Categories, factors, and colors

Author: Nicholas G Reich

Different kinds of variables

Give some examples of each

- Continuous: variables taking any real number value in a range
- Discrete: variables taking an integer value
- Categorical: variables taking one of a fixed set of values

Categorical variables in R often start as strings

By default, characters are read in as characters, not as factors, although you can force factors. A factor is a special type of R data type that can be used to represent a categorical variable with a fixed number of responses.

```
library(tidyverse)
co2 <- read_csv("../../data/co2emissions.csv")
head(co2)

## # A tibble: 6 x 3
##    Year   CO2 Type
##   <dbl> <dbl> <chr>
## 1  1980  81.2 Rural Diesel
## 2  1981  89.9 Rural Diesel
## 3  1982  89.9 Rural Diesel
## 4  1983  95.7 Rural Diesel
## 5  1984  95.7 Rural Diesel
## 6  1985  95.7 Rural Diesel
```

Tidy aggregation and summary by category

We can use group_by() and summarize() to aggregate and compute summaries by categories. (You will be asked to do this in a future coding challenge.)
For example, here we compute the average CO2 emissions across all years, for each type of vehicle.

```
co2 %>%
  group_by(Type) %>%
  summarize(mean_emissions = mean(CO2))
```

```
## # A tibble: 4 x 2
##   Type           mean_emissions
##   <chr>                   <dbl>
## 1 Rural Diesel             146.
## 2 Rural Gasoline           390.
## 3 Urban Diesel             127.
## 4 Urban Gasoline           669.
```

Tidy aggregation and summary by category

You can compute multiple summaries at once.

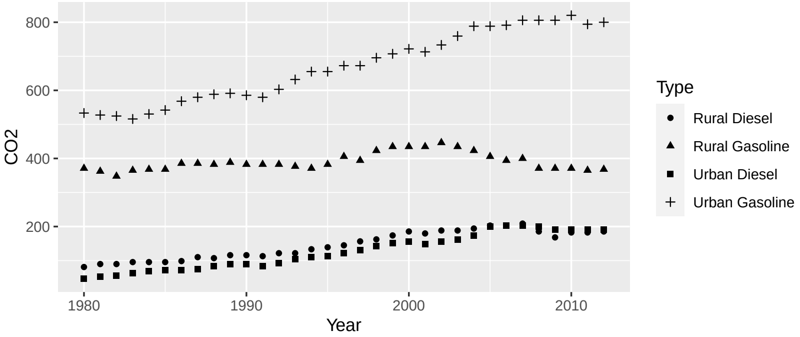
```
co2 %>%
  group_by(Type) %>%
  summarize(
    mean_emissions = mean(CO2),
    max_emissions = max(CO2),
    min_emissions = min(CO2))
```

```
## # A tibble: 4 x 4
##   Type           mean_emissions max_emissions min_emissions
##   <chr>                   <dbl>         <dbl>         <dbl>
## 1 Rural Diesel             146.          209.          81.2
## 2 Rural Gasoline           390.          446.         348.
## 3 Urban Diesel             127.          203.          46.4
## 4 Urban Gasoline           669.          820.         516.
```

Using categorical variables for aesthetics

Note that R translates the character variable into a factor for you without you doing anything.

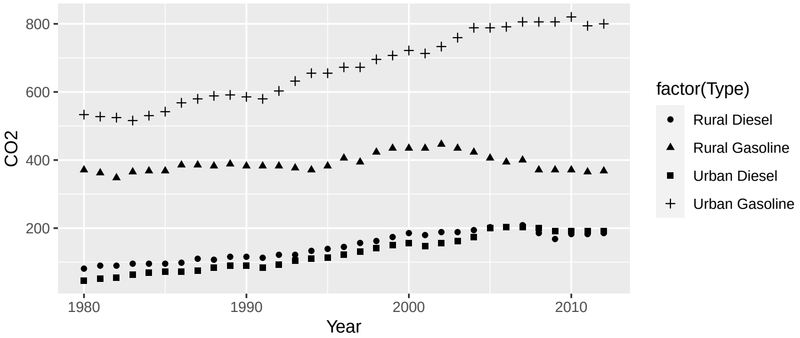
```
ggplot(co2, aes(x = Year, y = CO2, shape = Type, fill = Type))+
  geom_point()
```



Using factors for aesthetics

Note that you can get the same result by explicitly calling Type a `factor`.

