HDAT9800 Health Data Visualisation & Communication

Chapter 3 Interactive Tutorial -- more on visualisation with `ggplot2`

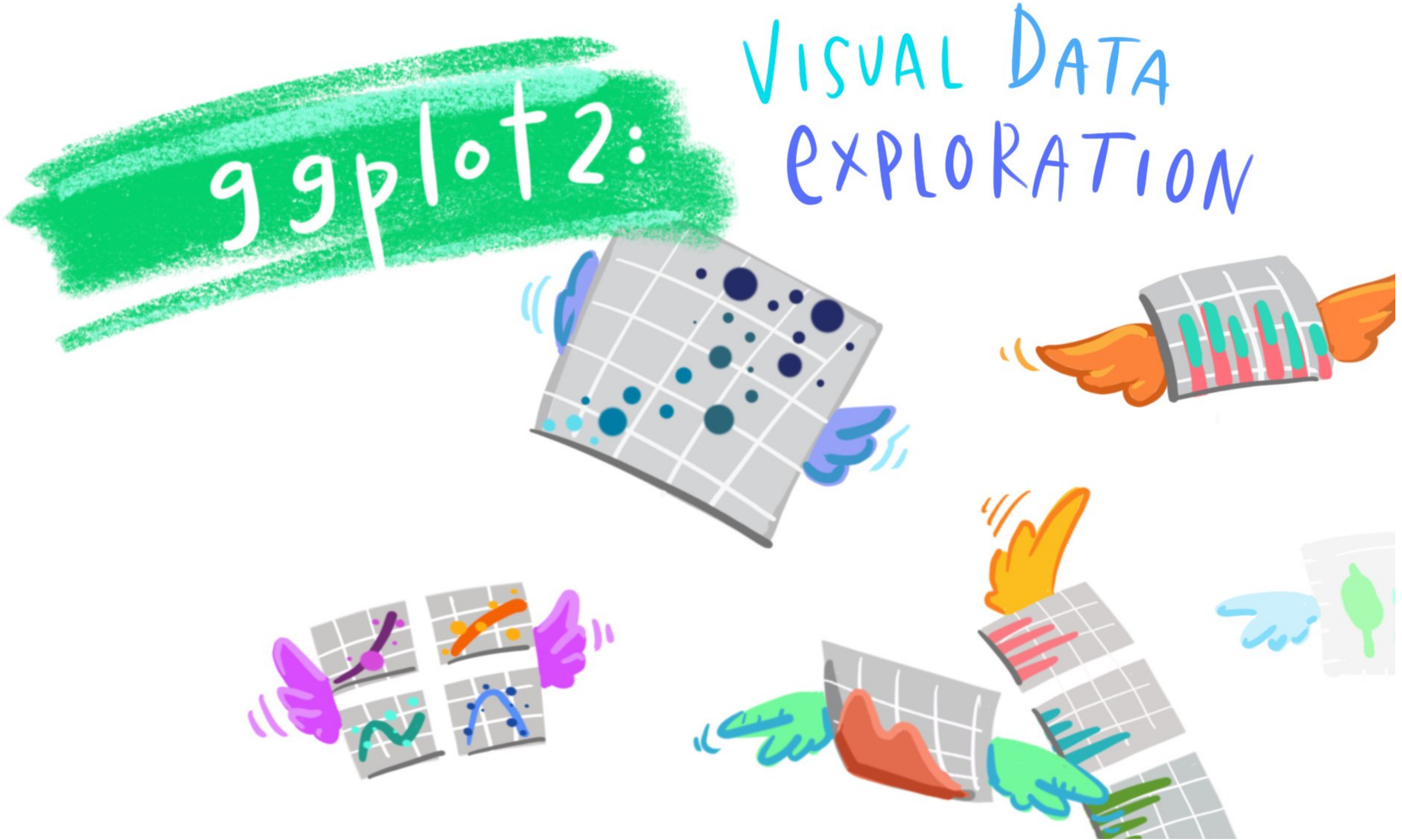
Tim Churches

UNSW Medicine

8th June 2022

Agenda for Chapter 3 interactive session

- Q & A
- core readings (Wilke Chapters 3, 4 & 5 recap)
- diving deeper into `ggplot2` with some hands-on coding
  - a quick look at the `patchwork` package


ggplot2:
Visual Data
Exploration

Chapter 2

- Q & A

Wilke Chapter 3 - coordinate systems and axes

- position scales are needed to determine where on a chart different data values should be shown
- for typical 2D charts, we need two position scales
    - usually these scales correspond to the *x* and *y* coordinates of a plot
    - the geometric arrangement of the scales also needs to be specified (even if implicitly)
    - conventionally the *x* axis is the horizontal axis and the *y* axis is the vertical axis
    - but it doesn't have to be so: the *y* axis could, for example, run at some angle other than 90 degrees to the *x* axis, or one axis could run in a circle and the other run radially to that circle
    - the combination of a set of position scales and their relative geometric arrangement is known as a *coordinate system*

Wilke Chapter 3 - Cartesian coordinates

- the most commonly used coordinate system
    - *x* and *y* axes are orthogonal (at 90 degrees to each other)
    - represent continuous scales, can show both positive and negative values
    - requires a range to be specified for each axis

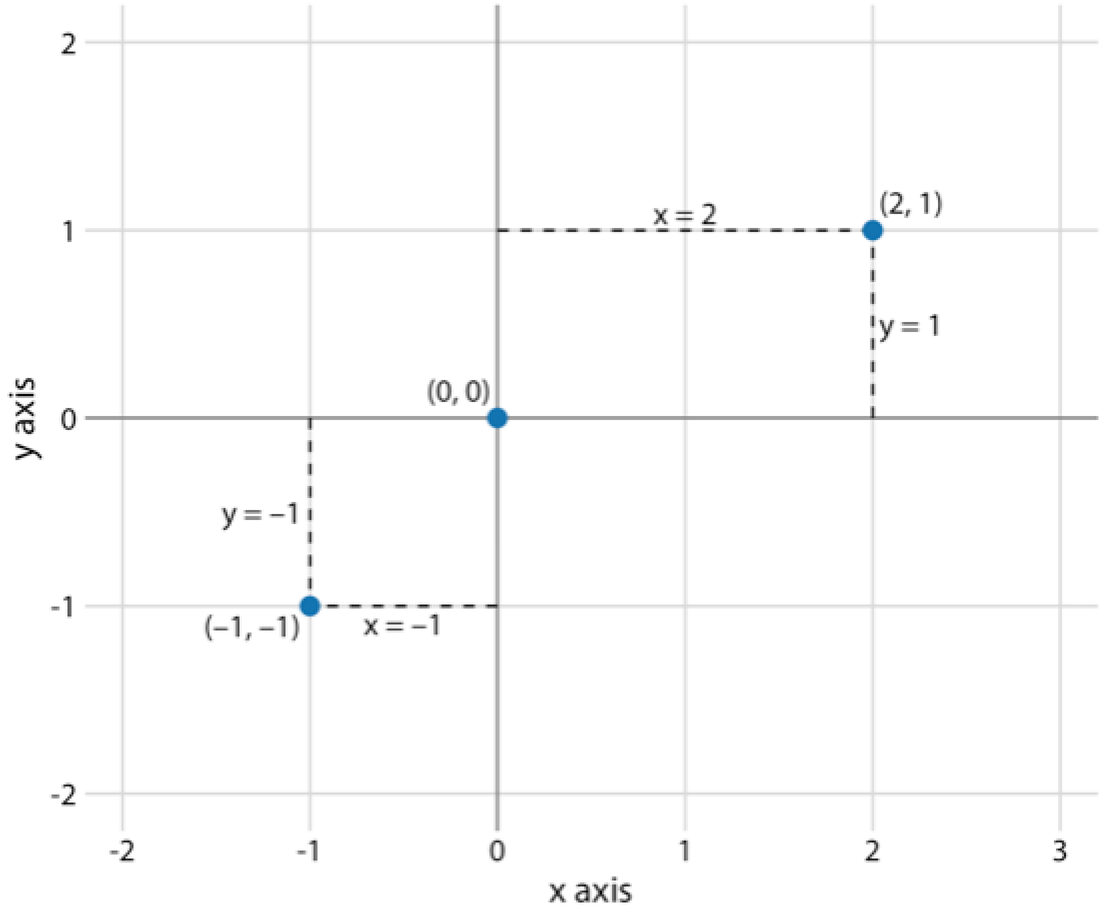


Figure 3.1: Standard cartesian coordinate system. The horizontal axis is conventionally called *x* and the vertical axis *y*. The two axes form a grid with equidistant spacing. Here, both the *x* and the *y* grid lines are separated by units of one. The point (2, 1) is located two *x* units to the right and one *y* unit above the origin (0, 0). The point (-1, -1) is located one *x* unit to the left and one *y* unit below the origin.

Wilke Chapter 3 - Cartesian coordinates, cont'd

- data values almost always have units:
  - temperature in degrees Celsius
  - distance in millimetres, metre, kilometres etc
  - time in minutes, hours, days or dates, months, years
- charts often use scales representing different units for each axis
  - the drawing of axes may be expanded or contracted
  - spacing of units doesn't need to be the same if the *x* and *y* units are different
  - Cleveland's principle of 45 degree "banking"
  - https://idl.cs.washington.edu/files/2006-Banking-InfoVis.pdf
  - Banking slopes to 45 degrees with `ggthemes`

