

Intro to R for Biologists

Session 4

Data Visualisation

Irina & Rao

02/02/2021

Hilary 2021

INTRO TO R FOR BIOLOGISTS

► Data Visualisation

- Intro to ggplot2
- Base R plotting vs ggplot2
- ggplot2 syntax (demo)
- Intro to factors
- Problem set from Session 3
- Merging and reshaping data.frames (demo)
- Dot/box/bar plots (demo with Covid vaccine data)
- Practical (breakout room)
 - OWID Covid data

How to choose the graph type

- Human perception is not uniformly good at distinguishing different physical aspects – e.g. length is better than volume
- Factors to help decide on viz:
 - Information – detail or summary?
 - Number of dimensions
 - Comparisons
 - Continuous vs categorical data
 - Scale – e.g. linear or log?
 - Shape and size
 - Colours – distinct? Colour-blind friendly?
 - Legends and labels
 - Subplots
- [From Data to Viz](#) – A handy tool to help you decide

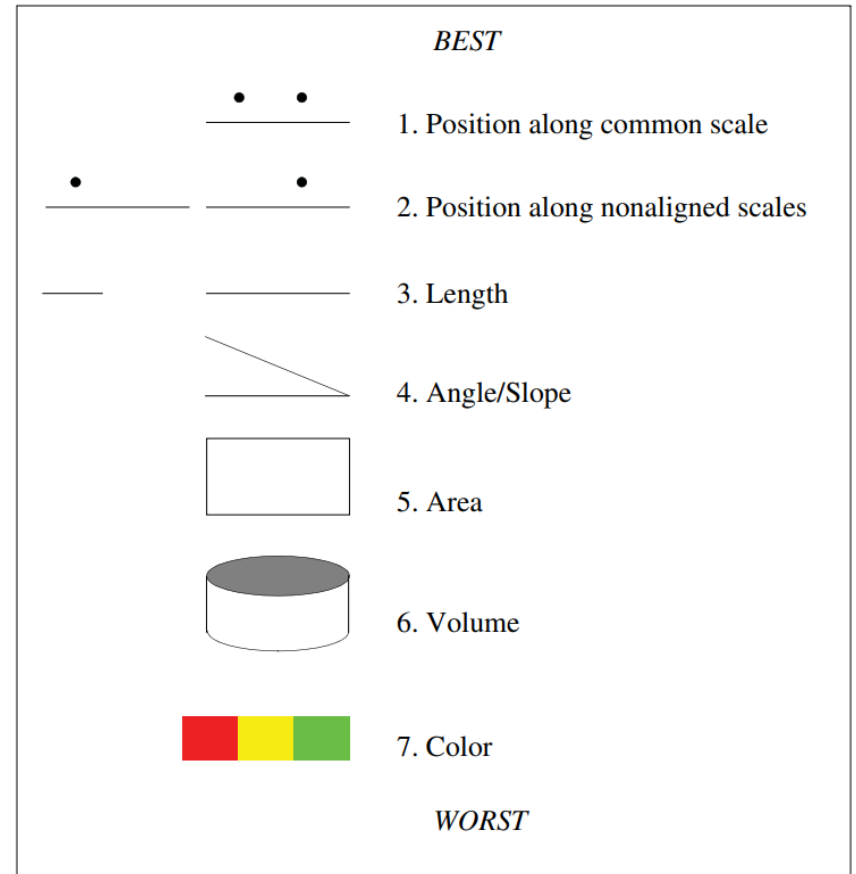
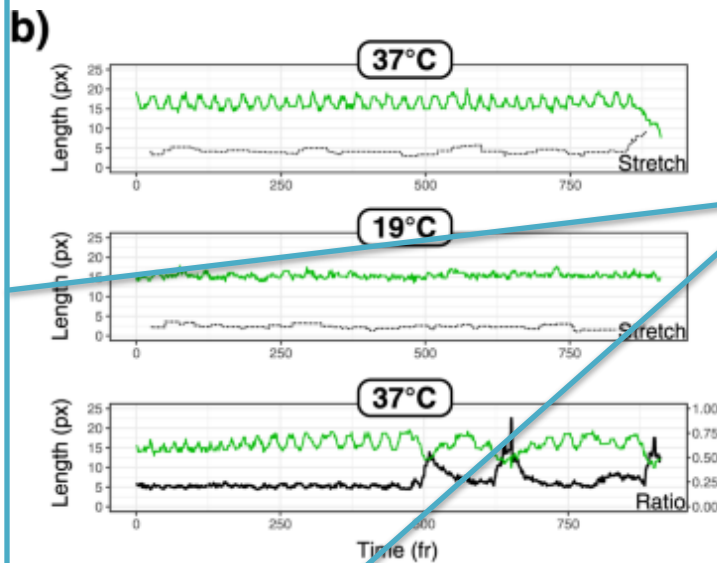
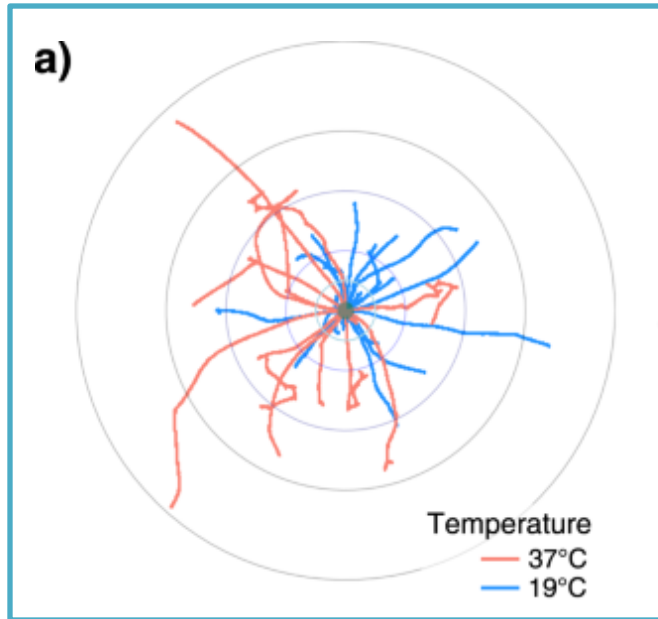
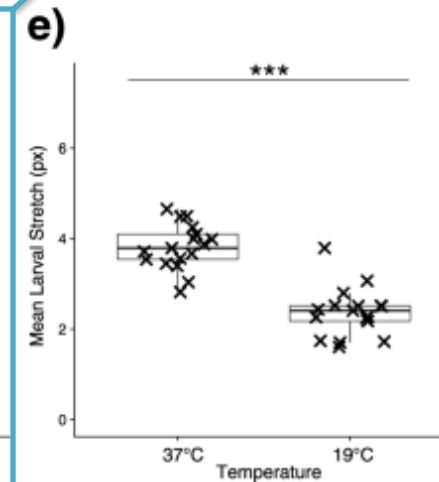
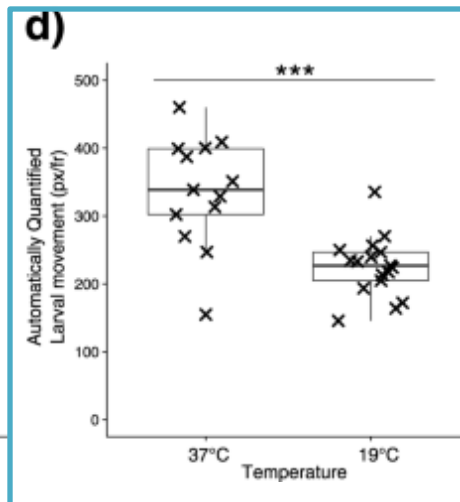
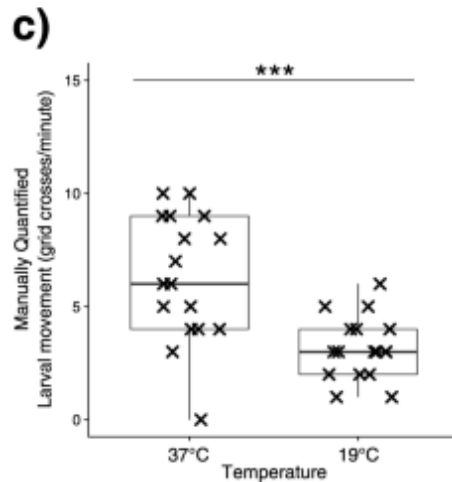


Figure 10.3 Cleveland graphic elements hierarchy

How to choose the graph type



Using the same data but making different points



Visualising data in R

- Base R plotting (e.g. plot() function)

```
plot(x = iris$Sepal.Length, y =  
iris$Petal.Length, col =  
iris$Species)
```

