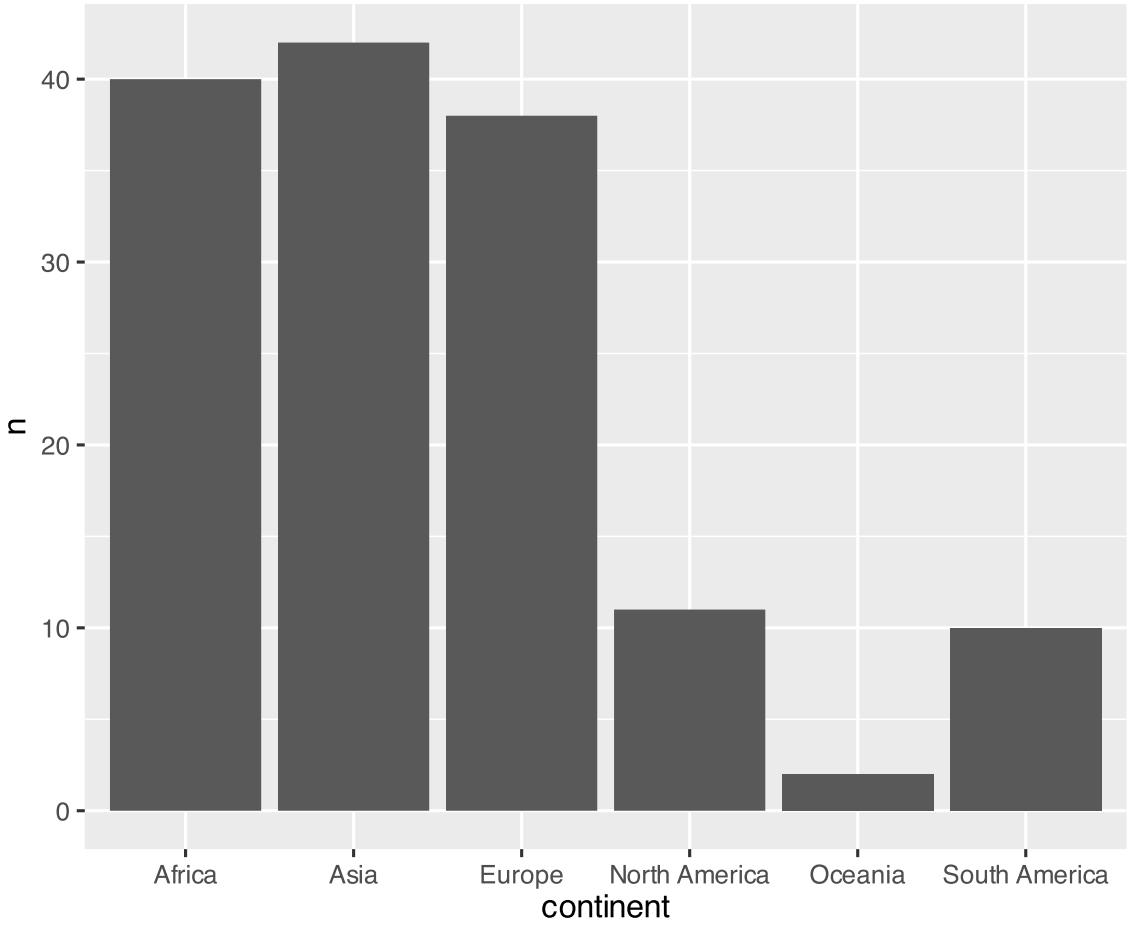
Brewer Scales



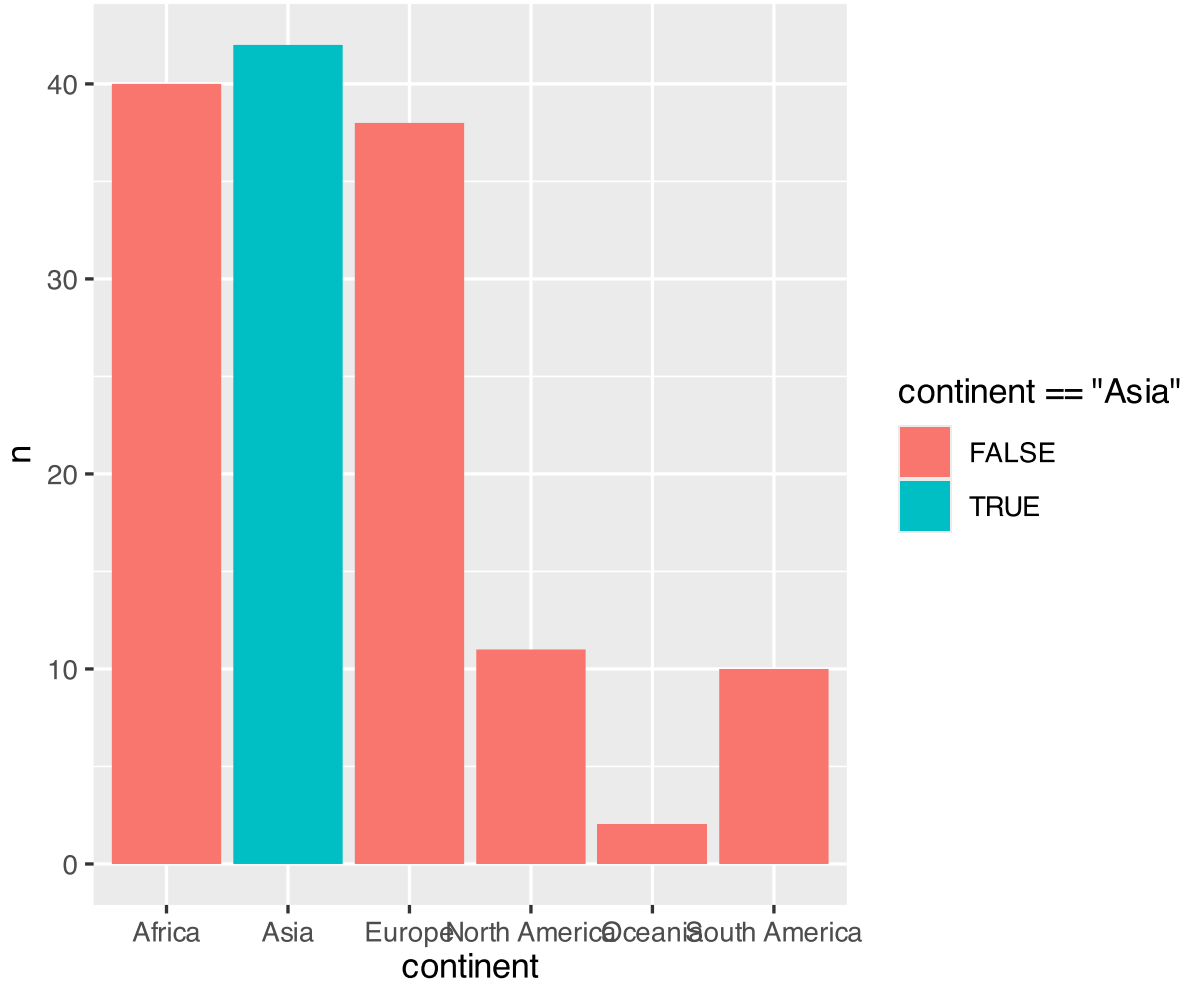
```
happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col()
```



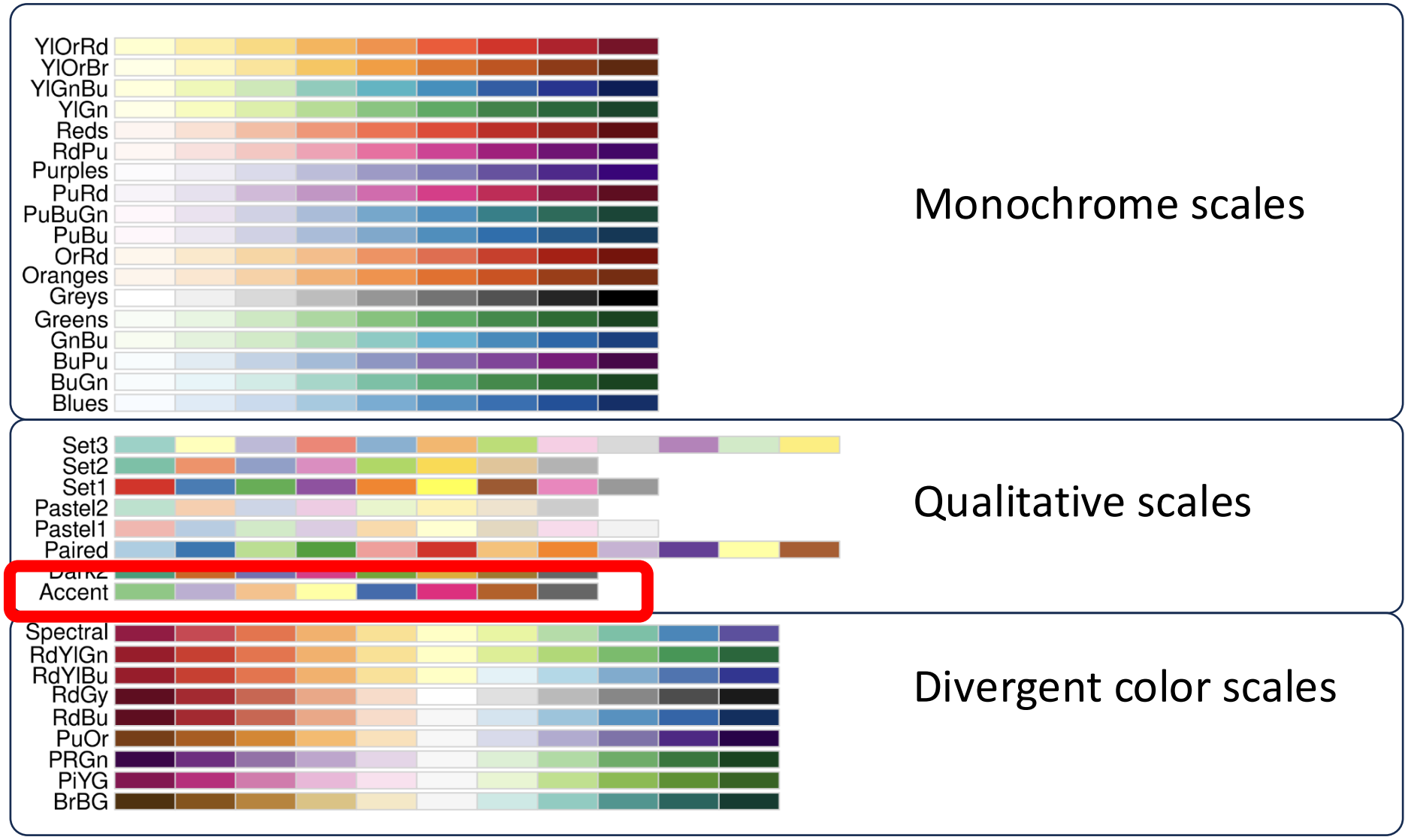
```
happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col(aes(fill=continent=="Asia")
```



