MONASH University

Advanced Data Visualisation with R

ggplot2 internals

Instructor: *Emi Tanaka*

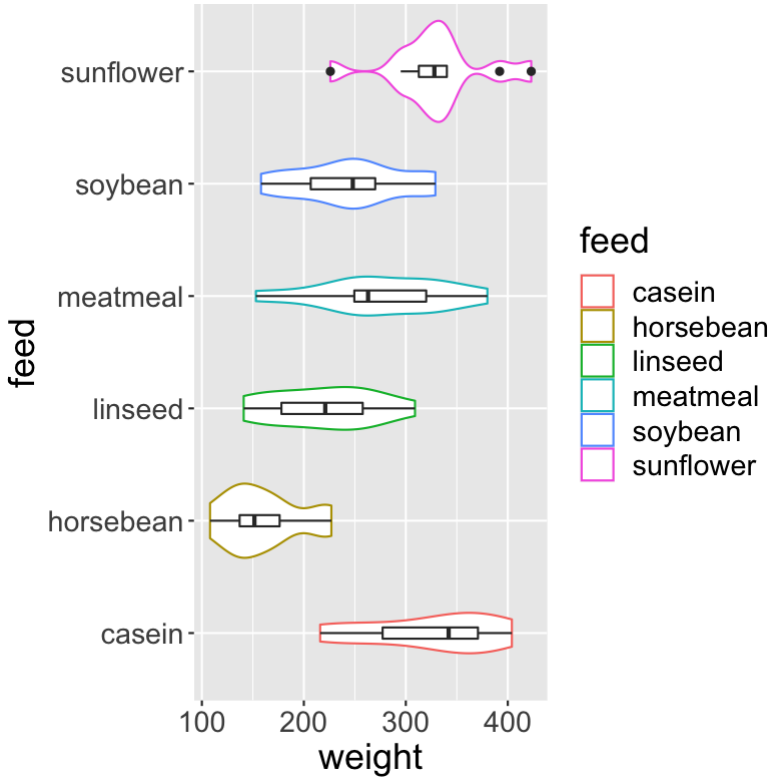
emi.tanaka@monash.edu

8th Dec 2021 @ Statistical Society of Australia Canberra Branch | Zoom

Quick overview of ggplot2

- `ggplot2` is one of the most popular packages for data visualisation
- It implements an interpretation of the **grammar of graphics** by Wilkinson

```
library(ggplot2)
ggplot(data = chickwts,
       aes(x = weight,
           y = feed)) +
  geom_violin(aes(color = feed)) +
  geom_boxplot(width = 0.1)
```

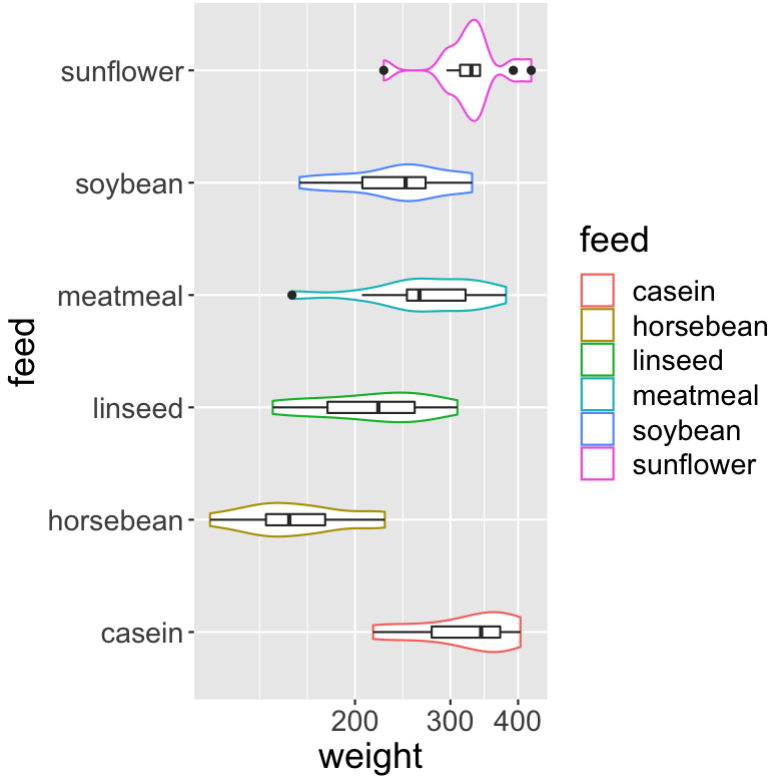


H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.
Wilkinson, Leland. 2005. The Grammar of Graphics. Springer. 2nd edition.

Quick overview of ggplot2

- `ggplot2` is one of the most popular packages for data visualisation
- It implements an interpretation of the **grammar of graphics** by Wilkinson

```
library(ggplot2)
ggplot(data = chickwts,
       aes(x = weight,
           y = feed)) +
  geom_violin(aes(color = feed)) +
  geom_boxplot(width = 0.1) +
  scale_x_continuous(trans = "log10")
```

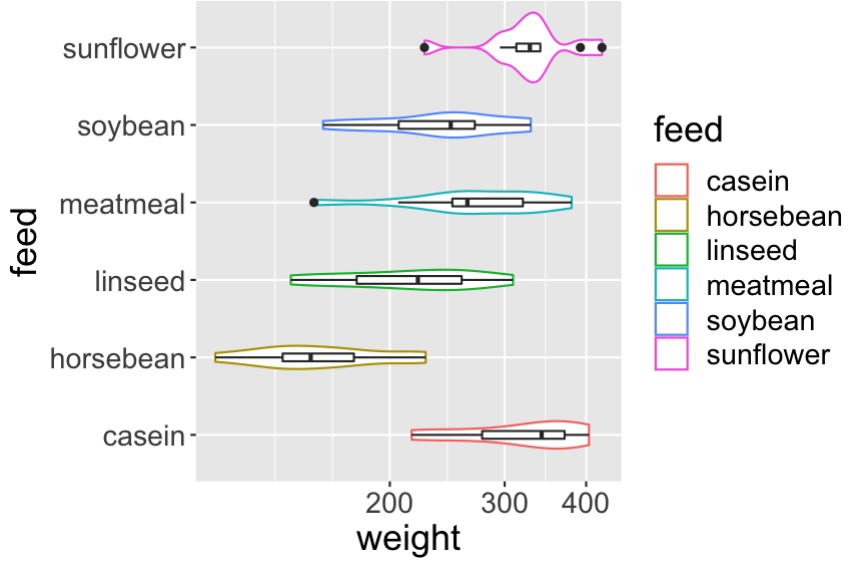


H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.
Wilkinson, Leland. 2005. The Grammar of Graphics. Springer. 2nd edition.