- ggplot2 package

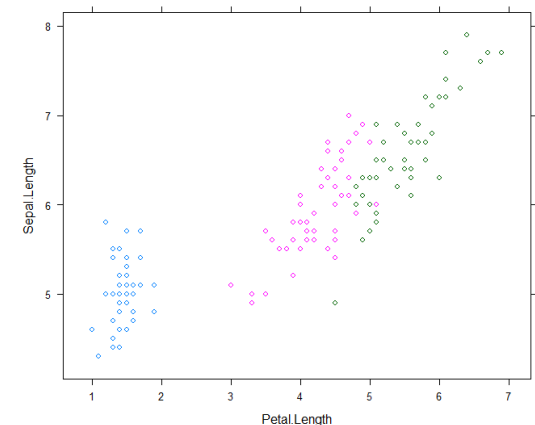
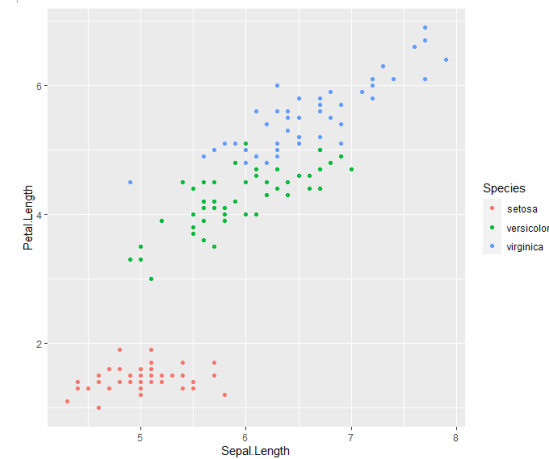
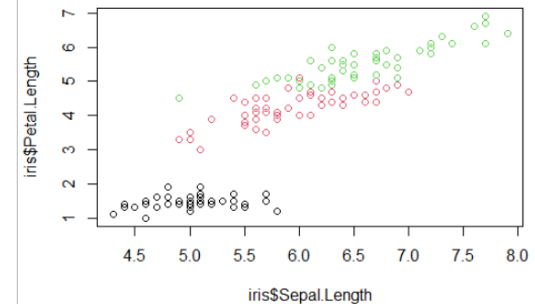
```
ggplot(iris, aes(x = Sepal.Length,  
y = Petal.Length, colour =  
Species)) + geom_point()
```

- Lattice package

```
xyplot(Sepal.Length ~ Petal.Length,  
data = iris, groups = Species)
```

- Grid package

- Low level plotting (both ggplot2 and lattice built on grid graphics)

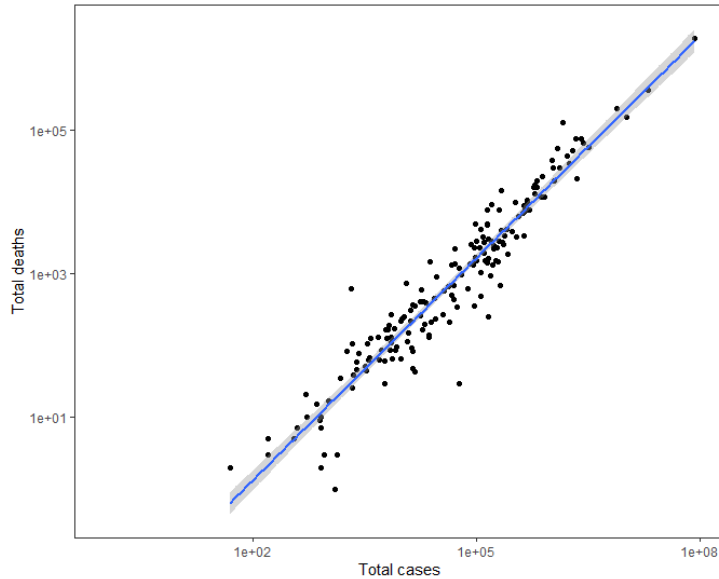


Graphics devices in R

- The plotting window in Rstudio
 - Needs to be saved manually
 - Size of the figure can be adjusted manually
 - Can be copied to clipboard from RStudio
 - For quick exploratory visualisation
- A file (svg, png, pdf, tiff, jpeg, bmp)
 - Saved to file from code
 - Size of the figure can be specified in the code
 - For creating reproducible figures
 - `png()`, `pdf()`, `svg()`, `ggsave()` are functions to save plots
- Start or close graphical devices with `dev.new()` or `dev.off()` respectively
 - Tip: If your plot is not showing up on the window or file that you are expecting it to, you are probably drawing on the wrong device. `dev.off()` may solve this issue.