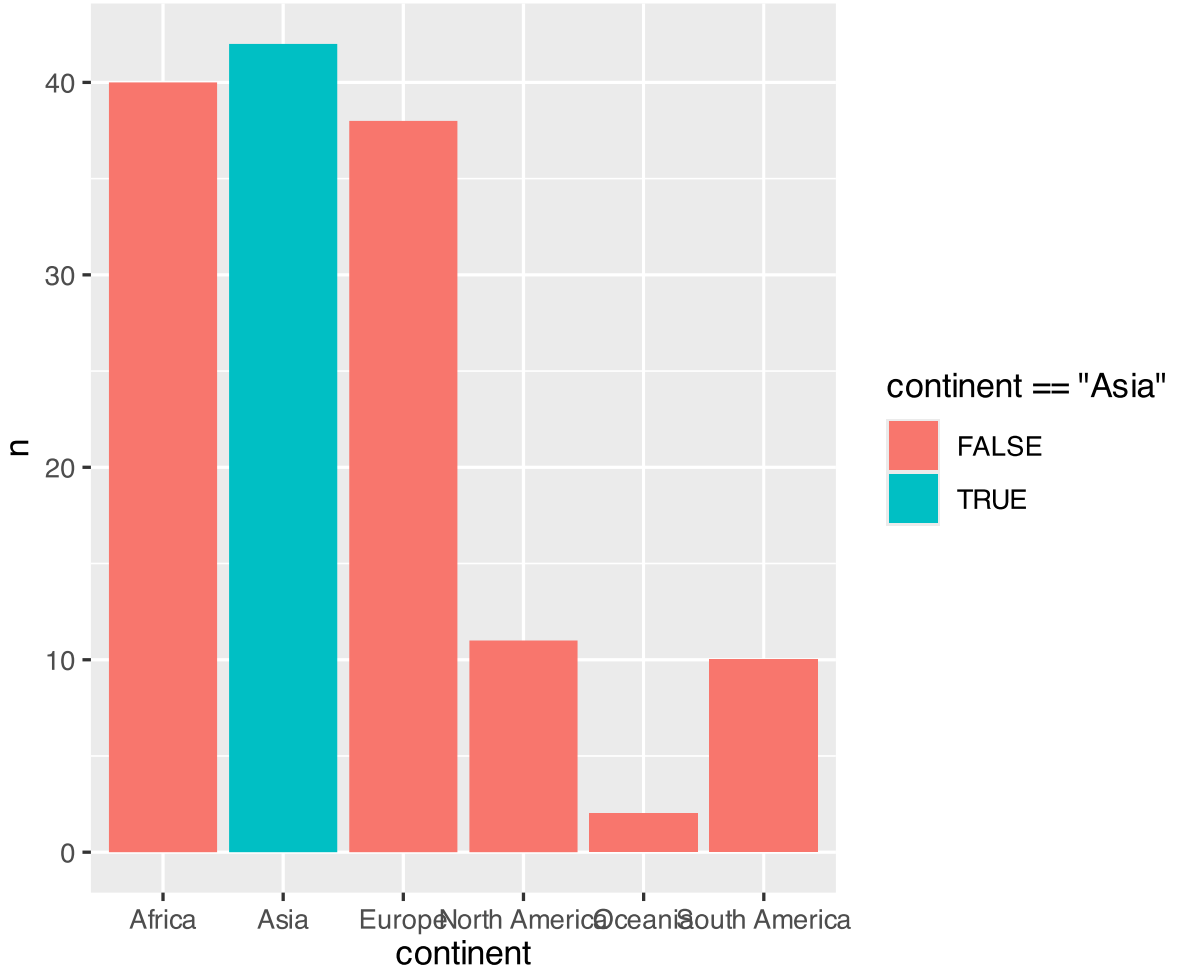
```
happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col(aes(fill=continent=="Asia"))
```



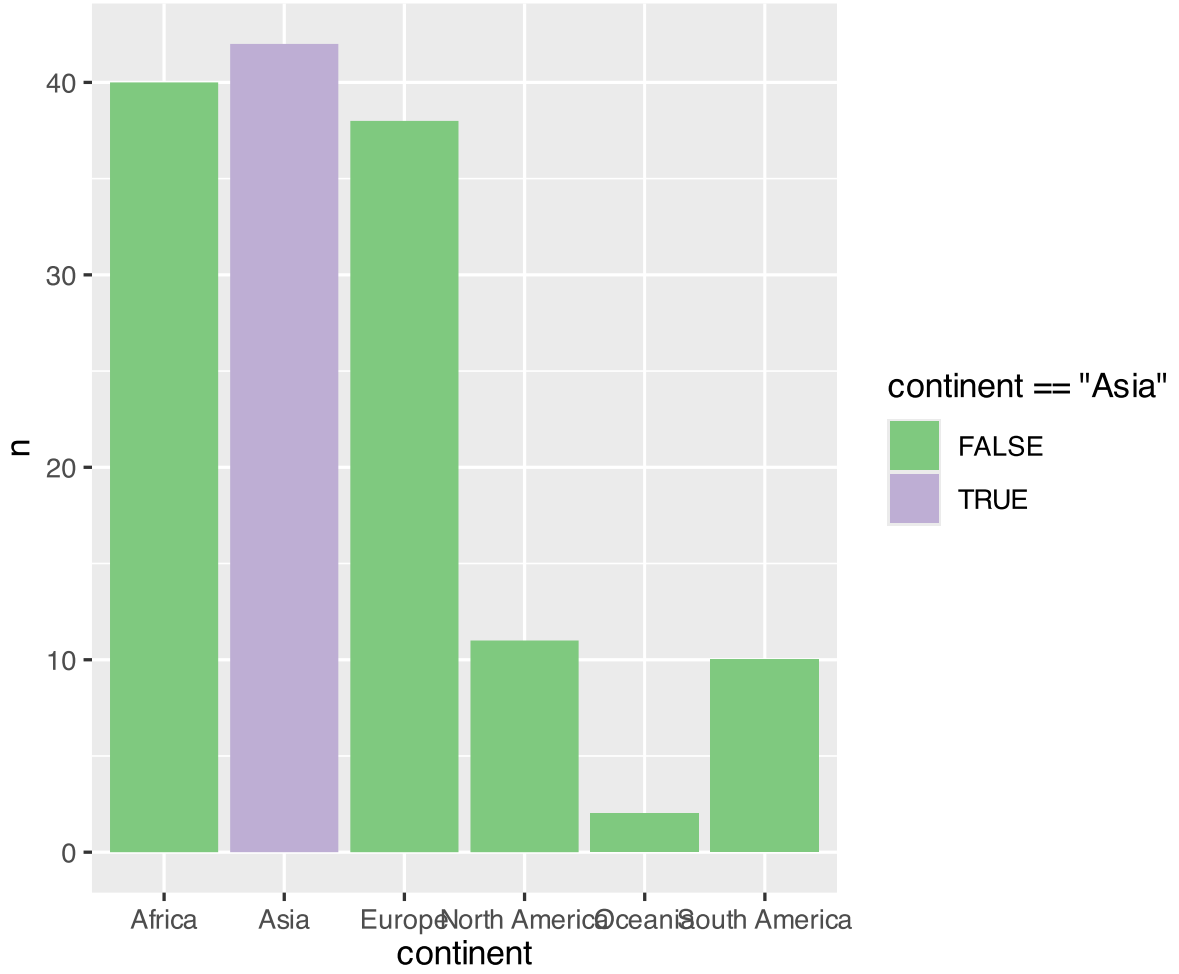
Brewer Scales



```
happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col(aes(fill=continent=="Asia") +
  scale_fill_brewer(palette='Accent')
```



```
happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col(aes(fill=continent=="Asia") +
  scale_fill_brewer(palette='Accent')
```



http://brand.duke.edu/colors

Extended Palette

The colors in Duke's extended palette are intended for use as secondary and tertiary colors in design projects. They were selected to complement Duke Navy Blue and should be used for a range of elements including graphic accents, typography, backgrounds, call-to-action buttons and more.

Copper	PMS	166 U / C	HEX	#C84E00	CMYK	0, 76, 100, 0	RGB	200, 78, 0
Persimmon	PMS	1375 U / C	HEX	#E89923	CMYK	0, 45, 95, 0	RGB	232, 153, 35
Dandelion	PMS	114 U / 121 C	HEX	#FFD960	CMYK	0, 8, 70, 0	RGB	255, 217, 96
Piedmont	PMS	382 U / 376 C	HEX	#A1B70D	CMYK	54, 0, 100, 0	RGB	161, 183, 13
Eno	PMS	3262 U / 326 C	HEX	#339898	CMYK	81, 0, 39, 0	RGB	51, 152, 152
Magnolia	PMS	328 U / 323 C	HEX	#1D6363	CMYK	96, 16, 42, 57	RGB	29, 99, 99
Prussian Blue	PMS	301 U / 7692 C	HEX	#005587	CMYK	100, 45, 0, 45	RGB	0, 85, 135
Shale Blue	PMS	Pantone Process Blue U / 7461 C	HEX	#0577B1	CMYK	100, 0, 1, 3	RGB	5, 119, 177
Ironweed	PMS	Pantone Purple U / 248 C	HEX	#993399	CMYK	35, 95, 0, 0	RGB	153, 51, 153
Hatteras	PMS	649 U / 656 C	HEX	#E2E6ED	CMYK	10, 2, 0, 0	RGB	226, 230, 237
Whisper Gray	PMS	Cool Gray 1 U / C	HEX	#F3F2F1	CMYK	4, 2, 4, 8	RGB	243, 242, 241
Ginger Beer	PMS	9060 U / C	HEX	#FCF7E5	CMYK	0, 2, 15, 0	RGB	252, 247, 229
Dogwood	PMS	7530 U / C	HEX	#988675	CMYK	10, 18, 25, 32	RGB	152, 134, 117
Shackleford	PMS	7527 U / 2527 C	HEX	#DAD0C6	CMYK	3, 4, 14, 8	RGB	218, 208, 198

Duke | Brand Guide

IDENTITY | LOGOS | BRANDING | COLORS | TYPOGRAPHY | IMAGERY

http://brand.duke.edu/colors

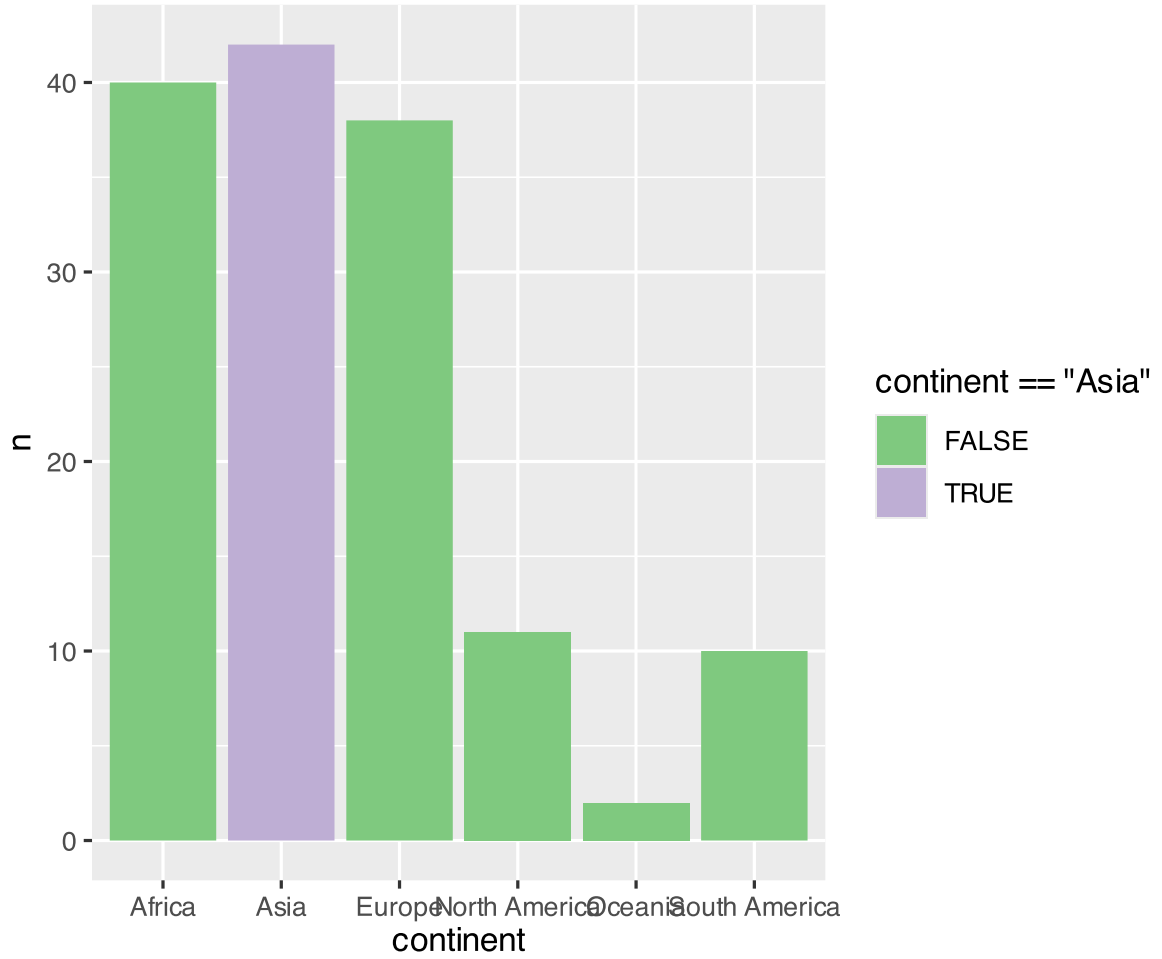
```
DukeColors = c("#DAD0C6","#C84E00")
```

