Filtering and Normalization of Transcriptomic Data

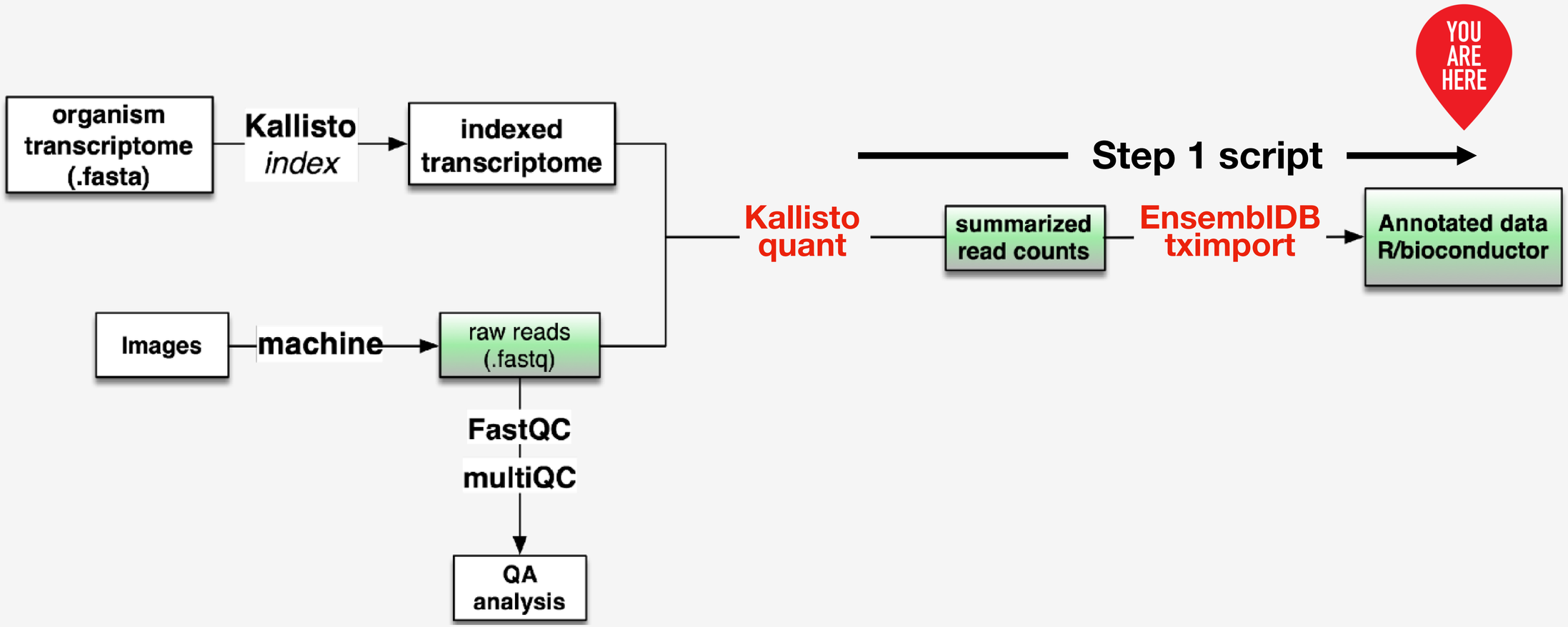
Understand the concept of a layered ‘grammar of graphics’ and how to use ggplot2