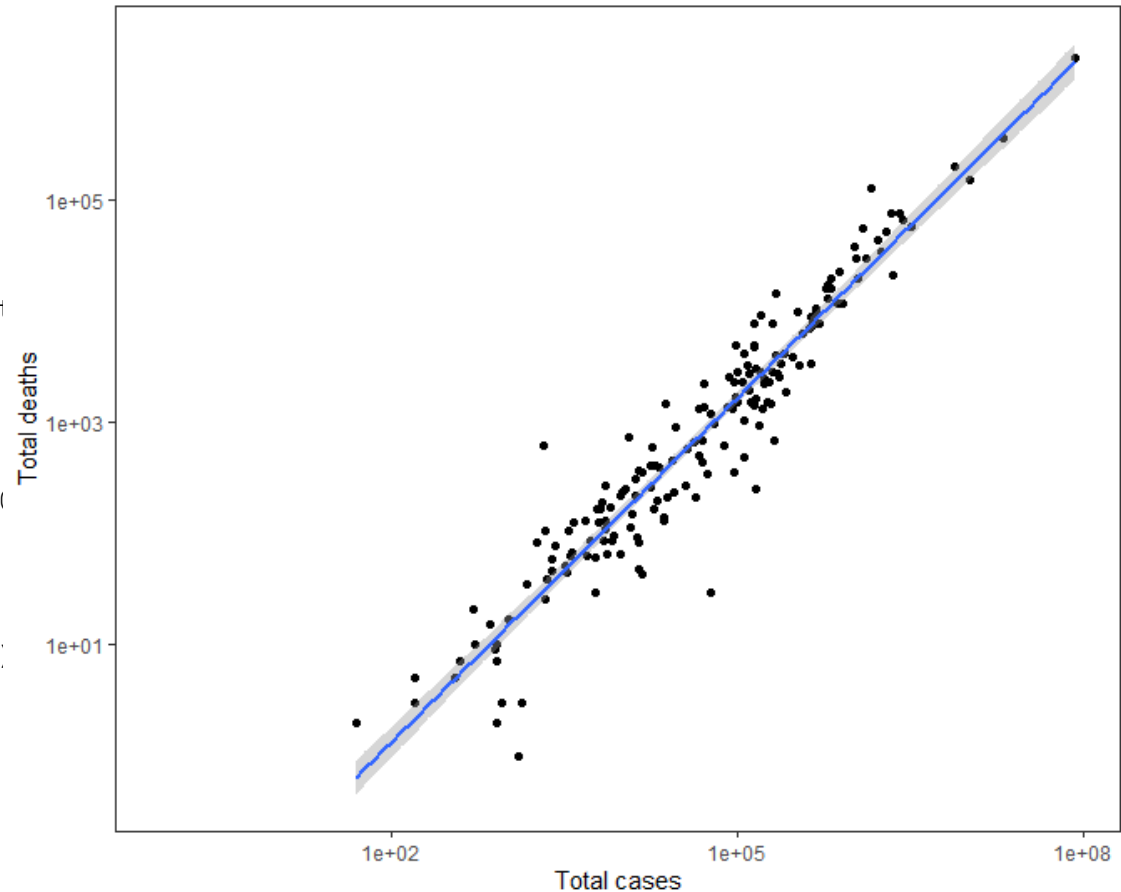
Building a graph step-by-step with ggplot2

OWID Covid data for Jan 01 2021: Total Cases vs. Total Deaths



```
ggplot(owid_covid_newyear, aes(x = total_cases, y = total_deaths)) +  
  geom_point() +  
  scale_x_log10() +  
  scale_y_log10() +  
  geom_smooth(method = "lm") +  
  ggtitle("OWID Covid data for Jan 01 2021: Total Cases vs. Total Deaths") +  
  xlab("Total cases") +  
  ylab("Total deaths") +  
  theme_bw() +  
  theme(panel.grid = element_blank())
```

OWID Covid data for Jan 01 2021: Total Cases vs. Total Deaths



ggplot2 syntax

```
ggplot(data, mapping) +  
  geom_point(mapping, stat, position, colour, fill, shape,  
size, alpha) +  
  coord_cartesian() +  
  scale_colour_discrete() +  
  facet_wrap() +  
  theme_bw() +  
  ggtitle() +  
  labs()
```

Global mapping –
applies to the
entire plot

Geom-specific

Where,

```
mapping = aes(x, y, colour, fill, shape, size, alpha)
```

Note: Colour, shape, alpha, etc. can be passed

- 1) to the **mapping** argument - changes attributes based on some variable in the dataset itself (e.g. colour = location, or shape = hdi_class)
- 2) directly as specific arguments in the geom function - changes the attributes to fixed values that are provided separately from the data (e.g. colour = "red" or size = 10)

**Let's
explore
practically**



How to plot your data?

The most common plots in publications:

box plots/violin plots, bar plots, dot plots, histograms/density plots, line graphs, networks, heatmaps, PCA, etc.

Hint: choose the appropriate representation based on the type of data you have.

