### Why plot?

#### Raw Data

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>55.38</td>
<td>97.18</td>
</tr>
<tr>
<td>51.53</td>
<td>96.02</td>
</tr>
<tr>
<td>46.15</td>
<td>94.49</td>
</tr>
<tr>
<td>42.82</td>
<td>91.41</td>
</tr>
<tr>
<td>40.76</td>
<td>88.33</td>
</tr>
<tr>
<td>38.71</td>
<td>84.87</td>
</tr>
<tr>
<td>35.64</td>
<td>79.87</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

#### Summaries

<table>
<thead>
<tr>
<th>Statistic</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>54.26</td>
<td>47.83</td>
</tr>
<tr>
<td>SD</td>
<td>16.7</td>
<td>26.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Regression Statistic Y=X</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>53.4</td>
</tr>
<tr>
<td>Slope</td>
<td>-0.10</td>
</tr>
<tr>
<td>Correlation</td>
<td>-0.06</td>
</tr>
</tbody>
</table>
Why plot?

Data from https://cran.r-project.org/package=datasauRus
## Installing ggplot2

- Even though the package is sometimes just referred to as "ggplot", the package name is "ggplot2"

- `ggplot` is included in the `tidyverse` package. To load the `tidyverse` package, run
  - `library(tidyverse)`

- If you get the message "there is no package 'tidyverse' " you must install it first
  - `install.packages("tidyverse")`
  - `library(tidyverse)`

- Be sure to load the package at the start of your session
ggplot2 help

• Use R help "?ggplot"
• Use website (has pictures)
  • http://ggplot2.tidyverse.org/reference/
  • [open now]
• Read Hadley's book

# Gapminder Data

- Dataset tracking life expectancy and per-capita GDP of 142 countries
- Data reported every five years from 1952-2007

- Available in R package on CRAN
  - `install.packages("gapminder")`
  - `library(gapminder)`
  - `View(gapminder)`
## Gapminder Data

A tibble: 1,704 × 6

<table>
<thead>
<tr>
<th>country</th>
<th>continent</th>
<th>year</th>
<th>lifeExp</th>
<th>pop</th>
<th>gdpPercap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afghanistan</td>
<td>Asia</td>
<td>1952</td>
<td>28.8</td>
<td>8425333</td>
<td>779.</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>Asia</td>
<td>1957</td>
<td>30.3</td>
<td>9240934</td>
<td>821.</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>Asia</td>
<td>1962</td>
<td>32.0</td>
<td>10267083</td>
<td>853.</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>Asia</td>
<td>1967</td>
<td>34.0</td>
<td>11537966</td>
<td>836.</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>Asia</td>
<td>1972</td>
<td>36.1</td>
<td>13079460</td>
<td>740.</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>Asia</td>
<td>1977</td>
<td>38.4</td>
<td>14880372</td>
<td>786.</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>Asia</td>
<td>1982</td>
<td>39.9</td>
<td>12881816</td>
<td>978.</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>Asia</td>
<td>1987</td>
<td>40.8</td>
<td>13867957</td>
<td>852.</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>Asia</td>
<td>1992</td>
<td>41.7</td>
<td>16317921</td>
<td>649.</td>
</tr>
<tr>
<td>Afghanistan</td>
<td>Asia</td>
<td>1997</td>
<td>41.8</td>
<td>22227415</td>
<td>635.</td>
</tr>
</tbody>
</table>

# ... with 1,694 more rows
Start Plot

ggplot(data = gapminder)
Add x variable

ggplot(data = gapminder) +
aes(x = gdpPercap)
Fix x label

ggplot(data = gapminder) +
aes(x = gdpPercap) +
labs(x = "GDP per capita")
Add y variable

ggplot(data = gapminder) +
  aes(x = gdpPercap) +
  labs(x = "GDP per capita") +
  aes(y = lifeExp)
Fix x label

```r
ggplot(data = gapminder) +
  aes(x = gdpPercap) +
  labs(x = "GDP per capita") +
  aes(y = lifeExp) +
  labs(y = "Life Expectancy")
```
**Draw points**

```r
ggplot(data = gapminder) + 
aes(x = gdpPercap) + 
labs(x = "GDP per capita") + 
aes(y = lifeExp) + 
labs(y = "Life Expectancy") + 
geom_point()
```