Discuss basics of ‘tidy’ vs messy data and the tidyr package

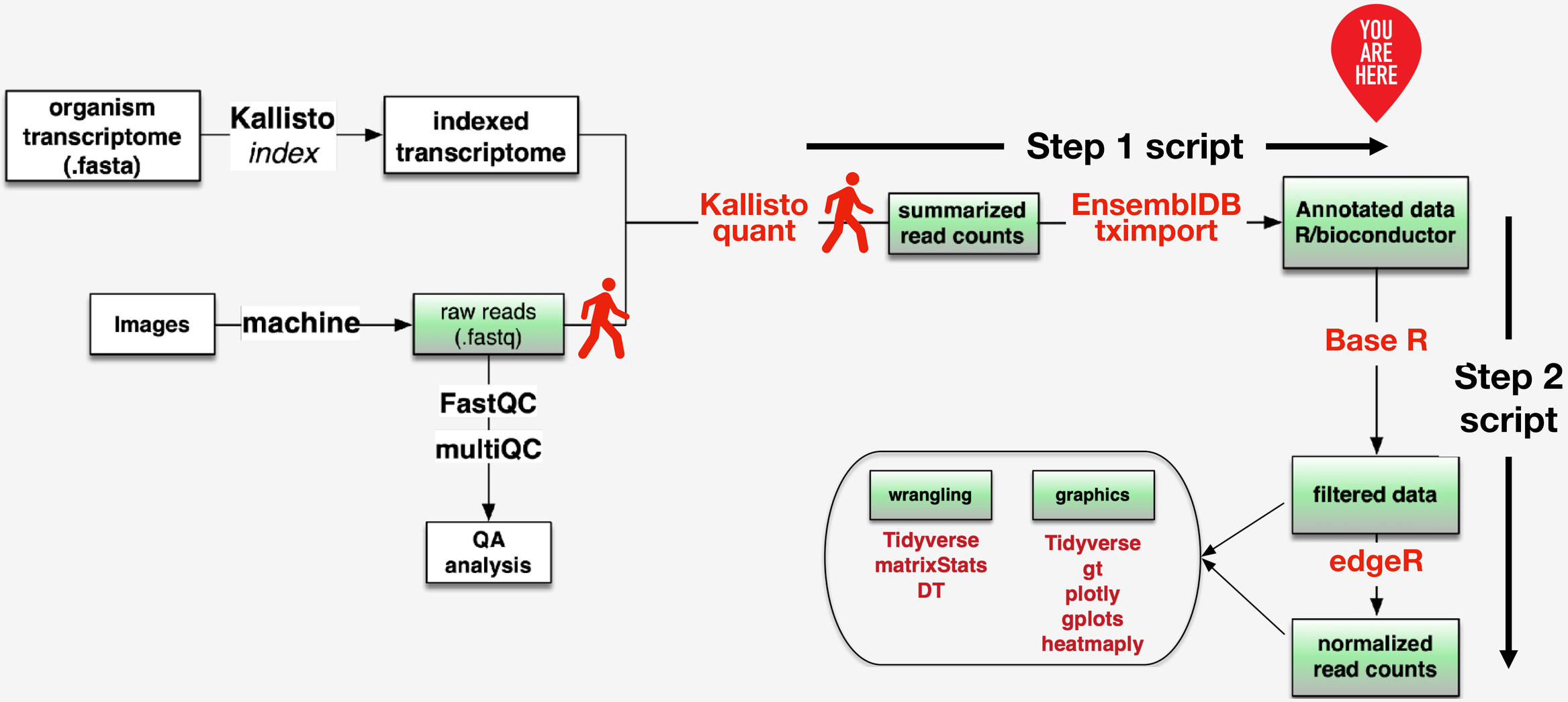
Filter data to remove lowly expressed genes

Normalize data (to allow between-sample comparisons)