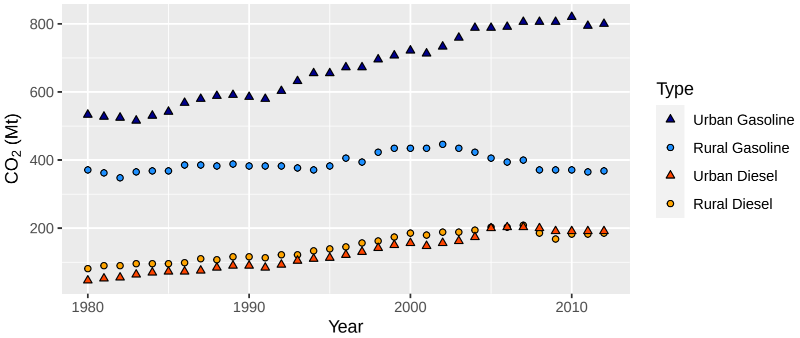
```
ggplot(co2, aes(x = Year, y = CO2, shape = factor(Type), fill = factor(Type)))+
  geom_point()
```



Using factors for aesthetics

And with just a few small tweaks, we can customize

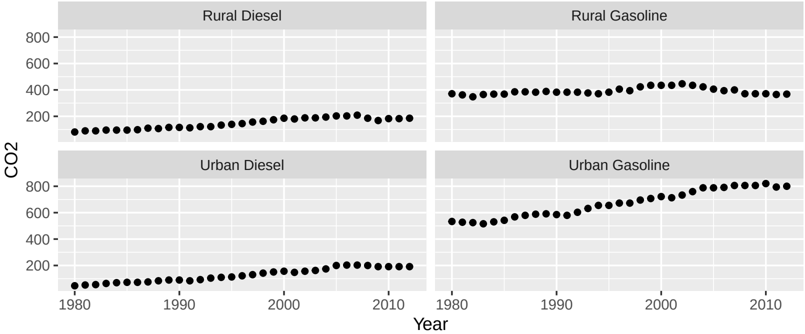
```r
levels <- c("Urban Gasoline", "Rural Gasoline", "Urban Diesel", "Rural Diesel")
ggplot(co2, aes(x = Year, y = CO2, shape = Type, fill = Type)) +
  geom_point() +
  scale_shape_manual(breaks=levels, values=c(24, 21, 24, 21)) +
  scale_fill_manual(breaks = levels,
                    values=c("blue4", "dodgerblue", "orangered", "orange")) +
  ylab(expression(CO[2]*" (Mt)"))
```



Using factors for faceting

Factors (or any variable with a small number of distinct values) can be used to create facets as well.

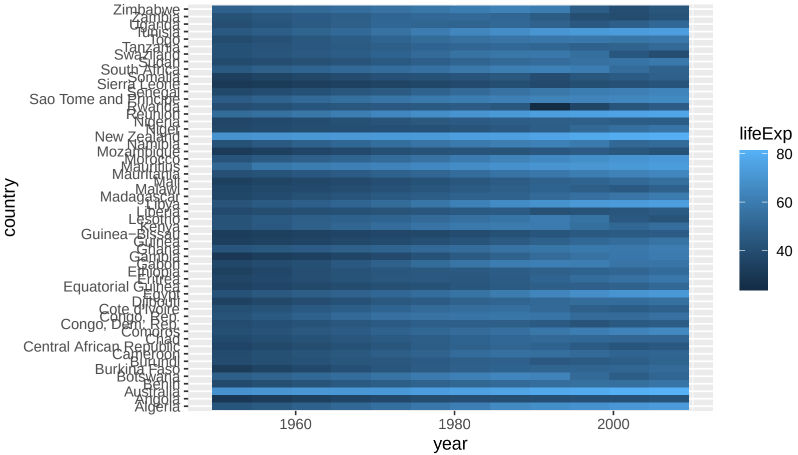
```
ggplot(co2, aes(x = Year, y = CO2)) +
  geom_point() +
  facet_wrap(~Type)
```



Advanced use of factors: ordering

By default, R will sort factors alphanumerically. Reordering factors might help you show more data.

