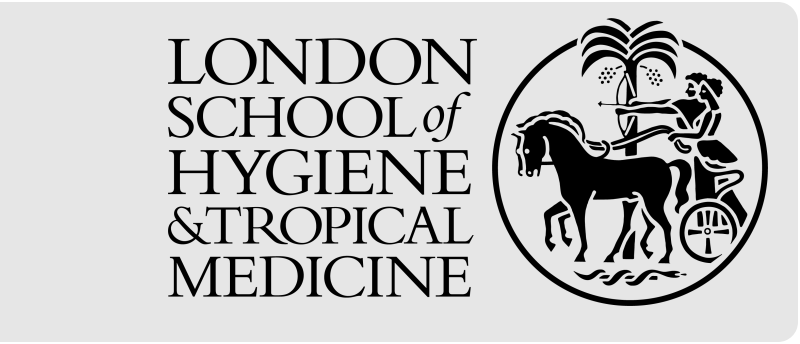


Exploratory Data Analysis

2491 - Data Challenge

Sam Clifford

Session 3 2022-01-13

Introduction

Tidyverse

- Suite of R packages to make working with data as easy as possible (Wickham 2020), including
  - ggplot2: for plotting data
  - dplyr: for manipulating data frames
  - tidyr: for making data tidy
  - forcats: for manipulating factor variables
  - *magrittr: for easy chaining of commands*

(Wickham and Grolemund 2017; Wickham et al. 2019)

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Summarising data

Summarising data

- Summary statistics one of most common data analysis tasks
- Consider the `Gestation` data from `mosaicData`

Birth weight, date, and gestational period collected as part of the Child Health and Development Studies in 1961 and 1962. Information about the baby's parents — age, education, height, weight, and whether the mother smoked is also recorded (Nolan and Speed 2001).

- We will use some functions from dplyr to choose, group and summarise data
- `verb(.data, ...)` applies a dplyr `verb` to a data frame

Summarising data

- `count` how many babies in data set

```
library(mosaicData)
library(tidyverse)
data(Gestation)

count(Gestation) # just like nrow()
## # A tibble: 1 × 1
##       n
##   <int>
## 1  1236
```

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Summarising data

- Can also `count` for a given grouping variable

```
count(Gestation, race) # nrow can't do this
## # A tibble: 6 × 2
##   race      n
##   <chr> <int>
## 1 asian    44
## 2 black   244
## 3 mex      40
## 4 mixed    25
## 5 white   870
## 6 <NA>     13
```

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Summarising data

- The `summarise` function allows us to calculate summary statistics of a variable
- Can (and should) give names to summary columns
- Calculate the mean birth weight in the data set

```
summarise(Gestation, wt_mean = mean(wt))
## # A tibble: 1 × 1
##   wt_mean
##     <dbl>
## 1    120.
```

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Summarising data

- We can calculate multiple summaries at once

```
summarise(Gestation,
          Mean = mean(wt),
          SD   = sd(wt),
          Low  = quantile(wt, 0.025),
          High = quantile(wt, 0.975))
## # A tibble: 1 × 4
##    Mean    SD   Low  High
##   <dbl> <dbl> <dbl> <dbl>
## 1  120.  18.2    81   155
```

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Summarising grouped data

- We can group the rows in our data and calculate summaries for each group
- `group_by` lets us pass variable names to set the structure
- Row order is maintained

```
Gestation_grouped_by_race <- group_by(Gestation, race)

Gestation_grouped_by_race
## # A tibble: 1,236 × 23
## # Groups:   race [6]
##       id pluralty   outcome  date       gestation sex      wt parity race    age
##    <dbl> <chr>      <chr>    <date>         <dbl> <chr> <dbl>  <dbl> <chr> <dbl>
##  1    15 single fe… live bi… 1964-11-11       284 male    120      1 asian    27
##  2    20 single fe… live bi… 1965-02-07       282 male    113      2 white    33
##  3    58 single fe… live bi… 1965-04-25       279 male    128      1 white    28
##  4    61 single fe… live bi… 1965-02-12        NA male    123      2 white    36
##  5    72 single fe… live bi… 1964-11-25       282 male    108      1 white    23
##  6   100 single fe… live bi… 1965-07-31       286 male    136      4 white    25
##  7   102 single fe… live bi… 1964-12-19       244 male    138      4 black    33
##  8   129 single fe… live bi… 1965-04-11       245 male    132      2 black    23
##  9   142 single fe… live bi… 1964-11-08       289 male    120      3 white    25
## 10   148 single fe… live bi… 1965-04-17       299 male    143      3 white    30
## # … with 1,226 more rows, and 13 more variables: ed <chr>, ht <dbl>,
## #   wt.1 <dbl>, drace <chr>, dage <dbl>, ded <chr>, dht <dbl>, dwt <dbl>,
## #   marital <chr>, inc <chr>, smoke <chr>, time <chr>, number <chr>
```

Summarising grouped data

- `summarise()` respects the grouping structure

```
summarise(Gestation_grouped_by_race,
          Mean = mean(wt),
          SD   = sd(wt),
          Low  = quantile(wt, 0.025),
          High = quantile(wt, 0.975))
## # A tibble: 6 × 5
##   race   Mean    SD   Low  High
##   <chr> <dbl> <dbl> <dbl> <dbl>
## 1 asian  110.  16.0  78.4  139.
## 2 black  113.  19.1  71    150
## 3 mex    124.  14.1  99.0  146.
## 4 mixed  120.  20.1  78.8  150.
## 5 white  122.  17.7  85    158
## 6 <NA>   117.  16.7  86.8  143.
```

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Summarising data

- We can summarise multiple variables (`.vars`) with the same function(s) (`.funs`)

```
summarise_at(Gestation,
             .vars = vars(gestation, wt, age),
             .funs = mean, na.rm = T)
## # A tibble: 1 × 3
##   gestation    wt   age
##       <dbl> <dbl> <dbl>
## 1      279.  120.  27.3
```

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Summarising data

- We can even do multiple summaries across multiple variables by specifying
  - which `.vars` we want to summarise, and
  - a list of which `.funs` we wish to summarise with

```
summarise_at(Gestation,
             .vars = vars(gestation, wt, age),
             .funs = list(mean = mean,
                          sd   = sd), na.rm = T)
## # A tibble: 1 × 6
##   gestation_mean wt_mean age_mean gestation_sd wt_sd age_sd
##            <dbl>   <dbl>    <dbl>        <dbl> <dbl>  <dbl>
## 1           279.    120.     27.3         16.0  18.2   5.78
```

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Summarising grouped data

- All `summarise*()` respects the grouping structure

```
summarise_at(Gestation_grouped_by_race,
             .vars = vars(gestation, wt, age),
             .funs = list(mean = mean), na.rm = T)
## # A tibble: 6 × 4
##   race  gestation_mean wt_mean age_mean
##   <chr>          <dbl>   <dbl>    <dbl>
## 1 asian           274.    110.     29.4
## 2 black           274.    113.     27.6
## 3 mex             278.    124.     26.5
## 4 mixed           279.    120.     26.4
## 5 white           281.    122.     27.1
## 6 <NA>            280     117.     30.7
```

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Rows and columns

Operating on columns

- create/modify/delete columns with dplyr's `mutate()`
- e.g. relabelling `race` so words start with a capital,

```
Gestation <- mutate(Gestation, race = str_to_title(race))

count(Gestation, race)
## # A tibble: 6 × 2
##   race      n
##   <chr> <int>
## 1 Asian    44
## 2 Black   244
## 3 Mex      40
## 4 Mixed    25
## 5 White   870
## 6 <NA>     13
```