Tracking our progress in this course



Tracking our progress in this course



Point at code

This is a challenging lecture

RNAseq

Filtering

Normalization

Data science

Tidy data

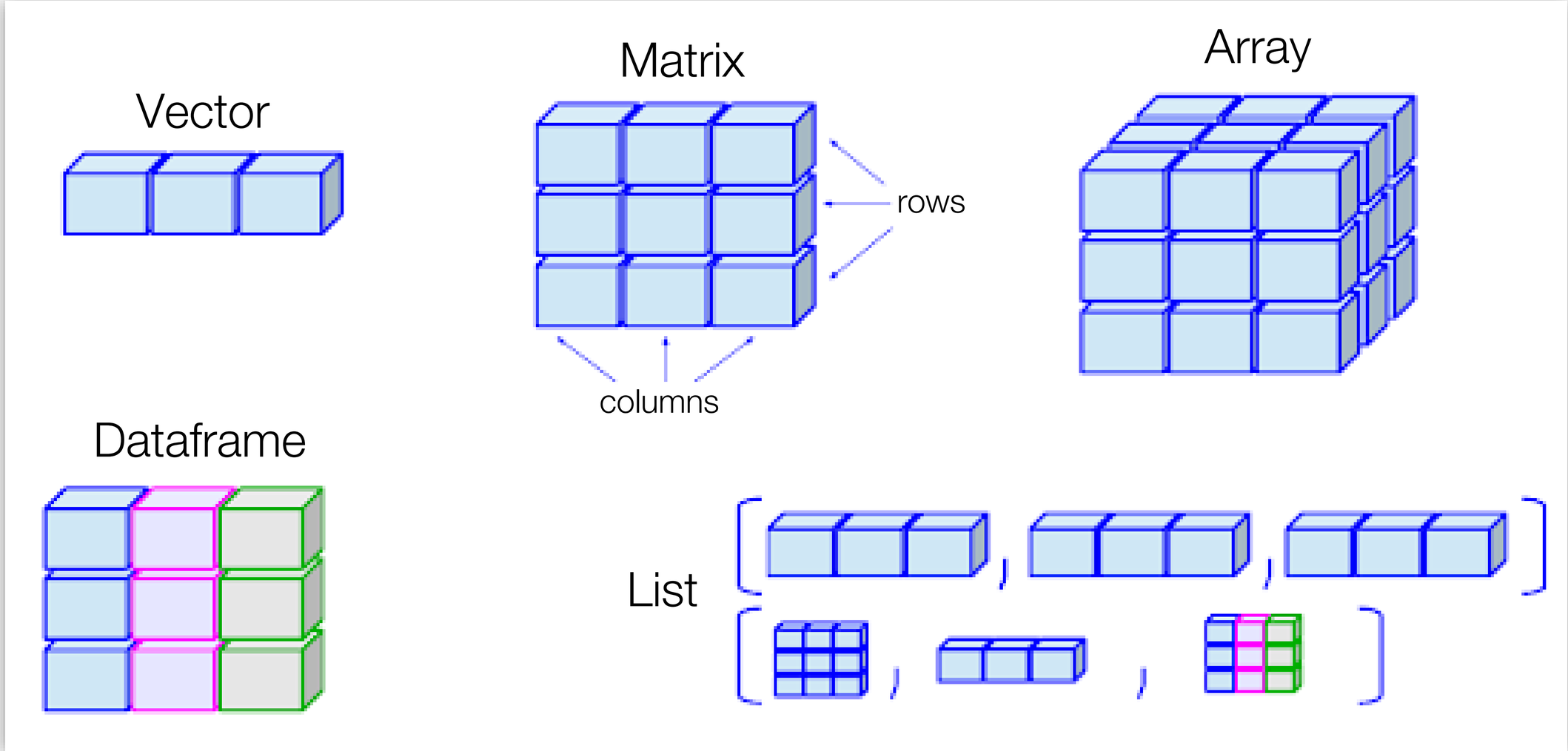
Plotting

Subsetting data

Open the Step 1 script

review ‘the essentials’ chunk

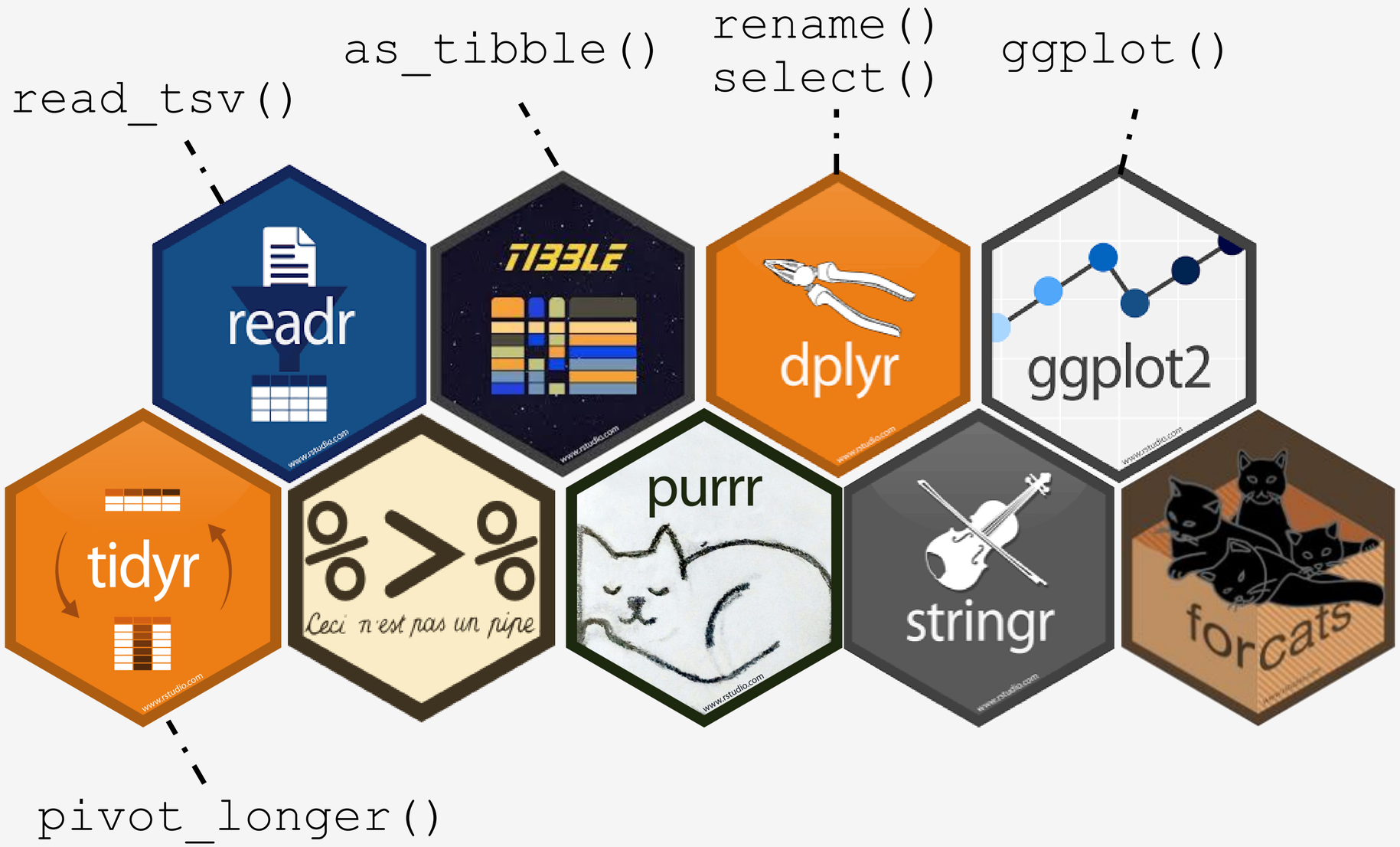
Data types in R



The Tidyverse

```
> library(tidyverse)
── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
✓ ggplot2 3.3.0     ✓ purrr   0.3.4
✓ tibble  3.0.1     ✓ dplyr   0.8.5
✓ tidyr   1.0.2     ✓ stringr 1.4.0
✓ readr   1.3.1     ✓ forcats 0.5.0
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
x dplyr::select() masks biomaRt::select()
```