Underlying mechanisms for drawing ggplot objects

```
library(ggplot2)
g <- ggplot(data = chickwts, aes(x = weight, y = feed)) +
  geom_violin(aes(color = feed)) +
  geom_boxplot(width = 0.1) +
  scale_x_continuous(trans = "log10")
```

```
print(g)
```



Drawing in `ggplot2` happens when you **`print` the ggplot object**

- Essentially this involves:

```
#1> data <- ggplot_build(g)
#2> gtable <- ggplot_gtable(data)
#3> grid::grid.newpage()
    grid::grid.draw(gtable)
```

Dissecting the ggplot object

```
str(g)
```

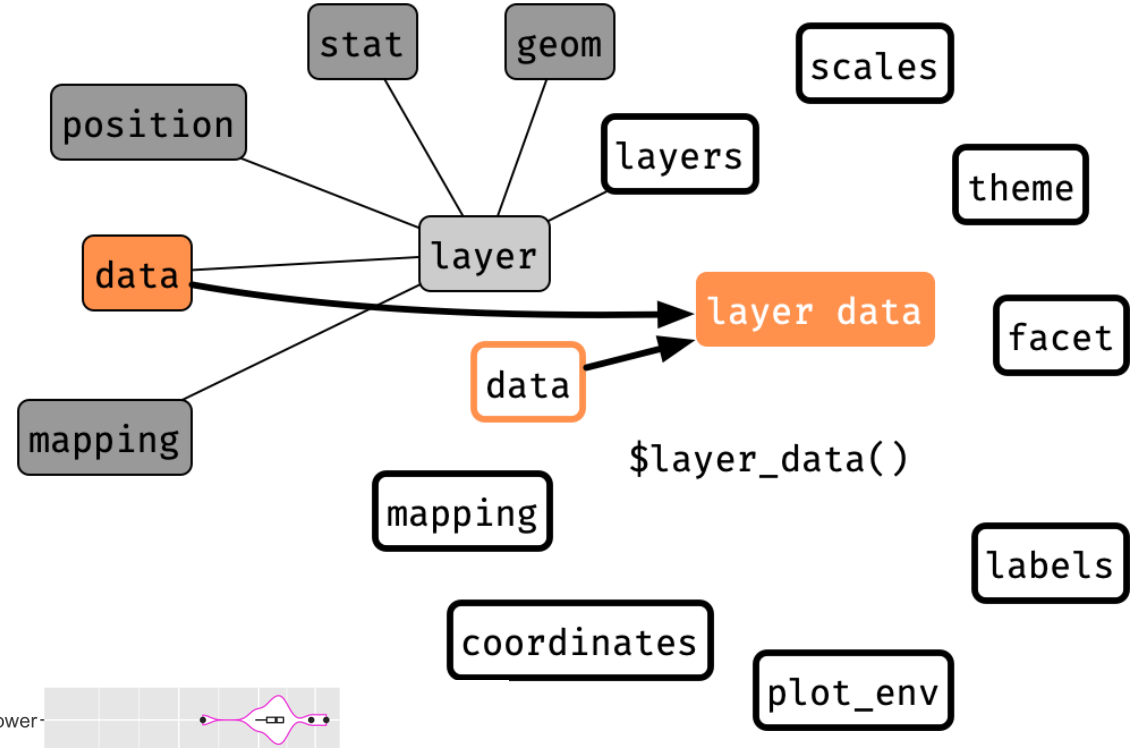
```
## List of 9
##  $ data       :'data.frame':    71 obs. of  2 variables:
##   ..$ weight: num [1:71] 179 160 136 227 217 168 108 124 143 140 ...
##   ..$ feed  : Factor w/ 6 levels "casein","horsebean",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ layers     :List of 2
##   ..$ :Classes 'LayerInstance', 'Layer', 'ggproto', 'gg' <ggproto object: Class LayerInstance,
##     aes_params: list
##     compute_aesthetics: function
##     compute_geom_1: function
```

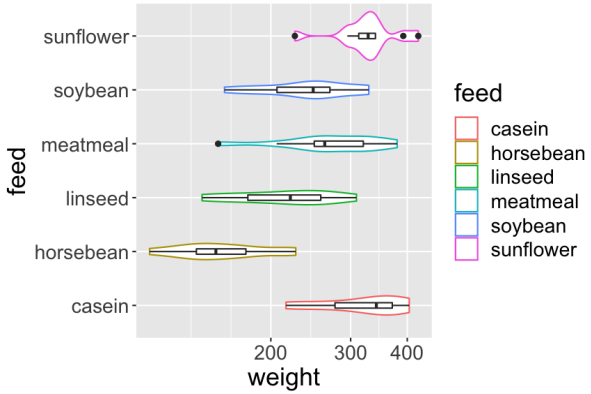
scroll ↓

- The `ggplot` object contains:
  - data
  - layers
  - scales
  - mapping
  - theme
  - coordinates
  - facet
  - plot environment
  - labels

#1> ggplot_build()

Data transformation 1 *Get the input data*





The data for each layer may be obtained:

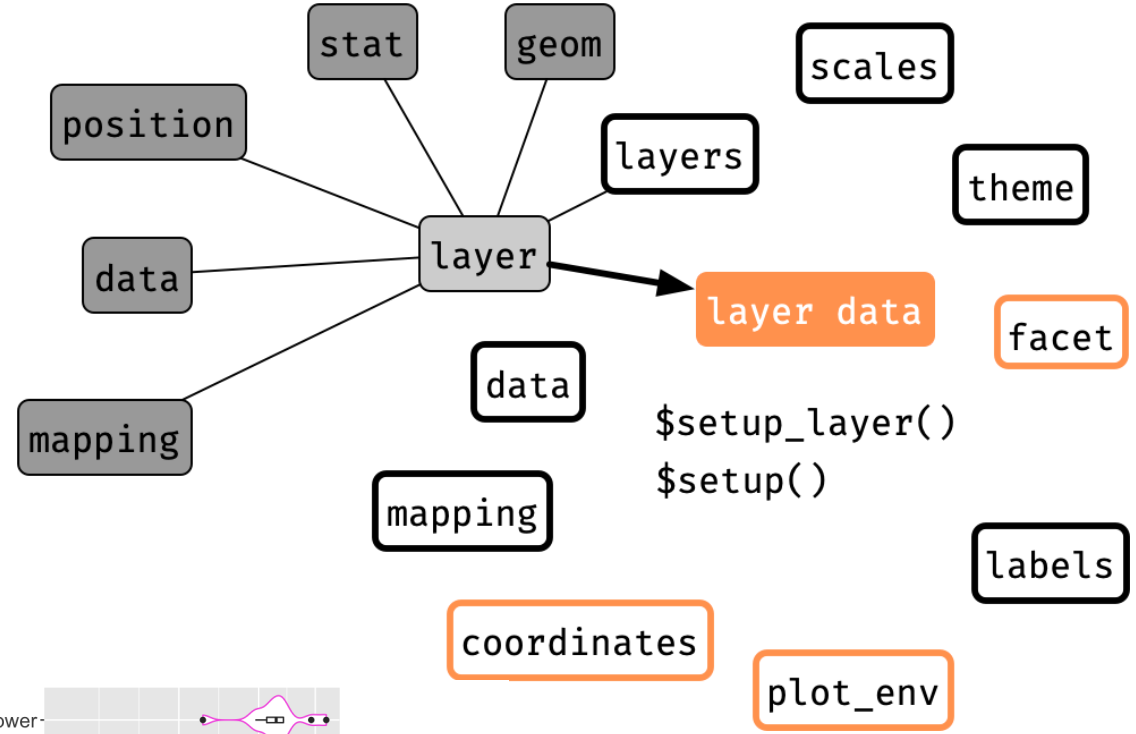
1. from the **`data` argument of the layer** where
   - `data` is a `data.frame` or
   - `data` is a function applied to the global data that returns a `data.frame`
2. by inheriting the **global data specified in `ggplot()`**