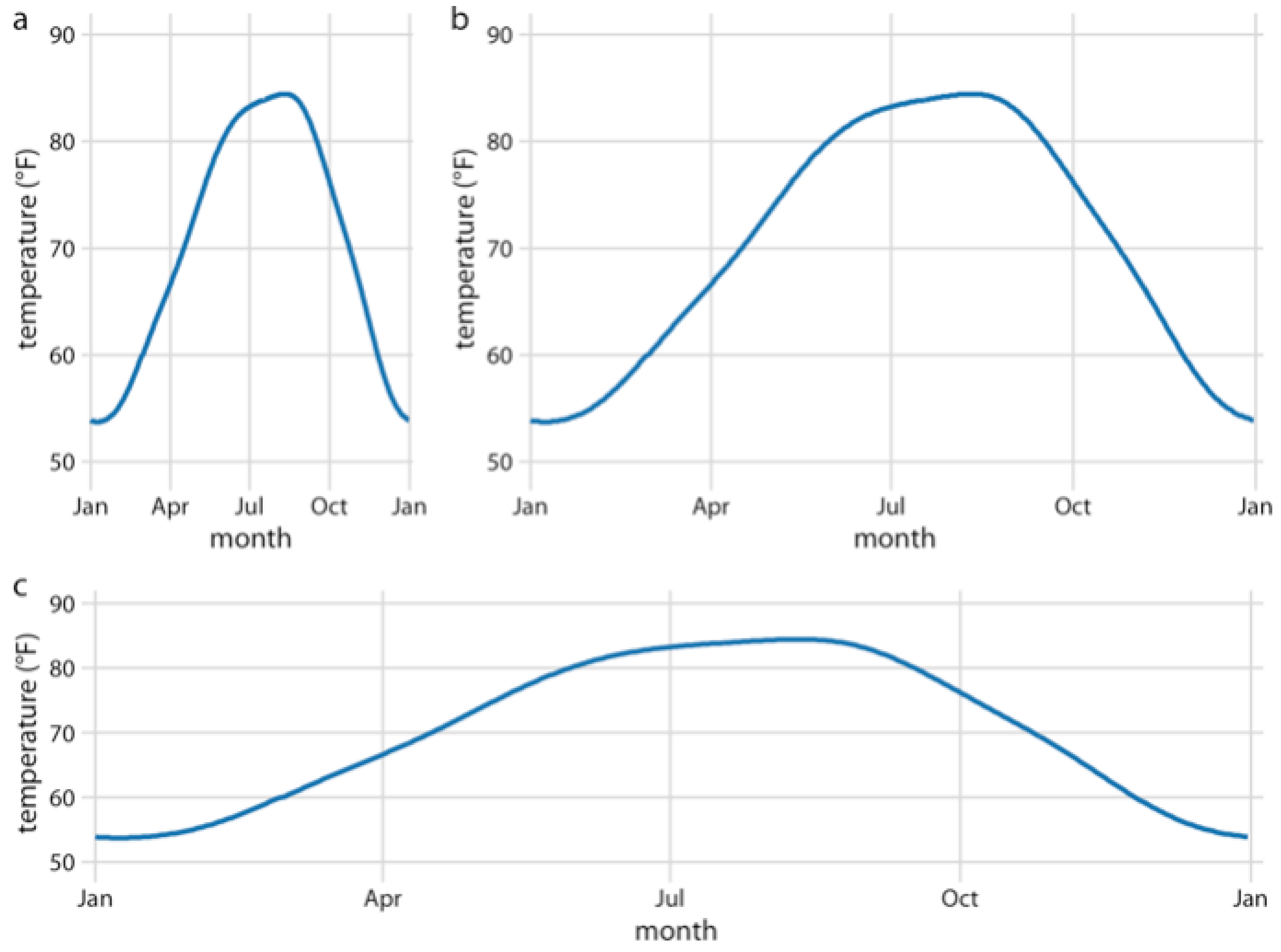


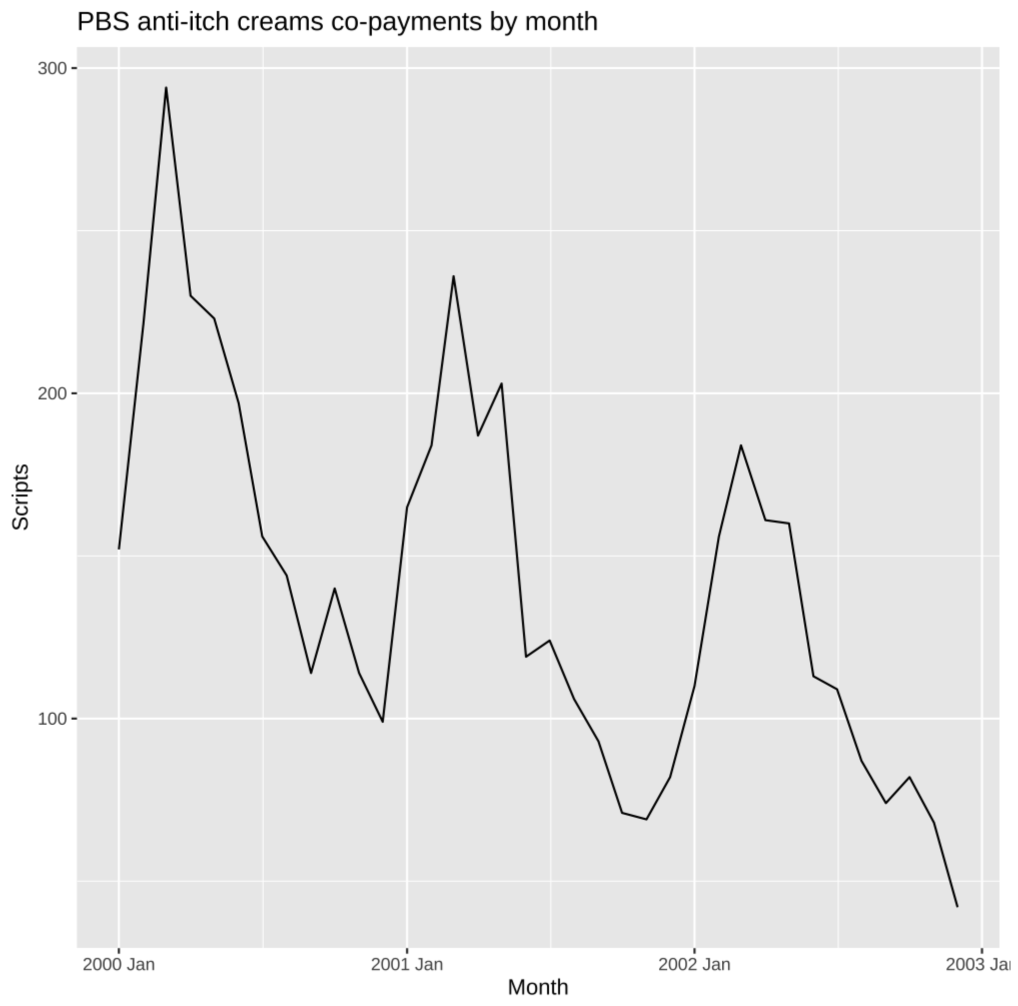
Figure 3.2: Daily temperature normals for Houston, TX. Temperature is mapped to the *y* axis and day of the year to the *x* axis. Parts (a), (b), and (c) show the same figure in different aspect ratios. All three parts are valid visualizations of the temperature data. Data source: NOAA.

Example of 45 degree banking

- use the `PBS` dataset of the Australian Pharmaceutical Benefits Scheme therapeutic drug supplies by type of drug by month
  - contained in the `tsibbledata` package
  - a `tsibble` is a special type of R data frame (tibble) optimised for time series analysis
- default plot of two years of prescriptions filled for "ANTIPRURITICS,INCL ANTIHIST,ANESTHET,ETC"

```r
library(tidyverse)
library(tsibbledata)
library(tsibble)
library(ggthemes)

PBS %>%
  filter(ATC2_desc == "ANTIPRURITICS,INCL ANTIHIST,ANESTHET,ETC",
         Month > make_yearmonth(year = 1999L, month = 12L),
         Month <= make_yearmonth(year = 2002L, month = 12L)) %>%
  ggplot(aes(x=Month, y=Scripts)) + geom_line() + scale_x_yearmonth()
```


PBS anti-itch creams co-payments by month
300
200
100
Scripts
2000 Jan
2001 Jan
2002 Jan
2003 Ja
Month

Add 45 degree banking

- using the `bank_slopes()` function in `ggthemes`

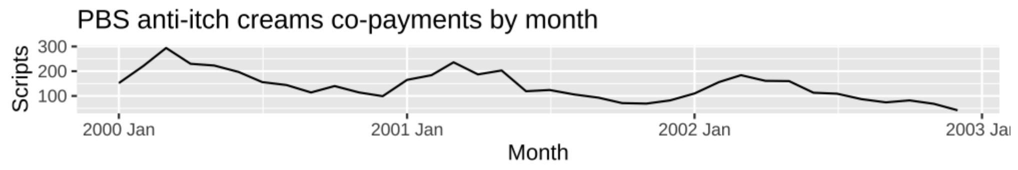
```r
anti_itch <- PBS %>%
  filter(ATC2_desc == "ANTIPRURITICS,INCL ANTIHIST,ANESTHET,ETC",
         Month > make_yearmonth(year = 1999L, month = 12L),
         Month <= make_yearmonth(year = 2002L, month = 12L),
         Type == "Co-payments") %>%
  arrange(Month) %>%
  mutate(Row = row_number()) %>%
  select(Row, Scripts)

aspect_ratio  <- bank_slopes(anti_itch$Row, anti_itch$Scripts)
```

Add 45 degree banking

- using the `bank_slopes()` function in `ggthemes`

```
PBS %>%
  filter(ATC2_desc == "ANTIPRURITICS,INCL ANTIHIST,ANESTHET,ETC",
         Month > make_yearmonth(year = 1999L, month = 12L),
         Month <= make_yearmonth(year = 2002L, month = 12L),
         Type == "Co-payments") %>%
  ggplot(aes(x=Month, y=Scripts)) +
  geom_line() +
  scale_x_yearmonth() +
  ggtitle("PBS anti-itch creams co-payments by month") +
  coord_fixed(ratio = aspect_ratio)
```


PBS anti-itch creams co-payments by month
Scripts
300
200
100
2000 Jan
2001 Jan
2002 Jan
2003 Jan
Month

Wilke Chapter 3 - same units on both axes

- if the units for the *x* and *y* axes are the same, then the scale should map them to the chart with the same grid spacings
- Cartesian coordinates are invariant under linear transformations (such as "centring")

Wilke Chapter 3 - nonlinear axes

- Cartesian coordinate systems are linear: grid lines and tick marks are spaced evenly
- but with a nonlinear scale, even spacing in data units corresponds to uneven spacing in the visualisation, or even spacing in the visualisation corresponds to unequal spacing in the underlying data units
- *logarithmic* scales are the most commonly used type of nonlinear scale
  - they are *multiplicative*, meaning they are linear in multiplication
  - need to log-transform the underlying data values and exponentiate the numbers shown on the chart
  - or leave the data values as they are and use a log scale on the chart
  - mathematically, there is no difference between plotting the log-transformed data on a linear scale or plotting the original data on a logarithmic scale
  - the only difference lies in the labelling for the individual axis ticks and for the axis as a whole

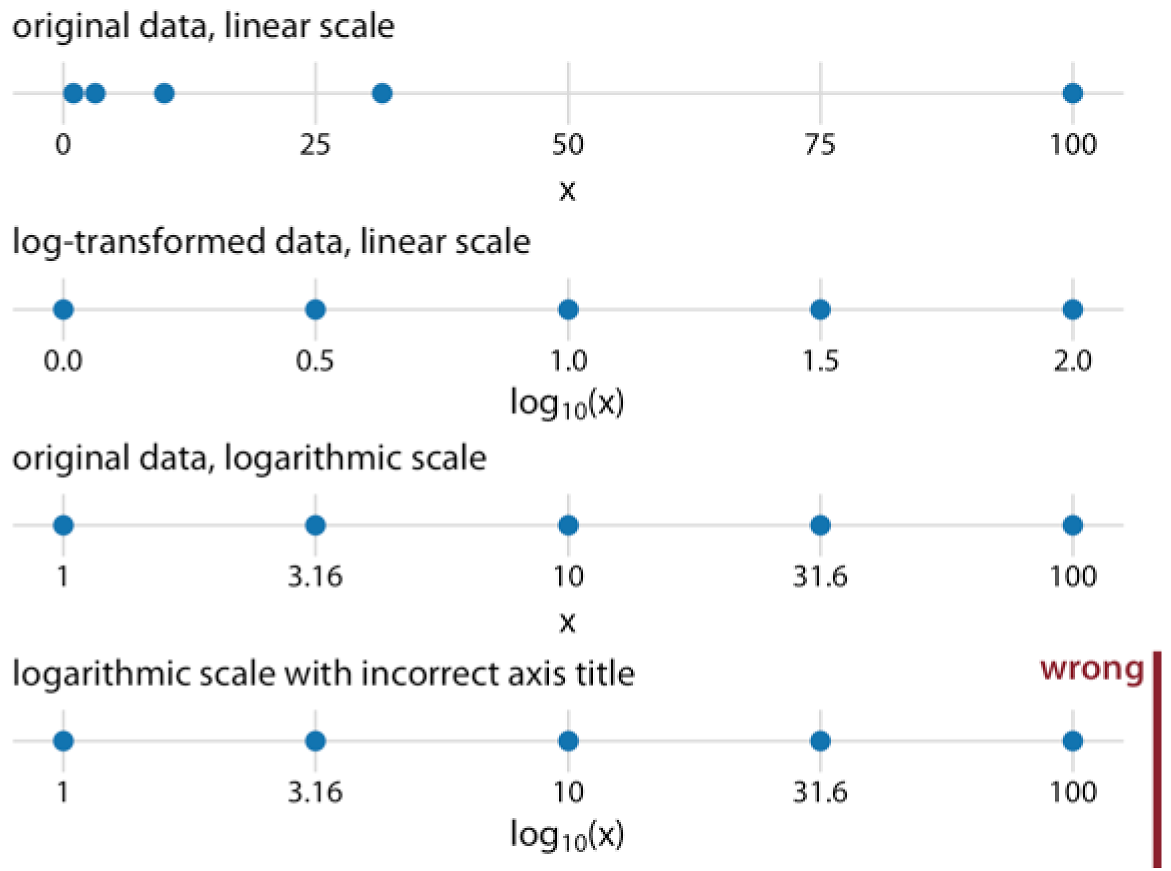


Figure 3.4: Relationship between linear and logarithmic scales. The dots correspond to data values 1, 3.16, 10, 31.6, 100, which are evenly-spaced numbers on a logarithmic scale. We can display these data points on a linear scale, we can log-transform them and then show on a linear scale, or we can show them on a logarithmic scale. Importantly, the correct axis title for a logarithmic scale is the name of the variable shown, not the logarithm of that variable.

Wilke Chapter 3 - plotting ratios

- multiplication on a log scale looks like addition on a linear scale
    - thus log scales are useful for data that have been obtained by multiplication or division, especially ratios

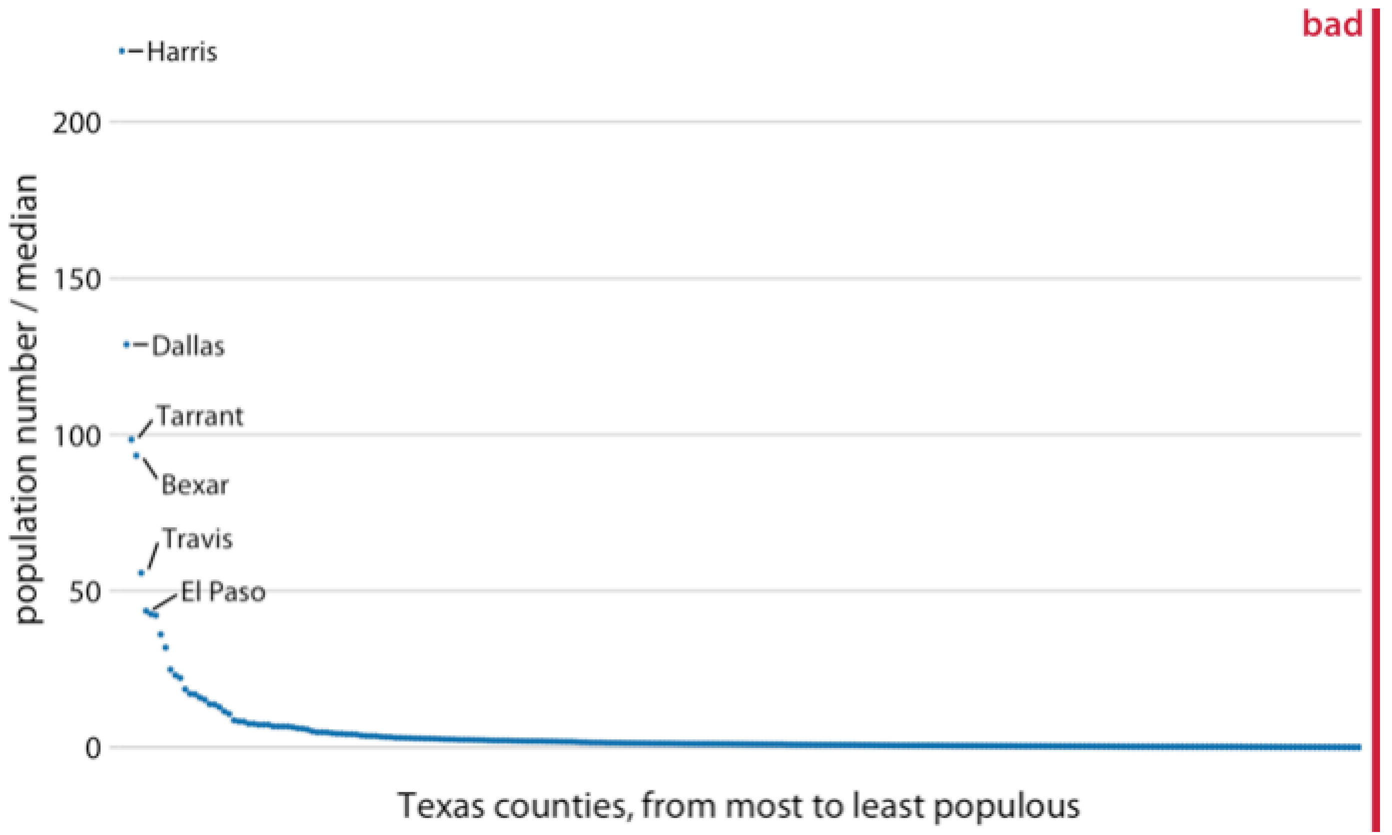


Figure 3.6: Population sizes of Texas counties relative to their median value. By displaying a ratio on a linear scale, we have overemphasized ratios > 1 and have obscured ratios < 1. As a general rule, ratios should not be displayed on a linear scale. Data source: 2010 Decennial U.S. Census.