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Choosing columns

- For one reason or another we may want to `select` only certain columns of our data frame

```
head(Gestation, 1)
## # A tibble: 1 × 23
##      id pluralty     outcome date       gestation sex      wt parity race    age
##   <dbl> <chr>        <chr>   <date>         <dbl> <chr> <dbl>  <dbl> <chr> <dbl>
## 1    15 single fetus live b… 1964-11-11       284 male    120      1 Asian    27
## # … with 13 more variables: ed <chr>, ht <dbl>, wt.1 <dbl>, drace <chr>,
## #   dage <dbl>, ded <chr>, dht <dbl>, dwt <dbl>, marital <chr>, inc <chr>,
## #   smoke <chr>, time <chr>, number <chr>

head(select(Gestation, race, wt, number), 1)
## # A tibble: 1 × 3
##   race     wt number
##   <chr> <dbl> <chr>
## 1 Asian   120 never
```

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Choosing and renaming columns

- We can also rename columns on the fly as we select them

```
select(Gestation,
       Race                = race,
       `Birthweight (oz)`  = wt,
       `Cigs. smoked`      = number)
## # A tibble: 1,236 × 3
##    Race  `Birthweight (oz)` `Cigs. smoked`
##    <chr>              <dbl> <chr>
##  1 Asian                120 never
##  2 White                113 never
##  3 White                128 1-4 per day
##  4 White                123 20-29 per day
##  5 White                108 20-29 per day
##  6 White                136 5-9 per day
##  7 Black                138 never
##  8 Black                132 never
##  9 White                120 never
## 10 White                143 15-19 per day
## # … with 1,226 more rows
```

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

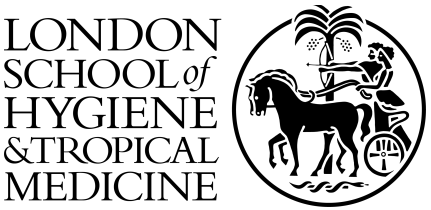
Choosing and renaming columns

- Alternatively we can `rename` columns without worrying about failing to select columns we haven't renamed

```
names(Gestation)
##  [1] "id"         "pluralty"   "outcome"    "date"       "gestation" "sex"
##  [7] "wt"         "parity"     "race"       "age"        "ed"        "ht"
## [13] "wt.1"       "drace"      "dage"       "ded"        "dht"       "dwt"
## [19] "marital"    "inc"        "smoke"      "time"       "number"

Gestation <- rename(Gestation, `Cigs. smoked` = number)

names(Gestation)
##  [1] "id"           "pluralty"     "outcome"      "date"         "gestation"
##  [6] "sex"          "wt"           "parity"       "race"         "age"
## [11] "ed"           "ht"           "wt.1"         "drace"        "dage"
## [16] "ded"          "dht"          "dwt"          "marital"      "inc"
## [21] "smoke"        "time"         "Cigs. smoked"
```



Choosing rows

- The dplyr equivalent of `subset` is `filter`
- Takes a logical statement and does non-standard evaluation of variable names
- `filter(data, A & B)` the same as `filter(data, A, B)`

```
Gestation2 <- select(Gestation,
                     Race                = race,
                     `Birthweight (oz)`  = wt,
                     `Cigs. smoked`)

filter(Gestation2, Race == "White", `Cigs. smoked` == "never")
## # A tibble: 352 × 3
##    Race  `Birthweight (oz)` `Cigs. smoked`
##    <chr>              <dbl> <chr>
##  1 White                113 never
##  2 White                120 never
##  3 White                144 never
##  4 White                125 never
##  5 White                122 never
##  6 White                113 never
##  7 White                134 never
##  8 White                128 never
##  9 White                129 never
## 10 White                110 never
## # … with 342 more rows
```

Choosing rows

- `slice*()` functions allow you to select rows based on their properties, e.g. which babies have lowest birth weight overall and in each race group?

```
slice_min(Gestation2, `Birthweight (oz)`)
## # A tibble: 1 × 3
##   Race  `Birthweight (oz)` `Cigs. smoked`
##   <chr>              <dbl> <chr>
## 1 Black                 55 never

slice_min(group_by(Gestation2, Race), `Birthweight (oz)`)
## # A tibble: 6 × 3
## # Groups:   Race [6]
##   Race  `Birthweight (oz)` `Cigs. smoked`
##   <chr>              <dbl> <chr>
## 1 Asian                 71 5-9 per day
## 2 Black                 55 never
## 3 Mex                   97 never
## 4 Mixed                 77 20-29 per day
## 5 White                 63 never
## 6 <NA>                  82 20-29 per day
```

Reshaping data frames

Long and wide tidy data

- Tidy **data frames** consist of a number of observations (rows) of variables (columns), they can be either **wide** or **long** (Wickham 2014)
- We can pivot between wide and long format with tidyr's `pivot_*()` functions
- Pivoting to a longer data frame helps put data in *key-value* pairs, useful for ggplot2
- The value is the `value` of the `name`d variable for a given `id`

Long and wide tidy data

To make this pivot, we specify

- which `cols` are to be converted from being k columns of length n to one column of length $n \times k$
- the `names_to` column name, which contains the names of the pivoted columns
- the `values_to` name of the column containing the `value` of each variable for each `id`

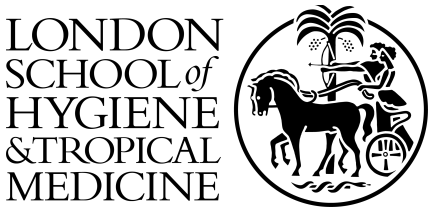
Long and wide tidy data

```
Gestation_igwa <- select(Gestation, id, gestation, wt, age)

Gestation_long <- pivot_longer(
  data            = Gestation_igwa,
  cols            = c(gestation, wt, age),
  names_to        = 'name',
  values_to       = 'value')

head(Gestation_long, 6)
## # A tibble: 6 × 3
##      id name      value
##   <dbl> <chr>     <dbl>
## 1    15 gestation   284
## 2    15 wt          120
## 3    15 age          27
## 4    20 gestation   282
## 5    20 wt          113
## 6    20 age          33
```

- NB: we need to use `'` quotes for `names_to` and `values_to` arguments because they are strings defining new columns



Long and wide tidy data

Or specify which columns *not* to pivot, e.g. `-id` selects all variables except `id`

```
Gestation_long <- pivot_longer(Gestation_igwa, -id)

head(Gestation_long, 6)
## # A tibble: 6 × 3
##      id name      value
##   <dbl> <chr>     <dbl>
## 1    15 gestation   284
## 2    15 wt          120
## 3    15 age          27
## 4    20 gestation   282
## 5    20 wt          113
## 6    20 age          33
```

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Long and wide tidy data

- To convert to a wider format, we use `pivot_wider`
- For example, we specify:
  - the *data* source
  - where we get the new column *names from*
  - where we get the new column *values from*

```
Gestation_wide <- pivot_wider(data        = Gestation_long,
                              names_from  = name,
                              values_from = value)

head(Gestation_wide, 3) # recovered original data frame
## # A tibble: 3 × 4
##      id gestation    wt   age
##   <dbl>     <dbl> <dbl> <dbl>
## 1    15       284   120    27
## 2    20       282   113    33
## 3    58       279   128    28
```