Extended Palette

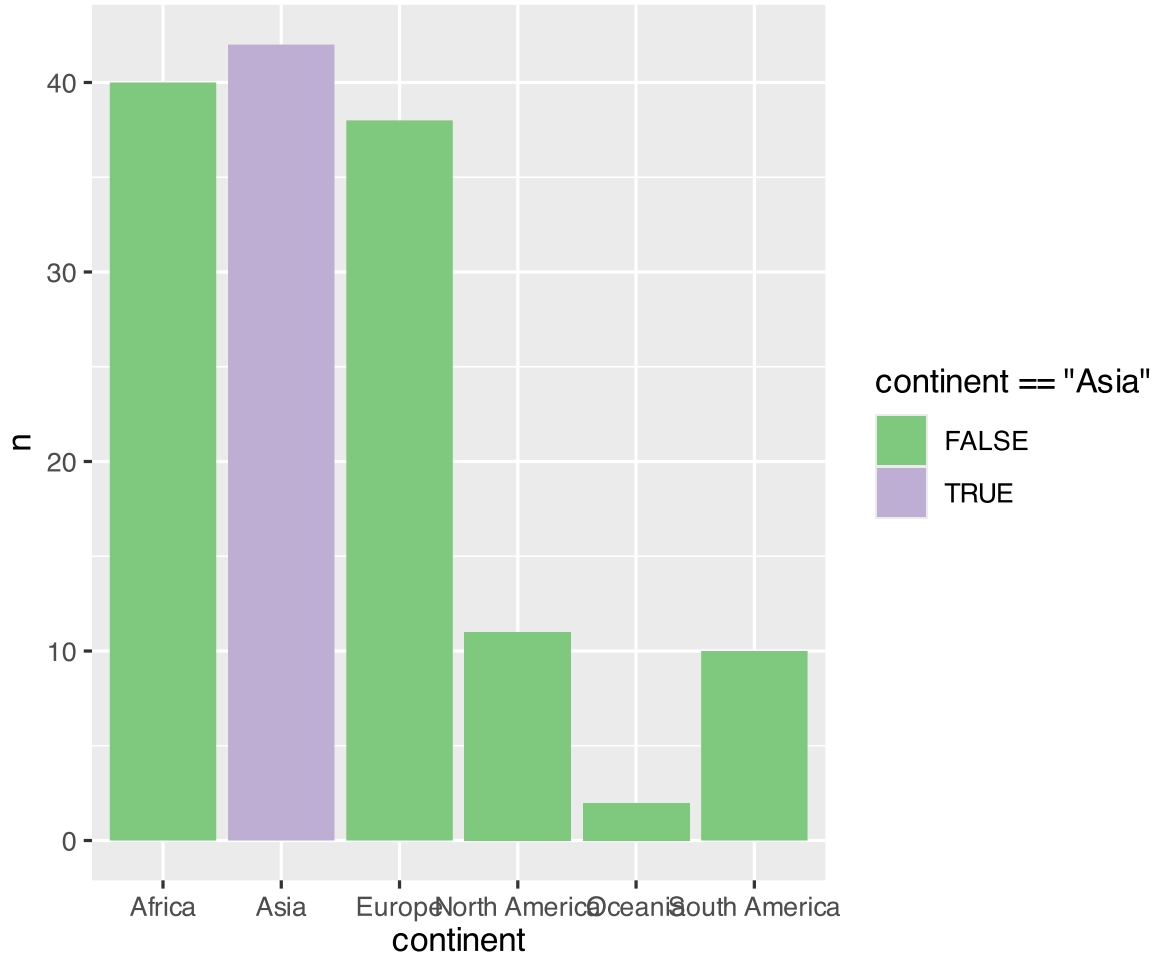
The colors in Duke's extended palette are intended for use as secondary and tertiary colors in design projects. They were selected to complement Duke Navy Blue and should be used for a range of elements including graphic accents, typography, backgrounds, call-to-action buttons and more.

Copper	PMS	166 U / C	HEX	#C84E00	CMYK	0, 76, 100, 0	RGB	200, 78, 0
Persimmon	PMS	1375 U / C	HEX	#E89923	CMYK	0, 45, 95, 0	RGB	232, 153, 35
Dandelion	PMS	114 U / 121 C	HEX	#FFD960	CMYK	0, 8, 70, 0	RGB	255, 217, 96
Piedmont	PMS	382 U / 376 C	HEX	#A1B70D	CMYK	54, 0, 100, 0	RGB	161, 183, 13
Eno	PMS	3262 U / 326 C	HEX	#339898	CMYK	81, 0, 39, 0	RGB	51, 152, 152
Magnolia	PMS	328 U / 323 C	HEX	#1D6363	CMYK	96, 16, 42, 57	RGB	29, 99, 99
Prussian Blue	PMS	301 U / 7692 C	HEX	#005587	CMYK	100, 45, 0, 45	RGB	0, 85, 135
Shale Blue	PMS	Pantone Process Blue U / 7461 C	HEX	#0577B1	CMYK	100, 0, 1, 3	RGB	5, 119, 177
Ironweed	PMS	Pantone Purple U / 248 C	HEX	#993399	CMYK	35, 95, 0, 0	RGB	153, 51, 153
Hatteras	PMS	649 U / 656 C	HEX	#E2E6ED	CMYK	10, 2, 0, 0	RGB	226, 230, 237
Whisper Gray	PMS	Cool Gray 1 U / C	HEX	#F3F2F1	CMYK	4, 2, 4, 8	RGB	243, 242, 241
Ginger Beer	PMS	9060 U / C	HEX	#FCF7E5	CMYK	0, 2, 15, 0	RGB	252, 247, 229
Dogwood	PMS	7530 U / C	HEX	#988675	CMYK	10, 18, 25, 32	RGB	152, 134, 117
Shackleford	PMS	7527 U / 2527 C	HEX	#DAD0C6	CMYK	3, 4, 14, 8	RGB	218, 208, 198

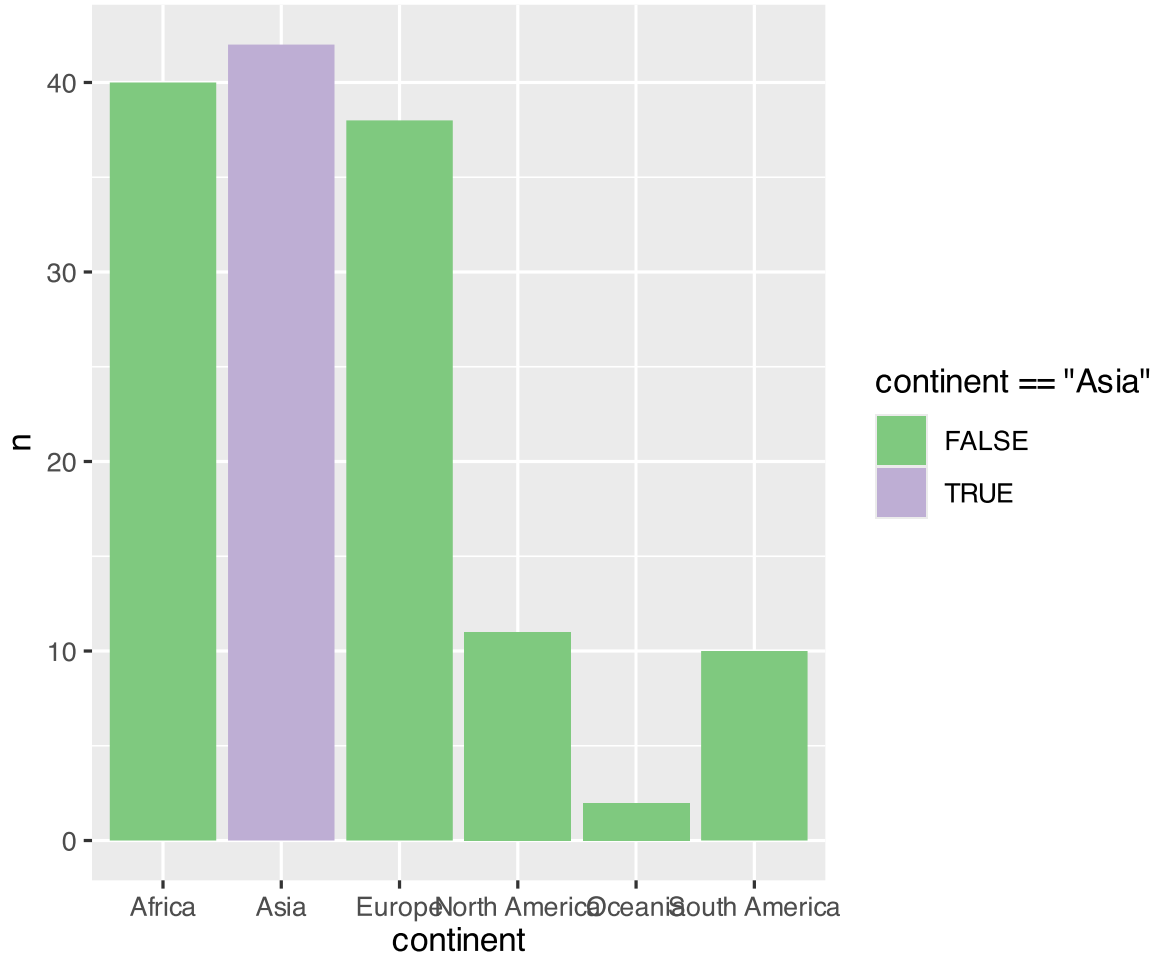
```
DukeColors <- c('#DAD0C6',"#C84E00")
```