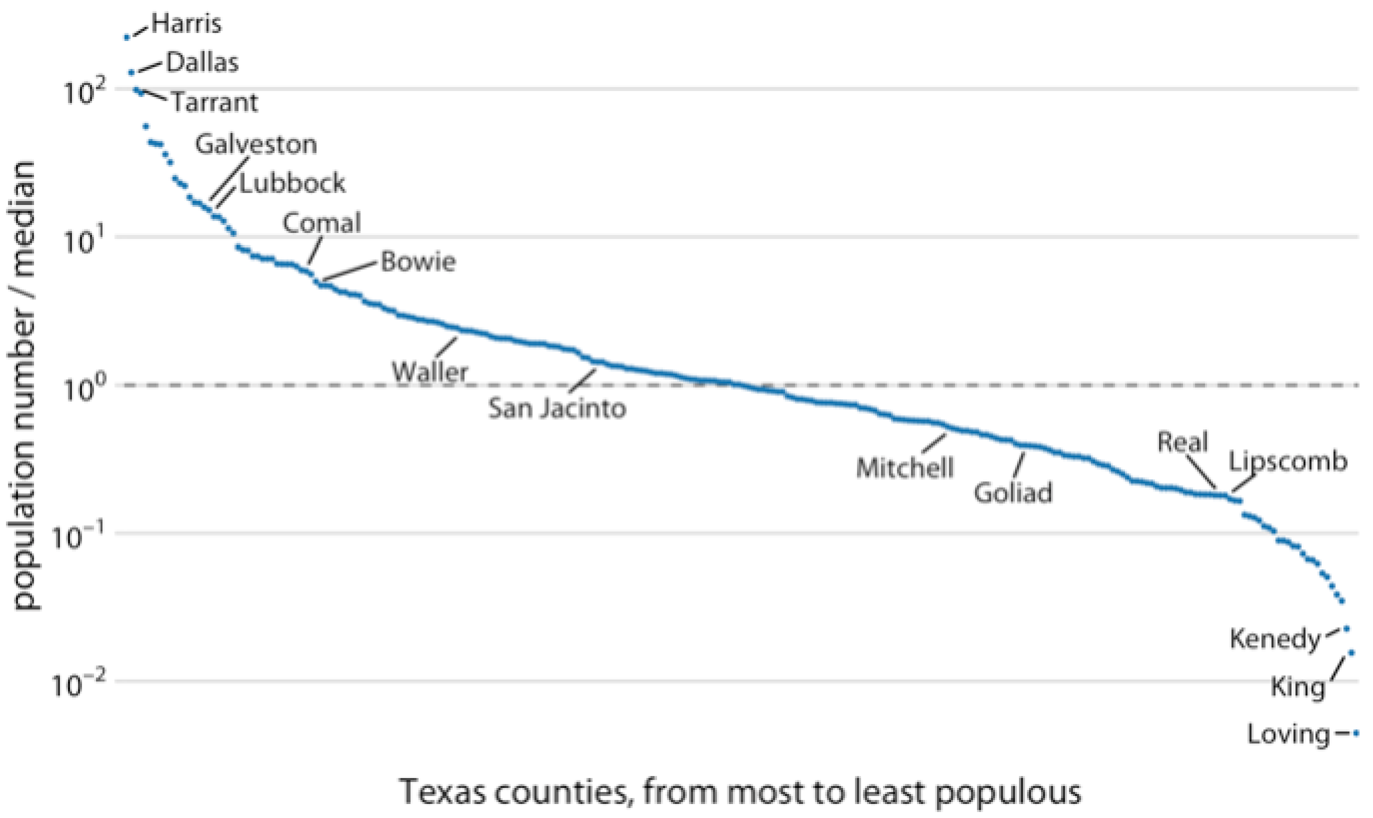


Figure 3.5: Population numbers of Texas counties relative to their median value. Select counties are highlighted by name. The dashed line indicates a ratio of 1, corresponding to a county with median population number. The most populous counties have approximately 100 times more inhabitants than the median county, and the least populous counties have approximately 100 times fewer inhabitants than the median county. Data source: 2010 Decennial U.S. Census.

Wilke Chapter 3 - plotting ratios

- what are some commonly visualised ratios in health data science?

Plotting odds ratios

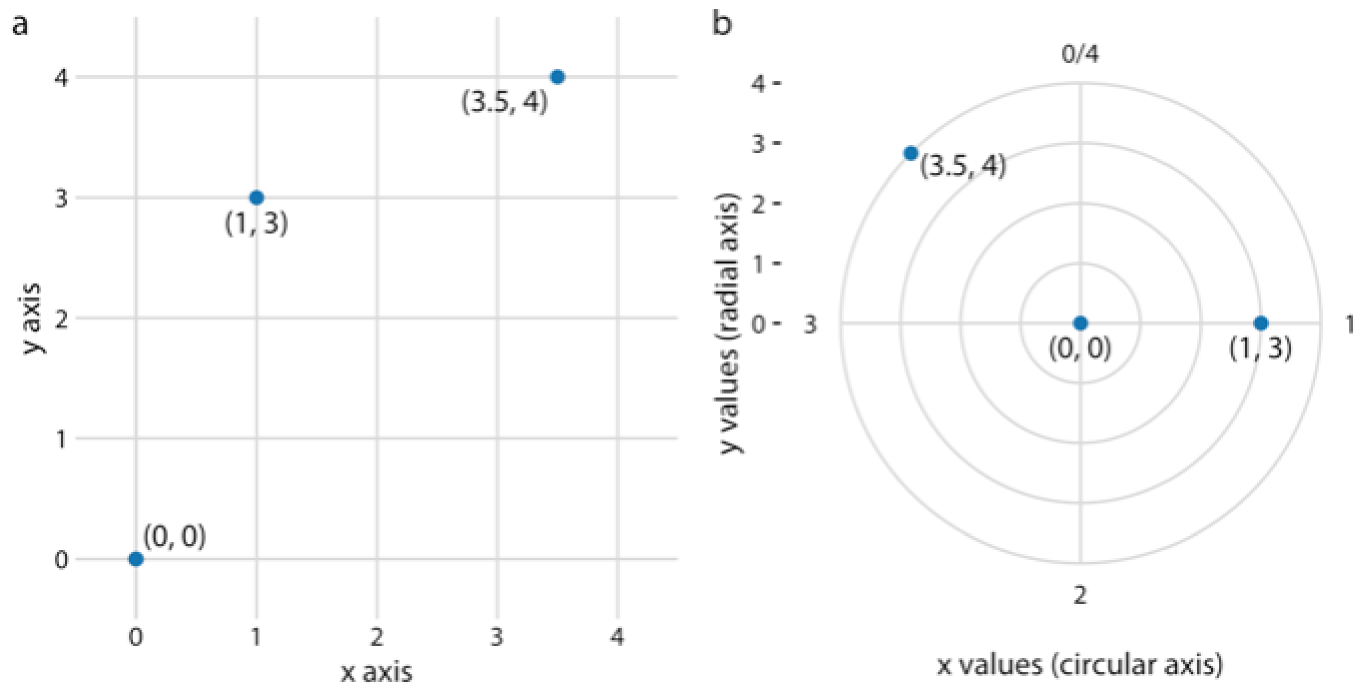
```r
library(finalfit)
explanatory <- c( "differ.factor", "age", "sex.factor",
                 "extent.factor", "obstruct.factor",
                 "nodes")
dependent <- "mort_5yr"
table2 <- colon_s %>%
  finalfit(dependent, explanatory,
           dependent_label_prefix = "")
```

	Mortality 5 year		Alive	Died	OR (univariable)	OR (multivariable)
4	Differentiation	Well	52 (56.5)	40 (43.5)	-	-
2		Moderate	382 (58.7)	269 (41.3)	0.92 (0.59-1.43, p=0.694)	0.62 (0.38-1.01, p=0.054)
3		Poor	63 (42.3)	86 (57.7)	1.77 (1.05-3.01, p=0.032)	1.00 (0.56-1.78, p=0.988)
1	Age (years)	Mean (SD)	59.8 (11.4)	59.9 (12.5)	1.00 (0.99-1.01, p=0.986)	1.01 (1.00-1.02, p=0.098)
12	Sex	Female	243 (55.6)	194 (44.4)	-	-
13		Male	268 (56.1)	210 (43.9)	0.98 (0.76-1.27, p=0.889)	0.97 (0.73-1.30, p=0.858)
8	Extent of spread	Submucosa	16 (80.0)	4 (20.0)	-	-
6		Muscle	78 (75.7)	25 (24.3)	1.28 (0.42-4.79, p=0.681)	1.25 (0.36-5.87, p=0.742)
7		Serosa	401 (53.5)	349 (46.5)	3.48 (1.26-12.24, p=0.027)	3.03 (0.96-13.36, p=0.087)
5		Adjacent structures	16 (38.1)	26 (61.9)	6.50 (1.98-25.93, p=0.004)	6.80 (1.75-34.55, p=0.010)
10	Obstruction	No	408 (56.7)	312 (43.3)	-	-
11		Yes	89 (51.1)	85 (48.9)	1.25 (0.90-1.74, p=0.189)	1.26 (0.88-1.82, p=0.206)
9	nodes	Mean (SD)	2.7 (2.4)	4.9 (4.4)	1.24 (1.18-1.30, p<0.001)	1.24 (1.18-1.31, p<0.001)

```r
colon_s %>%
  or_plot(dependent, explanatory,
          breaks = c(0.5, 1, 5, 10, 20, 30),
          table_text_size = 3.5)
```

Wilke Chapter 3 - curved axes

- *polar* coordinates specify the *x* axis as an angle around a central point and the *y* axis as a radial distance



Wilke Chapter 3 - periodic polar axes

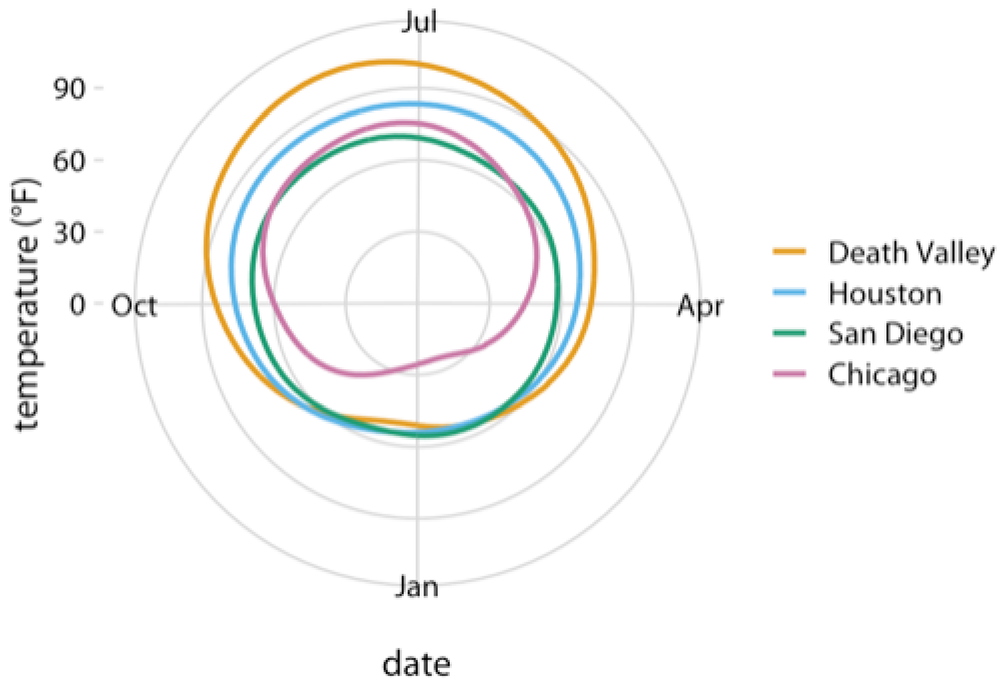


Figure 3.10: Daily temperature normals for four selected locations in the U.S., shown in polar coordinates. The radial distance from the center point indicates the daily temperature in Fahrenheit, and the days of the year are arranged counter-clockwise starting with Jan. 1st at the 6:00 position.