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Pipe (Optional)

Pipe

- dplyr imports the `%>%` pipe from magrittr
- `f(g(x))` is equivalent to `x %>% g %>% f`
- Makes it easier to chain operations together without storing temporary objects
- Output on left of `%>%` becomes first argument of function on right
    - by convention, all tidyverse functions take a data frame as their first argument

```
x %>% f_1 %>% f_2 %>% f_3

# rather than
f_3(f_2(f_1(x)))

# or even worse...
x_1 <- f_1(x)
x_2 <- f_2(x_1)
x_3 <- f_3(x_2)
```

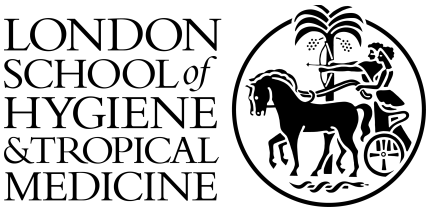
LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Pipe

- An example

```
Gestation %>% group_by(race, smoke) %>%
  summarise(wt = mean(wt)) %>%  pivot_wider(names_from  = race,
                                            values_from = wt)
## # A tibble: 5 × 7
##   smoke                Asian Black   Mex Mixed White  `NA`
##   <chr>                <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 never                114.   117.  123.  125.  125.  119.
## 2 now                   98.8  108.  127.  108.  116.  114.
## 3 once did, not now    111    117.   NA   134   126.   NA
## 4 until current pregnancy 117  113.  129.  111   127.   NA
## 5 <NA>                 126    119   115    NA   131.   NA
```

- NB the `smoke` variable is character and sorted alphabetically.
- We don't *expect* you to use pipes, but they're useful



Summary

Summary

- Summarising data
  - `group_by` to set a group structure
  - `summarise` to calculate summary stats across group structure
  - `count` to see how many rows in each group
- Reshaping data frames
  - `pivot_longer` from variables side by side to key-value
  - `pivot_wider` from key-value to named column variables

Summary

- Dealing with rows and columns
  - `mutate` to create/modify/delete columns
  - `select` to choose columns
  - `filter` to choose rows based on logical condition
  - `slice*` to choose rows based on position or property
- Pipe
  - `%>%` to chain operations
- Wickham (2014) on what tidy data is
- Wickham et al. (2019) for more explanation of tidyverse

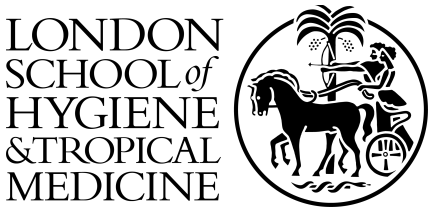
LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Visualisation with ggplot2

Why do we visualise?

Since the aim of exploratory data analysis is to learn what seems to be, it should be no surprise that pictures play a vital role in doing it well. There is nothing better than a picture for making you think of questions you had forgotten to ask (even mentally).

Tukey and Tukey (1985)



Principles

Tufte (1983) and Pantoliano (2012)

- Show the data
- Provide clarity
- Allow comparison where appropriate
    - use aesthetics to draw attention to important details
    - make clear that data has multiple levels of structure

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Principles

- Produce graphs with high data density
  - make every drop of ink count
  - careful use of whitespace
- Avoid excessive and unnecessary use of graphical effects
- Reader should be able to understand what the graph means and not be
  - misled into thinking something that is untrue
  - distracted from the main point

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Building plots

`ggplot2` uses a grammar of graphics (Wickham 2010)

- map variables in data frame to aesthetic options in the plot
- choose a geometry for how to display these variables
- adjustments to axis scales
- adjustments to colors, themes, etc.
- adding extra commands in a 'do this, then do this' manner
- python users have plotnine (Kibirige 2020) which is based on the same ideas

Building plots

How do we structure a call to `ggplot` to make a plot?

```
# library(ggplot2) # already loaded with tidyverse
ggplot(data = my.data.frame,
       aes(x = my.x.variable,
           y = my.y.variable)) +
  geom_point()
```

- load ggplot2 package
- Specify we want a `ggplot` object and which data frame we're going to use,
- Set **aesthetic options** to map to the x and y axes of the plot
- State geometry we're using to show variables

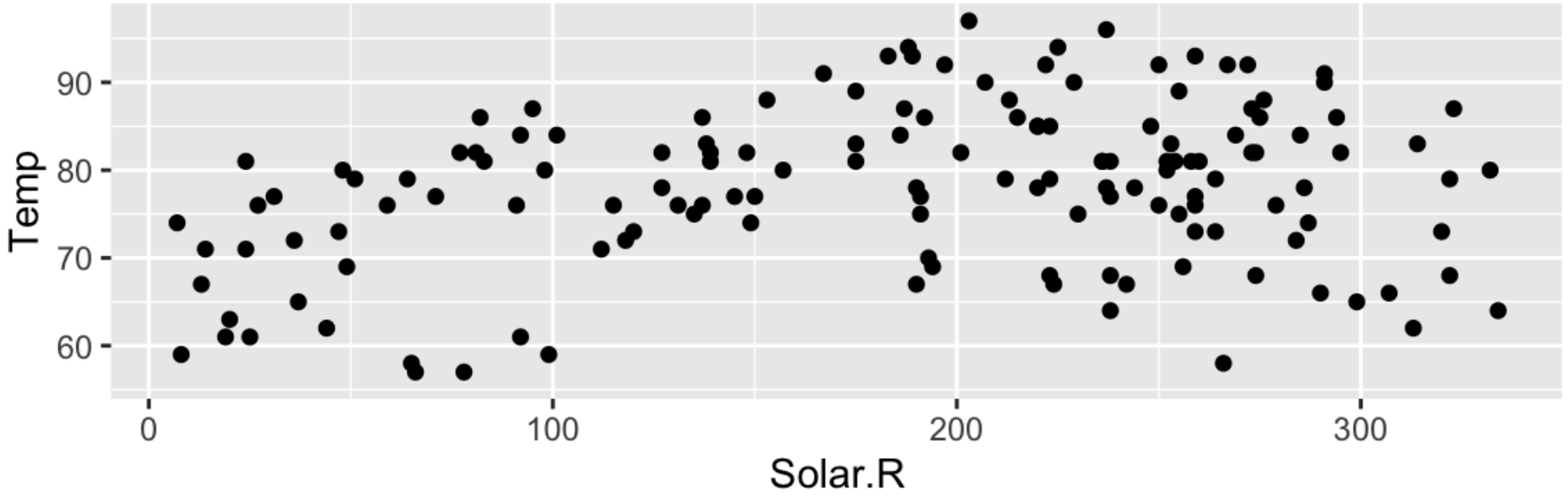
LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Building plots

- For example, consider daily maximum temperature varying with solar radiation in New York City 1973
- Each row is a pair of values (x, y), shown as a point

```
data(airquality)
solar_temp_plot <- ggplot(data = airquality,
       aes(x = Solar.R, y = Temp)) + geom_point()

solar_temp_plot
```