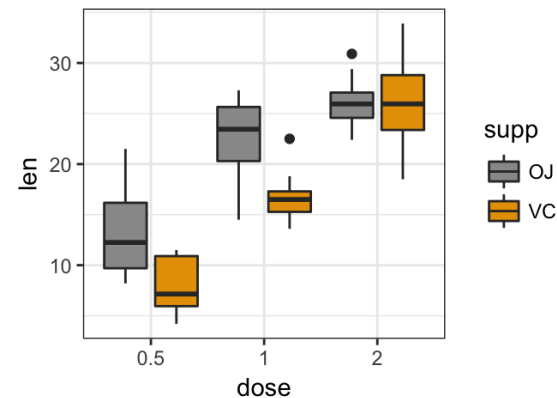
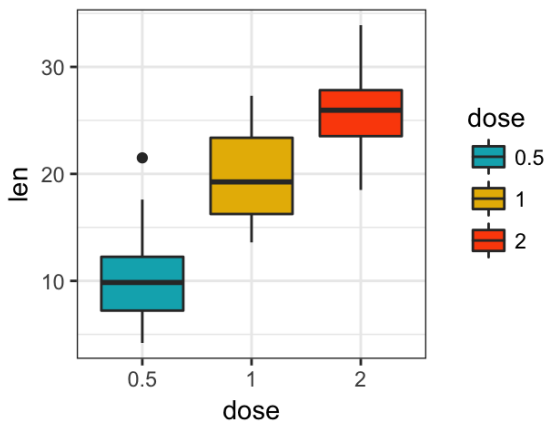
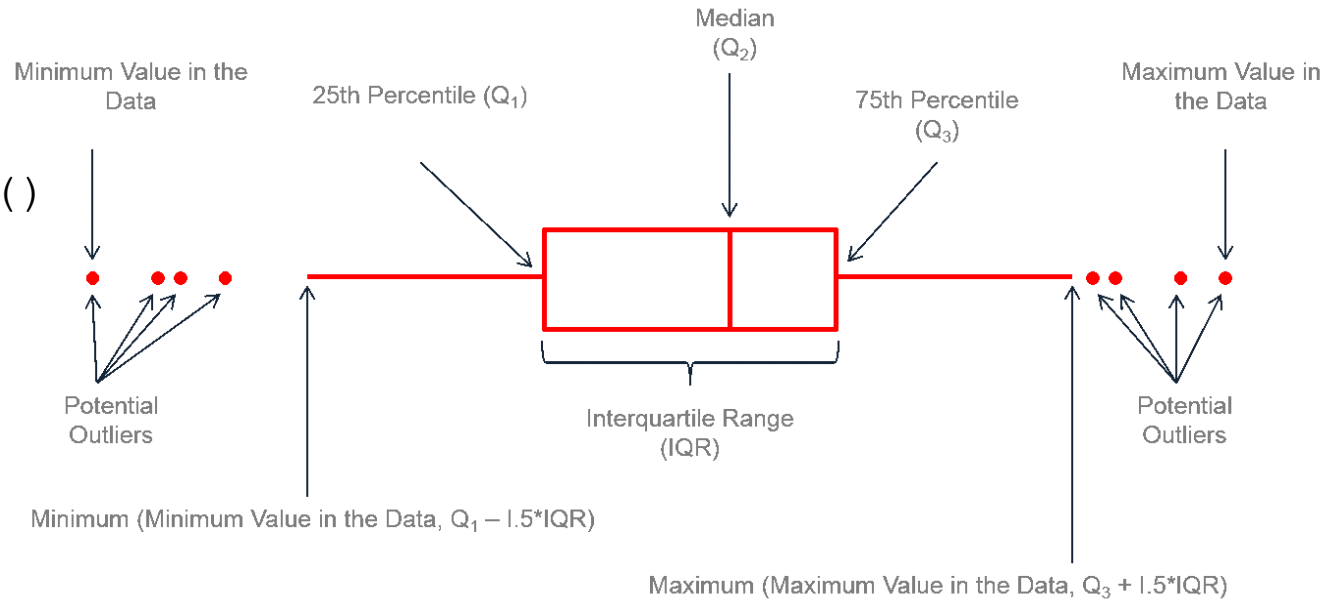
Typical data types: numeric, categoric, maps, network, time series
+combinations of those

[From Data to Viz](#) – Decision tree to choose an appropriate chart type for your data