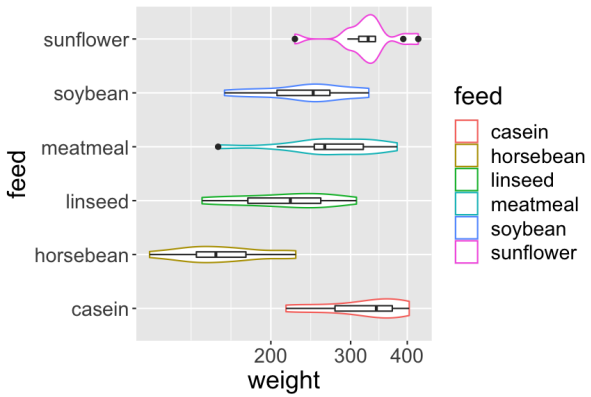
```
## [[1]]
## # A tibble: 71 × 2
##    weight feed
##     <dbl> <fct>
##  1    179 horsebean
##  2    160 horsebean
```

scroll ↓

Note: this is a rough illustration and some actual processes may differ

Data transformation 2 *Setup data*





- Adjustments made based on raw input data and plot info
- Initialise panels, add extra data for margins and missing faceting variables, and add on a PANEL variable to data

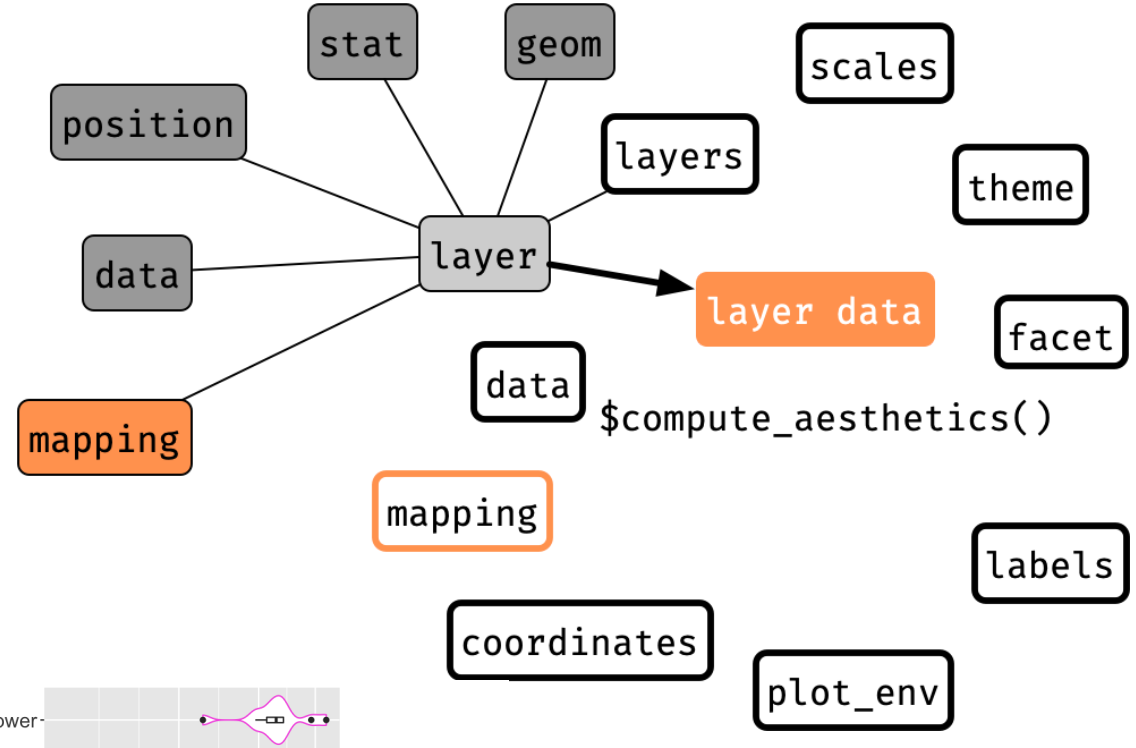
▶ Click to see difference to the previous data

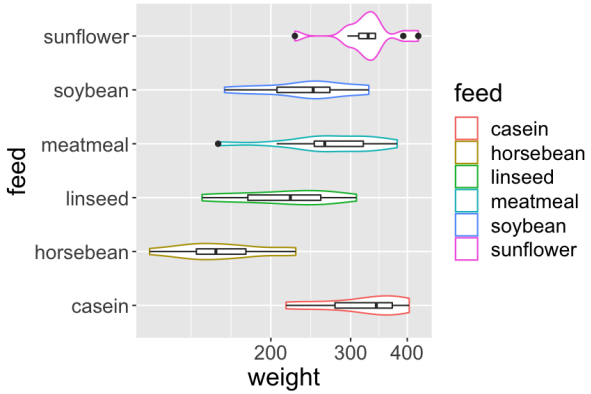
```
## [[1]]
## # A tibble: 71 × 3
##    weight feed      PANEL
##     <dbl> <fct>     <fct>
##  1    179 horsebean 1
##  2    160 horsebean 1
##  3    136 horsebean 1
##  4    227 horsebean 1
##  5    217 horsebean 1
##  6    168 horsebean 1
##  7    108 horsebean 1
```

scroll ↓

Note: this is a rough illustration and some actual processes may differ

Data transformation 3 *Compute aesthetics*





- Compute aesthetics to produce data with aesthetics variables
- Variables not specified in mapping are dropped from the data