A polar polar plot!

https://www.tylermw.com/polar-ice-data-in-r-with-rayrender/

Wilke Chapter 3 - geographic projections

Wilke Chapter 4 - colour scales

- three main use cases for colour in data visualisations
  - to distinguish groups from each other
  - to represent data values
  - to highlight particular data points

Wilke Chapter 4 - colour to distinguish groups

- qualitative colour scales
    - groups have no intrinsˆc order or ranking
    - colours should look distinct from each other
    - no particular colours should stand out
    - colours should not convey the impression of an order

Wilke Chapter 4 - colour to distinguish groups

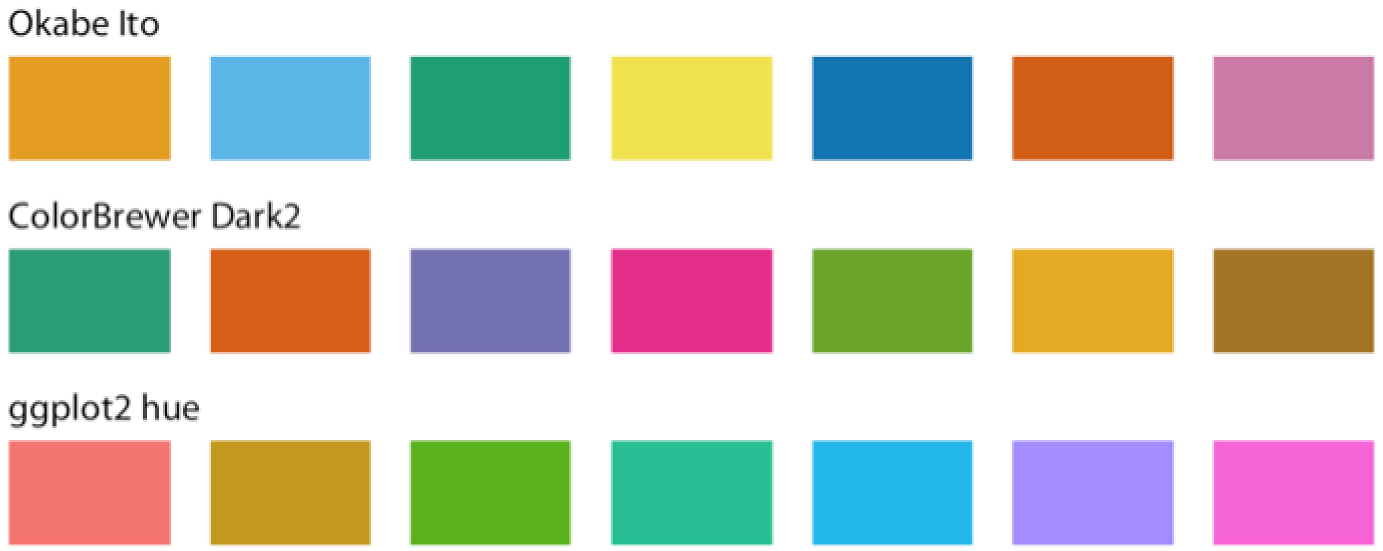
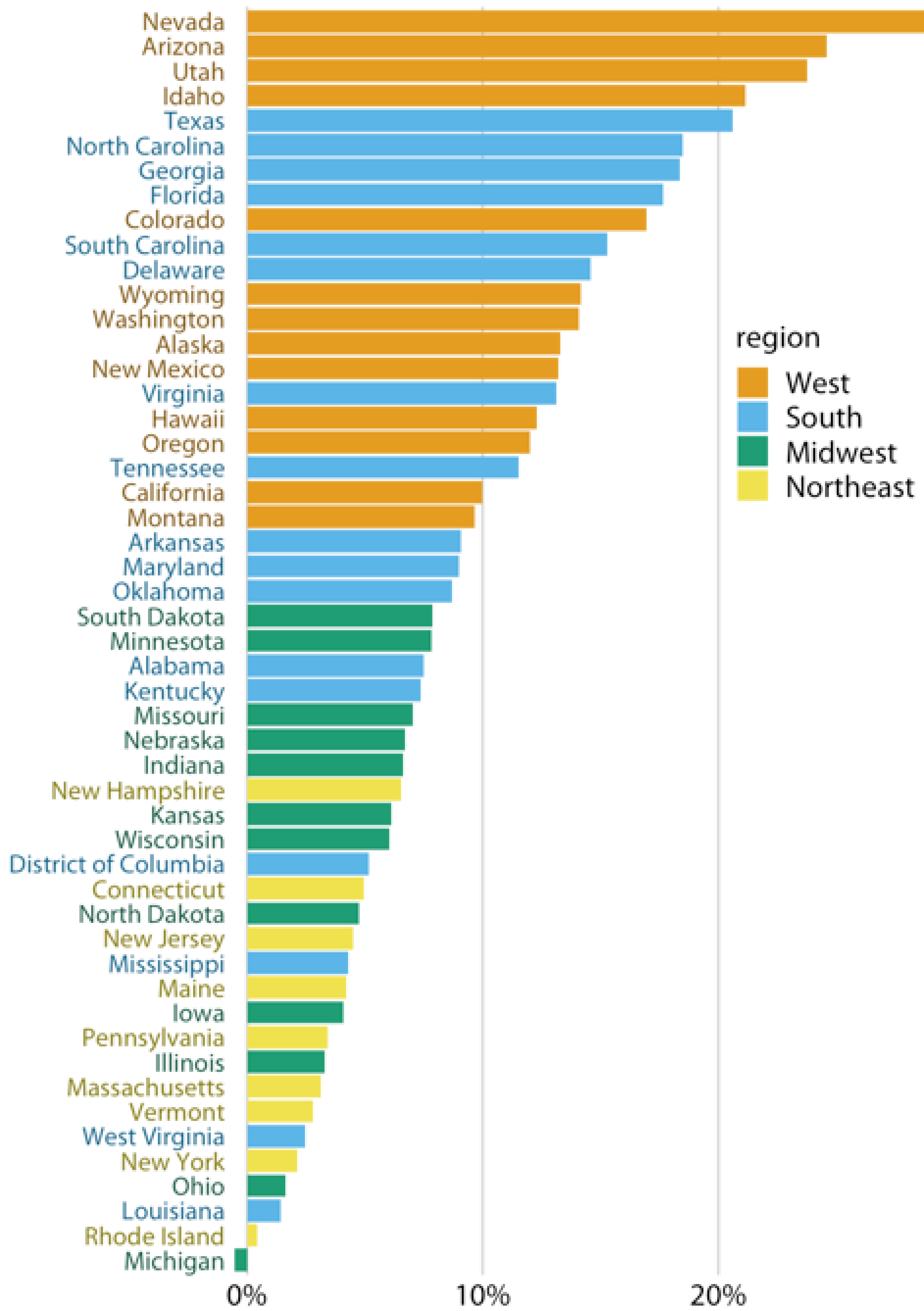


Figure 4.1: Example qualitative color scales. The Okabe Ito scale is the default scale used throughout this book (Okabe and Ito 2008). The ColorBrewer Dark2 scale is provided by the ColorBrewer project (Brewer 2017). The ggplot2 hue scale is the default qualitative scale in the widely used plotting software ggplot2.


Nevada
Arizona
Utah
Idaho
Texas
North Carolina
Georgia
Florida
Colorado
South Carolina
Delaware
Wyoming
Washington
Alaska
New Mexico
Virginia
Hawaii
Oregon
Tennessee
California
Montana
Arkansas
Maryland
Oklahoma
South Dakota
Minnesota
Alabama
Kentucky
Missouri
Nebraska
Indiana
New Hampshire
Kansas
Wisconsin
District of Columbia
Connecticut
North Dakota
New Jersey
Mississippi
Maine
Iowa
Pennsylvania
Illinois
Massachusetts
Vermont
West Virginia
New York
Ohio
Louisiana
Rhode Island
Michigan
0%
10%
20%
30%
region
West
South
Midwest
Northeast

Wilke Chapter 4 - colour to represent data values

- we use *sequential* colours to represent values on a data scale
- *sequential* colours need to clearly indicate which colours represent larger values and which colours represent smaller values
    - ideally these representations need to be perceived uniformly across the colour scale
    - that is difficult to achieve
    - there are individual and cultural differences
- *sequential colours scales* can be based on a single hue (eg light blue to dark blue)
- or on multiple hues (eg dark red to light yellow)
    - multi-hue scales tend to mimic the natural world eg dark red to light yellow
    - dark yellow to light red looks wrong

Wilke Chapter 4 - colour to represent data values

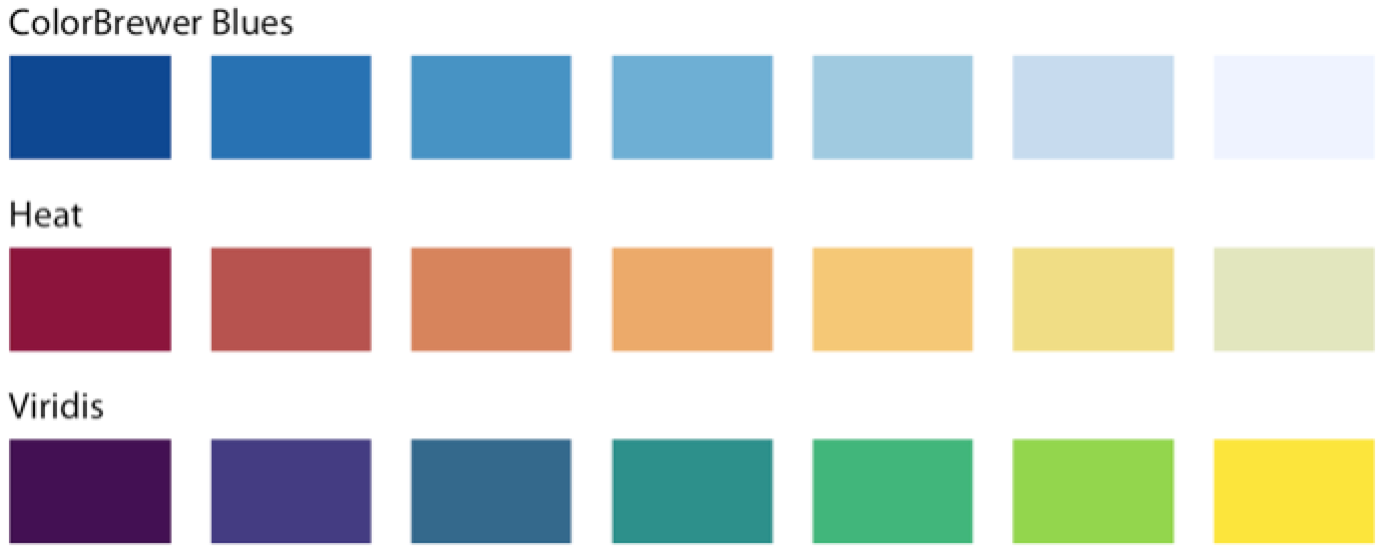


Figure 4.3: Example sequential color scales. The ColorBrewer Blues scale is a monochromatic scale that varies from dark to light blue. The Heat and Viridis scales are multi-hue scales that vary from dark red to light yellow and from dark blue via green to light yellow, respectively.