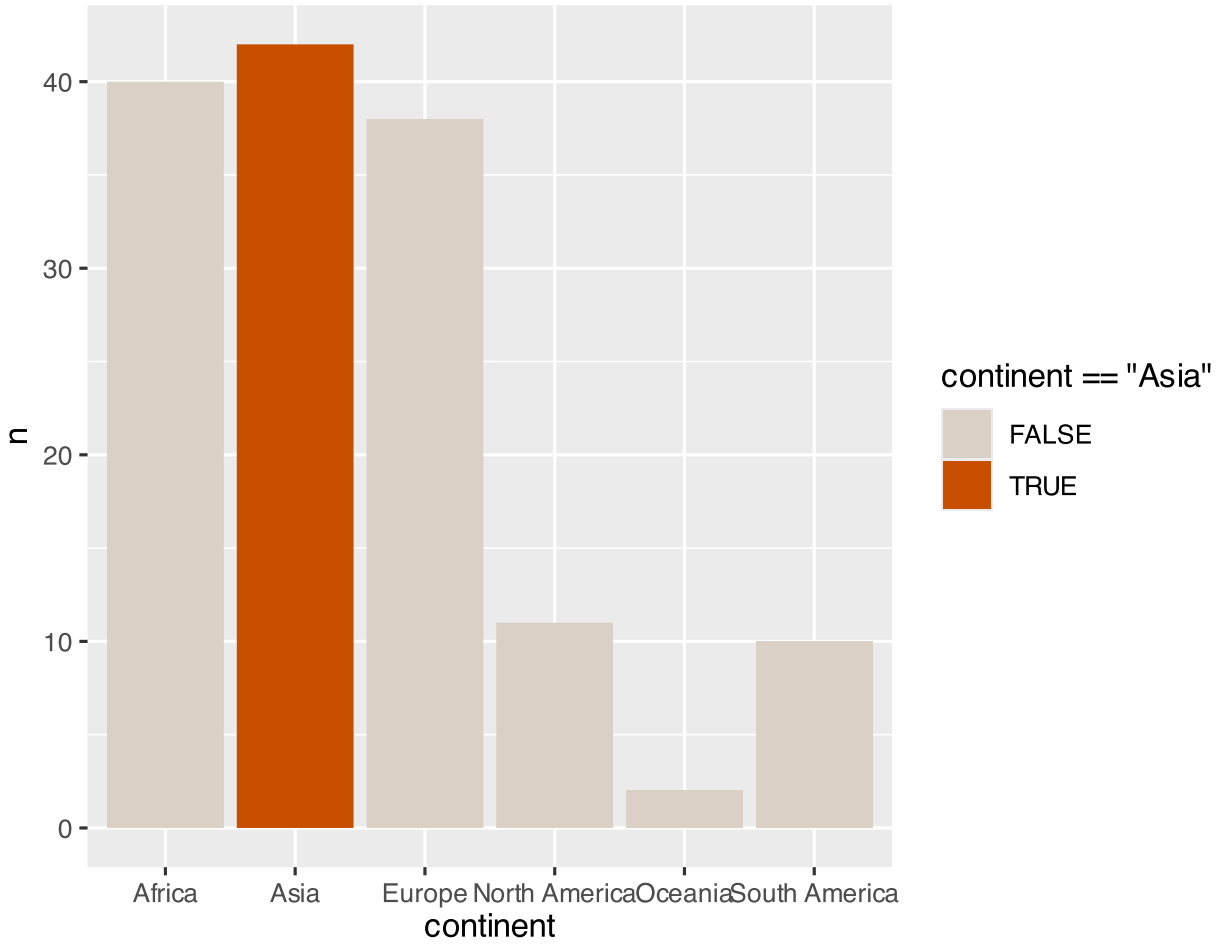
```
DukeColors <- c('#DAD0C6',"#C84E00")

happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col(aes(fill=continent=="Asia") +
  scale_fill_brewer(palette='Accent')
```

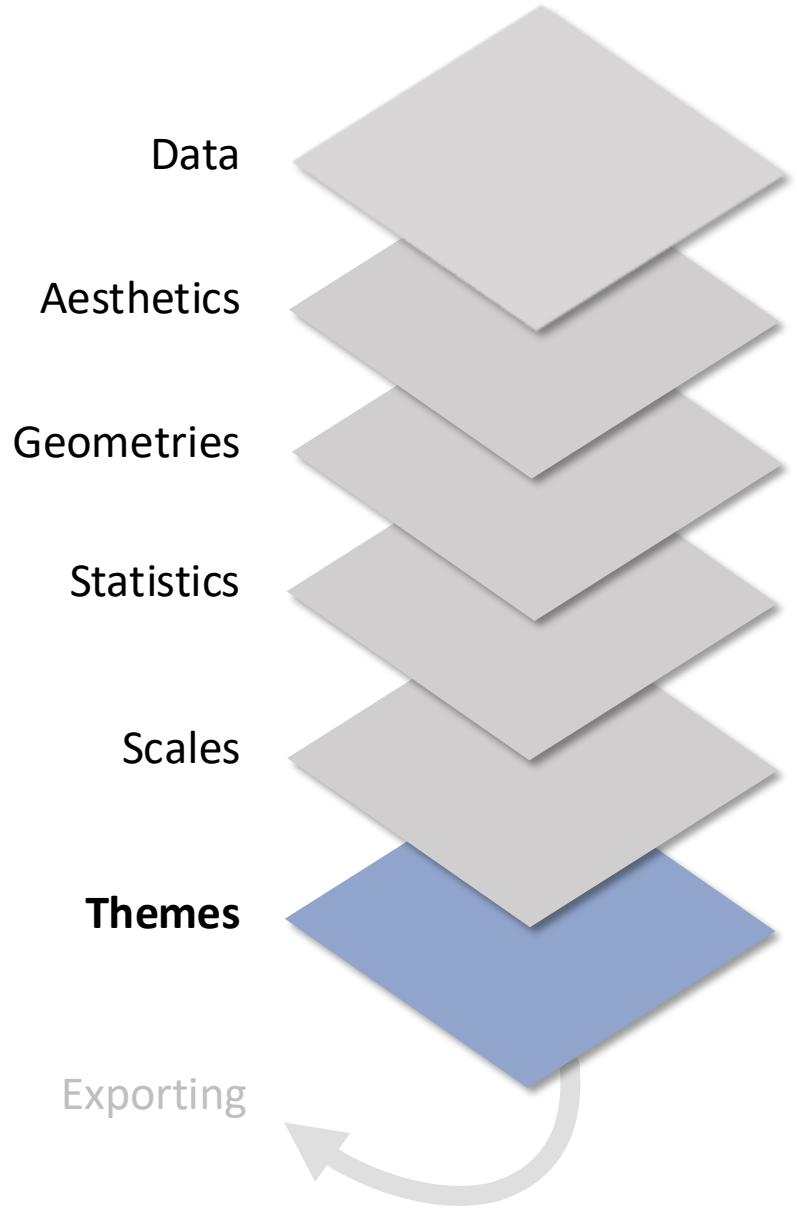


```
DukeColors <- c('#DAD0C6',"#C84E00")

happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col(aes(fill=continent=="Asia") +
  scale_fill_manual(values = DukeColors)
```

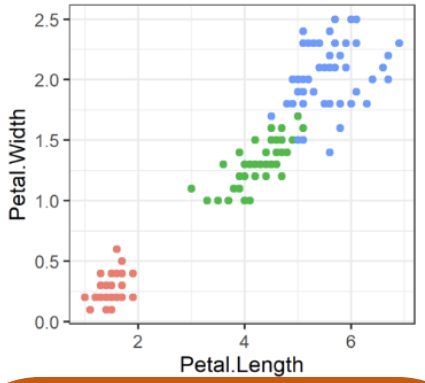


```
DukeColors <- c('#DAD0C6',"#C84E00")

happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col(aes(fill=continent=="Asia") +
  scale_fill_manual(values = DukeColors)
```

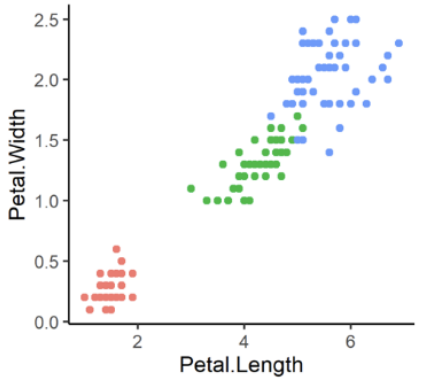



Data
Aesthetics
Geometries
Statistics
Scales
Themes
Exporting

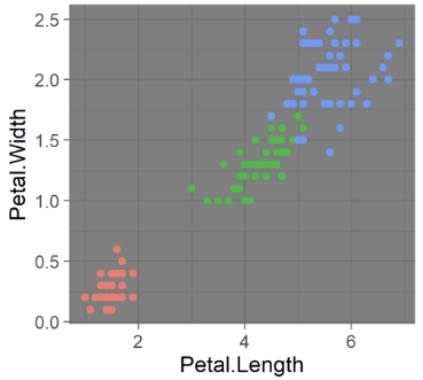
theme_bw()



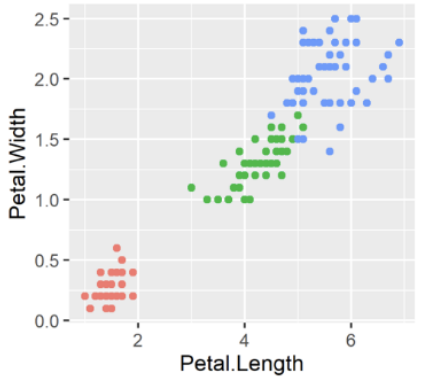
theme_classic()



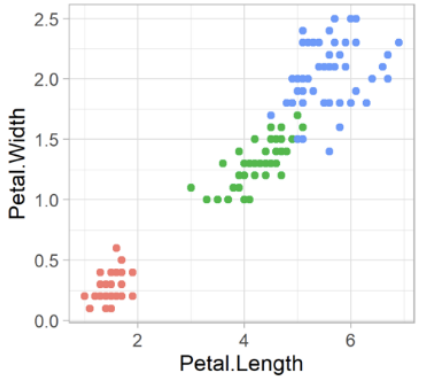
theme_dark()



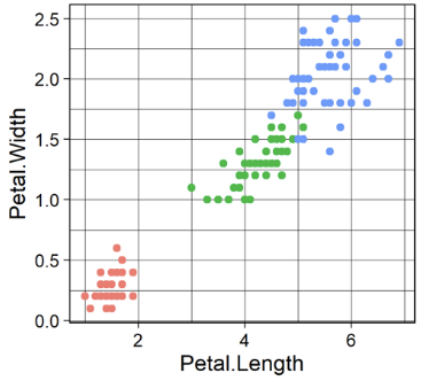
theme_grey()



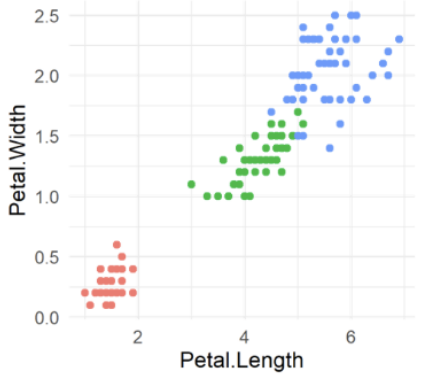
theme_light()



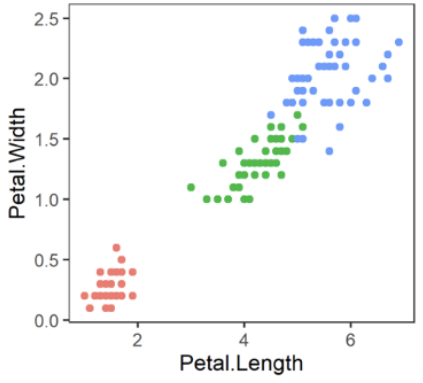
theme_linedraw()



theme_minimal()



theme_test()



theme_void()