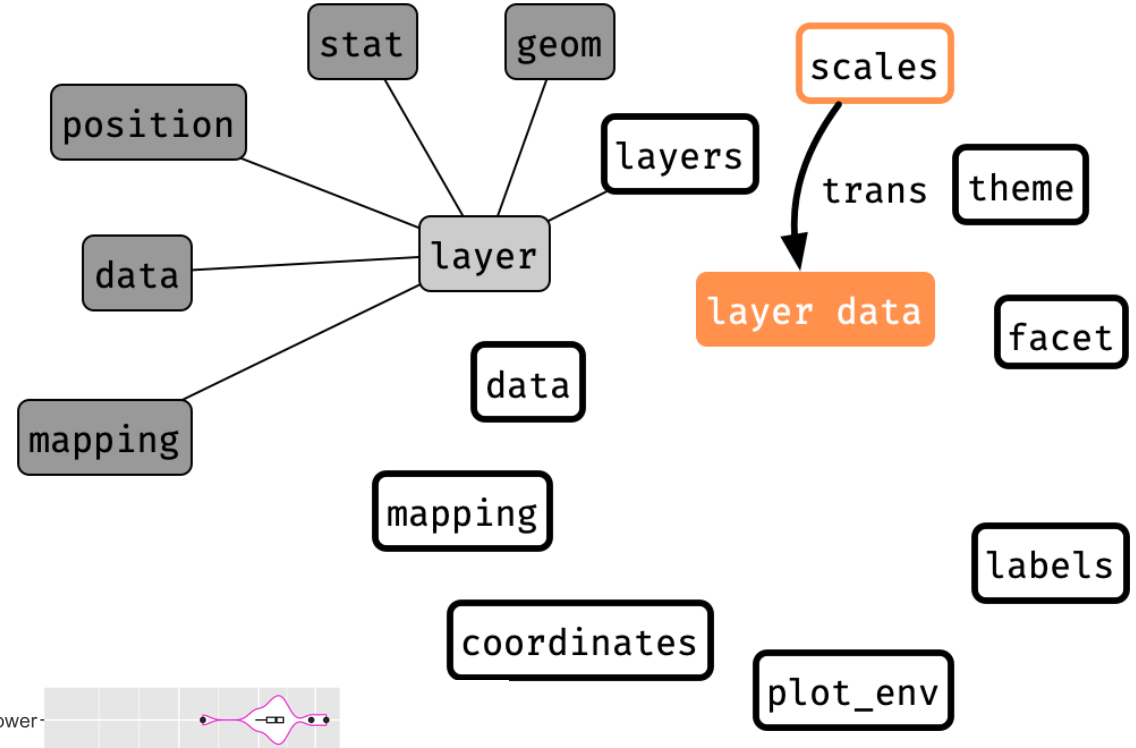
▶ Click to see difference to the previous data

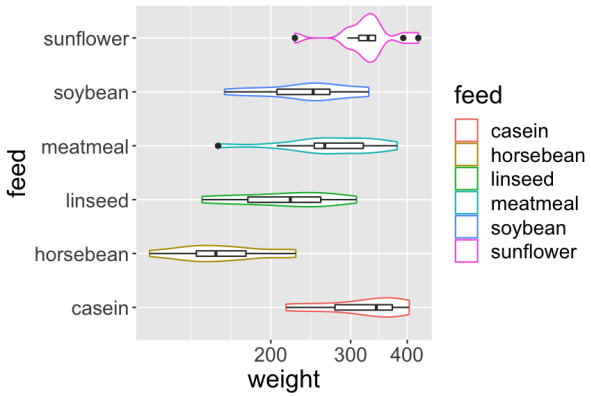
```
## [[1]]
## # A tibble: 71 × 5
##    colour        x y         PANEL group
##    <fct>     <dbl> <fct>     <fct> <int>
##  1 horsebean   179 horsebean 1         2
##  2 horsebean   160 horsebean 1         2
##  3 horsebean   136 horsebean 1         2
##  4 horsebean   227 horsebean 1         2
##  5 horsebean   217 horsebean 1         2
##  6 horsebean   168 horsebean 1         2
##  7 horsebean   108 horsebean 1         2
```

scroll ↓

Note: this is a rough illustration and some actual processes may differ

Data transformation 4 *Scale transformation*





- The `trans` argument in `scale_*` functions are applied to data here
- Subsequent rendering are based on this transformed space

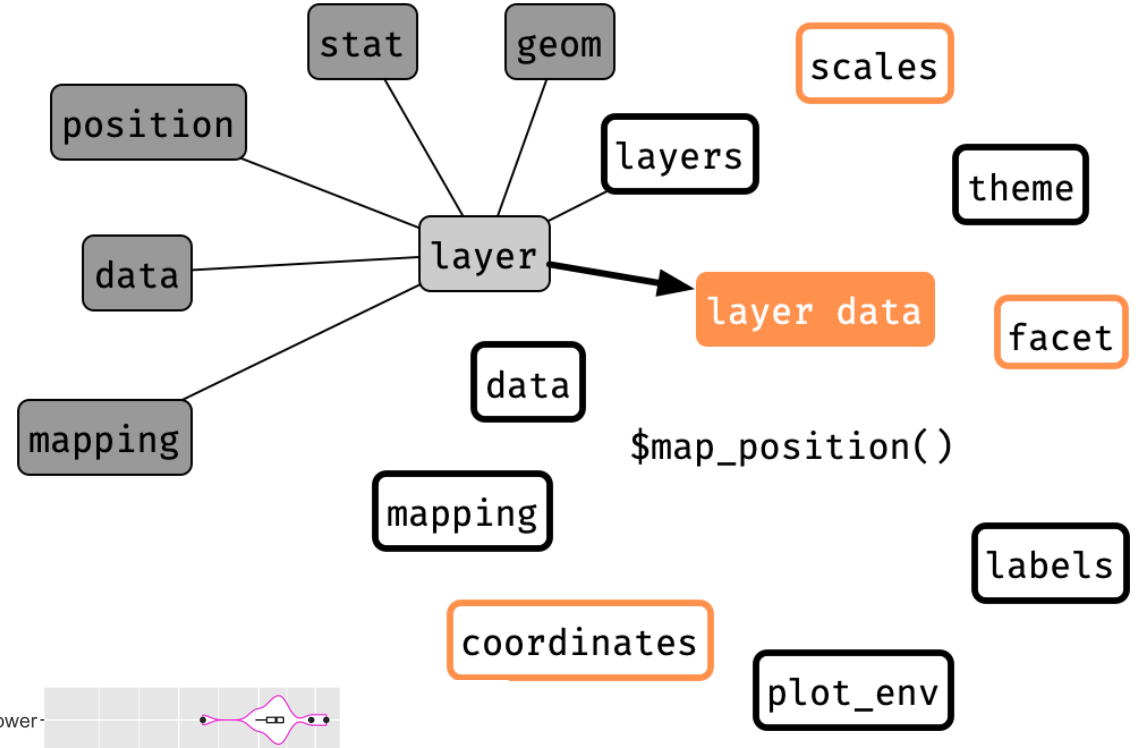
▶ Click to see difference to the previous data