Wilke Chapter 4 - sequential colour scales on choropleth maps

Wilke Chapter 4 - divergent colour scales

- sometimes we want to show divergence from some central or neutral value in both positive and negative directions
  - an OR (odds ratio) chart is an example, where the neutral point is OR=1.0
- a *divergent* colour scale is used
  - just two *sequential* colour scales stitched together
  - need to be balanced (difficult!)
  - saturation of colour is often used, as well as hue
  - we'll cover HLS and other colour models later

Wilke Chapter 4 - divergent colour scales

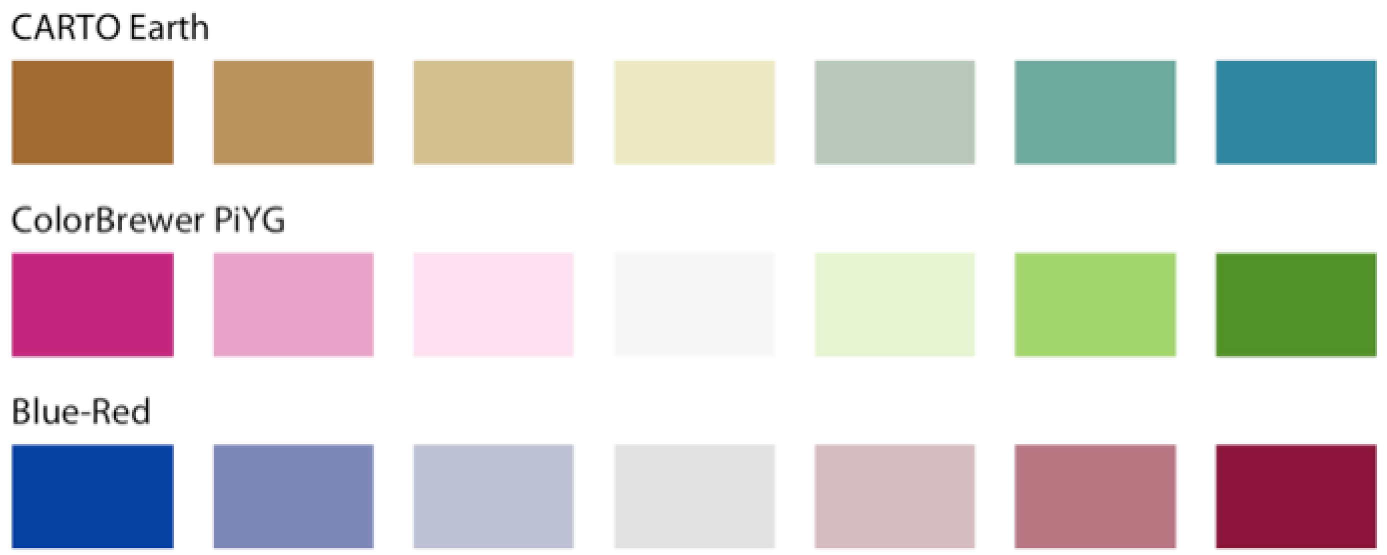


Figure 4.5: Example diverging color scales. Diverging scales can be thought of as two sequential scales stiched together at a common midpoint color. Common color choices for diverging scales include brown to greenish blue, pink to yellow-green, and blue to red.

Wilke Chapter 4 - divergent colour scale on a choropleth map

Wilke Chapter 4 - colour to highlight values

- colour can be used to highlight specific data values or groups of values
    - usually to emphasise some aspect of a story you want to tell
    - beware of personal bias!
- we use an *accent* colour scale for highlighting
    - contain both subdued colours and stronger colours
    - the baseline colours shouldn't compete with the highlight colour

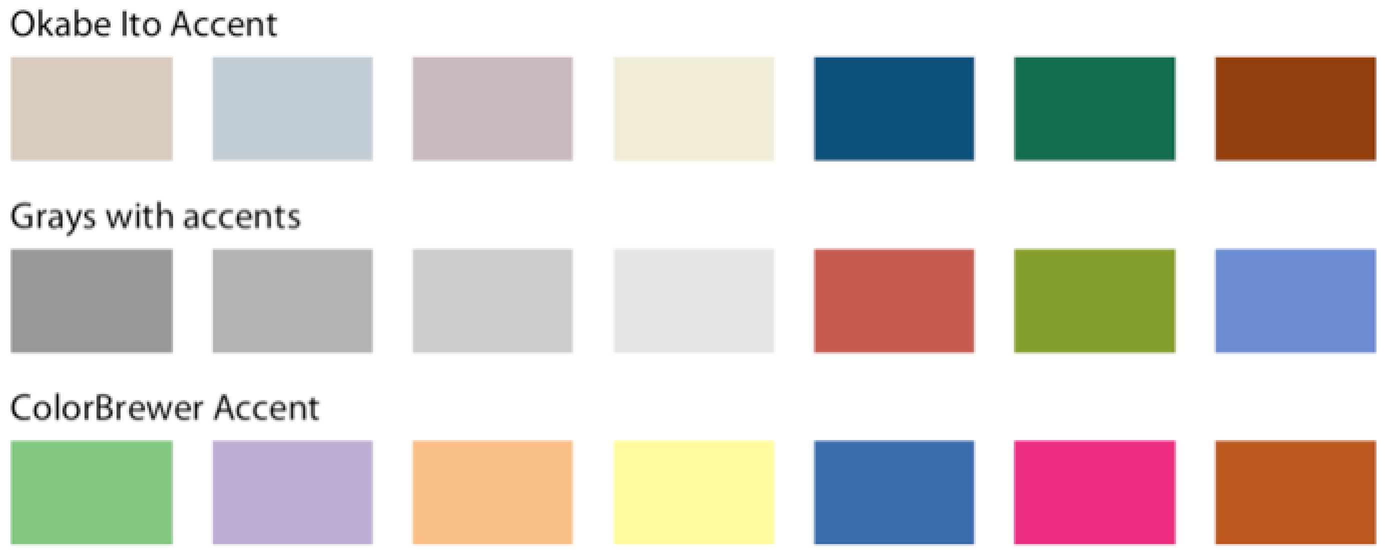
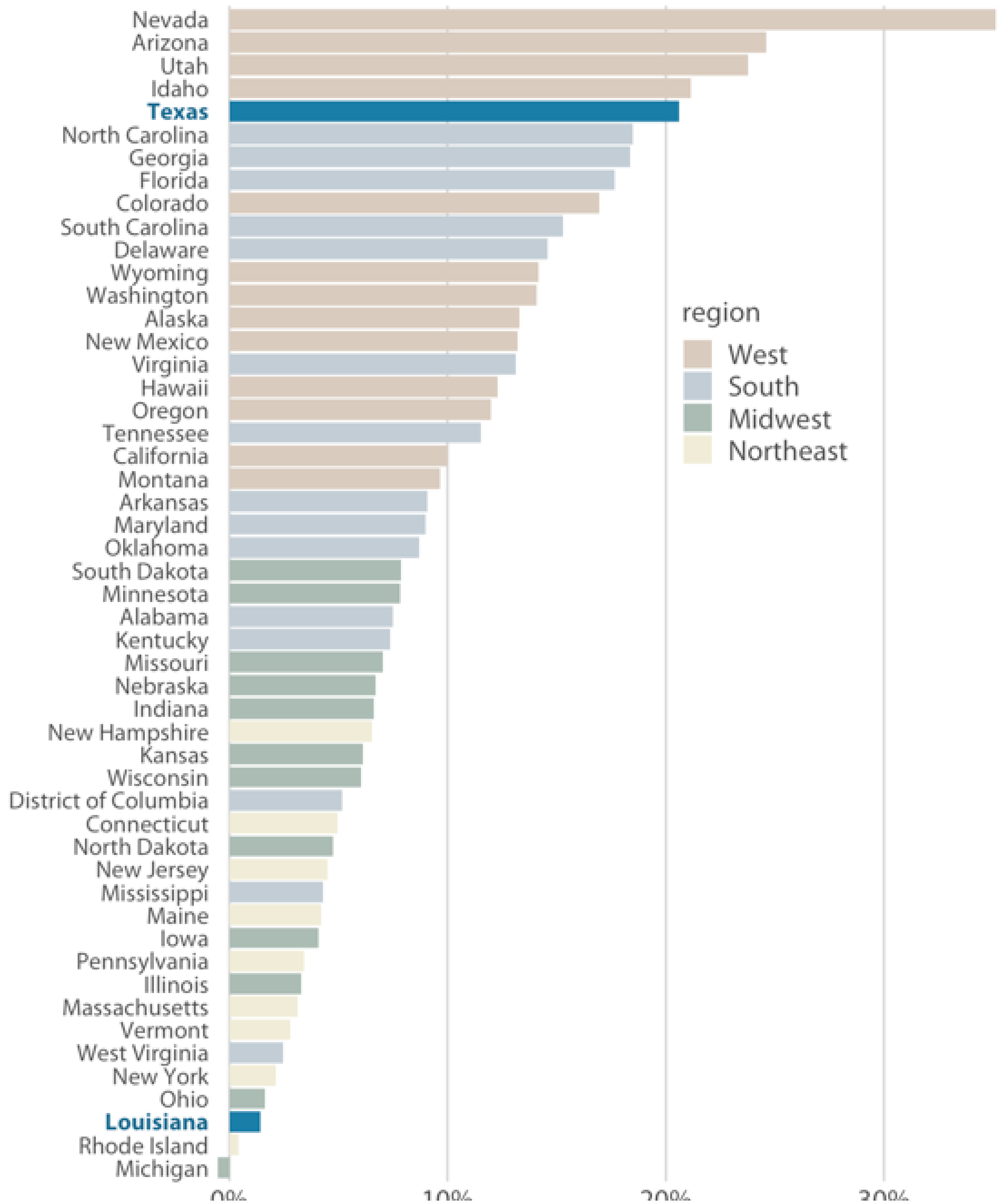


Figure 4.7: Example accent color scales, each with four base colors and three accent colors. Accent color scales can be derived in several different ways: (top) we can take an existing color scale (e.g., the Okabe Ito scale, Fig 4.1) and lighten and/or partially desaturate some colors while darkening others; (middle) we can take gray values and pair them with colors; (bottom) we can use an existing accent color scale, e.g. the one from the ColorBrewer project.


Nevada
Arizona
Utah
Idaho
Texas
North Carolina
Georgia
Florida
Colorado
South Carolina
Delaware
Wyoming
Washington
Alaska
New Mexico
Virginia
Hawaii
Oregon
Tennessee
California
Montana
Arkansas
Maryland
Oklahoma
South Dakota
Minnesota
Alabama
Kentucky
Missouri
Nebraska
Indiana
New Hampshire
Kansas
Wisconsin
District of Columbia
Connecticut
North Dakota
New Jersey
Mississippi
Maine
Iowa
Pennsylvania
Illinois
Massachusetts
Vermont
West Virginia
New York
Ohio
Louisiana
Rhode Island
Michigan
region
West
South
Midwest
Northeast
0%
10%
20%
30%

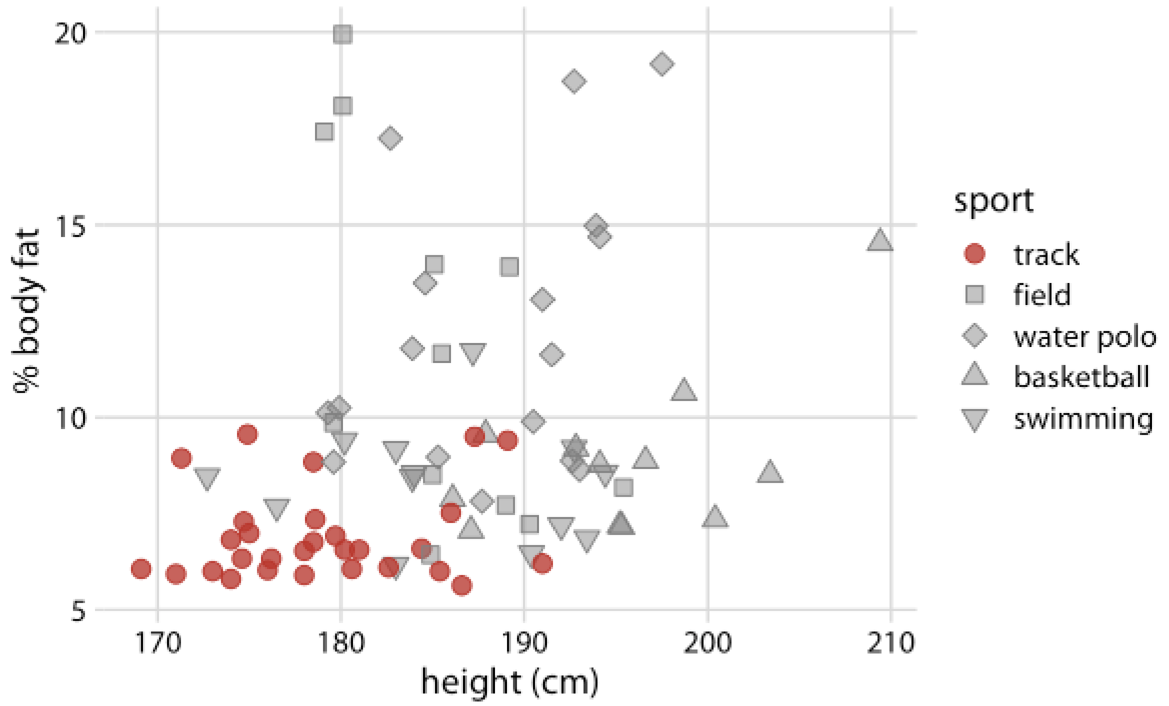


Figure 4.9: Track athletes are among the shortest and leanest of male professional athletes participating in popular sports. Data source: Telford and Cunningham (1991)

Wilke Chapter 5 - directory of visualisations

https://clauswilke.com/dataviz/directory-of-visualizations.html

Hands-on: more on `ggplot2`

- based on Chapter 3 of *R for Data Science* by Hadley Wickham and Garret Grolemund
  - available at https://r4ds.had.co.nz/index.html

Pre-requisites

- install the `ggplot2` package
    - or better, the `tidyverse` meta-package
- install a fork of the `medicaldata` package from GitHub

```
remotes::install_github("cbdrh-hdat9800/medicaldata")
```

```
## Skipping install of 'medicaldata' from a github remote, the SHA1 (be27ec88) has not changed since
##   Use `force = TRUE` to force installation
```