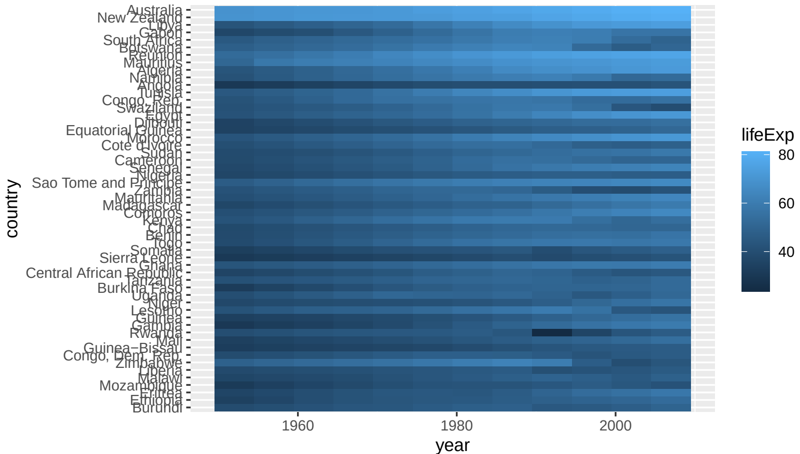
```
gapminder <- read_csv("../../data/gapminder.csv") %>%
    filter(continent %in% c("Africa", "Oceania"))
ggplot(gapminder, aes(x=year, y=country, fill=lifeExp)) +
    geom_tile()
```



Advanced use of factors: ordering

If “order matters” for your categorical variable, then turning it into an ordered factor might be useful.

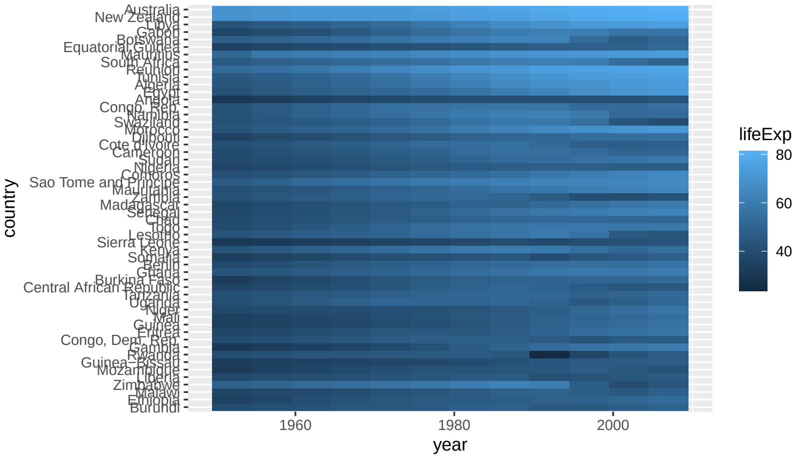
```r
## this redefines country based on average GDP
gapminder <- mutate(gapminder, country = reorder(country, gdpPercap, FUN=mean))

ggplot(gapminder, aes(x=year, y=country, fill=lifeExp)) +
    geom_tile()
```



Advanced use of factors: ordering

Here we order based on the maximum GDP rather than the mean.

```
## this redefines country based on max GDP
gapminder <- mutate(gapminder, country = reorder(country, gdpPercap, FUN=max))

ggplot(gapminder, aes(x=year, y=country, fill=lifeExp)) +
    geom_tile()
```



Using color

Three main types of color palettes

- sequential: a gradient in one direction
- divergent: a gradient away from a center
- qualitative: categorical groupings

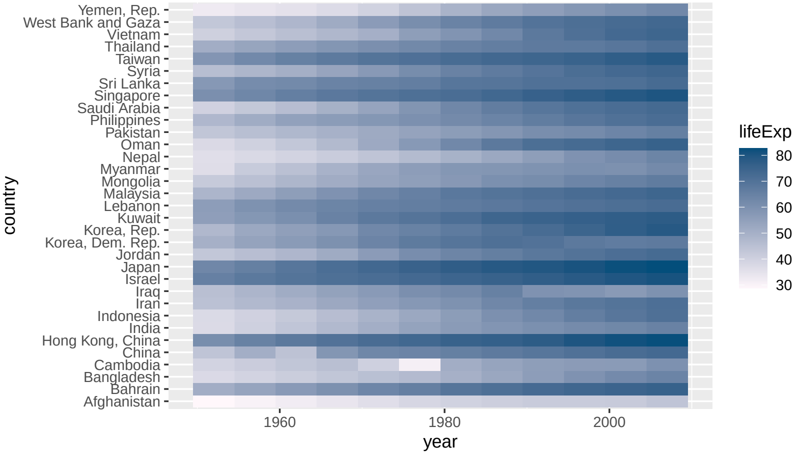
Setting up a small running example

```r
gapminder_asia <- read_csv("../../data/gapminder.csv") %>%
  filter(continent == "Asia")
```

Manual sequential color scale

Using color scales from ColorBrewer: colorbrewer2.org.