The Tidyverse

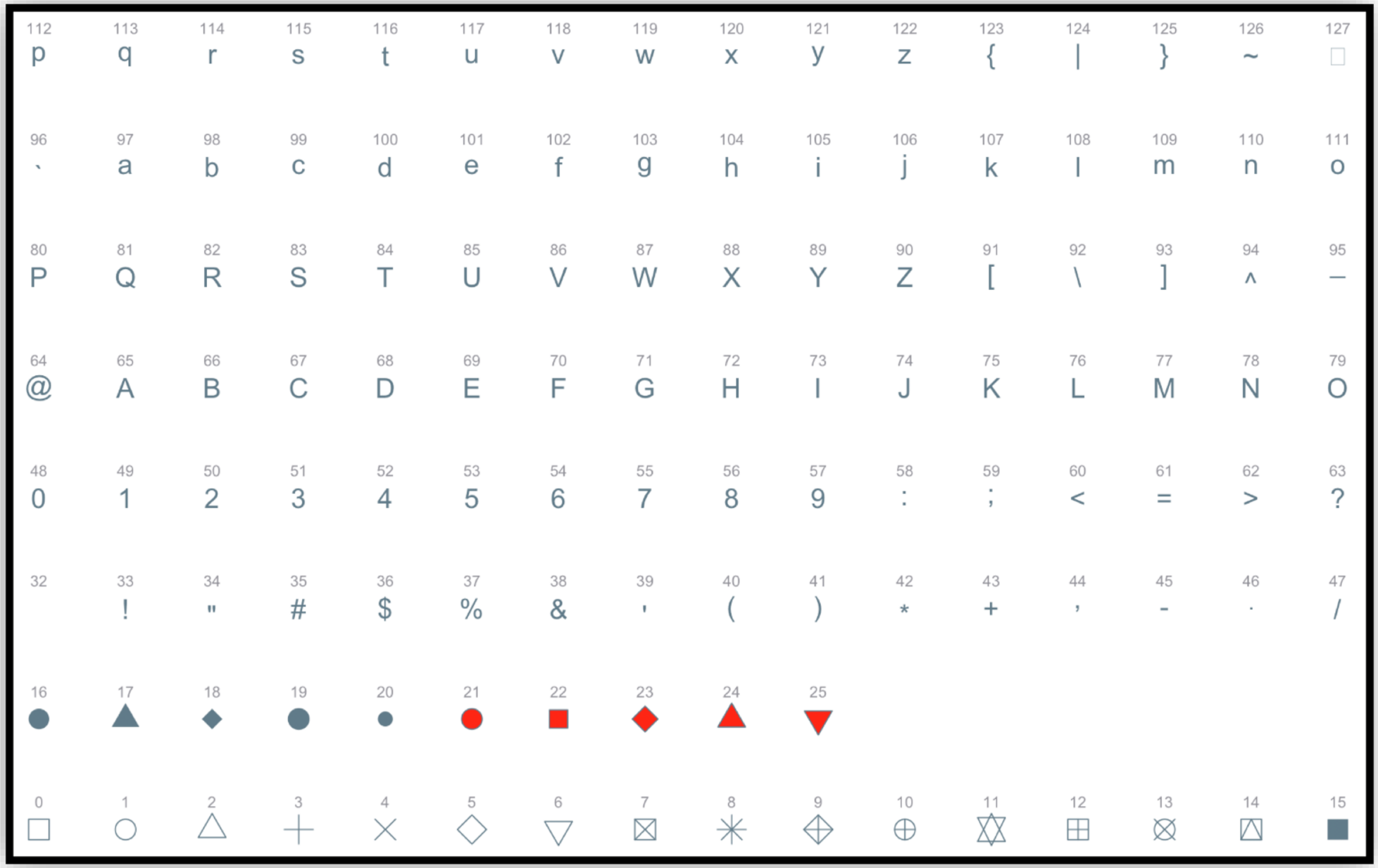


Hadley Wickham
Chief Scientist, RStudio

Open the Step 2 script

install packages, browse Txi lists

ggplot2 - shape parameter



ggplot2 themes

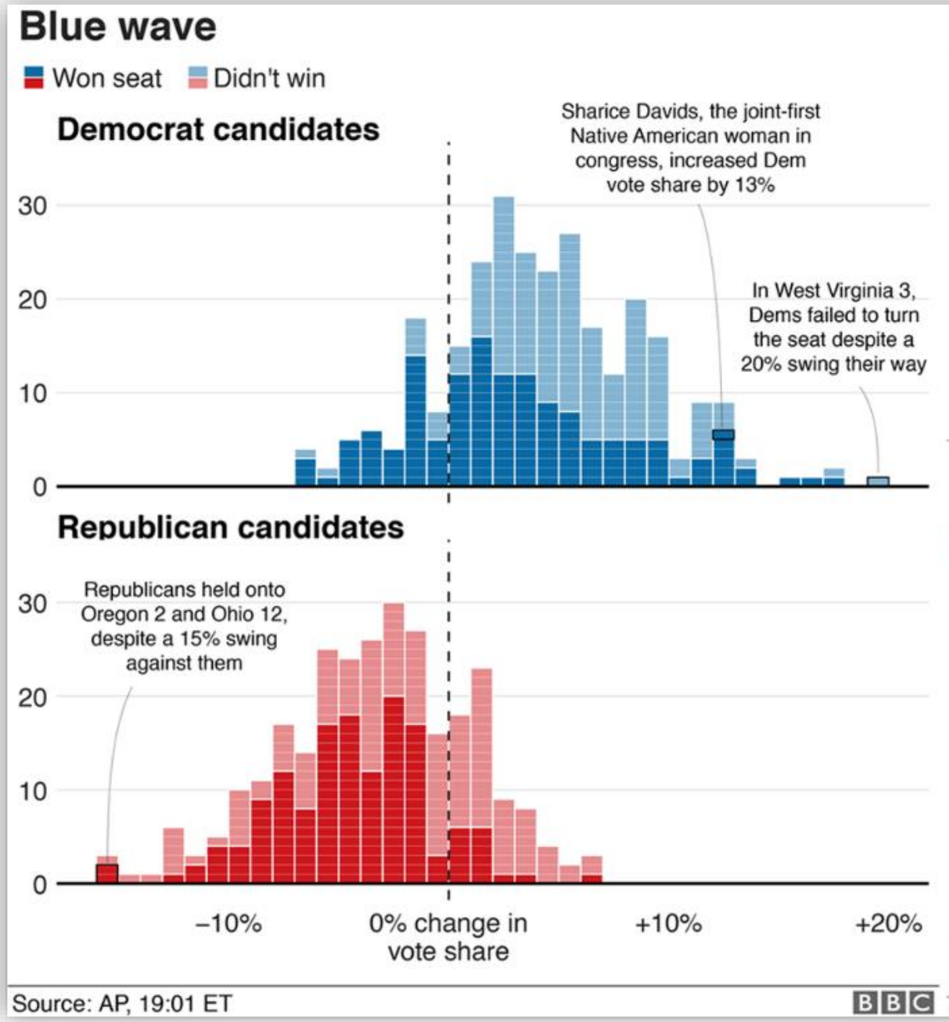
```
theme(
  line,
  rect,
  text,
  title,
  aspect.ratio,
  axis.title,
  axis.title.x,
  axis.title.x.top,
  axis.title.x.bottom,
  axis.title.y,
  axis.title.y.left,
  axis.title.y.right,
  axis.text,
  axis.text.x,
  axis.text.x.top,
  axis.text.x.bottom,
  axis.text.y,
  axis.text.y.left,
  axis.text.y.right,
  axis.ticks,
  axis.ticks.x,
  axis.ticks.x.top,
  axis.ticks.x.bottom,
  axis.ticks.y,
  axis.ticks.y.left,
  axis.ticks.y.right,
  axis.ticks.length,
  axis.ticks.length.x,
  axis.ticks.length.x.top,
  axis.ticks.length.x.bottom,
  axis.ticks.length.y,
  axis.ticks.length.y.left,
  axis.ticks.length.y.right,
  axis.line,
  axis.line.x,
  axis.line.x.top,
  axis.line.x.bottom,
  axis.line.y,
  axis.line.y.left,
  axis.line.y.right,
  legend.background,
  legend.margin,
  legend.spacing,
  legend.spacing.x,
  legend.spacing.y,
  legend.key,
  legend.key.size,
  legend.key.height,
  legend.key.width,
  legend.text,
  legend.text.align,
  legend.title,
  legend.title.align,
  legend.position,
  legend.direction,
  legend.justification,
  legend.box,
  legend.box.just,
  legend.box.margin,
  legend.box.background,
  legend.box.spacing,
  panel.background,
  panel.border,
  panel.spacing,
  panel.spacing.x,
  panel.spacing.y,
  panel.grid,
  panel.grid.major,
  panel.grid.minor,
  panel.grid.major.x,
  panel.grid.major.y,
  panel.grid.minor.x,
  panel.grid.minor.y,
  panel.ontop,
  plot.background,
  plot.title,
  plot.title.position,
  plot.subtitle,
  plot.caption,
  plot.caption.position,
  plot.tag,
  plot.tag.position,
  plot.margin,
  strip.background,
  strip.background.x,
  strip.background.y,
  strip.placement,
  strip.text,
  strip.text.x,
  strip.text.y,
  strip.switch.pad.grid,
  strip.switch.pad.wrap,
  ...,
  complete = FALSE,
  validate = TRUE
)
```