Plotting grouped data

Box-plots

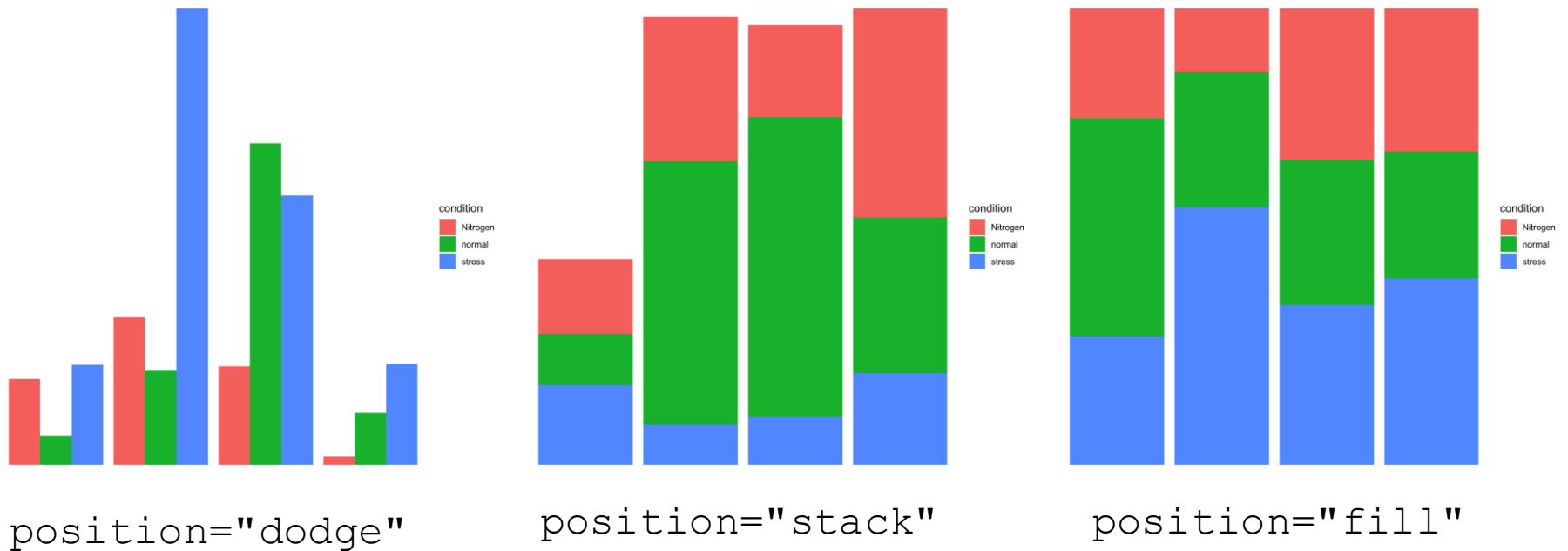
`geom_boxplot()`



Plotting grouped data

Bar plots

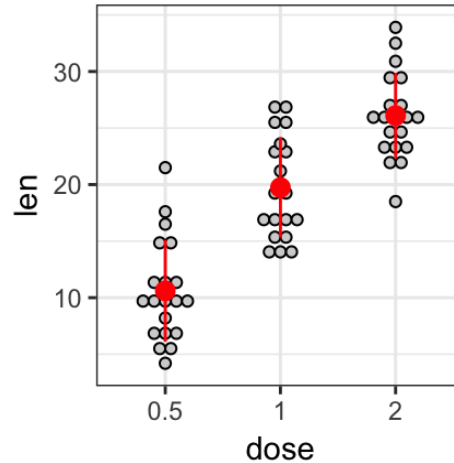
```
geom_bar(position=...)
```



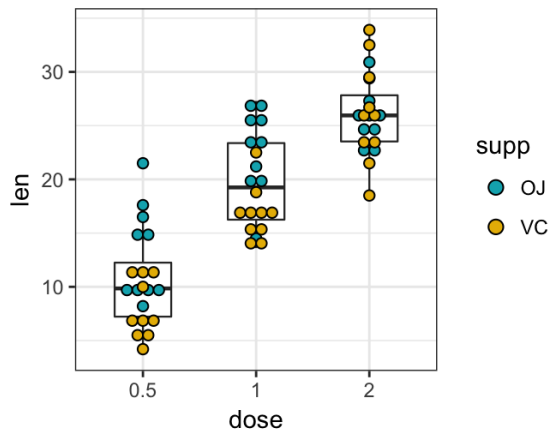
Plotting grouped data

Dot plots

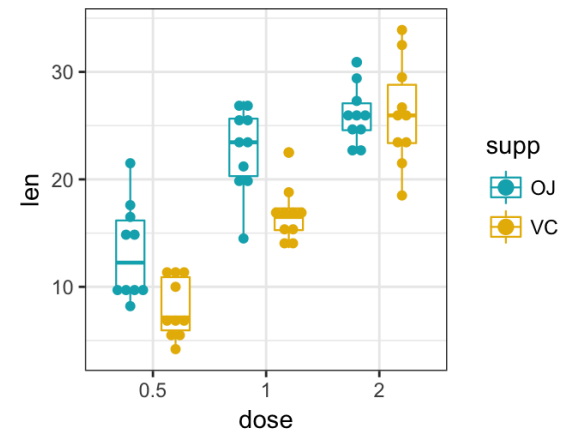
```
geom_dotplot()
```



```
ggplot(...) +  
geom_dotplot(...)
```



```
ggplot(...) +  
geom_boxplot(...) +  
geom_dotplot(...)
```



```
ggplot(...) +  
geom_boxplot(position = "dodge") +  
geom_dotplot(...)
```

Understanding factors

- Factors are used to represent categorical data.
- Factors don't have to be binary! You can have many categories.
- When you import a table to R, your categorical columns are treated as characters ("Germany", "Belgium", "Germany", etc.) or numbers (1, 2): using safest option - `stringsAsFactors = FALSE` argument in `read...` functions.