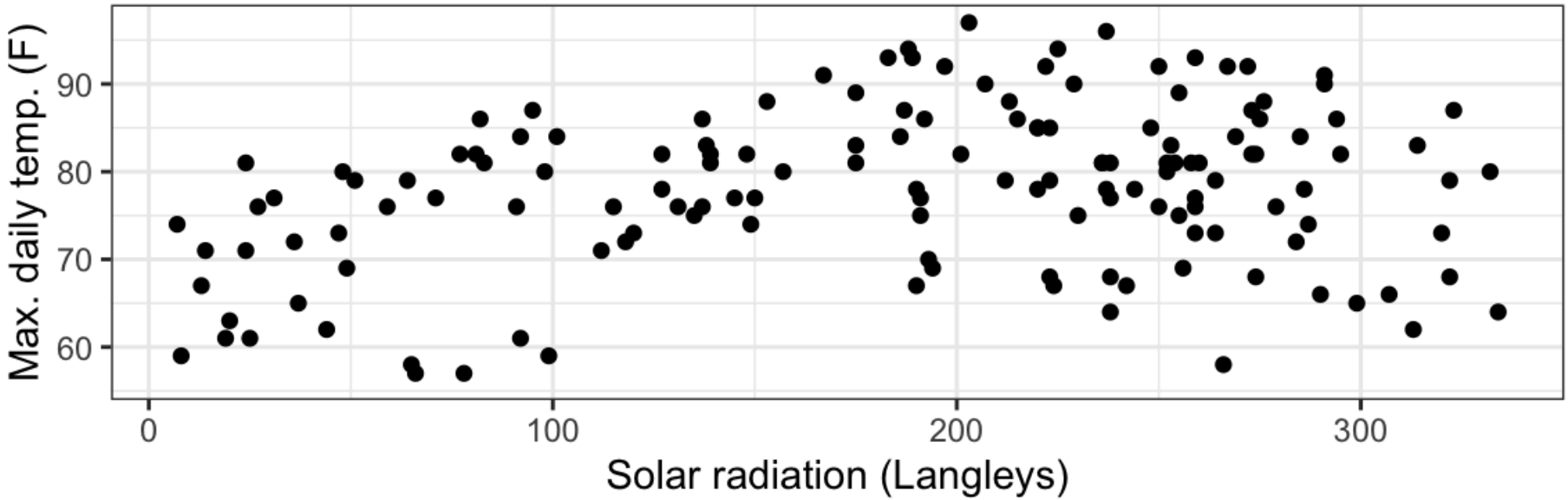
LOI SCHOOL of HYGIENE &TROPICAL MEDICINE

Scatter plot

- We can add some human-friendly labels and change the theme

```
solar_temp_plot <- solar_temp_plot + theme_bw() +
  labs(x = 'Solar radiation (Langleys)', y = 'Max. daily temp. (F)')

solar_temp_plot
```



LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Some more geometries

Line plot

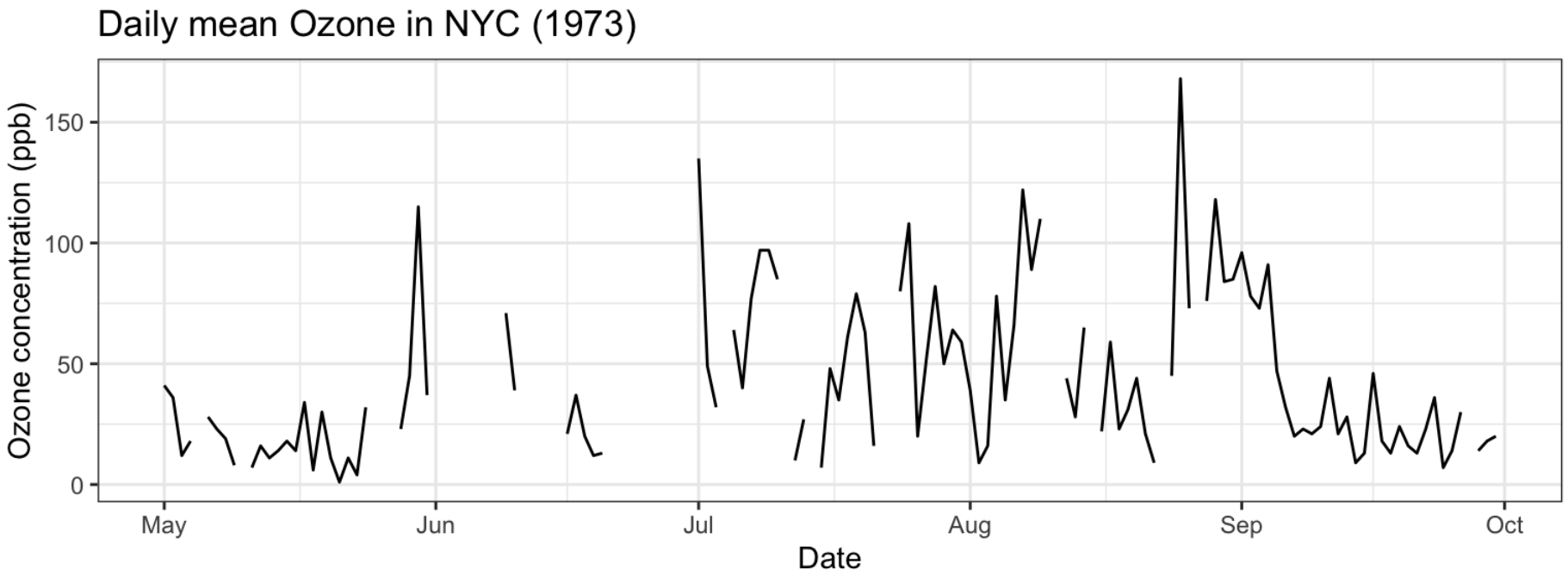
- Similar to scatter plot, but joins pairs of values
- Useful when showing how something changes over time
- If (x, y) pairs ordered by
  - x, use `geom_line()` (e.g. x is time)
  - row order, use `geom_path()`
  - nothing, don't use a line

```
airquality <- mutate(airquality, Date = as.Date(paste('1973', Month, Day, sep = '-')))

airquality_plot <- ggplot(data = airquality, aes(x=Date, y=Ozone)) +
  geom_line() + theme_bw() + labs(y     = 'Ozone concentration (ppb)',
                                  title = 'Daily mean Ozone in NYC (1973)')
```