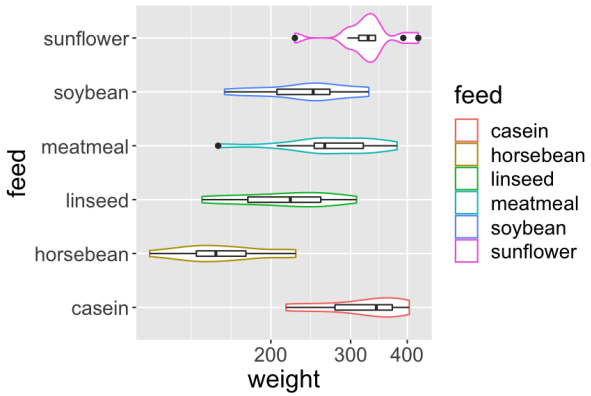
```
## [[1]]
## # A tibble: 71 × 5
##       x colour    y         PANEL group
##   <dbl> <fct>     <fct>     <fct> <int>
##  1  2.25 horsebean horsebean 1         2
##  2  2.20 horsebean horsebean 1         2
##  3  2.13 horsebean horsebean 1         2
##  4  2.36 horsebean horsebean 1         2
##  5  2.34 horsebean horsebean 1         2
##  6  2.23 horsebean horsebean 1         2
##  7  2.03 horsebean horsebean 1         2
```

scroll ↓

Note: this is a rough illustration and some actual processes may differ

Data transformation 5 *Map the position aesthetics*





- Map the position aesthetics using position scales
- These are often the x and y variables

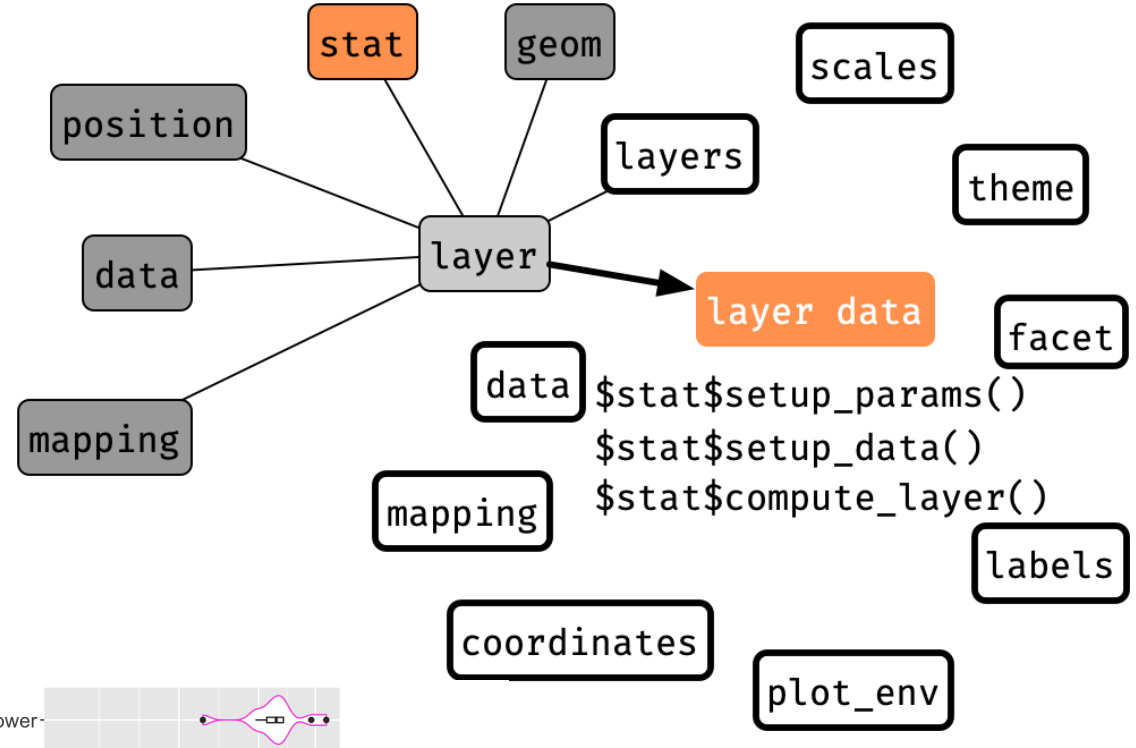
▶ Click to see difference to the previous data

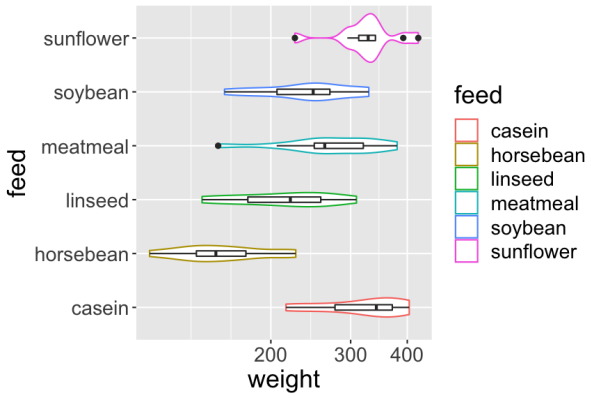
```
## [[1]]
## # A tibble: 71 × 5
##       x colour    y          PANEL group
##   <dbl> <fct>     <mppd_dsc> <fct> <int>
##  1  2.25 horsebean 2          1         2
##  2  2.20 horsebean 2          1         2
##  3  2.13 horsebean 2          1         2
##  4  2.36 horsebean 2          1         2
##  5  2.34 horsebean 2          1         2
##  6  2.23 horsebean 2          1         2
##  7  2.03 horsebean 2          1         2
```

scroll ↓

Note: this is a rough illustration and some actual processes may differ

Data transformation 6 *Compute and map statistics*





- Compute and map statistics
- Here for the:
  - violin plot: the density among other statistics are calculated,
  - boxplot: the five number summary among other statistics are calculated.