- load the libraries

```
library(tidyverse)
library(medicaldata)
```

the `diabetes` dataset

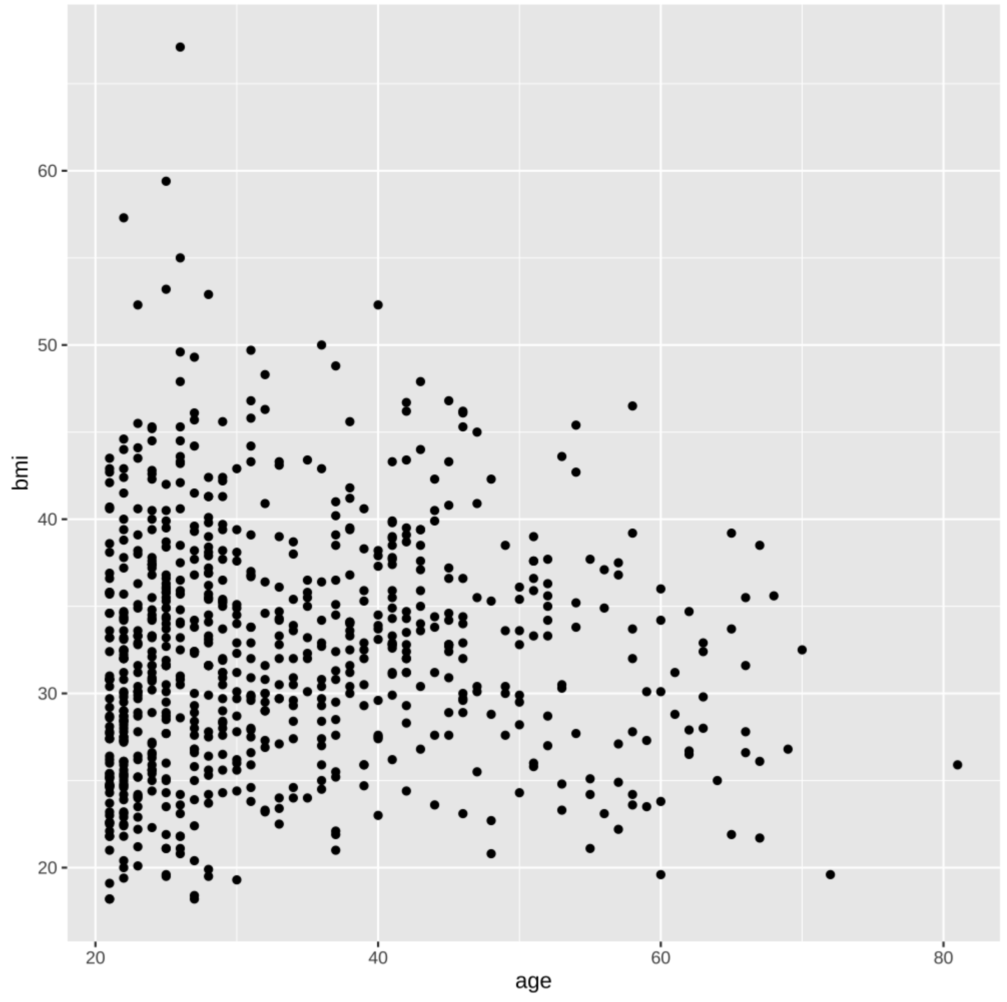
- at the R Console, type `?diabetes`
- examine the data with `head(dibetes)`
    - or `glimpse(diabetes)`

Create a ggplot

- using the `diabetes` dataset, write code to create a scatter plot of `age` versus `bmi`

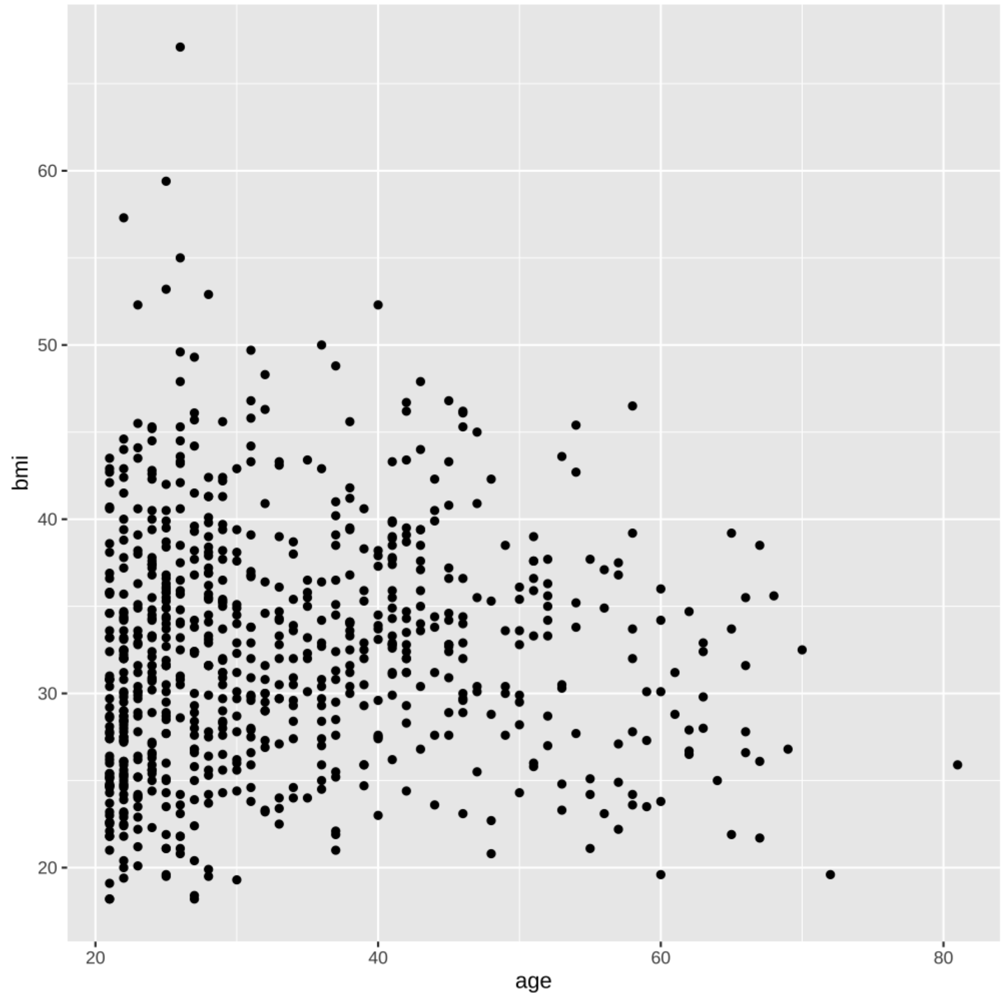
```r
ggplot(data = diabetes, aes(x = age, y = bmi)) +
  geom_point()
```

```
## Warning: Removed 11 rows containing missing values (geom_point).
```



```
ggplot(data = diabetes) +
  geom_point(mapping = aes(x = age, y = bmi))
```

```
## Warning: Removed 11 rows containing missing values (geom_point).
```



A graphing template

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Exercise 1

- make a scatter plot of `bmi` versus `glucose_mg-dl`

```
ggplot(data = diabetes, aes(x = bmi, y = glucose_mg-dl)) +
  geom_point()
```

We need to rename that column!

```
diab <- diabetes %>%
          rename(glucose_mg-dl = glucose_mg_dl)
```

A better way

```
library(janitor)
```

```
##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

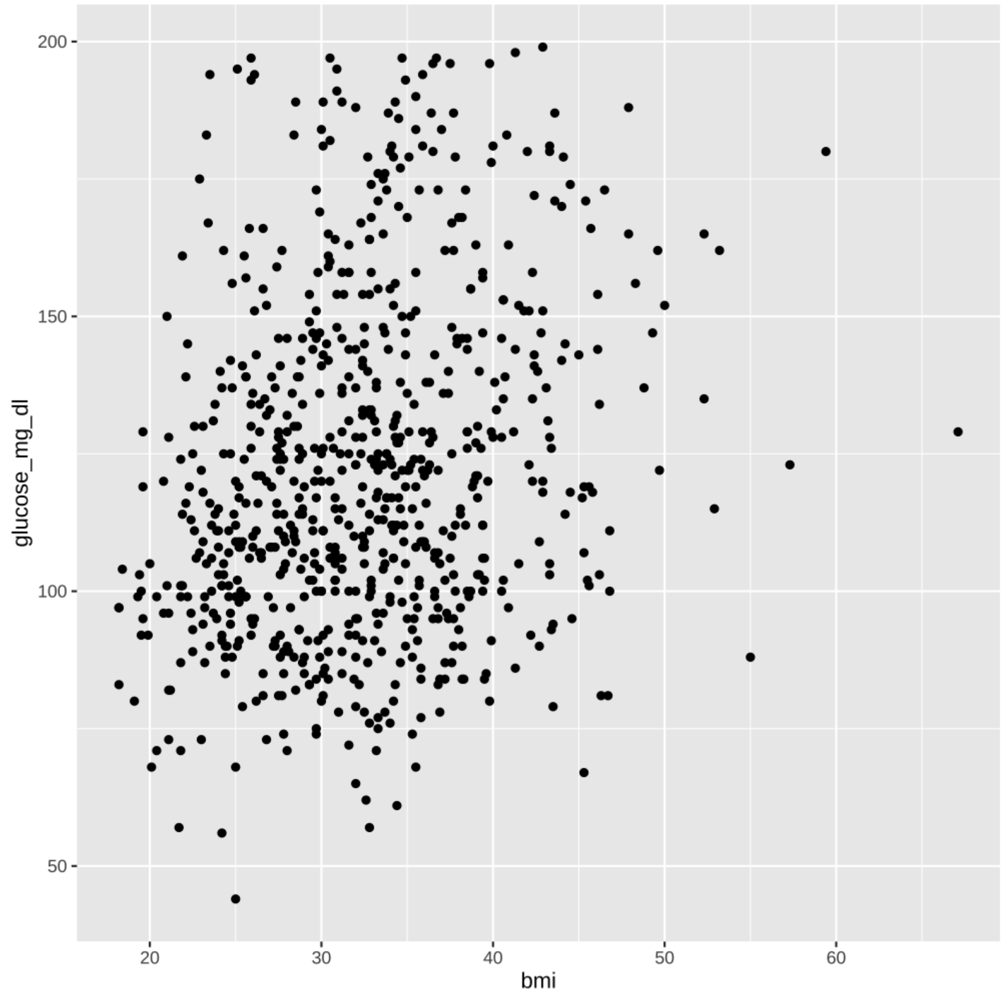
```
diab <- clean_names(diabetes)

glimpse(diab)
```

```
## Rows: 768
## Columns: 9
## $ pregnancy_num      <dbl> 6, 1, 8, 1, 0, 5, 3, 10, 2, 8, 4, 10, 10, 1, 5, 7, …
## $ glucose_mg_dl      <dbl> 148, 85, 183, 89, 137, 116, 78, 115, 197, 125, 110,…
## $ dbp_mm_hg          <dbl> 72, 66, 64, 66, 40, 74, 50, NA, 70, 96, 92, 74, 80,…
## $ triceps_mm         <dbl> 35, 29, NA, 23, 35, NA, 32, NA, 45, NA, NA, NA, NA,…
## $ insulin_microiu_ml <dbl> NA, NA, NA, 94, 168, NA, 88, NA, 543, NA, NA, NA, N…
```

```r
ggplot(data = diab, aes(x = bmi, y = glucose_mg_dl)) +
  geom_point()
```

```
## Warning: Removed 16 rows containing missing values (geom_point).
```

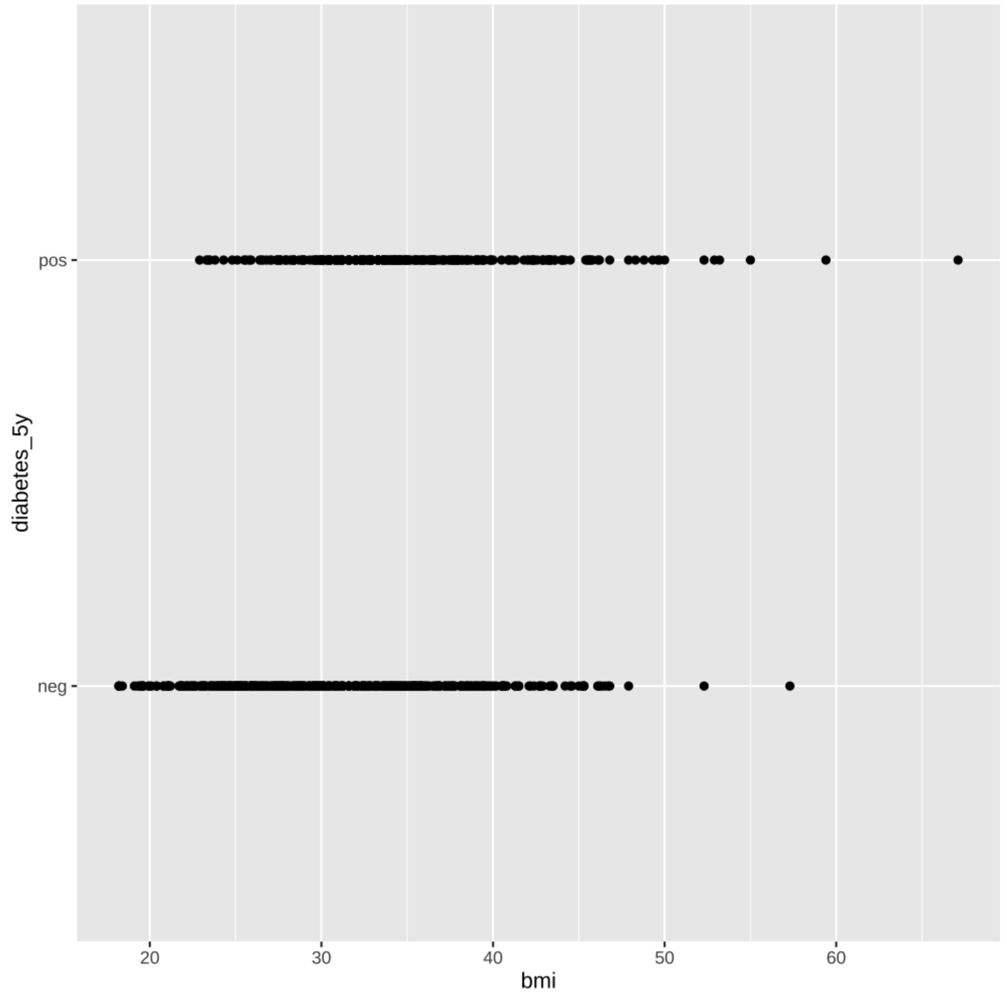


Exercise 2

- make a scatter plot of `bmi` versus `diabetes_5y`
- why is it not very useful?