Line plot

```
airquality_plot
```

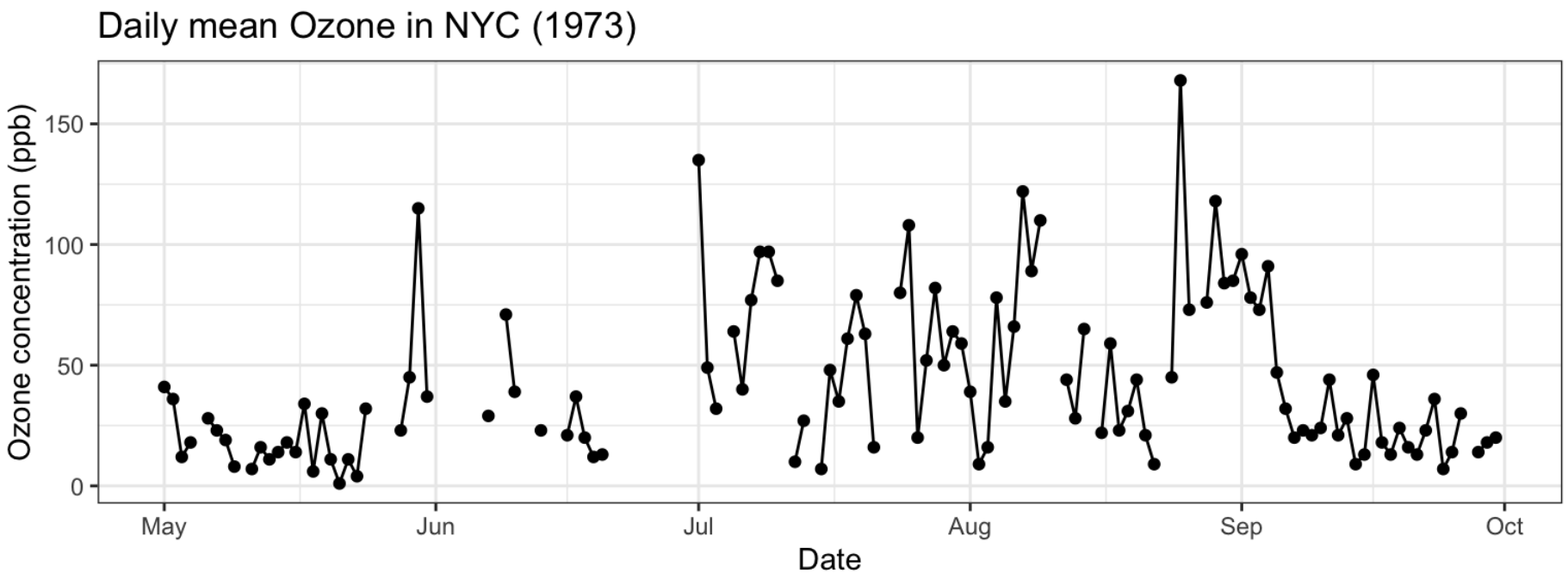


- You may see this referred to as a time series plot

Line plot

- Observations whose neighbours are `NA` values can't be plotted with a line
- Can layer multiple geometries for same aesthetic mapping

```
airquality_plot + geom_point()
```

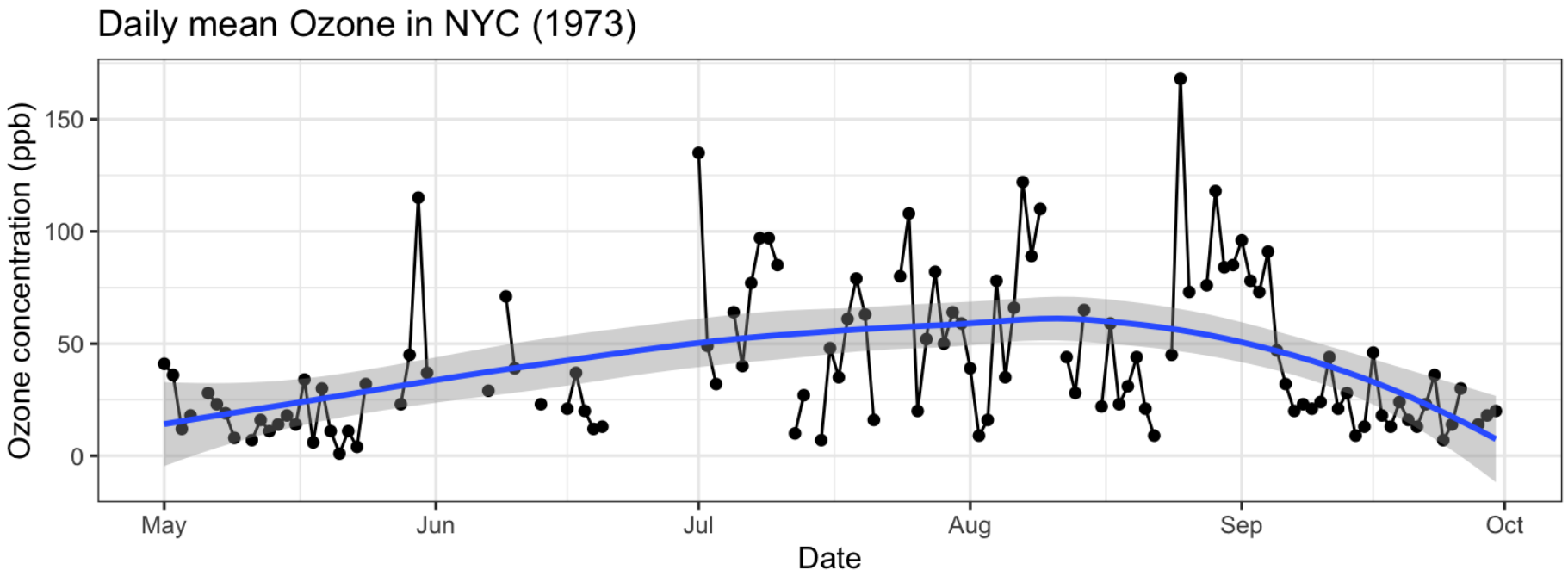


LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Scatterplot smoother

- Often too much data in a scatter plot to see pattern
- Maybe we want to highlight the trend in the data

```
airquality_plot + geom_point() + geom_smooth()
```

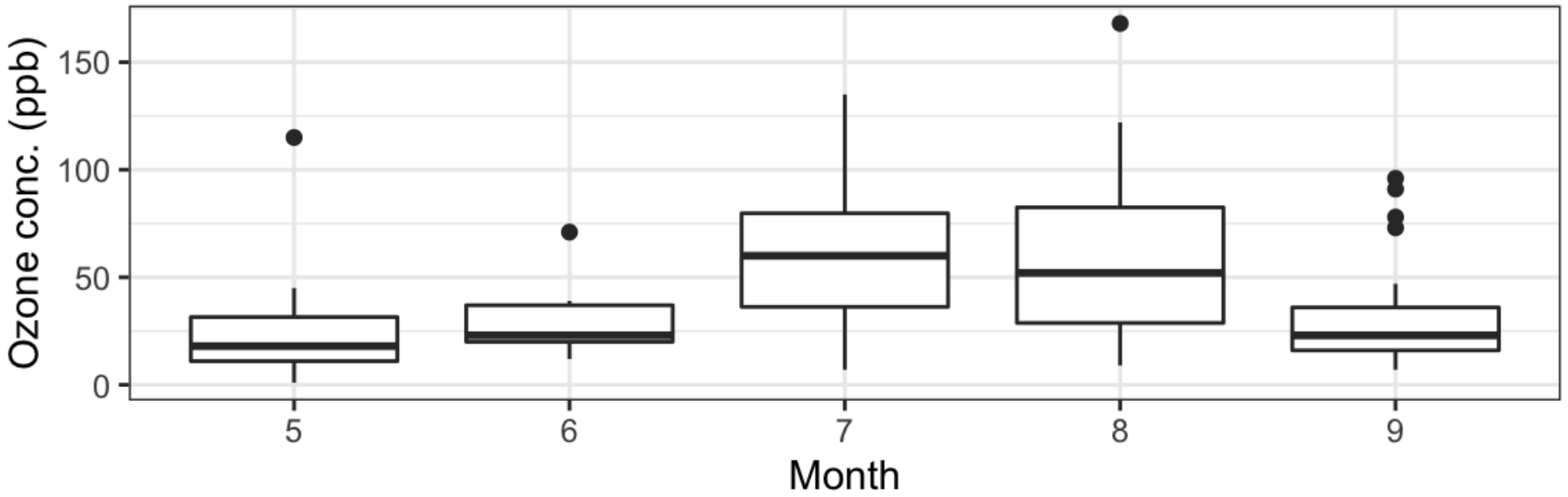


LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Boxplot

- continuous y, discrete x
- outliers (> 1.5 IQR from median) shown as points automatically

```
ggplot(data = airquality, aes(x = factor(Month), y = Ozone)) +
  geom_boxplot() + theme_bw() +
  labs(y = 'Ozone conc. (ppb)', x = 'Month')
```