Data

Aesthetics

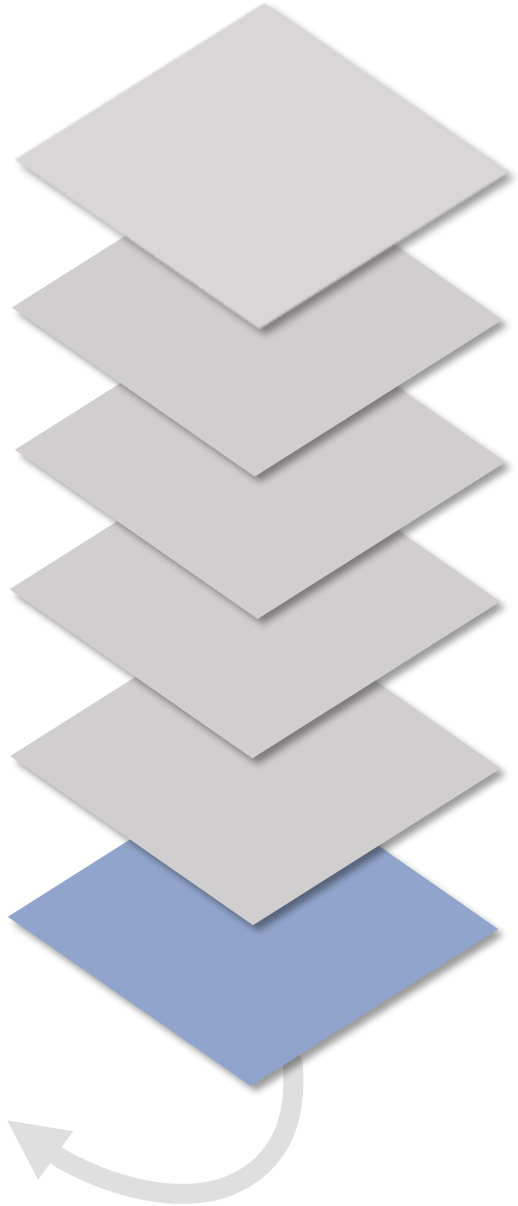
Geometries

Statistics

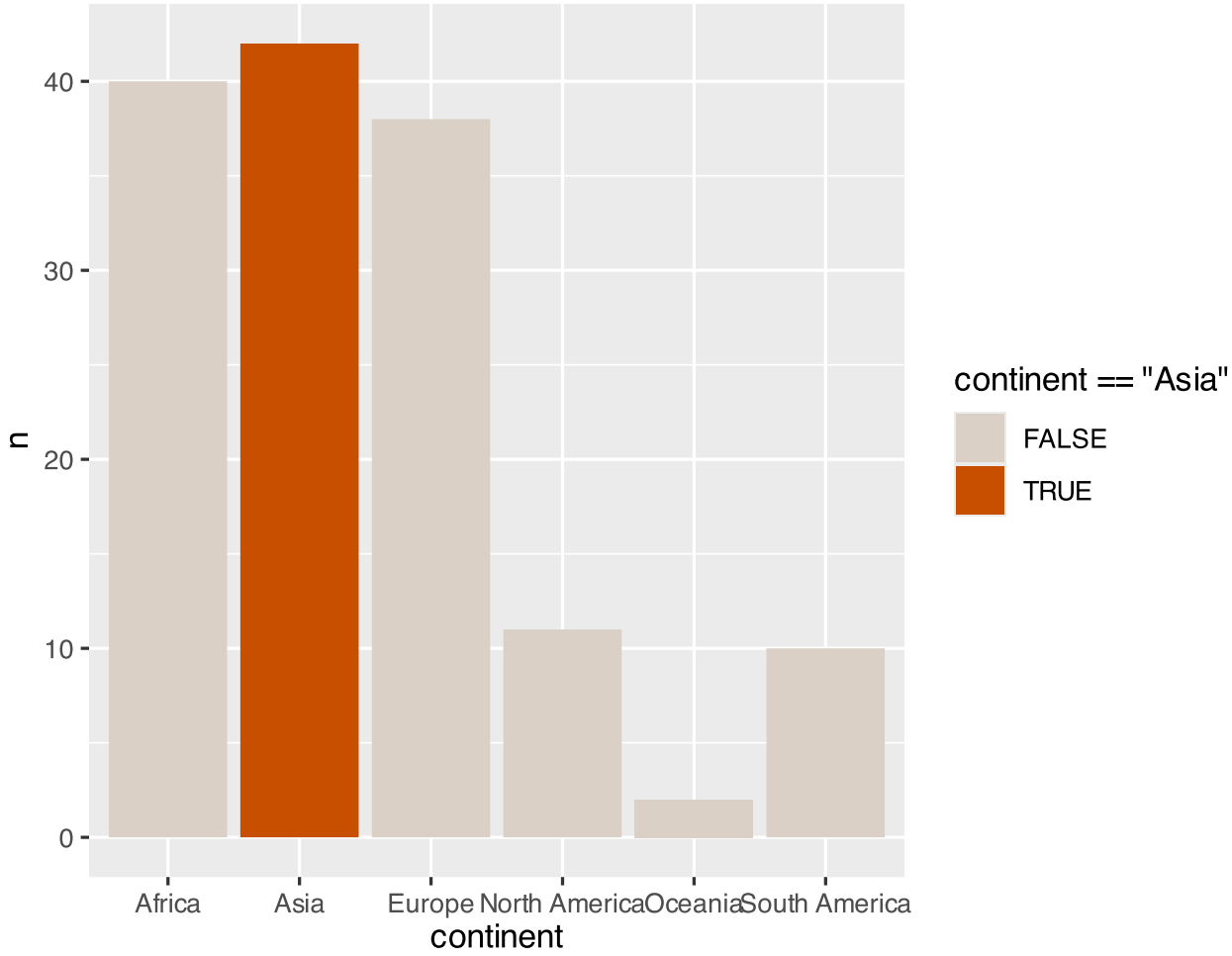
Scales

Themes

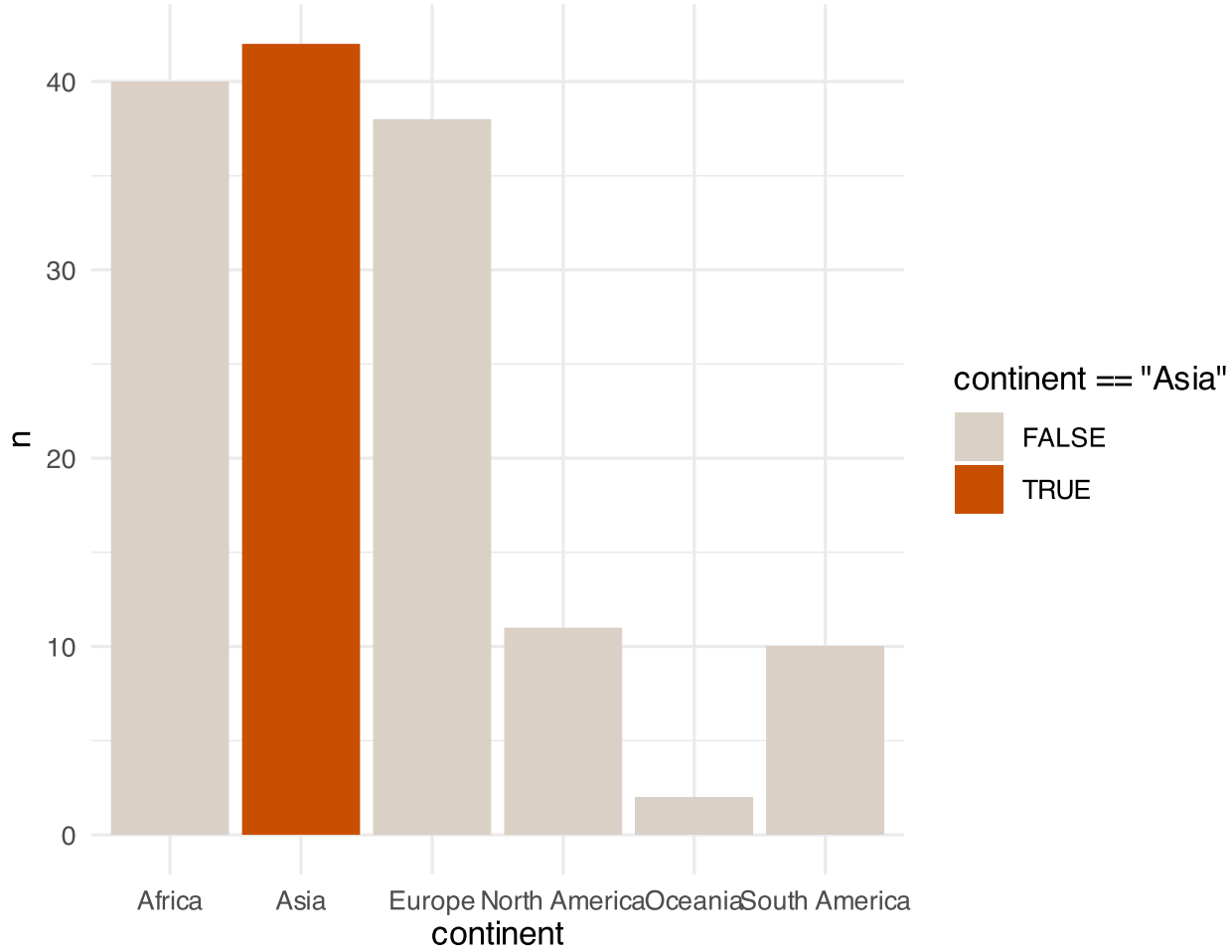
Exporting



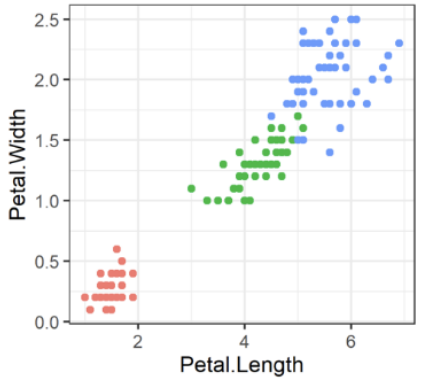
```
DukeColors <- c('#DAD0C6',"#C84E00")

happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col(aes(fill=continent=="Asia") +
  scale_fill_manual(value = DukeColors)+
  theme_minimal()
```



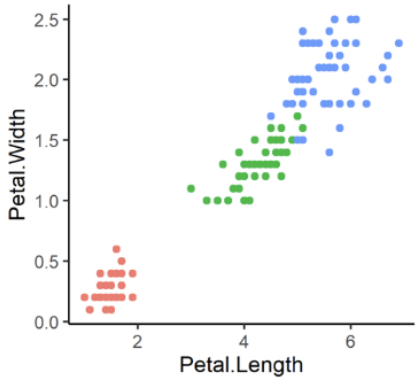
```
DukeColors <- c('#DAD0C6',"#C84E00")

happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col(aes(fill=continent=="Asia") +
  scale_fill_manual(value = DukeColors)+
  theme_minimal()
```



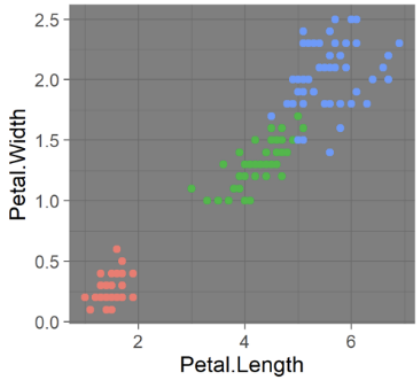
theme_bw()



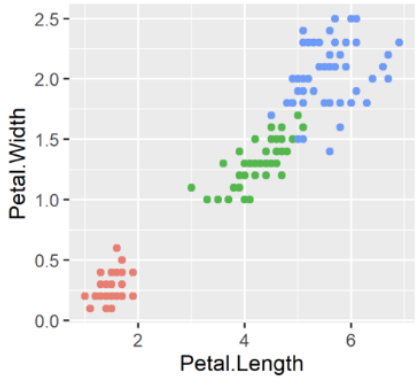
theme_classic()



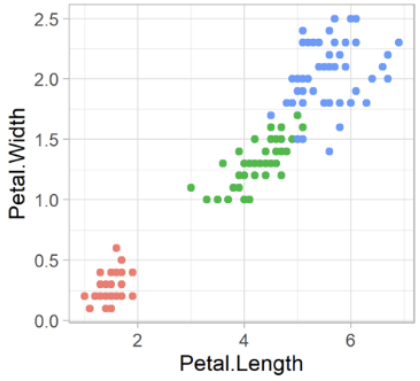
theme_dark()



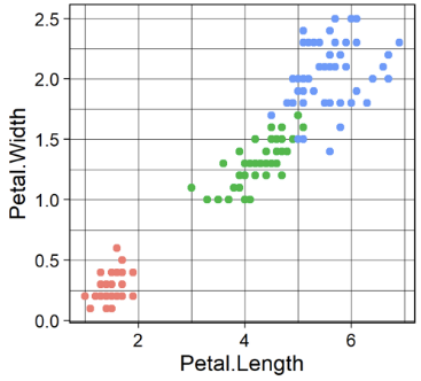
theme_grey()



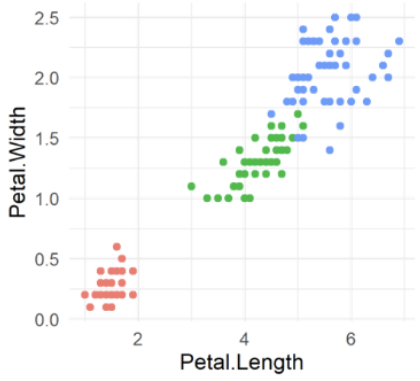
theme_light()