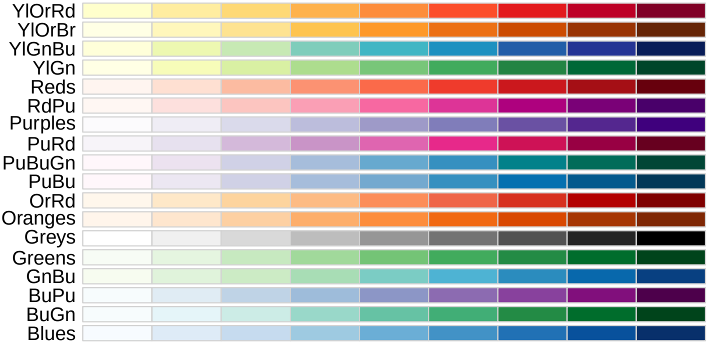
```
ggplot(gapminder_asia, aes(x=year, y=country, fill=lifeExp)) +
    geom_tile() +
    scale_fill_gradient(low="#fff7fb", high="#034e7b")
```



Use `RColorBrewer` palettes like a design pro

Here are all of the sequential palettes available in `RColorBrewer`. Good for showing variables that have a single natural direction:

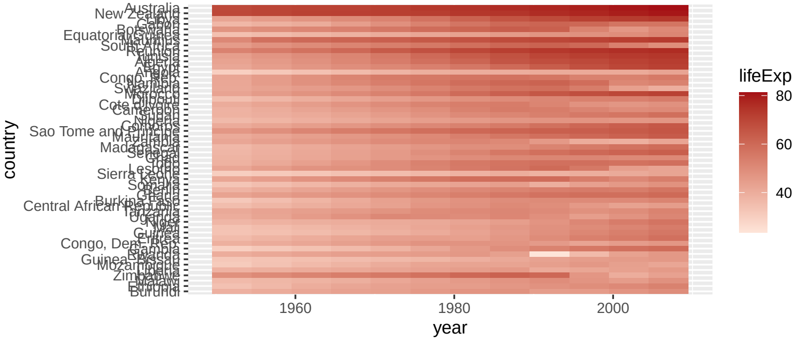
```r
RColorBrewer::display.brewer.all(n=NULL, type="seq", select=NULL, exact.n=TRUE)
```



Using RColorBrewer to pick sequential palettes

Picking palette colors using RColorBrewer.

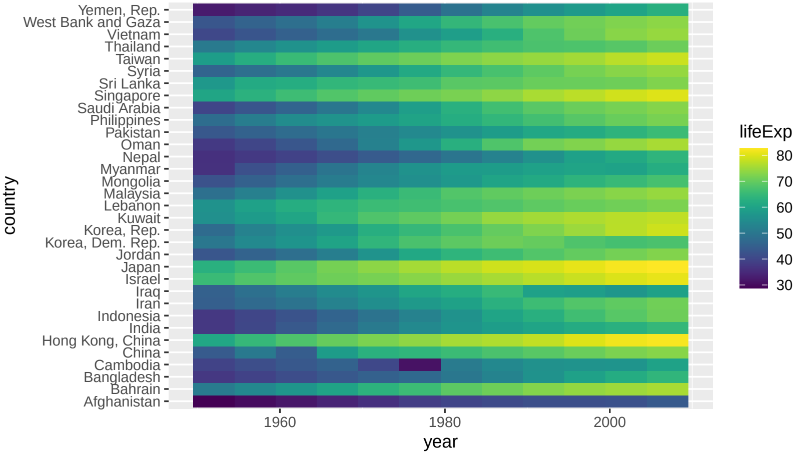
```
(pal <- RColorBrewer::brewer.pal(n=5, name="Reds"))

## [1] "#FEE5D9" "#FCAE91" "#FB6A4A" "#DE2D26" "#A50F15"

ggplot(gapminder, aes(x=year, y=country, fill=lifeExp)) +
    geom_tile() +
    scale_fill_gradient(low=pal[1], high=pal[5])
```