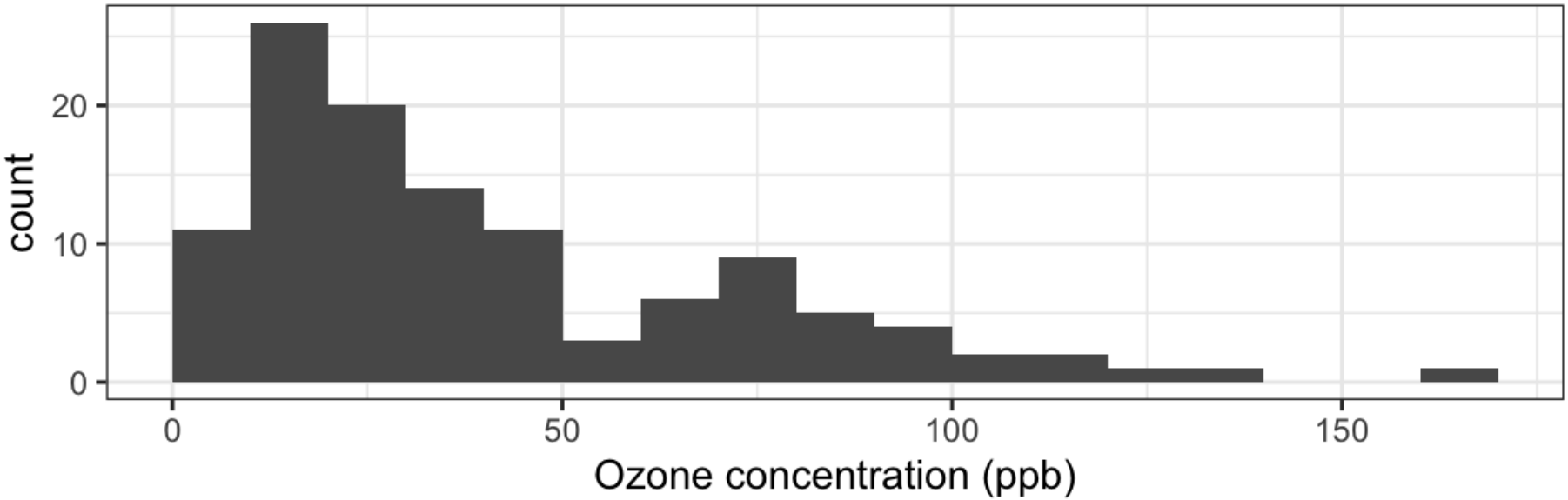
LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Histograms

- univariate graphical summary needs only one aesthetic, **`x`**

```
ozone_hist <- ggplot(data = airquality, aes(x = Ozone)) +
  geom_histogram(binwidth = 10, boundary = 0) +
  labs(x = 'Ozone concentration (ppb)') + theme_bw()

ozone_hist
```



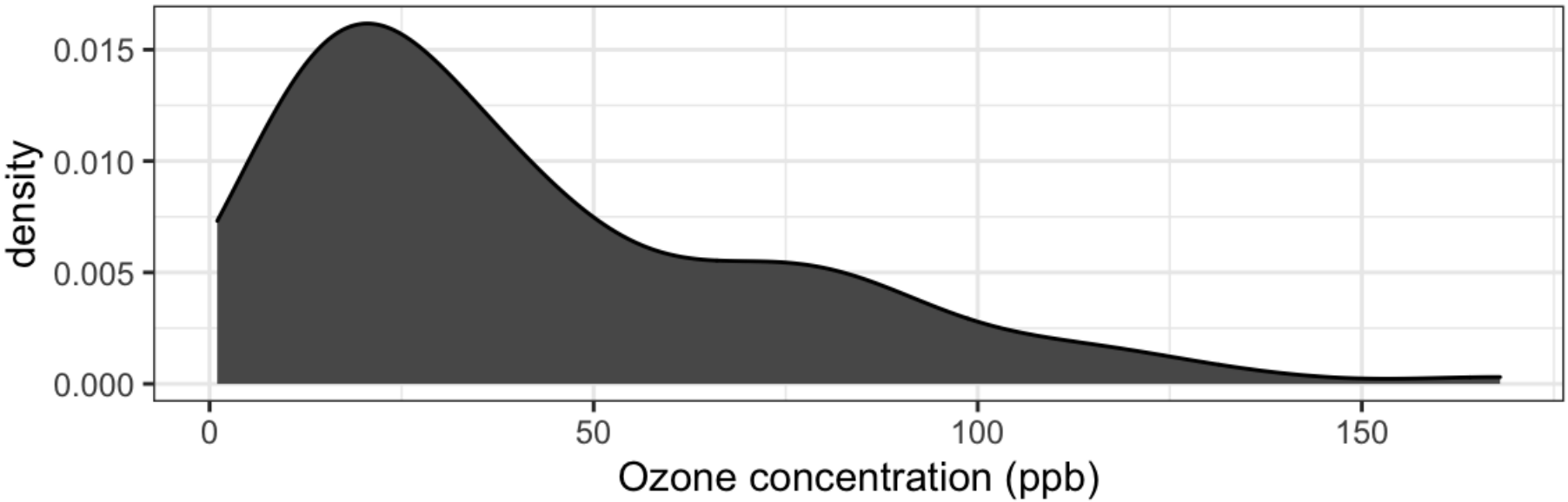
LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Density plots

- Kernel smoothing (continuous analogue of histogram)

```
ozone_dens <- ggplot(data = airquality, aes(x = Ozone)) +
  geom_density(fill = 'grey35') +
  labs(x = 'Ozone concentration (ppb)') + theme_bw()

ozone_dens
```



LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

More on aesthetics

More on aesthetics

Aesthetic	What it effects
`size`	points, lines
`shape`	points
`linetype`	lines
`colour`	points, lines, boundary
`alpha`	transparency
`fill`	interior
`group`	repeats geometry

- If these (except group) are *outside* `aes()` they fix the value for all parts of that geometry
- Aesthetics specified inside `ggplot()` are inherited by all geometries for that plot
- Not all geometries accept all aesthetics (e.g. `geom_line()` has no fill)
- Some point shapes admit a `colour` and a `fill`