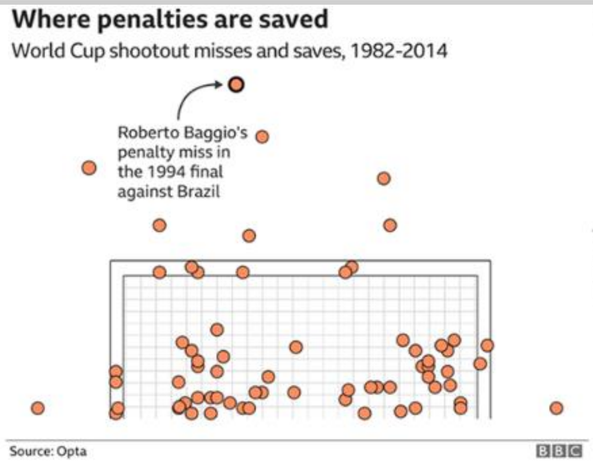
Complete themes

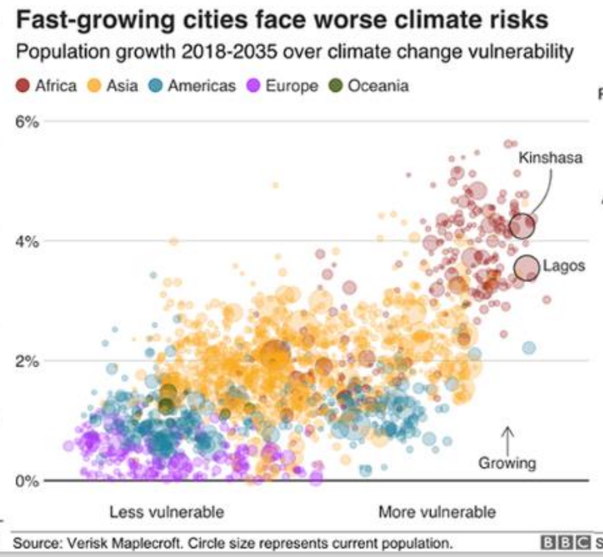
- theme_gray()
- theme_bw()
- theme_light()
- theme_dark()
- theme_minimal()
- theme_classic()
- theme_void()

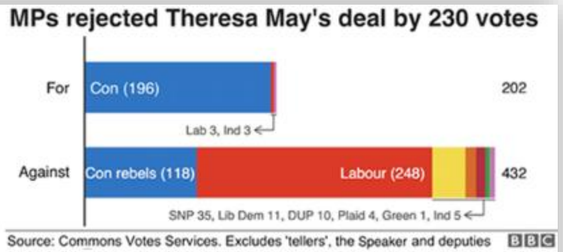
You are *not* restricted to ggplot's 'complete themes'