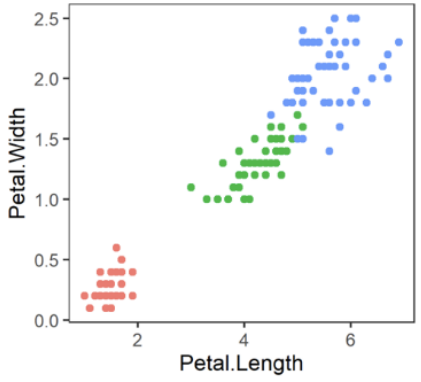
theme_linedraw()



theme_minimal()



theme_test()



theme_void()

Data

Aesthetics

Geometries

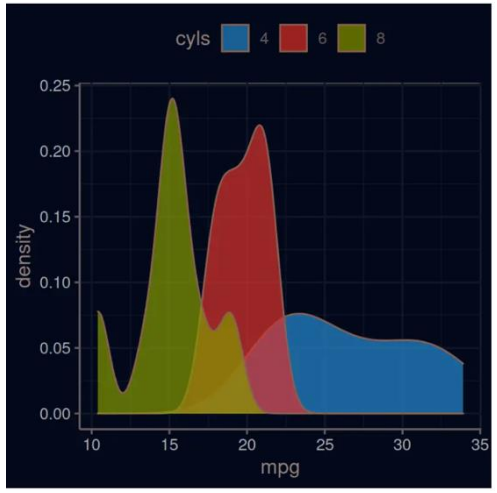
Statistics

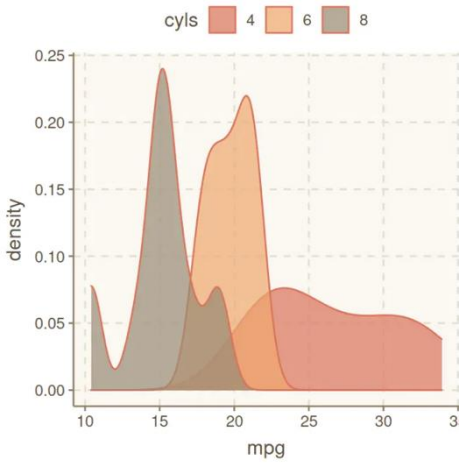
Scales

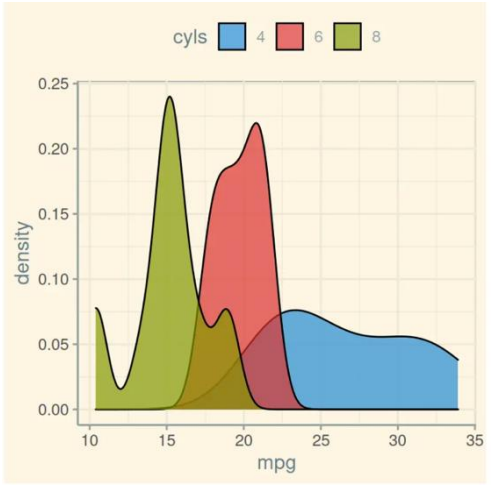
Themes

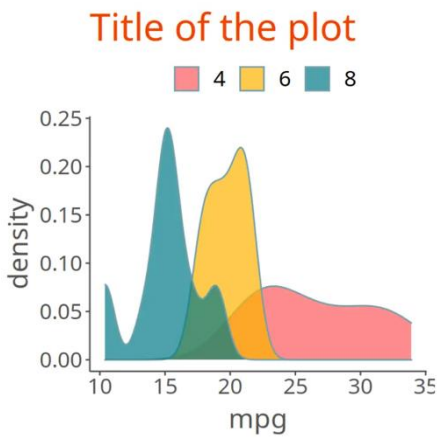
Exporting

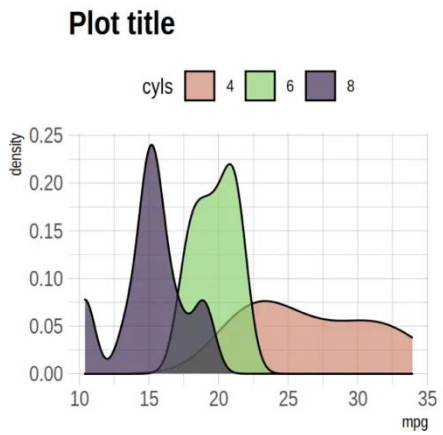
Over 60 pre-existing themes

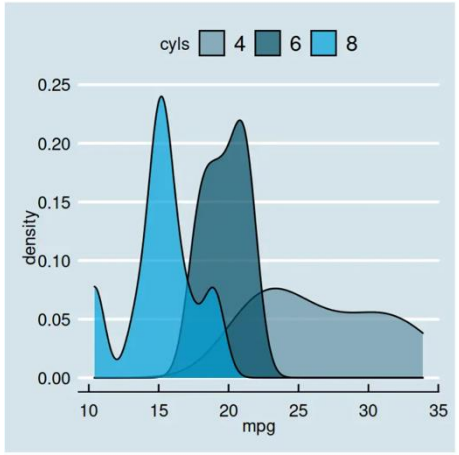




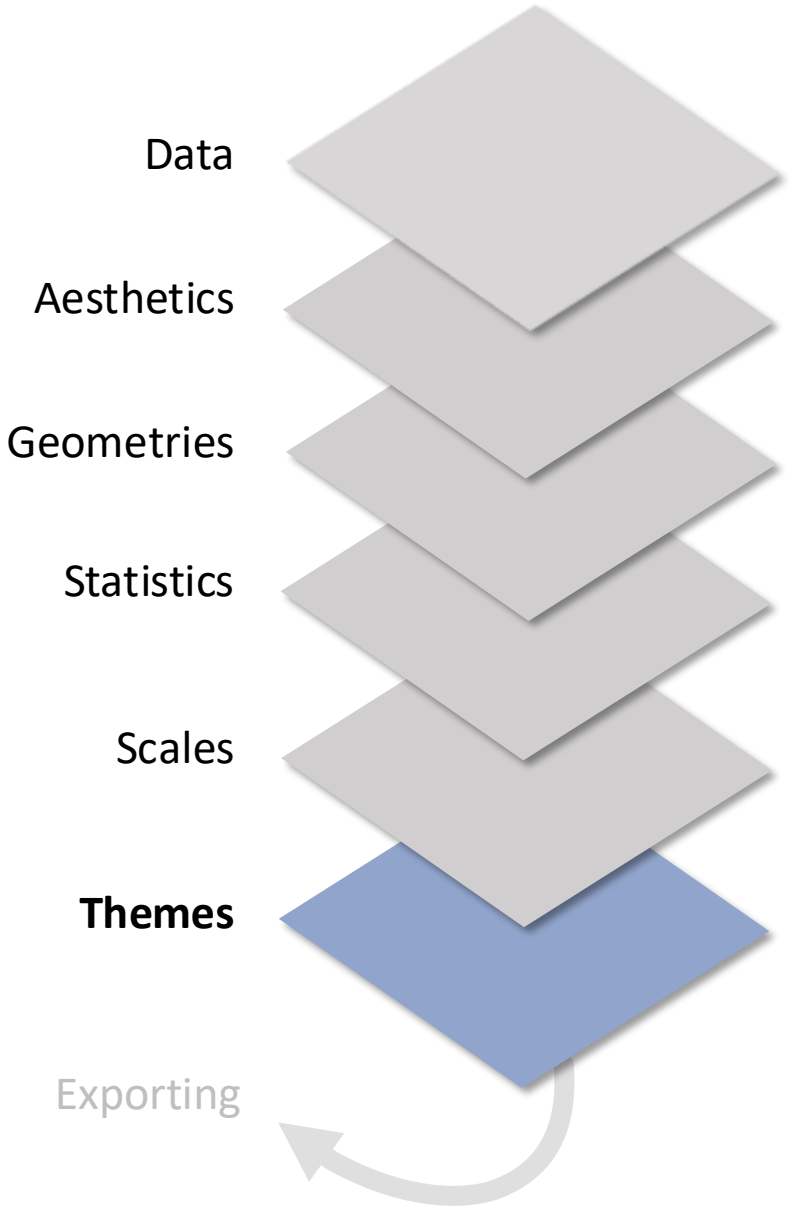








https://r-charts.com/ggplot2/themes/



?theme()

```
theme(
  line,
  rect,
  text,
  title,
  aspect.ratio,
  axis.title,
  axis.title.x,
  axis.title.x.top,
  axis.title.x.bottom,
  axis.title.y,
  axis.title.y.left,
  axis.title.y.right,
  axis.text,
  axis.text.x,
  axis.text.x.top,
  axis.text.x.bottom,
  axis.text.y,
  axis.text.y.left,
  axis.text.y.right,
  axis.ticks,
```

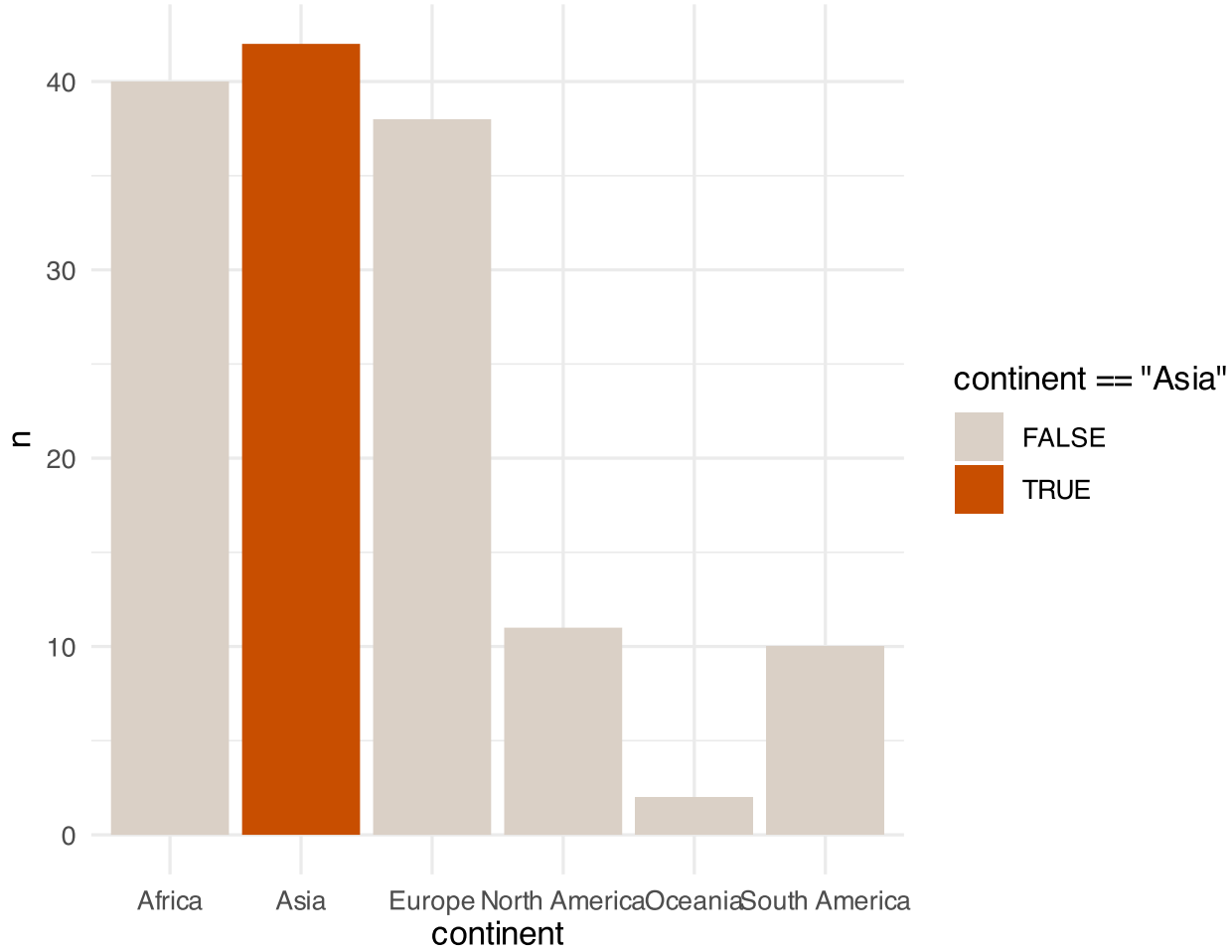
```
axis.ticks.x,
axis.ticks.x.top,
axis.ticks.x.bottom,
axis.ticks.y,
axis.ticks.y.left,
axis.ticks.y.right,
axis.ticks.length,
axis.ticks.length.x,
axis.ticks.length.x.top,
axis.ticks.length.x.bottom,
axis.ticks.length.y,
axis.ticks.length.y.left,
axis.ticks.length.y.right,
axis.line,
axis.line.x,
axis.line.x.top,
axis.line.x.bottom,
axis.line.y,
axis.line.y.left,
axis.line.y.right,
legend.background,
legend.margin,
legend.spacing,
legend.spacing.x,
legend.spacing.y,
```

```
legend.key,
legend.key.size,
legend.key.height,
legend.key.width,
legend.text,
legend.text.align,
legend.title,
legend.title.align,
legend.position,
legend.direction,
legend.justification,
legend.box,
legend.box.just,
legend.box.margin,
legend.box.background,
legend.box.spacing,
panel.background,
panel.border,
panel.spacing,
panel.spacing.x,
panel.spacing.y,
panel.grid,
panel.grid.major,
panel.grid.minor,
panel.grid.major.x,
```