#add-points
Color by continent

ggplot(data = gapminder) +
aes(x = gdpPercap) +
labs(x = "GDP per capita") +
aes(y = lifeExp) +
labs(y = "Life Expectancy") +
geom_point() +
aes(color=continent)
Size by population

ggplot(data = gapminder) +
aes(x = gdpPercap) +
labs(x = "GDP per capita") +
aes(y = lifeExp) +
labs(y = "Life Expectancy") +
geom_point() +
aes(color=continent) +
aes(size = pop)
Small plots by year

ggplot(data = gapminder) +
aes(x = gdpPercap) +
labs(x = "GDP per capita") +
aes(y = lifeExp) +
labs(y = "Life Expectancy") +
geom_point() +
aes(color=continent) +
aes(size = pop) +
facet_wrap(vars(year))
Filter years with dplyr

gapminder %>%
  filter(
    year %in% c(1957, 2007)
  ) %>%
ggplot(data = .) +
aes(x = gdpPercap) +
labs(x = "GDP per capita") +
aes(y = lifeExp) +
labs(y = "Life Expectancy") +
geom_point() +
aes(color = continent) +
aes(size = pop) +
facet_wrap(vars(year), nrow = 2)
Reorganize code (optional)

gapminder %>%
  filter(
    year %in% c(1957, 2007)
  ) %>%
  ggplot(data = .) +
  geom_point(mapping = aes(  
    x = gdpPercap, y = lifeExp,  
    color=continent, size = pop)) +
  facet_wrap(vars(year), nrow=2) +
  labs(x = "GDP per capita", y = "Life Expectancy")

Combine all the aes() options into one. Pass as the mapping= arguments of the geometry

Combine the labs() together
Aesthetics, aes()

- Mappings between a column of your data and some property of the geometry being drawn

- Can pass as the `mapping=` argument
  - `ggplot(data=, mapping=)`
  - `geom_xxx(mapping=, data=, ...)`

- If unnamed, `aes()` assumes the first two arguments are x and y
  - `aes(gdpPercap, lifeExp)`
  - `aes(x = gdpPercap, y = lifeExp)`
  - `aes(y = lifeExp, x = gdpPercap)`

- Never use "$" inside `aes()`
What other aes() does geom_point() know?

- Help page: "?geom_point"
- Check out the “Aesthetics” section of the help page
- Running vignette("ggplot2-specs") brings up more documentation

<table>
<thead>
<tr>
<th> </th>
<th> </th>
<th> </th>
<th> </th>
</tr>
</thead>
<tbody>
<tr>
<td>x (required)</td>
<td>y (required)</td>
<td>alpha</td>
<td>colour (color)</td>
</tr>
<tr>
<td>fill</td>
<td>group</td>
<td>shape</td>
<td>size</td>
</tr>
<tr>
<td>stroke</td>
<td> </td>
<td> </td>
<td> </td>
</tr>
</tbody>
</table>
Change point shape?

gap2007 <- gapminder %>%
  filter(year == 2007)

ggplot(data = gap2007) +
aes(x = gdpPercap) +
aes(y = lifeExp) +
geom_point() +
aes( ?? = ?? )

How can you get different shapes for each continent?
Fix values outside aes()

```r
ggplot(data = gap2007) +
  aes(x = gdpPercap) +
  aes(y = lifeExp) +
  geom_point(
    size = 3,
    color = "blueviolet"
  )
```

If you want to set a value not related to your data, do so in the geometry layer, outside of aes()
Pick your favorite color

ggplot(data = gap2007) +
aes(x = gdpPerCap) +
aes(y = lifeExp) +
geom_point(
  size = 3,
  color = " 
)