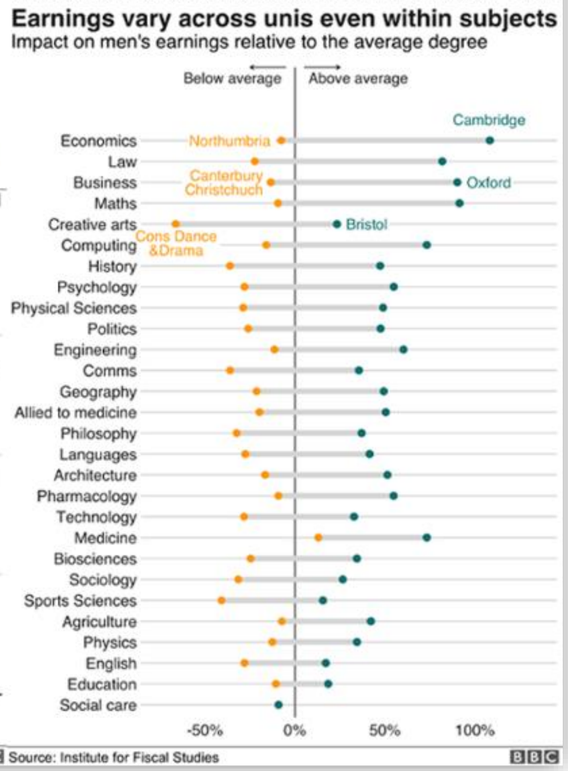
BBC theme





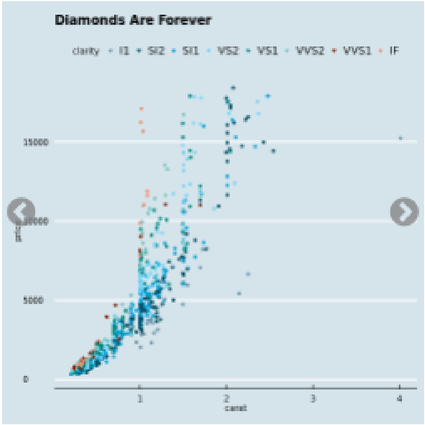






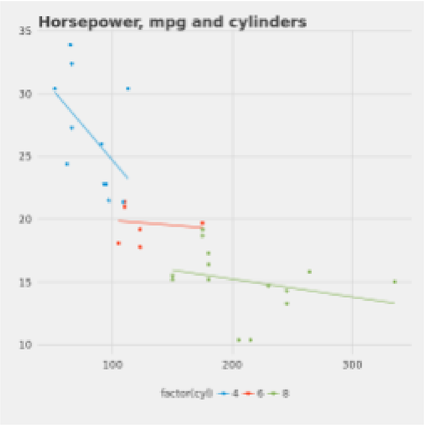
bbplot R package

ggthemes package



theme_economist

ggplot color theme based on the Economist

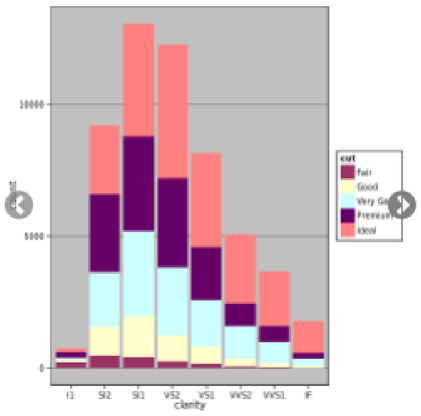


theme_fivethirtyeight

Theme inspired by fivethirtyeight.com plots

theme_wsj

Wall Street Journal theme



theme_excel

ggplot color theme based on old Excel plots

ggthemes R package (available on CRAN)

hrbrthemes package

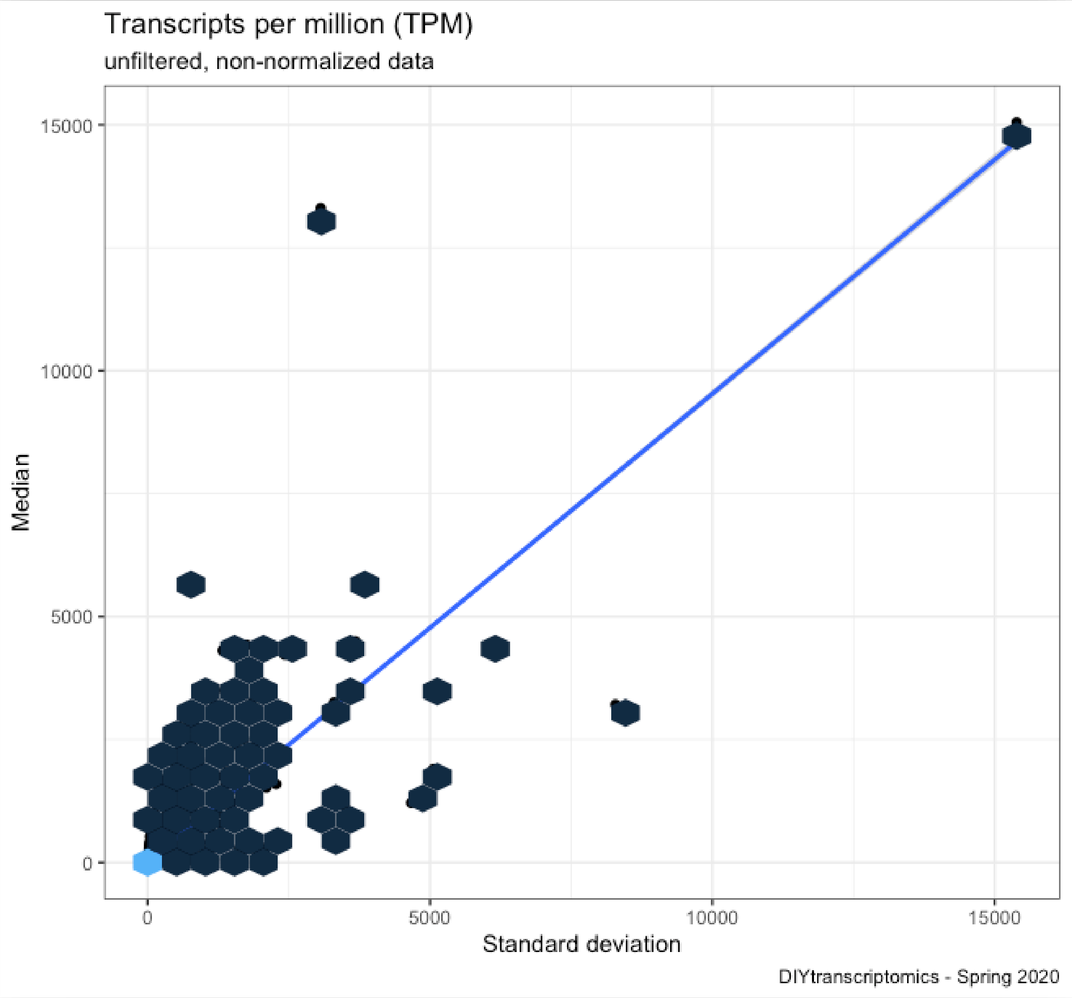
There are nigh countless tomes written about the colors you should consider using in charts as well as how to produce “minimal” charts (i.e. eliminate “chart junk”). One area that is rarely discussed is the use of fonts in charts (i.e. chart typography). The `hrbrthemes` package is *very opinionated* about chart typography:

To use a gross oversimplification, there are two basic font types: Serif and Sans-serif (that’s a lie of convenience, there are more than two). `hrbrthemes` suggests using sans-serif fonts since they have a more “modern” feel to them, you’re not producing long-form text and labels on charts may need to scale down to small sizes. Typography nerds on either side of the serif vs sans-serif debate can point to 100+ years of research-based evidence supporting their particular “side”.