```r
ggplot(data = diab) +
  geom_point(mapping = aes(x = bmi, y = diabetes_5y))
```

```
## Warning: Removed 11 rows containing missing values (geom_point).
```

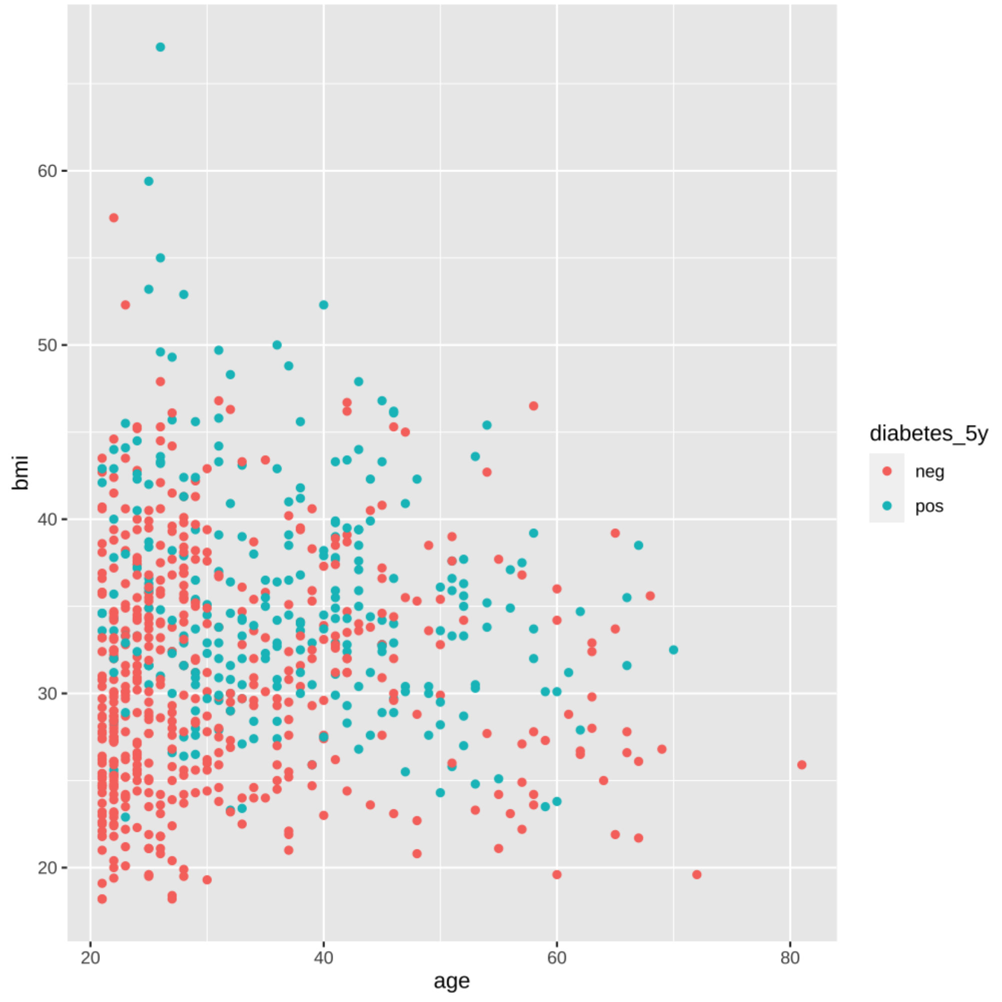


Add further aesthetic mappings

- start with our earlier plot of `age` vs `bmi`
- add a additional aesthetic mapping of `diabetes_5y` to the `colour` aesthetic

```
ggplot(data = diab) +
  geom_point(mapping = aes(x = age, y = bmi, colour = diabetes_5y))
```

```
## Warning: Removed 11 rows containing missing values (geom_point).
```



Experiment 3

- what happens if we map `diabetes_5y` to the `size` aesthetic instead of the `colour` aesthetic?

```r
ggplot(data = diab) +
  geom_point(mapping = aes(x = age, y = bmi, size = diabetes_5y))
```

```
## Warning: Using size for a discrete variable is not advised.

## Warning: Removed 11 rows containing missing values (geom_point).
```

Warning: Using size for a discrete variable is not advised.

- repeat that but mapping `diabetes_5y` to the `alpha` aesthetic and then to the `shape` aesthetic

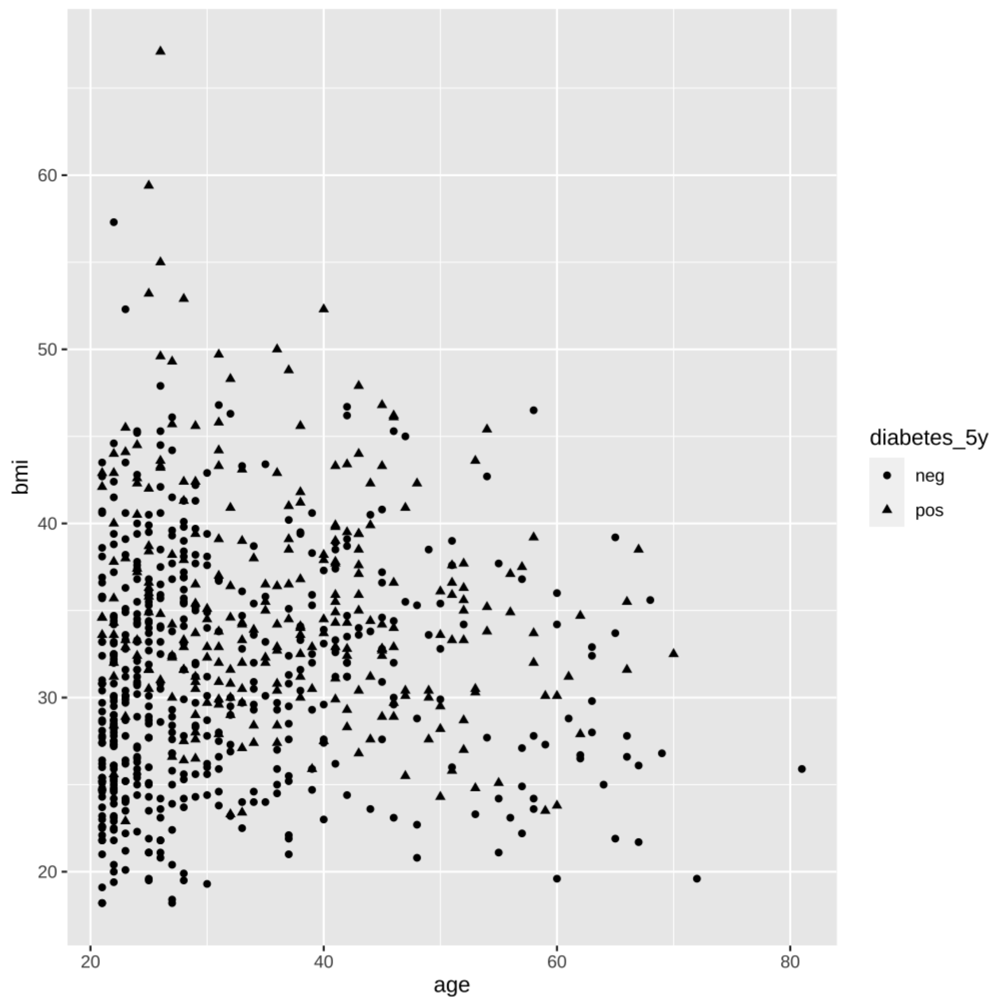
```r
ggplot(data = diab) +
  geom_point(mapping = aes(x = age, y = bmi, alpha = diabetes_5y))
```

```
## Warning: Using alpha for a discrete variable is not advised.

## Warning: Removed 11 rows containing missing values (geom_point).
```

```
ggplot(data = diab) +
  geom_point(mapping = aes(x = age, y = bmi, shape = diabetes_5y))
```

```
## Warning: Removed 11 rows containing missing values (geom_point).
```

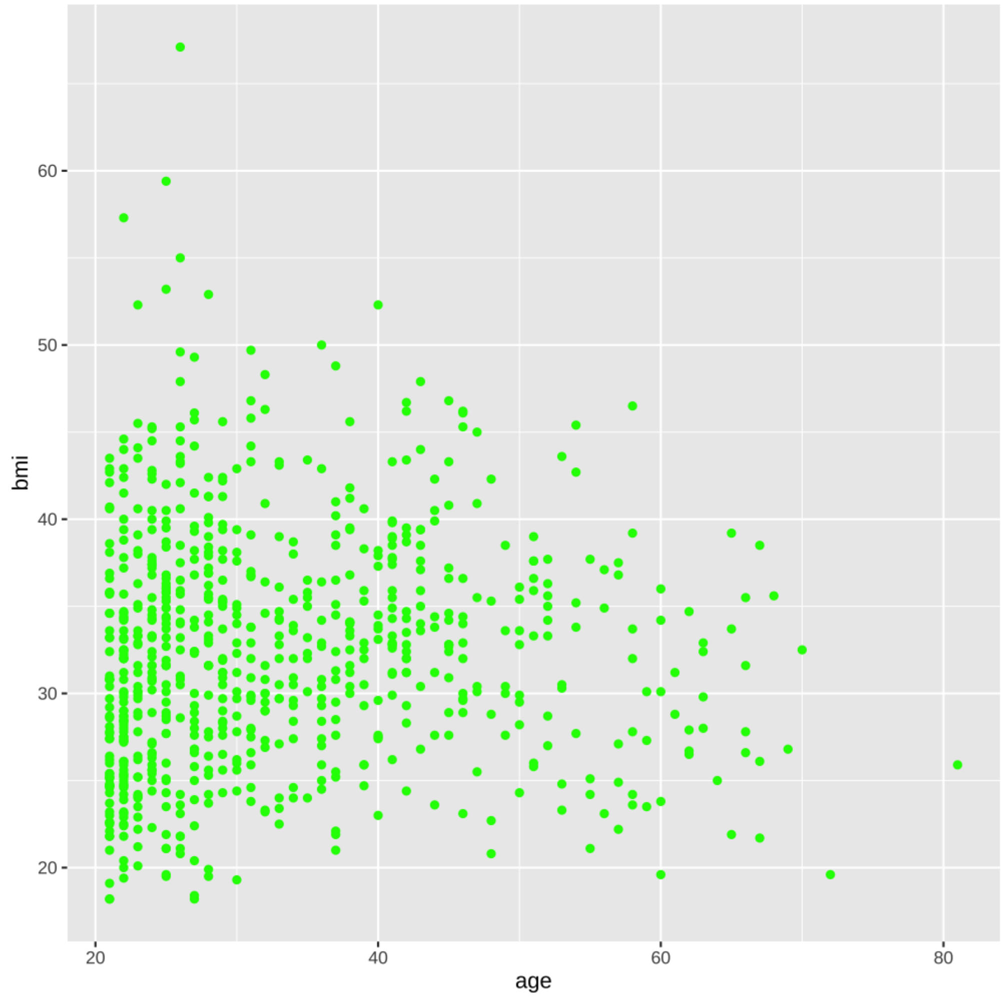


Manually setting an aesthetic

- how would you make the colour of all the points green?

```r
ggplot(data = diab) +
  geom_point(mapping = aes(x = age, y = bmi), colour = "green")
```

```
## Warning: Removed 11 rows containing missing values (geom_point).
```



Why does this happen?