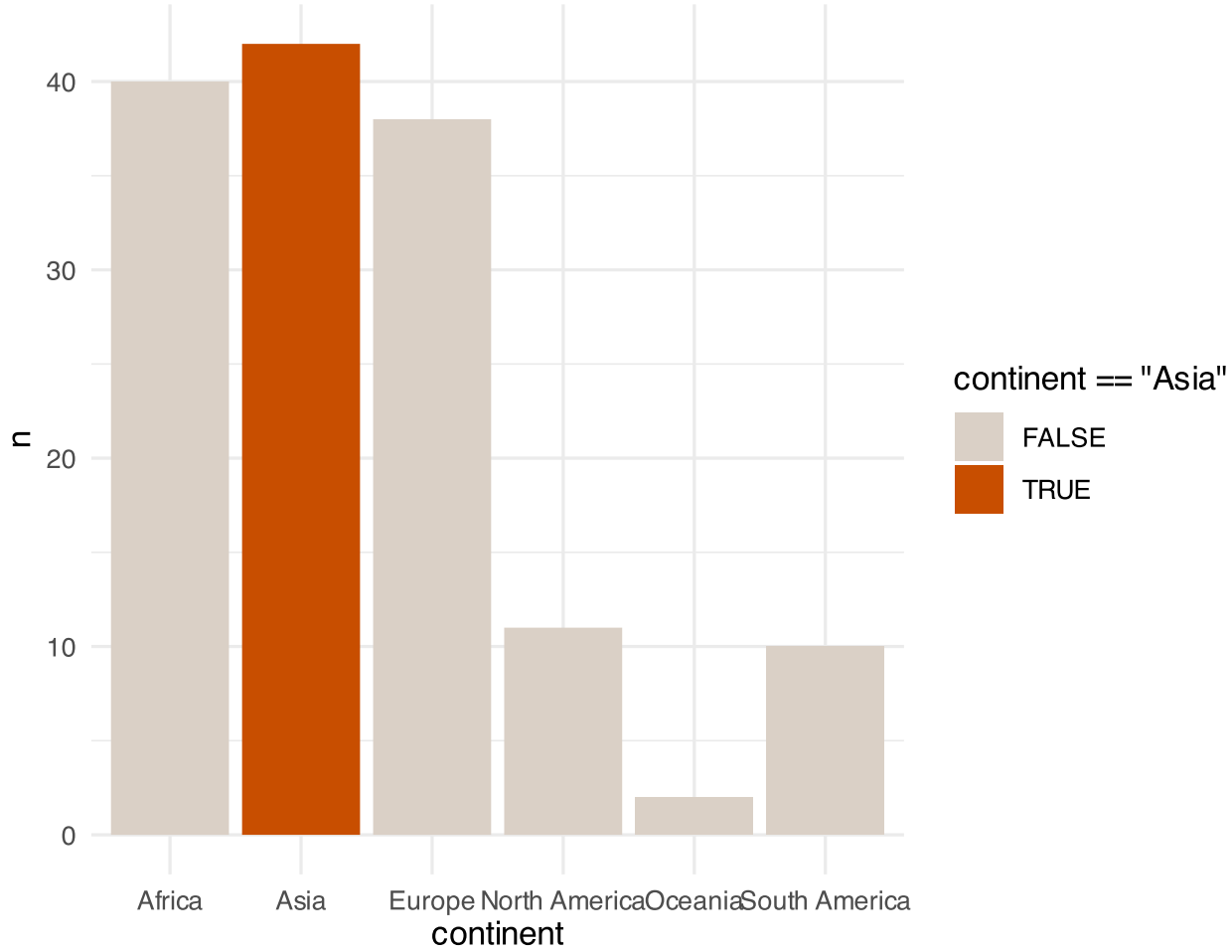
```
panel.grid.major.y,
panel.grid.minor.x,
panel.grid.minor.y,
panel.ontop,
plot.background,
plot.title,
plot.title.position,
plot.subtitle,
plot.caption,
plot.caption.position,
plot.tag,
plot.tag.position,
plot.margin,
strip.background,
strip.background.x,
strip.background.y,
strip.clip,
strip.placement,
strip.text,
strip.text.x,
strip.text.x.bottom,
strip.text.x.top,
strip.text.y,
strip.text.y.left,
strip.text.y.right,
strip.switch.pad.grid,
```

...

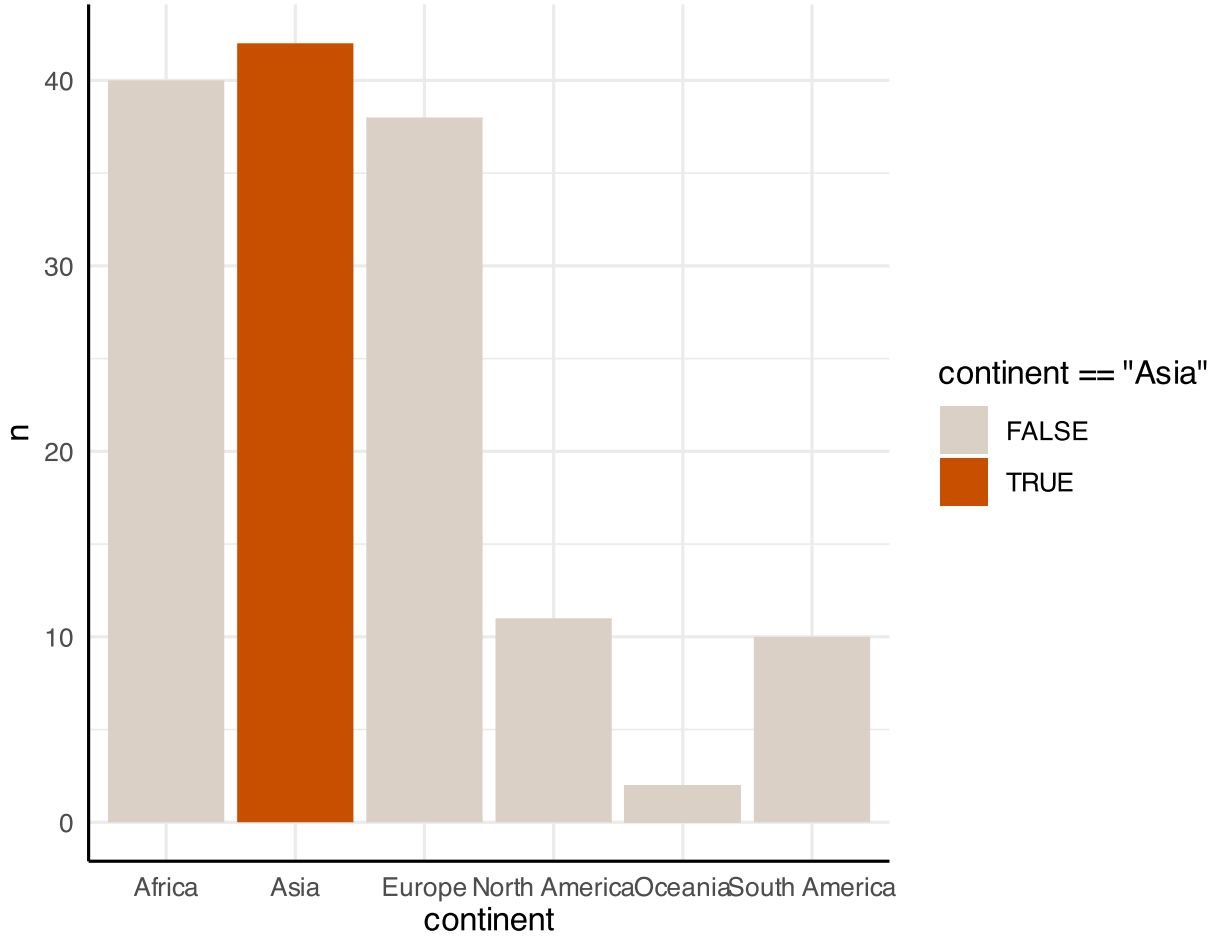
```
DukeColors <- c('#DAD0C6',"#C84E00")

happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col(aes(fill=continent=="Asia") +
  scale_fill_manual(value = DukeColors)+
  theme_minimal()
```