- You know where your categories are presented. Convert the relevant column into factor:

```
df1$col1 <- as.factor(df1$col1)
```
- Factors have levels, you can check all the categories within the factor:

```
levels(df1$col1)
```

You need to have your categories as factors in order to analyse and to plot the data by group!

Reshaping data

```
library(data.table)
```

```
melt(data=chrData, id.vars=c("region"))
```

Long Format

region	variable	value
chr1	gc.content	0.43
chr4	gc.content	0.38
chr8	gc.content	40.00
chr1	mappability	63.00
chr4	mappability	47.00
chr8	mappability	40.00

Legend

- Value
- Variable Name
- ID Variable

region	gc.content	mappability
chr1	0.43	63
chr4	0.38	47
chr8	40.00	40

Wide Format

```
dcast(data=chrData, region ~ variable, value.var="value")
```

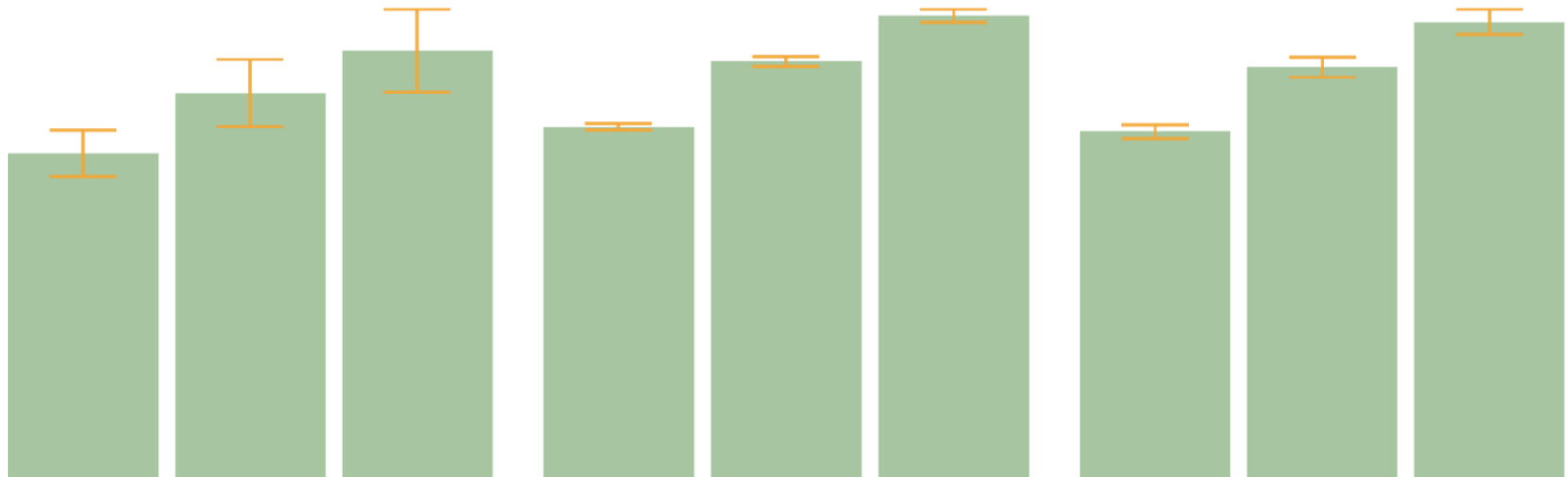
To plot by groups we need our data to be in a long format!