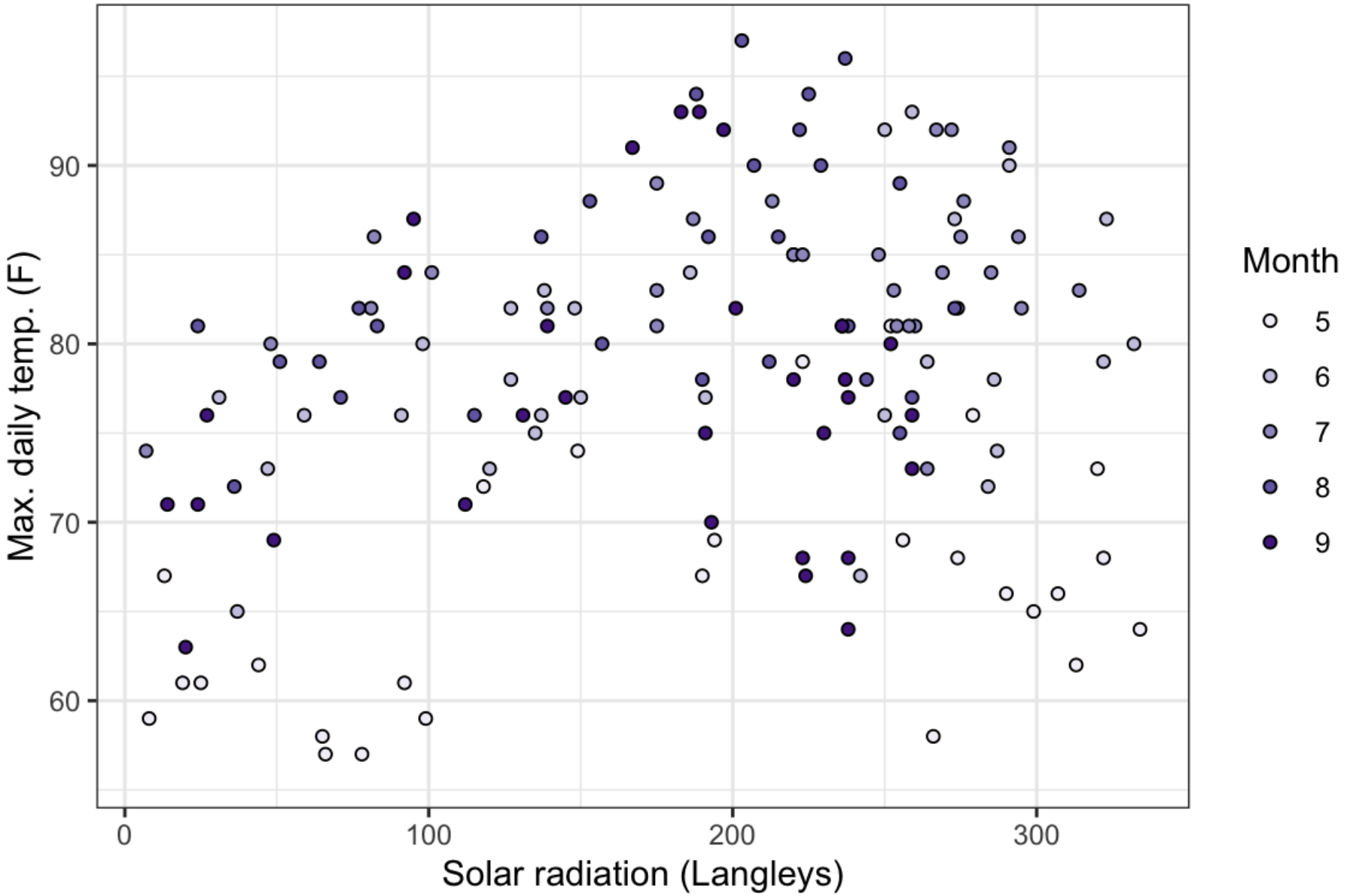
LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

More on aesthetics

```
data(airquality)
solar_temp_plot_colored <-
  ggplot(data = airquality,
       aes(x = Solar.R, y = Temp)) +
  geom_point(aes(fill = factor(Month)),
             shape = 21,
             color = 'black')  +
  labs(x = 'Solar radiation (Langleys)',
       y = 'Max. daily temp. (F)') +
  theme_bw() +
  scale_fill_brewer(palette = "Purples",
                    name   = 'Month')
```

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

More on aesthetics



Small multiples

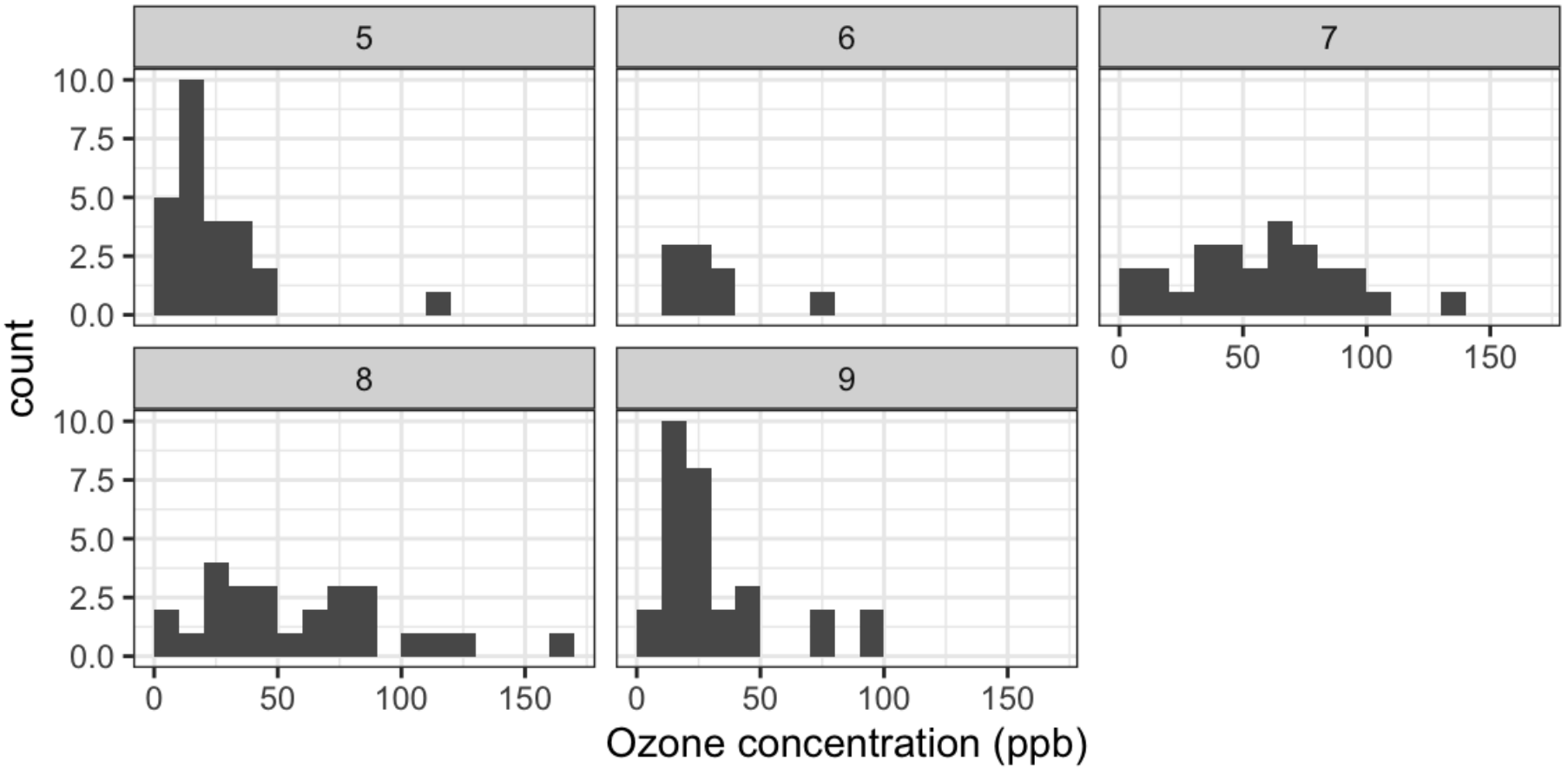
Small multiples

- Group a plot by some categorical variable
- Repeat a basic graph for groups in the data
    - air quality data has information about, e.g. months
- Can view 3-5 dimensions in the data on a 2D page
    - Often a better alternative to 3D, since it doesn't distort comparisons
    - Inner axes relate to the smallest X-Y plots
    - Outer axes relate to the grouping variables
- Avoids writing loops