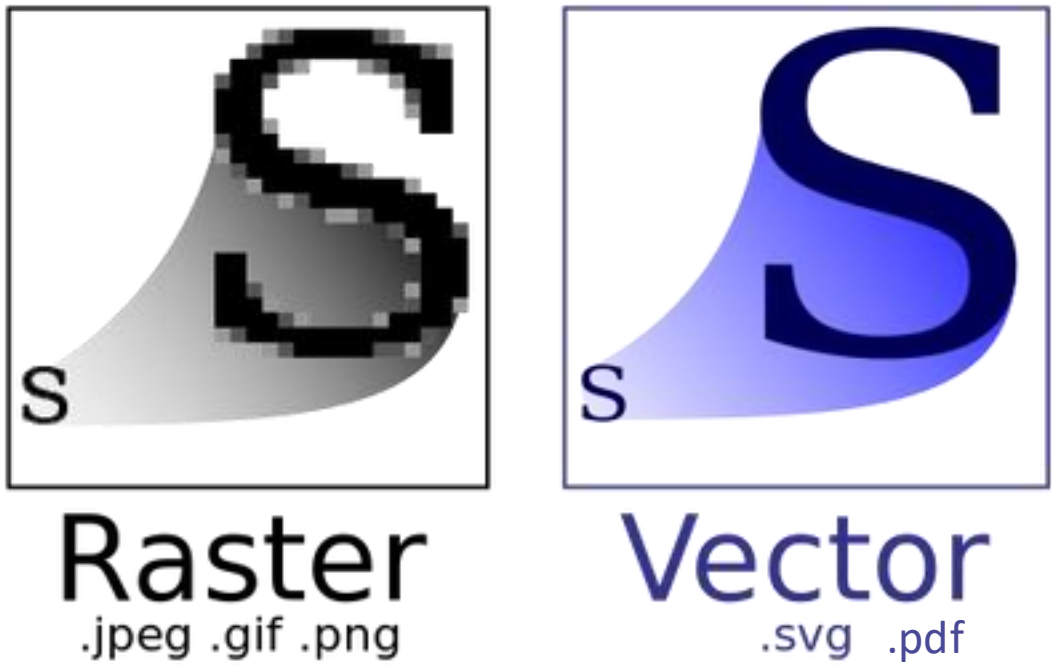


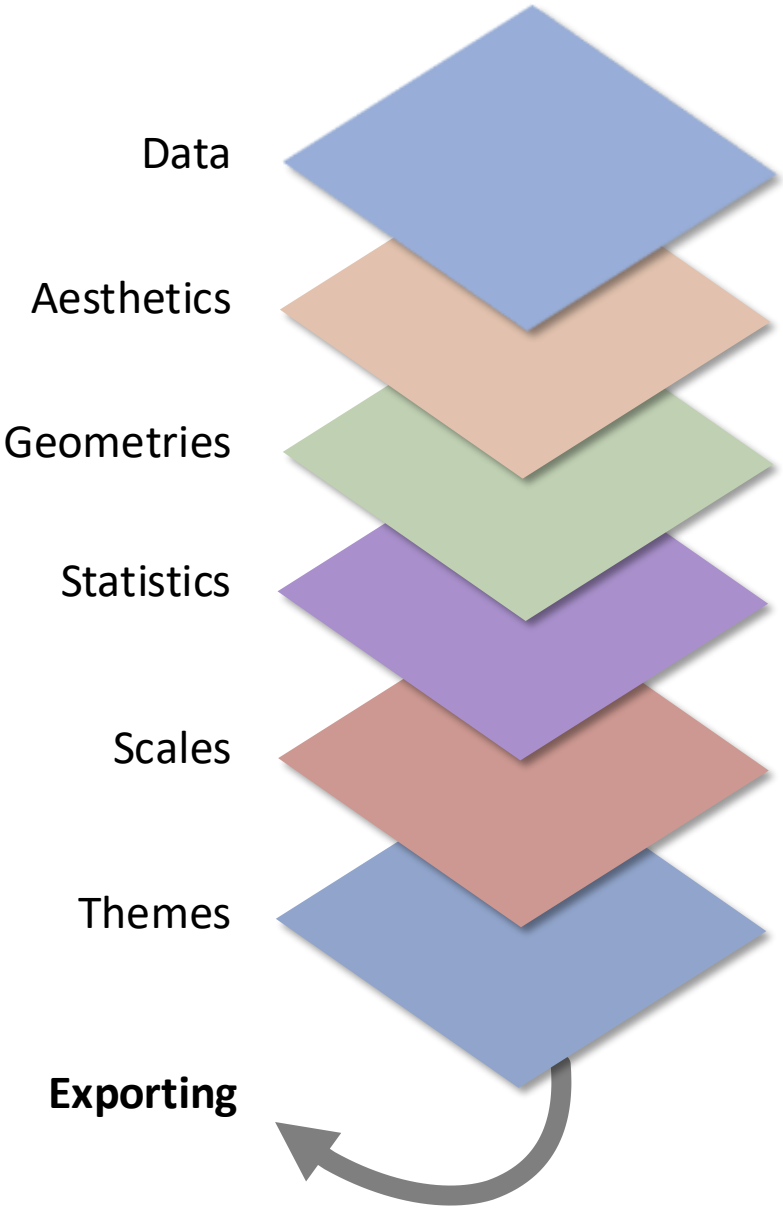
```
DukeColors <- c('#DAD0C6',"#C84E00")

happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col(aes(fill=continent=="Asia") +
  scale_fill_manual(value = DukeColors)+
  theme_minimal() +
  theme(
     axis.line.x.bottom=element_line(),
     axis.line.y.left=element_line()
              )
```

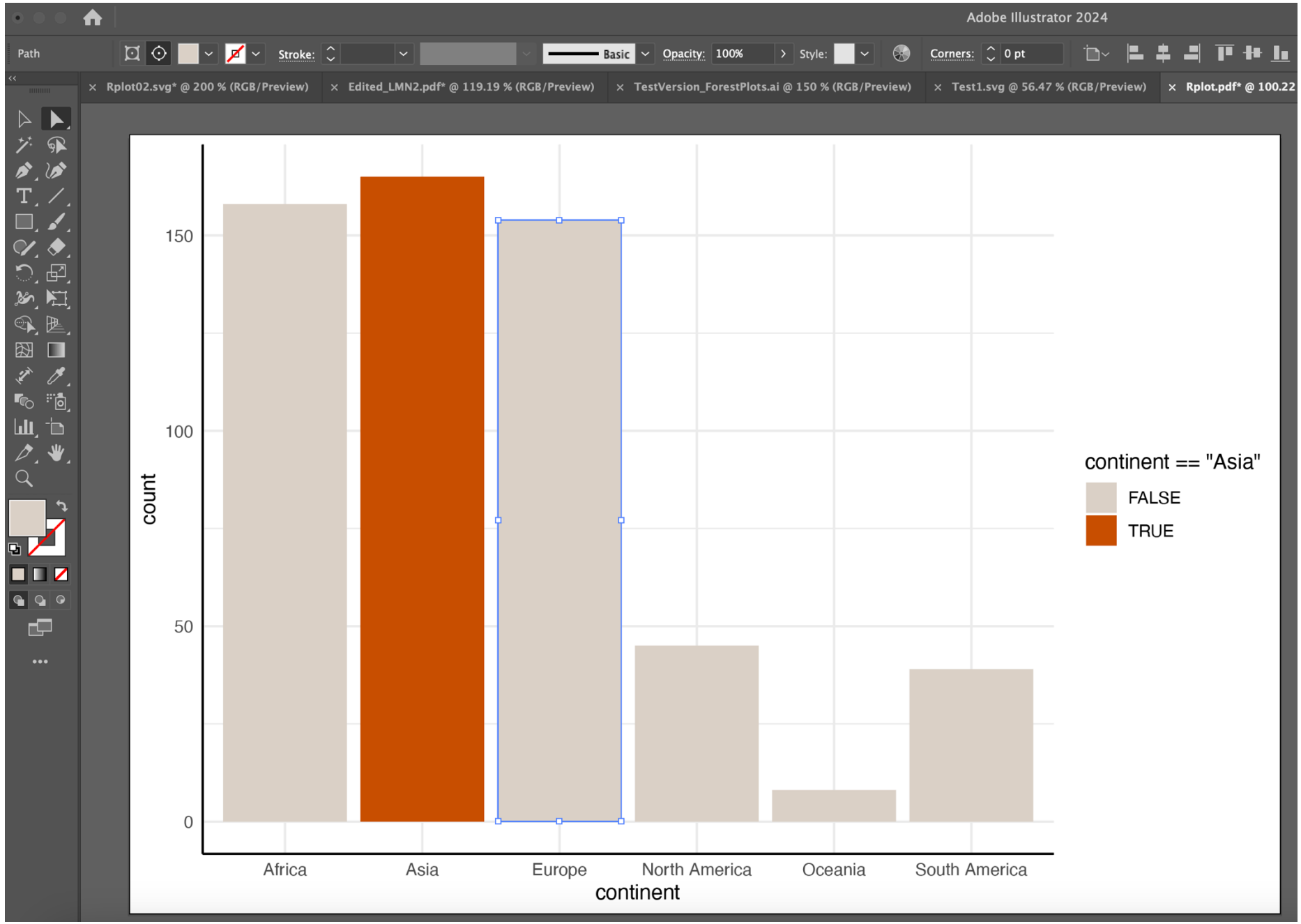


```
DukeColors <- c('#DAD0C6',"#C84E00")

happy |>
  filter(year == 2019) |>
  group_by(continent) |>
  summarize(n=n()) |>
  ggplot(aes(continent, n) +
  geom_col(aes(fill=continent=="Asia") +
  scale_fill_manual(value = DukeColors)+
  theme_minimal() +
  theme(
     axis.line.x.bottom=element_line(),
     axis.line.y.left=element_line()
              )
```



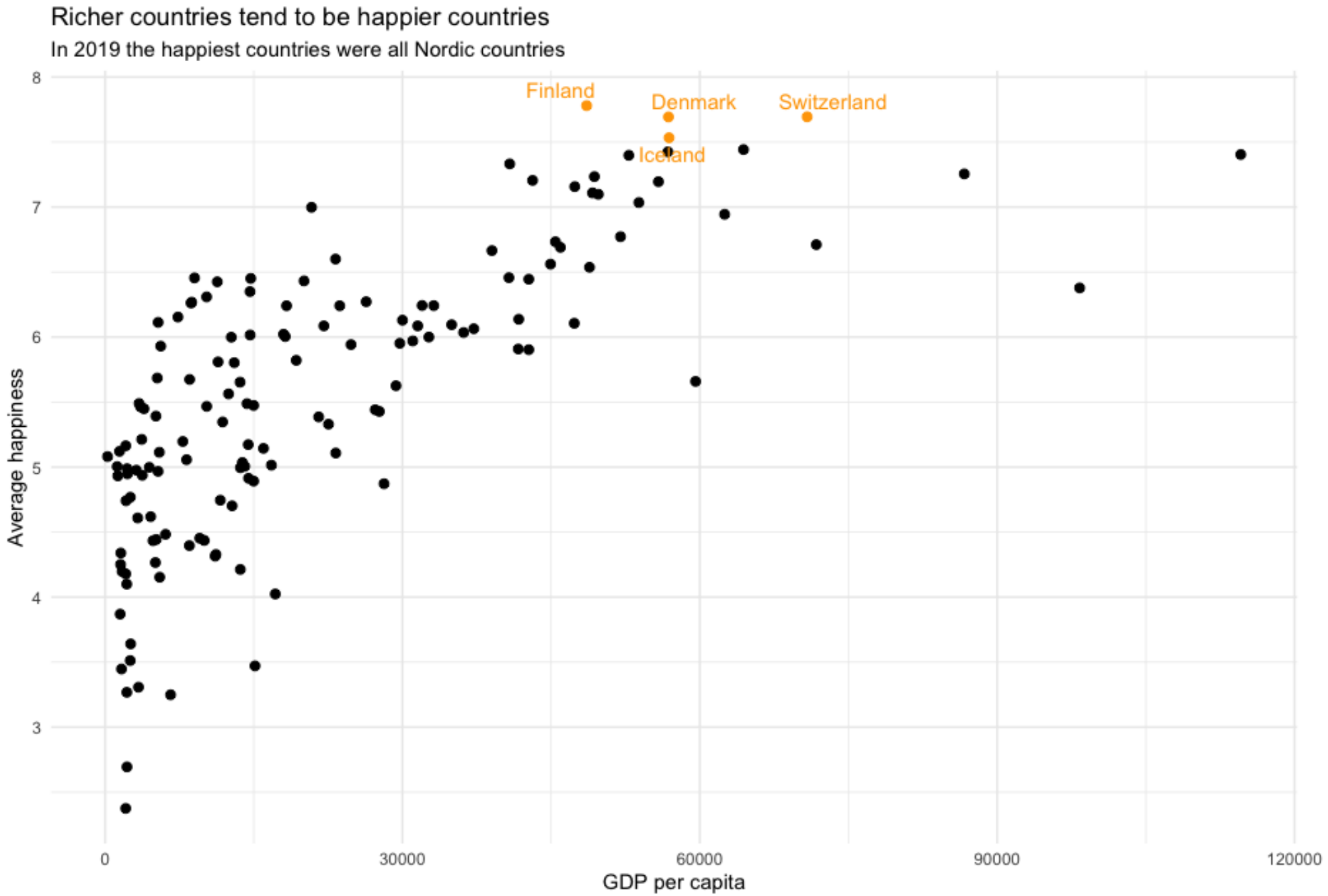

S
Raster
.jpeg .gif .png
S
Vector
.svg .pdf


Data
Aesthetics
Geometries
Statistics
Scales
Themes
Exporting


Adobe Illustrator 2024
Path
Stroke:
Basic
Opacity:
100%
Style:
Corners:
0 pt
Rplot02.svg* @ 200 % (RGB/Preview)
Edited_LMN2.pdf* @ 119.19 % (RGB/Preview)
TestVersion_ForestPlots.ai @ 150 % (RGB/Preview)
Test1.svg @ 56.47 % (RGB/Preview)
Rplot.pdf* @ 100.22
150
100
50
0
count
Africa
Asia
Europe
North America
Oceania
South America
continent
continent == "Asia"
FALSE
TRUE

Lets give it a try

https://duke.is/ggplot2


Richer countries tend to be happier countries
In 2019 the happiest countries were all Nordic countries
Finland
Denmark
Switzerland
Iceland
Average happiness
8
7
6
5
4
3
0
30000
60000
90000
120000
GDP per capita