hrbrthemes R package (available on CRAN)

ggplots are constructed in layers

```
ggplot(myTPM.stats) +
  aes(x = SD, y = MED) +
  geom_point(shape=16, size=2) +
  geom_smooth(method=lm) +
  geom_hex(show.legend = FALSE) +
  labs(y="Median", x = "Standard deviation",
       title="Transcript per million (TPM)",
       subtitle="unfiltered, non-normalized data",
       caption="DIYtranscriptomics - Spring 2020") +
  theme_classic() +
  theme_dark() +
  theme_bw()
```



Advantages of a layered grammar of graphics

- Users can iteratively update a plot, changing specific elements as needed
- Even high-level aspects of a plot can be changed
- Layers allow plots to be overplayed (e.g. box plot with points shown)
- Layers also make it easy to use # to toggle layers on/off
- Default arguments for each layer minimize the need for users to dictate details
- For software developer, easy to develop new functionality (e.g. stat transformation)

Tidy data philosophy

“A dataset is messy or tidy depending on how rows, columns and tables are matched up with observations, variables and types”

1. Each variable forms a column
2. Each observation forms a row
3. Each type of observational unit forms a table

“All happy families are alike; each unhappy family is unhappy in its own way”

- Leo Tolstoy, Anna Karenina

gene expression data is always ‘messy’

Gene	treatment 1	treatment 2	treatment 3
gene A	5	10	10
gene B	1000	1100	1050

messy

1. Each variable forms a column
2. Each observation forms a row
3. Each type of observational unit forms a table

gene expression data is always ‘messy’

Gene	treatment 1	treatment 2	treatment 3
gene A	5	10	10
gene B	1000	1100	1050

messy

tidyr

`pivot_longer()`

tidy

Gene	treatment	Expression
gene A	treatment 1	5
gene A	treatment 2	10
gene A	treatment 3	10
gene B	treatment 1	1000
gene B	treatment 2	1100
gene B	treatment 3	1050

gene expression data is always ‘messy’

Gene	treatment 1	treatment 2	treatment 3
gene A	5	10	10
gene B	1000	1100	1050

messy

tidyr

`pivot_wider()`

tidy

Gene	treatment	Expression
gene A	treatment 1	5
gene A	treatment 2	10
gene A	treatment 3	10
gene B	treatment 1	1000
gene B	treatment 2	1100
gene B	treatment 3	1050

Tidy data is ideal for graphing

Each variable gets its own column

Each observation gets its own row

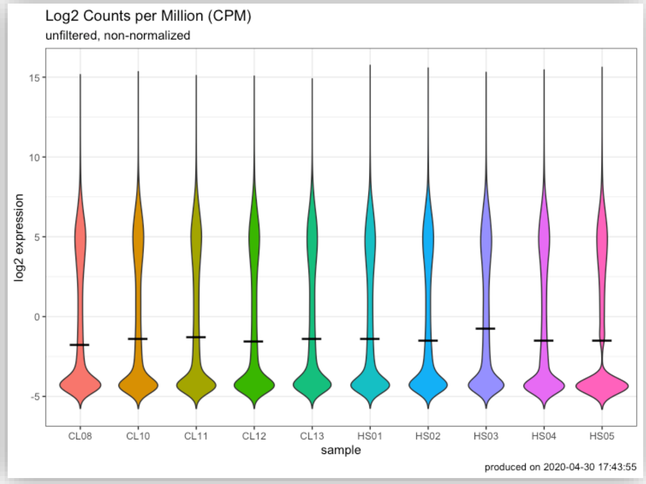
Gene	treatment	Expression
gene A	treatment 1	5
gene A	treatment 2	10
gene A	treatment 3	10
gene B	treatment 1	1000
gene B	treatment 2	1100
gene B	treatment 3	1050

Values are easily retrieved as single vector

Visualizing the Step 2 script

Raw

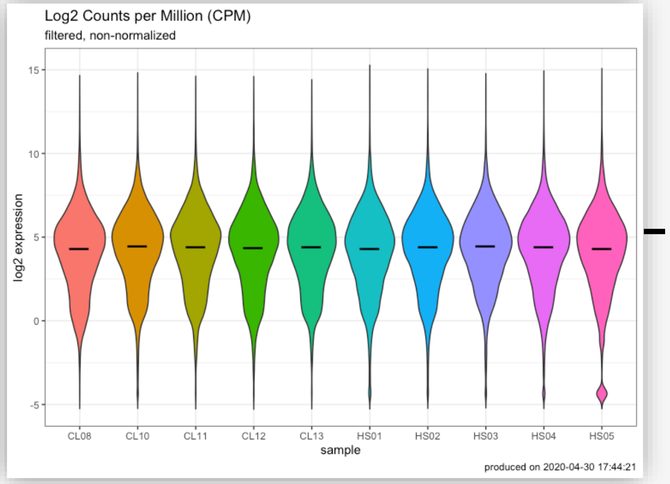
as_tibble()
pivot_longer()
ggplot()



Base R
rowSums()

Filtered

as_tibble()
pivot_longer()
ggplot()



edgeR::
calcnormfactors

Normalized

as_tibble()
pivot_longer()
ggplot()