# list all R color names
colors()

# choose 10 random colors
sample(colors(), 10)

# or specify a hex value
"#8A2BE2"

Find a cool color
Geometries

- `geom_point()` is just one of many geometries
- It is used to make scatter plots
- Works best with two continuous variables
- What if we wanted to look at a distribution for a single continuous variable?
`ggplot(gap2007) + aes(x=lifeExp) + geom_histogram()`

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
Single Variable Plot

ggplot(gap2007) + aes(x=lifeExp) + geom_density()
Single Variable Plot

ggplot(gap2007) + aes(x=lifeExp) + geom_density(color="firebrick")
ggplot(gap2007) + aes(x=lifeExp) + geom_density(fill="firebrick")
Single Variable Plot

```
library(ggplot2)

ggplot(gap2007) + aes(x=lifeExp, fill=continent) + geom_density(alpha=.2)
```
Single Variable Across Groups

```
#box-plot

ggplot(gap2007) + aes(x=continent, y=lifeExp) + geom_boxplot()
```
How can you get this plot?

ggplot(gap2007) +
  aes(x = continent) +
  aes(y = lifeExp) +
  geom_???()
Single Variable Across Groups

```r
ggplot(gap2007) + aes(x=continent, y=lifeExp) + geom_violin() + geom_jitter()
```
Change Across Time

```r
ggplot(gapminder) + aes(x=year, y=lifeExp) + geom_point()
```
Change Across Time

`ggplot(gapminder) + aes(x=year, y=lifeExp) + geom_line()`

What Happened??
Change Across Time

ggplot(gapminder) + aes(x=year, y=lifeExp) + geom_line(aes(group=country))
Smoothing Trends

```r
ggplot(gap2007) + aes(x=gdpPercap, y=lifeExp) + geom_point() + geom_smooth()
```
Smoothing Trends

```
library(ggplot2)

# Create a data frame
gap2007 <- gapminder[, c("country", "year", "lifeExp", "gdpcpp")]

# Plot life expectancy against GDP per capita
ggplot(gap2007) + 
  aes(x = gdpcpp, y = lifeExp) + 
  geom_point() + 
  geom_smooth(method = "lm")
```
Trend per continent?

ggplot(gapminder) +
  aes(??) +
  geom_??(??)

How can we plot a trend line per continent?
# Plot the number of countries in each continent

ggplot(gap2007) +
  aes(x=continent) +
  geom_bar()
Tidy data

- ggplot works best with tidy data
- Rules of tidy data
  - Each variable in the data set is placed in its own column
  - Each observation is placed in its own row
  - Each value is placed in its own cell
- You data (usually) should be in a single data.frame (or tibble)
- You may need to summarize or transform your data prior to plotting
- Some geoms will do basic summarization for you
# Plot the average life expectancy for each continent

gapminder %>%
  filter(year == 2007) %>%
  group_by(continent) %>%
  summarize(avgle = mean(lifeExp)) %>%
  ggplot(data = .) +
  aes(x = continent, y = avgle) +
  geom_col()
Column charts (for other values)

# Plot the average life expectancy for each continent for 1952 & 2007

gapminder %>%
  filter(year == 2007 | year == 1952) %>%
  group_by(continent, year) %>%
  summarize(avgle = mean(lifeExp)) %>%
  ggplot(data = .) +
  aes(x = continent, y = avgle) +
  geom_col(aes(fill = year))
Column charts (for other values)

# Plot the average life expectancy for each continent for 1952 & 2007

gapminder %>%
  filter(year == 2007 | year == 1952) %>%
  group_by(continent, year) %>%
  summarize(avgle = mean(lifeExp)) %>%
  ggplot(data = .) +
  aes(x = continent, y = avgle) +
  geom_col(aes(fill = factor(year)))
Column charts (for other values)

# Plot the average life expectancy for each continent for 1952 & 2007

```r
gapminder %>%
  filter(year==2007 | year==1952) %>%
  group_by(continent, year) %>%
  summarize(avgle = mean(lifeExp)) %>%
  ggplot(data = .) +
  aes(x=continent, y=avgle) +
  geom_col( aes(fill=factor(year)) ,
  position="dodge")
```
What does `ggplot()` do?