```r
ggplot(data = diab) +
  geom_point(mapping = aes(x = age, y = bmi, colour = "green"))
```

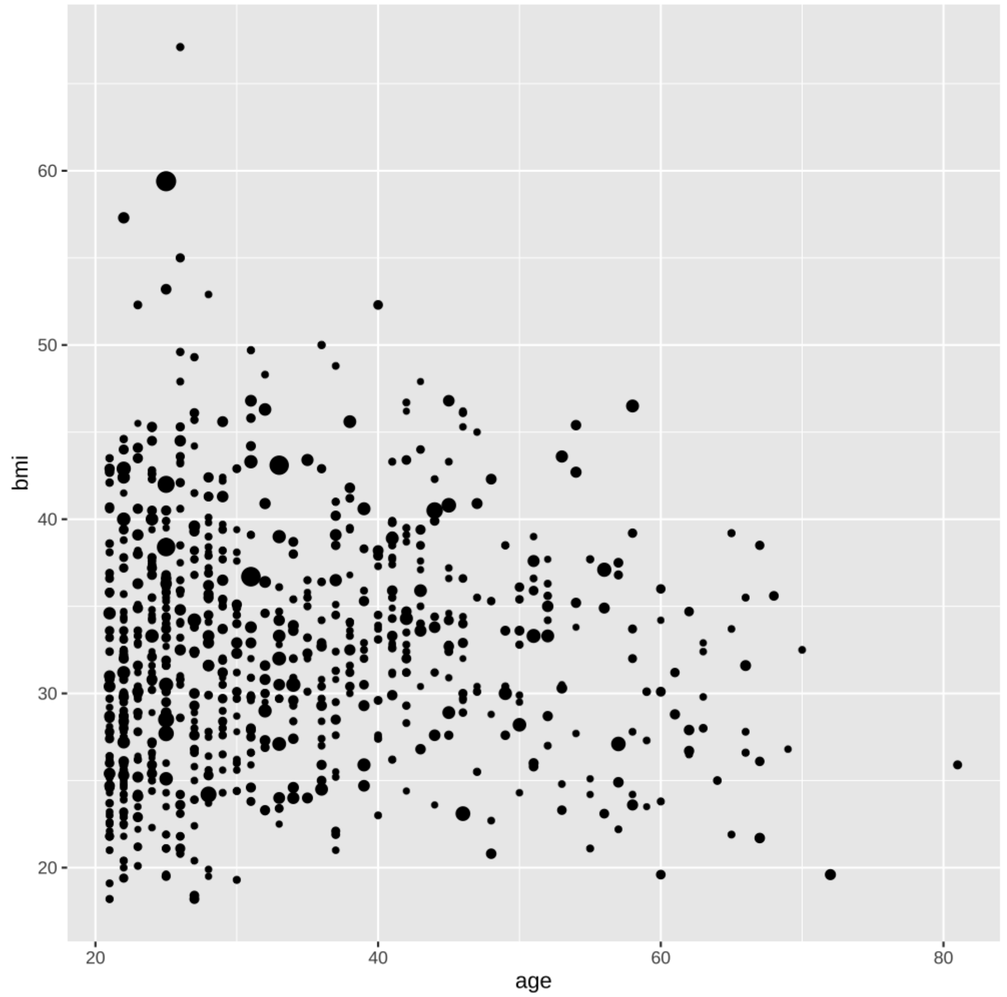
```
## Warning: Removed 11 rows containing missing values (geom_point).
```

Experiment 4

- write some code to see what the `stroke` aesthetic does, using `age` for the x-axis, `bmi` for the y-axis and `pedigree` for the `stroke` aesthetic mapping

```
ggplot(data = diab) +
  geom_point(mapping = aes(x = age, y = bmi, stroke = pedigree))
```

```
## Warning: Removed 11 rows containing missing values (geom_point).
```



Facets

- an alternative to mapping additional variables to aesthetics
- create lots of smaller plots broken down (conditioned on) additional variables instead
- Tufte's "small multiples"

Add a one-dimensional facetting

- write some code to add faceting by the `diabetes_5y` variable to a scatter plot of `age` for the x-axis and `bmi` for the y-axis
- hint: look at `facet_wrap()` in `ggplot2`
  - use `?facet_wrap` at the Console prompt of visit the ggplot2 documentation web site

```r
ggplot(data = diab) +
  geom_point(mapping = aes(x = age, y = bmi)) +
  facet_wrap(~ diabetes_5y)
```

```
## Warning: Removed 11 rows containing missing values (geom_point).
```

Adjust the number of facetting rows (or columns)

- add `nrow = 2` as an additional argument to the `facet_wrap()` function

```r
ggplot(data = diab) +
  geom_point(mapping = aes(x = age, y = bmi)) +
  facet_wrap(~ diabetes_5y, nrow = 2)
```

```
## Warning: Removed 11 rows containing missing values (geom_point).
```

Facet by two variables

- facet the plot by both `diabetes_5y` and `pregnancy_num`
  - hint: see `?facet_grid` at the Console prompt or on the `ggplot2` web site

```
ggplot(data = diab) +
  geom_point(mapping = aes(x = age, y = bmi)) +
  facet_grid(pregnancy_num ~ diabetes_5y)
```

```
## Warning: Removed 11 rows containing missing values (geom_point).
```

Too many facets!

- we need to reduce the number of categories for `pregnancy_num`
- many ways to do that, but here's one way
    - make a new column called `preg_group`

```r
diab2 <- diab %>%
          mutate(preg_group = case_when(pregnancy_num == 0 ~ "None",
                                        pregnancy_num >= 1 & pregnancy_num <= 3 ~ "1 to 3",
                                        pregnancy_num >= 4 & pregnancy_num <= 7 ~ "4 to 7",
                                        pregnancy_num >= 8 ~ "8 or more",
                                        TRUE ~ "other"))
```

```r
ggplot(data = diab2) +
  geom_point(mapping = aes(x = age, y = bmi)) +
  facet_grid(preg_group ~ diabetes_5y)
```

```
## Warning: Removed 11 rows containing missing values (geom_point).
```

How can we fix the ordering of the `preg_group` facet?

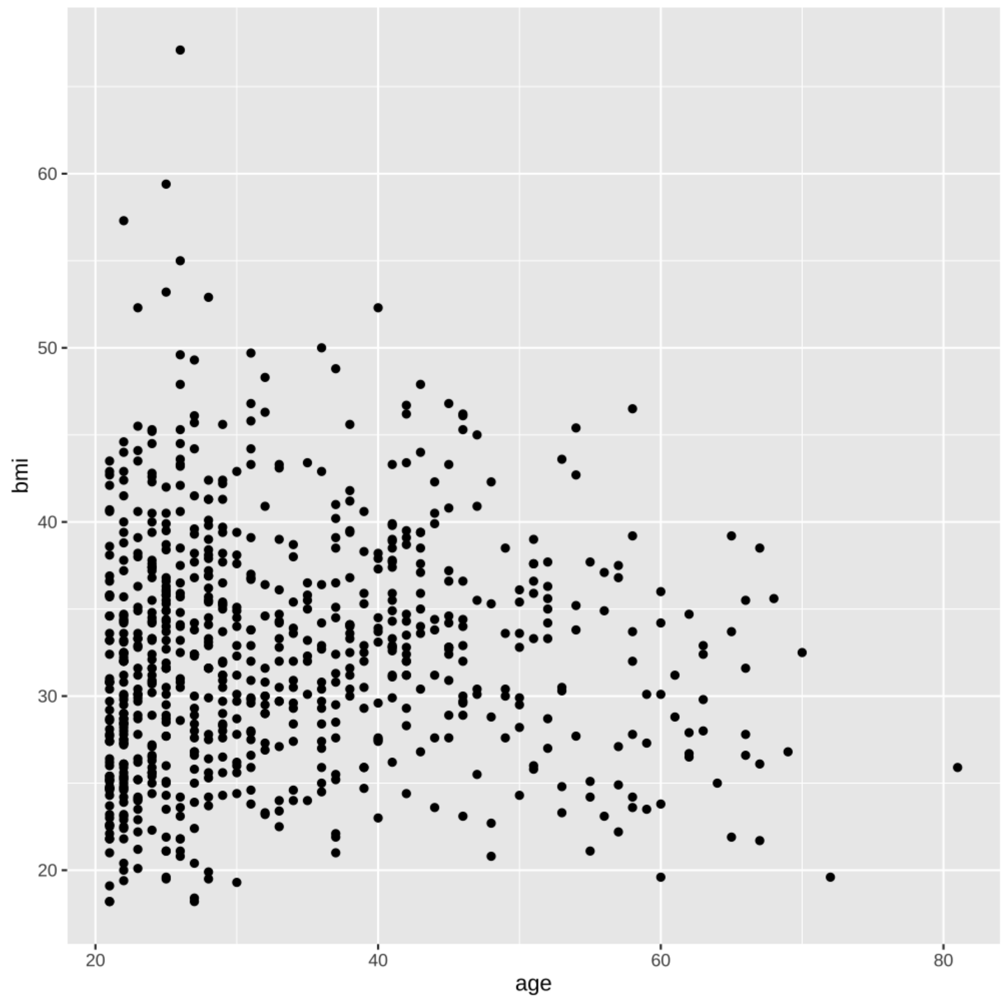
```
diab3 <- diab2 %>%
          mutate(preg_group = ordered(preg_group,
                                      levels = c("None", "1 to 3", "4 to 7", "8 or more")))

ggplot(data = diab3) +
  geom_point(mapping = aes(x = age, y = bmi)) +
  facet_grid(preg_group ~ diabetes_5y)
```

```
## Warning: Removed 11 rows containing missing values (geom_point).
```

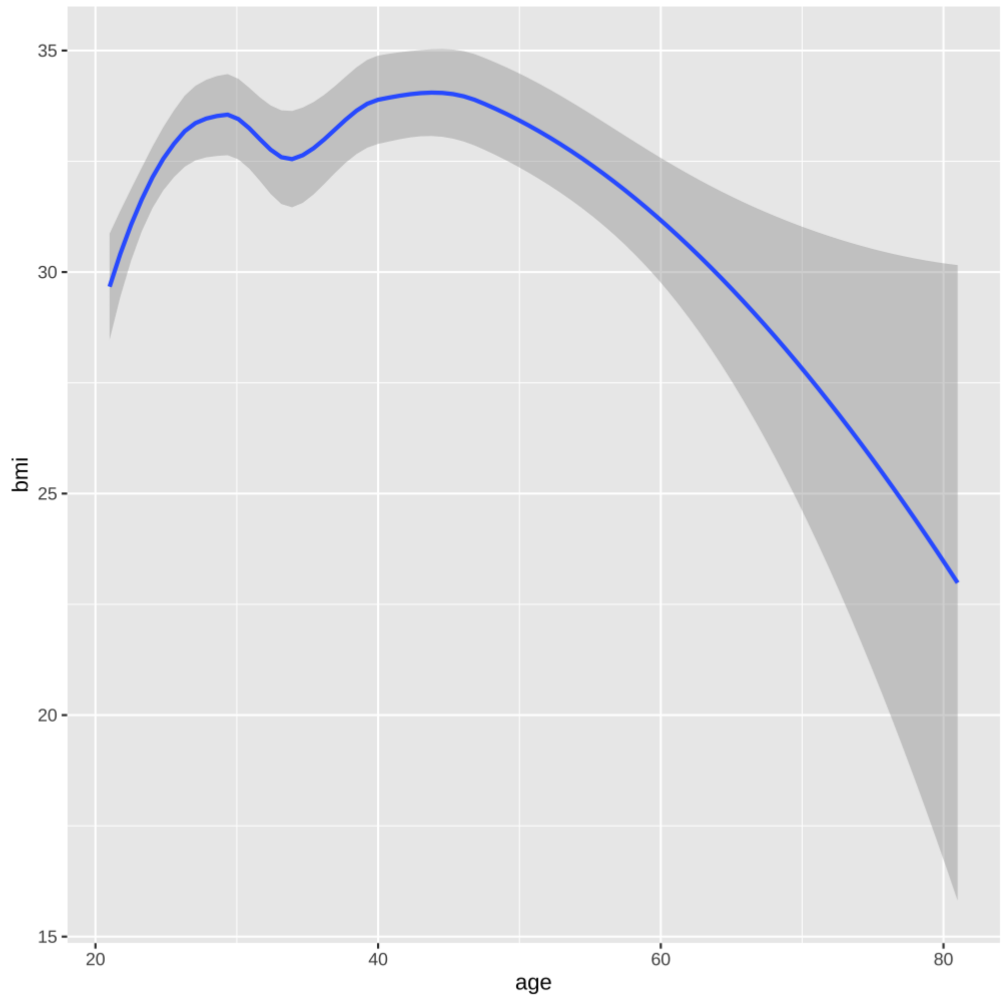
Exploring geometric objects (geoms)

- how are these two plots similar?



bmi
60
50
40
30
20
20
40
60
80
age

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Exploring geometric objects (geoms)

- how are these two plot similar?

```
ggplot(data = diab3) +
  geom_point(mapping = aes(x = age, y = bmi))

ggplot(data = diab3) +
  geom_smooth(mapping = aes(x = age, y = bmi))
```

Add an additional aesthetic mapping to `geom_smooth()`

- map `diabetes_5y` to the `linetype` aesthetic

```r
ggplot(data = diab3) +
  geom_smooth(mapping = aes(x = age, y = bmi, linetype = diabetes_5y))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

## Warning: Removed 11 rows containing non-finite values (stat_smooth).
```

More next week!

- we'll have a regular `ggplot2` hands-on session almost every week from now on
    - by the end of the course you'll be wizards!