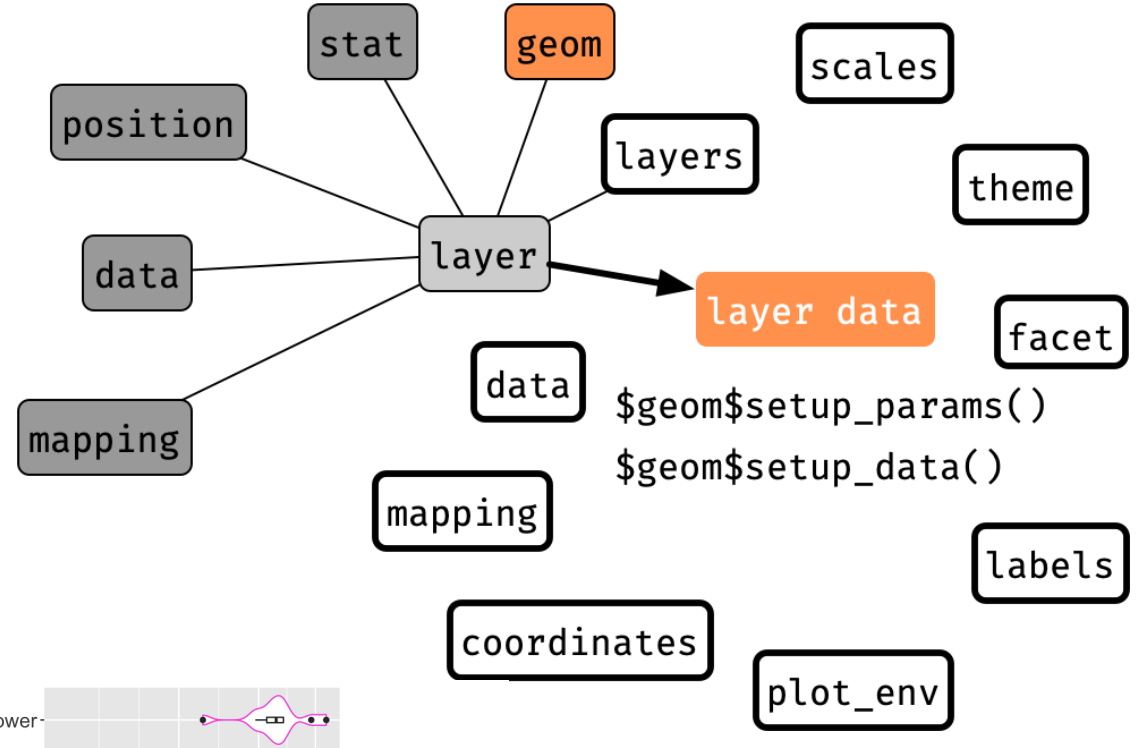
▶ Click to see difference to the previous data

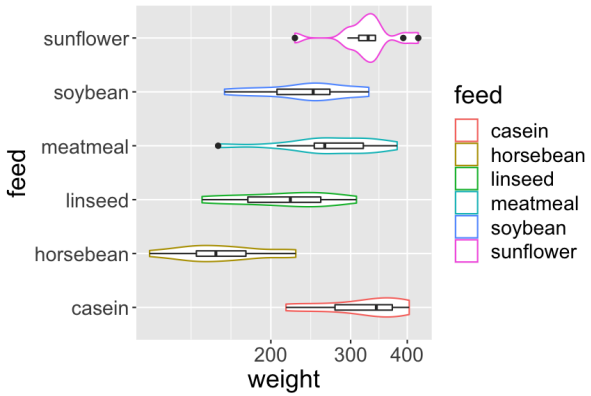
```
## [[1]]
## # A tibble: 3,072 × 12
##        y density scaled ndensity count     n
##    <dbl>   <dbl>  <dbl>    <dbl> <dbl> <int> <
##  1     1    1.52  0.345    0.345  18.3    12
##  2     1    1.53  0.346    0.346  18.3    12
##  3     1    1.53  0.347    0.347  18.4    12
##  4     1    1.54  0.348    0.348  18.5    12
##  5     1    1.54  0.350    0.350  18.5    12
##  6     1    1.55  0.351    0.351  18.6    12
##  7     1    1.55  0.352    0.352  18.7    12
```

scroll ↓

Note: this is a rough illustration and some actual processes may differ

Data transformation 7 *Reparametrise variables based on geom*





- Reparametrise variables based on the `geom`
- This may include calculating `width`, `ymin`, `ymax` and so on

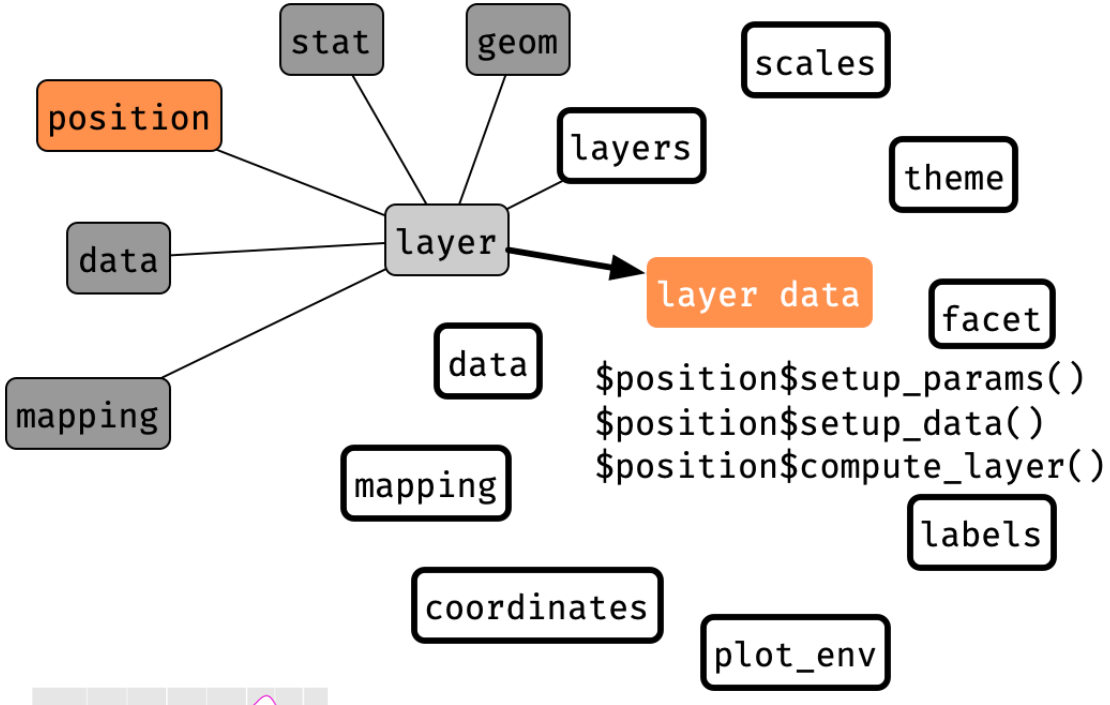
▶ Click to see difference to the previous data