- The `ggplot()` function creates a "gg/ggplot" object
- You use (+) to add additional instructions to the object to build your plot (Note: do not use `%>%` to add layers)
- Can be saved to a variable
- Doesn't actually "draw" the plot, that only happen

```r
p <- ggplot(data = gap2007) +
aes(x = gdpPercap, y = lifeExp) +
geom_point()
# nothing happens until
p
print(p)
```
Global options vs Layer options

**ggplot object**
- Global Data
- Global Mapping (aes)
- Facets
- **Layers**
- Scales
- Theme

**Layer**
- Geometry
- Mapping
- Data

**Layer**
- Geometry
- Mapping
- Data

**Layer**
- Geometry
- Mapping
- Data
More about aes()

• By default, layer aes() values are inherited from ggplot()
• Disable inheritance with geom_<name>(…, inherit.aes=FALSE)
• You may also add, override, or remove aes() values

<table>
<thead>
<tr>
<th>global mapping</th>
<th>layer mapping</th>
<th>Resulting aesthetics</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>aes(x=year, y=pop)</td>
<td>aes(color=continent)</td>
<td>aes(x=year, y=pop, color=continent)</td>
<td>Add</td>
</tr>
<tr>
<td>aes(x=year, y=pop)</td>
<td>aes(y=lifeExp)</td>
<td>aes(x=year, y=lifeExp)</td>
<td>Override</td>
</tr>
<tr>
<td>aes(x=year, y=pop)</td>
<td>aes(y=NULL)</td>
<td>aes(x=year)</td>
<td>Remove</td>
</tr>
</tbody>
</table>

```r
ggplot(mapping = aes(<global>)) +
aes(<global>) +
geom_<name>(mapping = aes(<layer>))
```
What is the final mapping?

ggplot(gapminder, aes(x=year, y=pop)) +
aes(y=country) +
geom_point(mapping=aes(y=gdpPercap)) +
aes(y=lifeExp)

A) aes(x=year, y=pop)
B) aes(x=year, y=country)
C) aes(x=year, y=gdpPercap)
D) aes(x=year, y=lifeExp)

Which will be the final mapping for the points?
Different data for different layers

# Only label large populations

to_label <- gap2007 %>%
  filter(pop > 200000000)

gap2007 %>%
  ggplot(data = .) +
  aes(x=gdpPerCap, y=lifeExp) +
  geom_point(aes(size=pop, color=continent)) +
  geom_text(aes(label=country), data=to_label)
Faceting

- Divide plot into subgroups and draw layers for each set
- Two primary options
  - Grid – up to two variables: one row rows, one for cols
  - Wrap – no panel structure
- Each facet gets all layers
- Each layer's data split on same variables
## facet_wrap()

- Requires a list of column names wrapped in `vars()`.
- Each combination of columns gets its own panel.
- `scales=` options:
  - "fixed" scales same in all
  - "free_x" x range can vary
  - "free_y" y range can vary
  - "free" domain and range can change for each panel
- `ncol=` number of columns

```r
ggplot(gapminder) +
aes(x=lifeExp) +
geom_density(fill="grey40") +
facet_wrap(vars(continent))
```
You can specify a list of column names wrapped in `vars()` to, `rows=`, `cols=`, or both

- Share axis across panels
- Results in “rectangular” output

```r
p <- gapminder %>%
  mutate(decade=year%%10*10) %>%
  group_by(continent, decade, country) %>%
  select(-year) %>%
  summarize_all(mean) %>%
  ggplot() +
  aes(gdpPercap, lifeExp) +
  geom_point()

p + facet_grid(rows=vars(decade),
               cols=vars(continent))
p + facet_grid(rows=vars(decade))
p + facet_grid(cols=vars(continent))
```
# Wrap vs Grid