Error bars – geom_bar()

using standard deviation

using standard error

using confidence interval



```
ggplot(data)
+ geom_bar( aes(x=group, y=mean), stat="identity",
fill="forestgreen", alpha=0.5)
+ geom_errorbar( aes(x=group, ymin=mean-sd, ymax=mean+sd),
width=0.4, colour="orange", alpha=0.9, size=1.5) +
ggtitle("using standard deviation")
```


SD, SE, CI

- Standard Deviation (SD) represents the amount of dispersion of the variable. Calculated as the root square of the variance:

```
sd <- sd(vec)
```

```
sd <- sqrt(var(vec) )
```

- Standard Error (SE) is the standard deviation of the vector sampling distribution. Calculated as the SD divided by the square root of the sample size.

```
se = sd(vec) / sqrt(length(vec) )
```

- Confidence Interval (CI).

This interval is defined so that there is a specified probability that a value lies within it. It is calculated as $t * SE$. Where t is the value of the Student's t-distribution for a specific alpha.

```
alpha=0.05
```

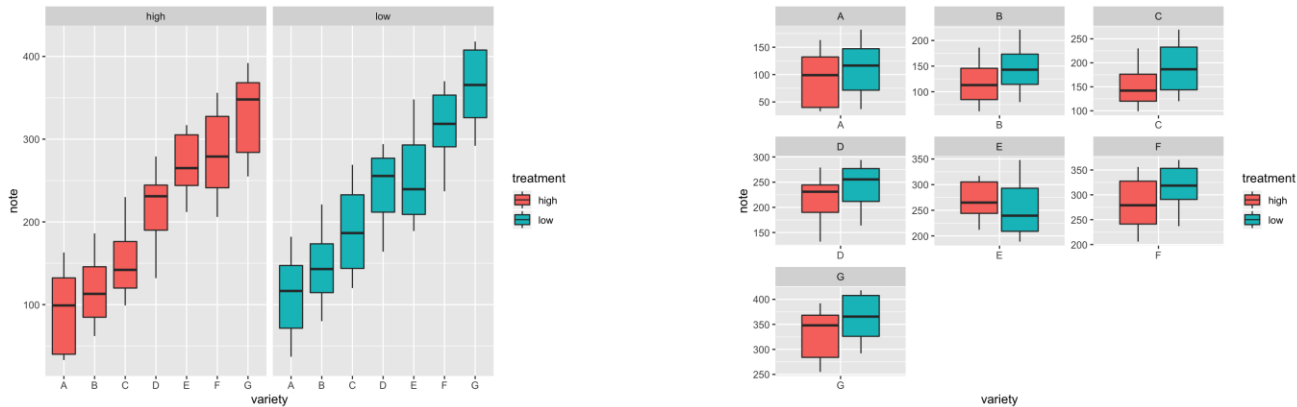
```
t=qt((1-alpha)/2 + .5, length(vec)-1)    # tend to 1.96 if  
sample size is big enough
```

```
CI=t*se
```

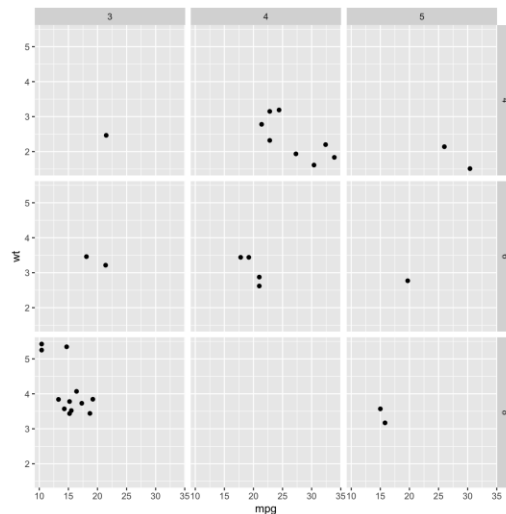
Hint: easier way to add error bars without calculation! Explore **stat_summary()**

Faceting

`facet_wrap(~var)` builds a new chart for each level of a categorical variable



`facet_grid(var1~var2)` builds one chart for each combinations of 2 categorical variables



**Let's
explore
practically**



**Let's
explore
practically**



Address the tasks in breakout rooms!

Useful references

- [R Graph Gallery](#) – Examples of the kinds of graphs and plots possible in R (+ the code to create them)
- [The Grammar of Graphics \(Leland Wilkinson\)](#) – Theory of graphical visualisation (SpringerLink book – free access through Uni of Oxford Bodleian subscription)
- [ggplot2 reference](#) – Help and cheatsheet
- [ggplot2 extensions](#) – Additional packages that extend ggplot2 functionality
- [From Data to Viz](#) – Decision tree to choose an appropriate chart type for your data