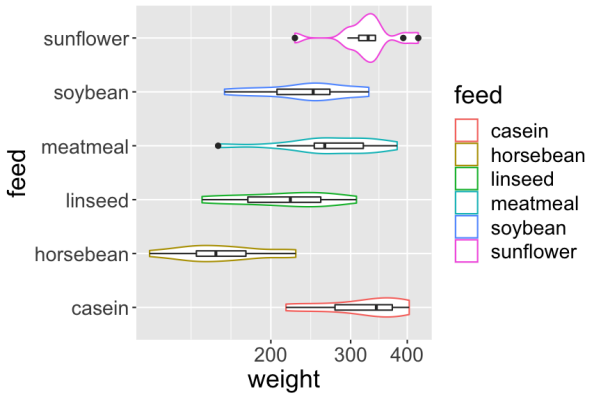
```
## [[1]]
## # A tibble: 3,072 × 15
##        y density scaled ndensity count     n
##    <dbl>   <dbl>  <dbl>    <dbl> <dbl> <int> <
##  1     1    1.52  0.345    0.345  18.3    12
##  2     1    1.53  0.346    0.346  18.3    12
##  3     1    1.53  0.347    0.347  18.4    12
##  4     1    1.54  0.348    0.348  18.5    12
##  5     1    1.54  0.350    0.350  18.5    12
##  6     1    1.55  0.351    0.351  18.6    12
##  7     1    1.55  0.352    0.352  18.7    12
```

scroll ↓

Note: this is a rough illustration and some actual processes may differ

Data transformation 8 *Compute position adjutmnets*





- Compute position adjustments

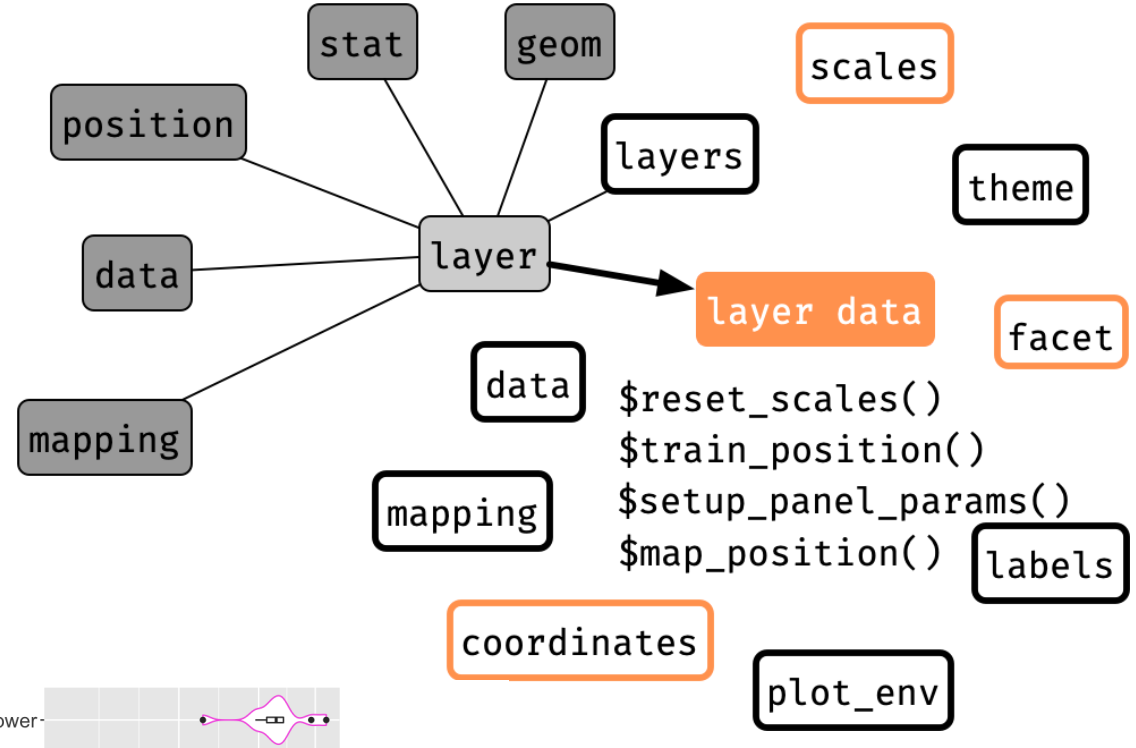
▶ Click to see difference to the previous data

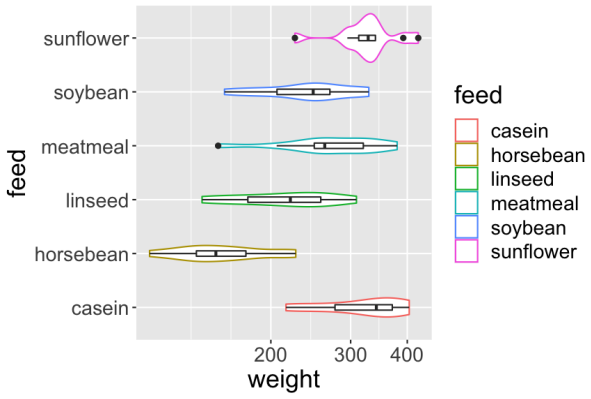
```
## [[1]]
## # A tibble: 3,072 × 16
##        y density scaled ndensity count     n
##    <dbl>   <dbl>  <dbl>    <dbl> <dbl> <int> <
##  1     1    1.52  0.345    0.345  18.3    12
##  2     1    1.53  0.346    0.346  18.3    12
##  3     1    1.53  0.347    0.347  18.4    12
##  4     1    1.54  0.348    0.348  18.5    12
##  5     1    1.54  0.350    0.350  18.5    12
##  6     1    1.55  0.351    0.351  18.6    12
##  7     1    1.55  0.352    0.352  18.7    12
```

scroll ↓

Note: this is a rough illustration and some actual processes may differ

Data transformation 9 *Retrain position scales*





- Reset position scales
- Re-train and map position once again.

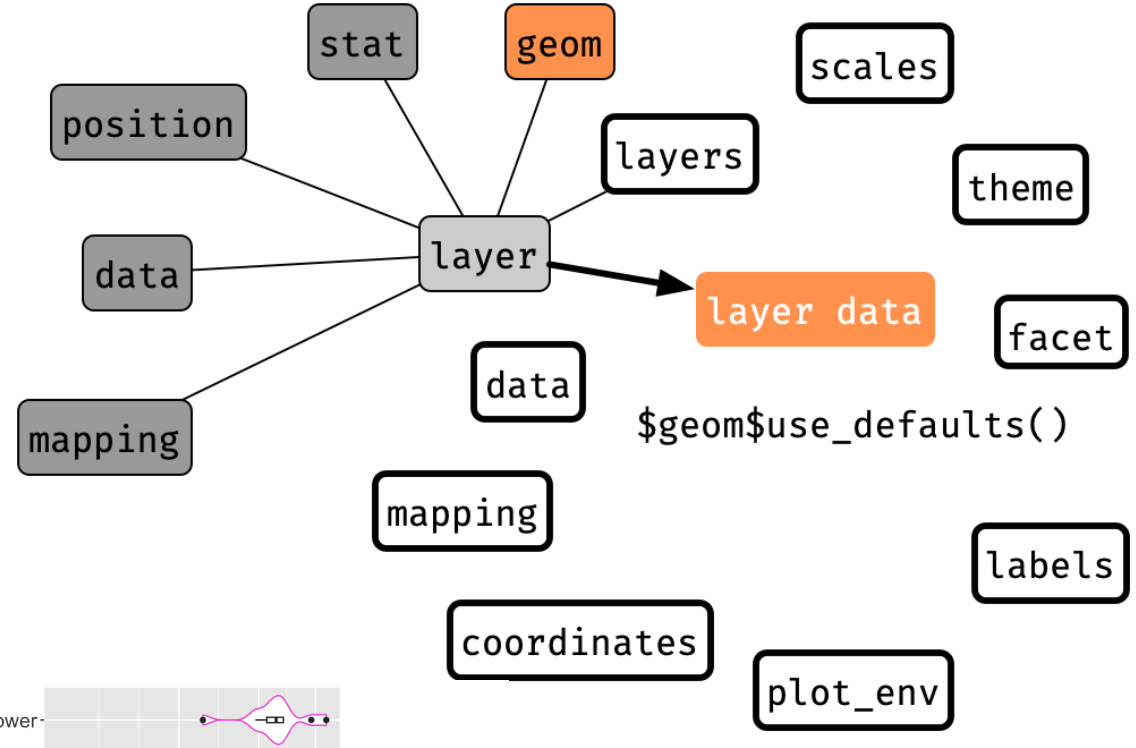
▶ Click to see difference to the previous data