<table>
<thead>
<tr>
<th>Grid</th>
<th>Wrap</th>
</tr>
</thead>
<tbody>
<tr>
<td>• rows and columns have meaning</td>
<td>• rows and columns do not have meaning</td>
</tr>
<tr>
<td>• each rows/column represents a single level of a discrete variable</td>
<td>• Labels are at the top of each sub plot</td>
</tr>
<tr>
<td>• Labels are at the top of each row/column</td>
<td>• Labels are at the top of each sub plot</td>
</tr>
</tbody>
</table>
## Scales

- Scales describe how raw data values should be converted to aesthetic values
- Default scales are determined by the class of the variables in your data
- Each aesthetic (e.g., color, fill, size, shape) can have at most one scale
- Scales can have guides
  - Axes for positions
  - Legends for everything else
Scaling based on data type

- Color is mapped differently depending if year is a numeric or character vector.
- Default color scales:
  - Numeric: `scale_color_continuous()`
  - Factor: `scale_color_discrete()`

```r
p <- gapminder %>%
  filter(country=="United States") %>%
  ggplot(aes(gdpPercap, lifeExp))

# Compare output
p + geom_point(aes(color=year))
p + geom_point(aes(color=factor(year)))
```
Manually setting discrete colors

<table>
<thead>
<tr>
<th>You can customize the default color scales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Or you can create your own manual scale</td>
</tr>
<tr>
<td>• <a href="http://colorbrewer2.org/">http://colorbrewer2.org/</a></td>
</tr>
<tr>
<td>RColorBrewer::display.brewer.all()</td>
</tr>
<tr>
<td>• Get RGB values from anywhere</td>
</tr>
<tr>
<td>• <a href="http://colormind.io/">http://colormind.io/</a></td>
</tr>
<tr>
<td>• <a href="http://color.adobe.com">http://color.adobe.com</a></td>
</tr>
<tr>
<td>• scale_color_manual expects</td>
</tr>
<tr>
<td>• Vector of color values=</td>
</tr>
<tr>
<td>• Named vector of color values=</td>
</tr>
<tr>
<td>• Vector of color values= for levels named in breaks=</td>
</tr>
</tbody>
</table>

```r
# scale_color_manual

p <- ggplot(mpg, aes(displ, hwy)) + geom_point(aes(color = drv))
p + scale_color_manual(values=c("4"="#F2CED8", "f"="#88B8B8", "r"="#DE7E68"))
p + scale_color_brewer(palette="Paired")
```
Expanding `scale_color_manual()`

- `+ scale_color_manual(
  values=c(
    "4"="#F2CED8",
    "f"="#88B8B8",
    "r"="#DE7E68"))

- `+ scale_color_manual(
  values=c("#F2CED8", "#88B8B8", "#DE7E68"),
  breaks=c("4","f","r"),
  labels=c("4 wheel", "front", "rear"),
  name="Drive")`
Use continent colors

gap2007 %>%
ggplot() +
aes(x=gdpPercap) +
aes(y=lifeExp) +
aes(color = ??) +
geom_point() +
scale_color_(??)( ?? )

How can you use the built-in vector continent_colors to change the colors of the points for each continent?
Manually setting continuous colors

<table>
<thead>
<tr>
<th>Continuous values plotted with gradients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two color gradient</td>
</tr>
<tr>
<td>• <code>scale_color_gradient()</code></td>
</tr>
<tr>
<td>Three color diverging gradient</td>
</tr>
<tr>
<td>• <code>scale_color_gradient2()</code></td>
</tr>
<tr>
<td>N-color gradient</td>
</tr>
<tr>
<td>• <code>scale_color_gradientn()</code></td>
</tr>
<tr>
<td>See all color names in R</td>
</tr>
<tr>
<td>• <code>colors()</code></td>
</tr>
</tbody>
</table>

```r
p <- ggplot(mpg, aes(cty, hwy)) + geom_point(aes(color = scale(displ)))
p + scale_color_gradient(low="white", high="orchid")
p + scale_color_gradient2(low="white", high="orchid", mid="tomato")
p + scale_color_gradientn(colors=c("blue","wheat","green"))
```
Setting vs mapping