Special hot-cold sequential scales

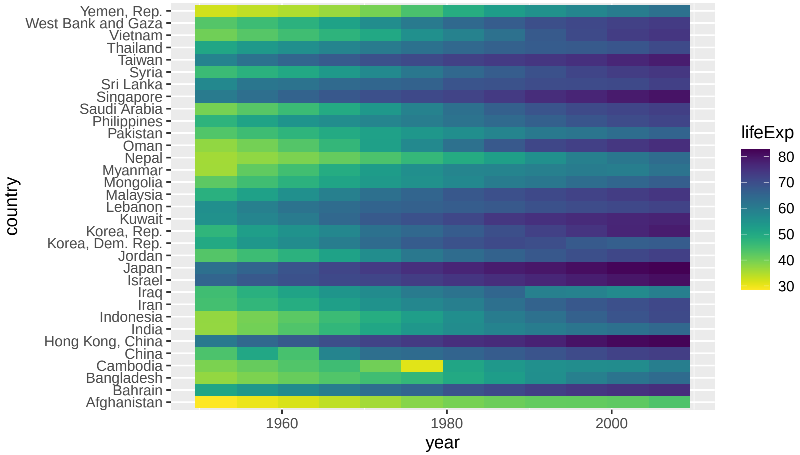
And from the `viridis` package.

```
library(viridis)
ggplot(gapminder_asia, aes(x=year, y=country, fill=lifeExp)) +
    geom_tile() +
    scale_fill_viridis()
```



Special hot-cold sequential scales

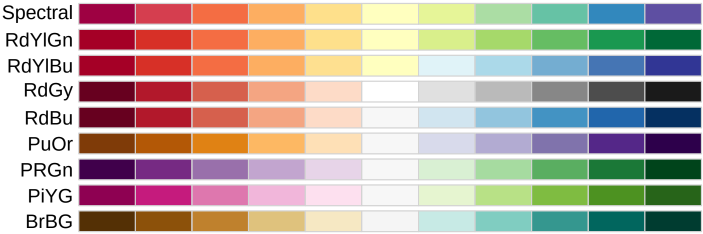
```
ggplot(gapminder_asia, aes(x=year, y=country, fill=lifeExp)) +
    geom_tile() +
    scale_fill_viridis(direction=-1)
```



Divergent palettes

Here are the RColorBrewer divergent palettes: good for showing variables that have a natural midpoint to facilitate a “baseline” comparison.

```
RColorBrewer::display.brewer.all(n=NULL, type="div", select=NULL, exact.n=TRUE)
```



Using divergent palettes to compare values to a baseline

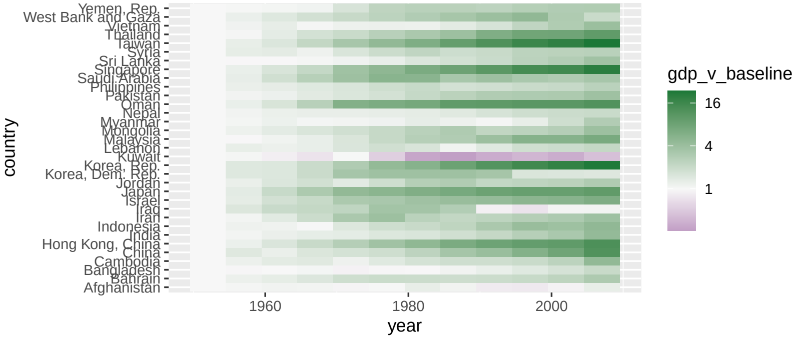
To create a “baseline” comparison for plotting a divergent variable, we create a new GDP variable in the dataset, where we set a country’s GDP in 1952 as the “basline” (equal to 1).

```
pal <- RColorBrewer::brewer.pal(n=9, name="PRGn")

## extract 1952 life expectancy as a baseline
gapminder_asia_1952 <- gapminder_asia %>%
  filter(year==1952) %>%
  select(country, gdp1952 = gdpPercap)

## merge back into original dataset, compute new variable that shows change in life expectancy relative to
gapminder_asia <- gapminder_asia %>%
  left_join(gapminder_asia_1952) %>%
  mutate(gdp_v_baseline = gdpPercap/gdp1952)

head(gapminder_asia)

## # A tibble: 6 x 8
##   country     continent  year lifeExp      pop gdpPercap gdp1952 gdp_v_baseline
##   <chr>       <chr>     <dbl>   <dbl>    <dbl>     <dbl>   <dbl>          <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.    779.          1
## 2 Afghanistan Asia       1957    30.3  9240934      821.    779.          1.05
## 3 Afghanistan Asia       1962    32.0 10267083      853.    779.          1.09
## 4 Afghanistan Asia       1967    34.0 11537966      836.    779.          1.07
## 5 Afghanistan Asia       1972    36.1 13079460      740.    779.          0.949
## 6 Afghanistan Asia       1977    38.4 14880372      786.    779.          1.01
```

Using divergent palettes to compare values to a baseline

Note: when plotting a ratio, it can be useful to use a log transformation on the scale because this will make 0.5 and 2 have the same intensity color away from 1.