Small multiples

- Repeat histogram for each value of `Month`, one per facet

```
ozone_hist + facet_wrap( ~ Month)
```

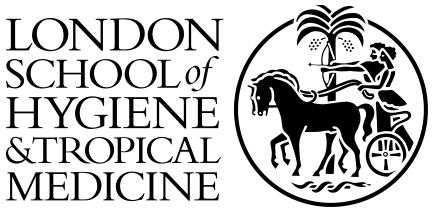


Small multiples

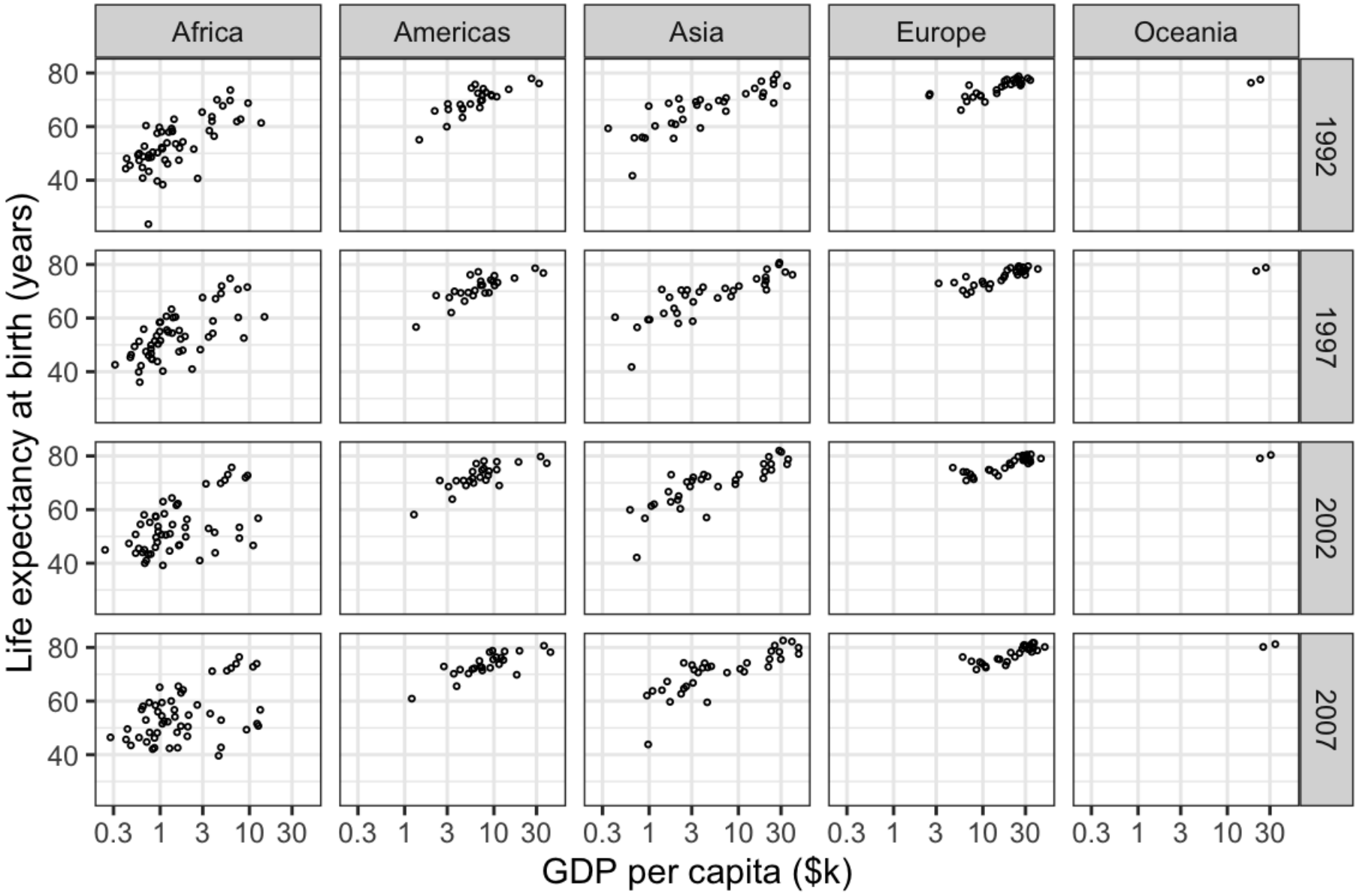
We can also use `facet_grid()` to repeat the aesthetic and geometries for specified `rows` and `cols` variables

```
library(gapminder)
data(gapminder)

gapminder_plot <-
  ggplot(data = subset(gapminder, year >= 1992),
       aes(x = gdpPercap/1e3,
           y = lifeExp)) +
    geom_point(shape = 1, size = 0.5) +
    facet_grid(rows = vars(year),
               cols = vars(continent)) +
    scale_x_log10(labels = ~sprintf("%g", .)) +
  xlab("GDP per capita ($k)") +
  ylab("Life expectancy at birth (years)") +
  theme_bw() +
  theme(panel.grid.minor.x = element_blank())
```



Small multiples



Summary

Summary

- We make graphs to tell a story with data
- Should draw reader in and explain what they're seeing
- Plots are built from
    - geometric objects
    - axis scales
    - coordinate systems (linear or logarithmic scale, 2D, 3D, etc.)
    - annotations (e.g. heading in small multiples)

Summary

- Successively building a plot with a grammar of graphics allows development of complex plots from simple elements and small changes
- Choose a plotting geometry that helps tell the story
- Meaningful labels remove ambiguity and confusion
- Be careful not to put too much in

Further reading

The `#r4ds` community have TidyTuesday which makes use of the ideas in Wickham and Grolemund (2017)

- History of visualisation
  - Friendly (2005)
  - Friendly (2006)
- Visualisation to help decision making
  - Tufte (1997)
- ggplot2 resources
  - RStudio (2021)
  - Chang (2017)

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

References

Chang, Winston. 2017. *R Graphics Cookbook: Practical Recipes for Visualizing Data*. 2nd ed. O'Reilly Media. http://www.cookbook-r.com/Graphs/.

Friendly, M. 2005. "Milestones in the History of Data Visualization: A Case Study in Statistical Historiography." In *Classification: The Ubiquitous Challenge*, edited by C. Weihs and W. Gaul, 34–52. New York: Springer. http://www.math.yorku.ca/SCS/Papers/gfkl.pdf.

———. 2006. "A Brief History of Data Visualization." In *Handbook of Computational Statistics: Data Visualization*, edited by C. Chen, W. Härdle, and A Unwin. Vol. III. Heidelberg: Springer-Verlag. http://www.datavis.ca/papers/hbook.pdf.

Kibirige, Hassan. 2020. *A Grammar of Graphics for Python*. https://plotnine.readthedocs.io/en/stable/index.html.

Nolan, Deborah, and Terry P Speed. 2001. *Stat Labs: Mathematical Statistics Through Applications*. Springer Science & Business Media.

Pantaliano, Mike. 2012. "Data Visualization Principles: Lessons from Tufte." https://moz.com/blog/data-visualization-principles-lessons-from-tufte