- Notice the difference that `color=` makes inside vs outside the `aes()` function.
- Only things inside an `aes()` get a legend (only the mappings).
- If you have a column that has color values, use `scale_color_identity()` to prevent remapping.

```r
# "odd" behavior
ggplot(mpg, aes(cty, hwy)) + geom_point(aes(color = "darkblue"))
ggplot(mpg, aes(cty, hwy)) + geom_point(color = "darkblue")
```
Literal string mappings

• Specifying a literal mapping can be useful if using multiple layers
• Here we add two layers with different smoothers
• We specify a color= in the aes() so we get a nice legend

```r
# mapping a literal value
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_smooth(aes(color = "loess"),
              method = "loess", se = FALSE) +
  geom_smooth(aes(color = "lm"),
              method = "lm", se = FALSE) +
  labs(color = "Method")
```
Axes are just scales as well

<table>
<thead>
<tr>
<th>You can change transformations of x/y axes via scales</th>
</tr>
</thead>
<tbody>
<tr>
<td>• scale_x_log10()</td>
</tr>
<tr>
<td>• scale_x_sqrt()</td>
</tr>
<tr>
<td>• scale_x_reverse()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Can also take finer control over tick marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>• scale_x_continuous() – numeric values</td>
</tr>
<tr>
<td>• scale_x_datetime() – date/time values</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control display of factor levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>• scale_x_discrete()</td>
</tr>
<tr>
<td>• Choose new labels for factor levels</td>
</tr>
</tbody>
</table>
Discrete axes plotting order

- Discrete axes are drawn in the order of the levels() of the corresponding factor.
- You can change that order by changing the axes scale.
- Or you can re-order the factor itself (see "?reorder")

```r
p <- ggplot(mpg, aes(y=hwy))
# default
p + geom_boxplot(aes(drv))
# use scale
p + geom_boxplot(aes(drv)) + scale_x_discrete(limits=c("f","r","4"))
# use data
p + geom_boxplot(aes(reorder(drv, hwy)))
```
### Labeling your plot and axes

- You can label your x and y axes:
  ```r
  ggplot(mpg, aes(cyl)) +
  geom_bar() +
  labs(
    title=expression(y==alpha+beta*x),
    x="Cylinder", y="Count"
  )
  ```

- You can also include mathematical expressions (see "?plotmath"):
  ```r
  + labs(title=expression(y==alpha+beta*x))
  ```

- Setting values to "" shows no label, setting values to NULL removes space for label as well.
### Formatting your tick labels

- **Format 0-1 as percents**
  - `+ scale_y_continuous(labels = scales::percent_format())`

- **Format as dollar amounts**
  - `+ scale_y_continuous(labels = scales::dollar_format("\$"))`

- **Format in thousands**
  - `+ scale_fill_continuous(labels = scales::unit_format("k", 1e-3))`

- **You can pass any function as the labels= argument**
## Formatting your tick marks

- You can set where your tick marks fall with `scale_x_continuous`
  - `breaks` = vector of values where to draw major tick marks
  - `labels` = vector of values with what to draw at those tick marks
  - `minor_breaks` = vector of values where to draw minor (unlabeled) tick marks
  - `trans` = optional transformation to apply to axis
  - `expand` = how far to extend axis past observed data
  - `limits` = lower and upper bound for tick marks