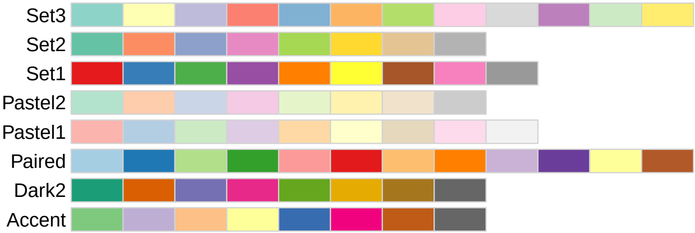
```r
ggplot(gapminder_asia, aes(x=year, y=country, fill=gdp_v_baseline)) +
    geom_tile() +
    scale_fill_gradient2(low=pal[1], mid=pal[5], high=pal[9], midpoint = 0, trans="log2")
```



Qualitative palettes

Here are the available palettes from RColorBrewer that are qualitative. These are good for showing unordered categories for comparison.

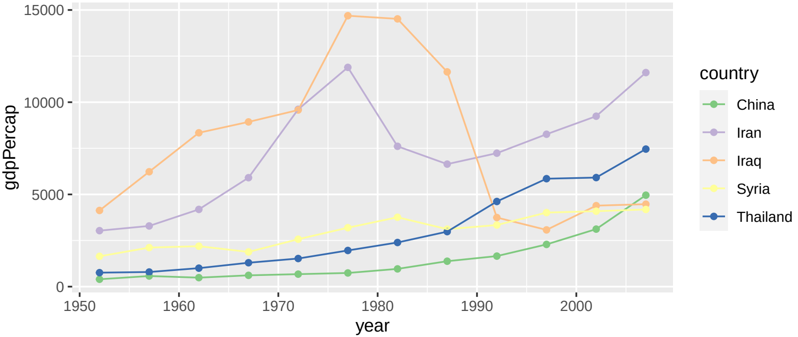
```
RColorBrewer::display.brewer.all(n=NULL, type="qual", select=NULL, exact.n=TRUE)
```



Using qualitative palettes from RColorBrewer

For low-number palettes (usually less than 12 colors) you can request a qualitative palette type.

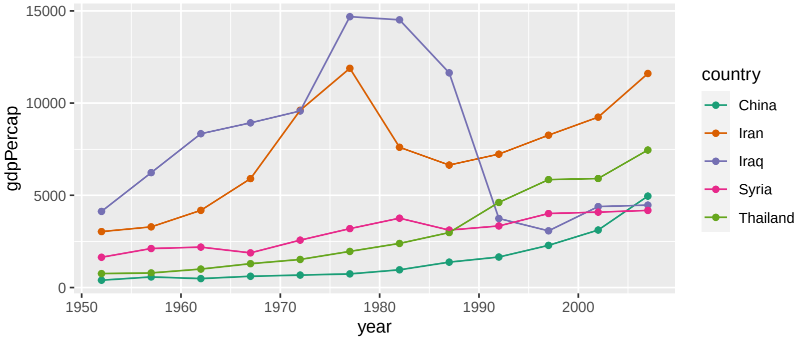
```
gapminder_asia %>%
  filter(country %in% c("Syria", "Iraq", "Iran", "China", "Thailand")) %>%
  ggplot(aes(x=year, y=gdpPercap, color=country)) +
  geom_point() + geom_line() +
  scale_color_brewer(type = "qual")
```



Using qualitative palettes from RColorBrewer

You can also specify a palette directly.

```
gapminder_asia %>%
  filter(country %in% c("Syria", "Iraq", "Iran", "China", "Thailand")) %>%
  ggplot(aes(x=year, y=gdpPercap, color=country)) +
  geom_point() + geom_line() +
  scale_color_brewer(palette = "Dark2")
```


gdpPercap
year
country
China
Iran
Iraq
Syria
Thailand