- **For datetime axes**
  - `date_minor_breaks` = units like "1 month" or "2 years"
Coordinate Transformations

coord_flip(): swap x & y axes

```r
p <- ggplot(mpg, aes(drv)) + geom_bar()

P + coord_flip()
```

coord_polar(): make "pie" charts

```r
ggplot(mpg) + geom_bar(aes(factor(1), fill=drv), width=1) + coord_polar(theta="y")
```
Coordinate Transformations

- `coord_cartesian()`: limit the plotting window
  - `xlim` = range of x values `c(lower, upper)`
  - `ylim` = range of y values `c(lower, upper)`
  - Differs from changing limits on scales which will subset data

- `coord_fixed()`: fix the distance ratio for x and y
### Setting a theme

- The overall "look" of a plot is set by the theme.
- Just call one of the theme functions to see all the values you can customize.
- You can create your own theme objects.
- The "ggthemes" package has additional themes to try out.

```r
p <- ggplot(mpg, aes(cty, hwy, color = factor(cyl))) +
  geom_jitter() +
  geom_abline(color="grey50", size=2) +
  ggtitle("My Plot!")

#default
p + theme_grey()
# try these
p + theme_bw()
p + theme_linedraw()
p + theme_light()
p + theme_dark()
p + theme_minimal()
```
Customizing a theme

- **Increase font size for presentation slides**
  - `p + theme_grey(base_size=18)`

- You can customize parts of themes
  - `p + theme(plot.title = element_text(color="red", margin=margin(t=20, b=20)))`
  - `p + theme(panel.background = element_rect(fill = "linen"))`

- Set default theme for session
  - `theme_set(theme(…))`

- Read the ggplot2 book or the “theme” help page on the ggplot2 website for more info
Does anybody have a map?

mapdata <- map_data("world") %>%
  mutate(region = recode(region,
    USA="United States",
    UK="United Kingdom"))

gap2007 %>%
  ggplot() +
  geom_map(aes(map_id=country,
    fill=lifeExp), map=mapdata) +
  expand_limits(x = mapdata$long, y = mapdata$lat) +
  coord_map(projection = "mollweide",
    xlim = c(-180, 180)) +
  ggthemes::theme_map()
ggsave()

- Once you’ve made your masterpiece, use ggsave() to save it
- It will create a file in your current working directory (getwd()/setwd())
- Saves last plot printed
- Looks at file name to determine type
  - `ggsave("plot.pdf")`; `ggsave("plot.eps")`
  - `ggsave("plot.png")`; `ggsave("plot.jpg")`
  - `ggsave("plot.tiff")`; `ggsave("plot.svg")`
- Can also pass plot object to save any plot
  - `ggsave("plot.png",
    ggplot(mpg, aes(cty, hwy)) + geom_point())`
### Vector vs raster

<table>
<thead>
<tr>
<th>Two main image format categories</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vector images</strong></td>
</tr>
<tr>
<td>• Remembers the shapes drawn</td>
</tr>
<tr>
<td>• Infinitely zoom-able</td>
</tr>
<tr>
<td>• pdf, svg, eps,</td>
</tr>
<tr>
<td><strong>Raster/bitmap images</strong></td>
</tr>
<tr>
<td>• Remembers just the pixels of the image</td>
</tr>
<tr>
<td>• Number of pixels depends on the dots per inch (DPI) of your image</td>
</tr>
</tbody>
</table>

Typically vector is better, but if you have lots of points, raster may be easier to work with.
Programming with aesthetics

• The aes() function requires symbols, not variables
• aes_string() allows you to specify columns as characters
• aes() can expand quosures
• Best descriptions of Hadley's POV on this is: vignette("programming", package="dplyr")

```r
# won't work
f <- function(y) {
  ggplot(mpg, aes(cty, y)) + geom_point()
}
f(hwy)

# works

g <- function(y) {
  ggplot(mpg, aes_string("cty", y)) + geom_point()
}
g("hwy")

h <- function(y) {
  ggplot(mpg, aes(cty, !!enquo(y))) + geom_point()
} h(hwy)
```
q()