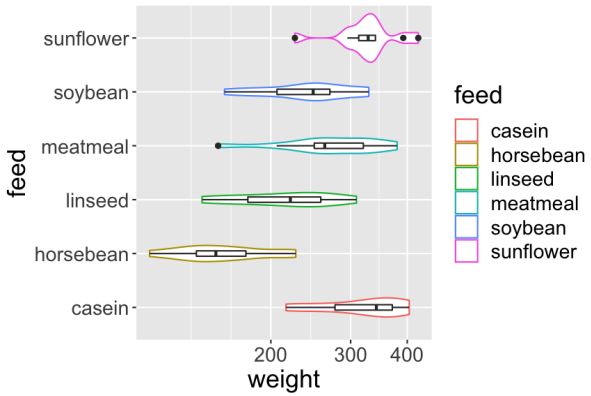
```
## [[1]]
## # A tibble: 3,072 × 16
##    y          density scaled ndensity count
##    <mppd_dsc>   <dbl>  <dbl>    <dbl> <dbl> <i
##  1 1             1.52  0.345    0.345  18.3
##  2 1             1.53  0.346    0.346  18.3
##  3 1             1.53  0.347    0.347  18.4
##  4 1             1.54  0.348    0.348  18.5
##  5 1             1.54  0.350    0.350  18.5
##  6 1             1.55  0.351    0.351  18.6
##  7 1             1.55  0.352    0.352  18.7
```

scroll ↓

Note: this is a rough illustration and some actual processes may differ

Data transformation 9 *Retrain data based on geom defaults*





- Re-compute data based on geom defaults

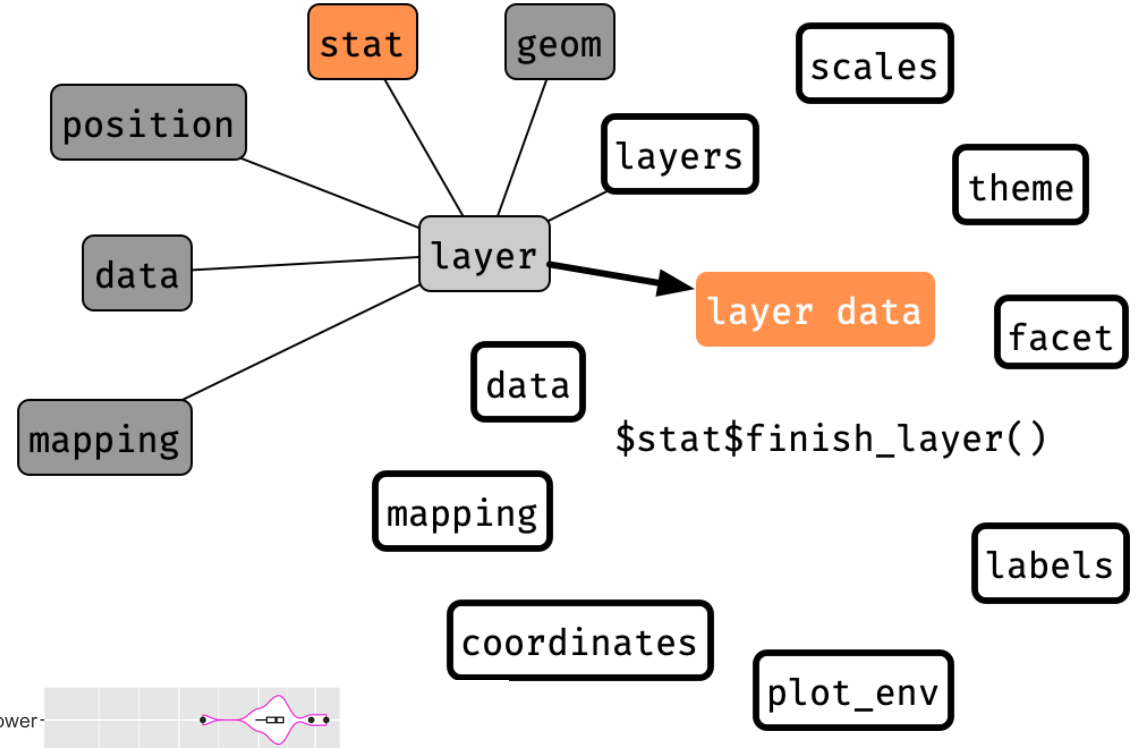
▶ Click to see difference to the previous data

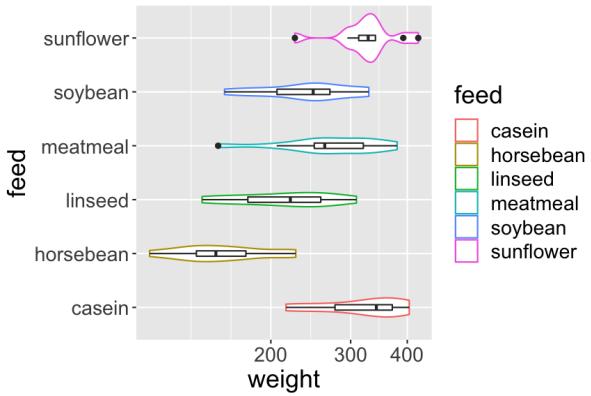
```
## [[1]]
## # A tibble: 3,072 × 21
##    colour  y          density scaled ndensity
##    <chr>   <mppd_dsc>   <dbl>  <dbl>    <dbl>
##  1 #F8766D 1             1.52  0.345    0.345
##  2 #F8766D 1             1.53  0.346    0.346
##  3 #F8766D 1             1.53  0.347    0.347
##  4 #F8766D 1             1.54  0.348    0.348
##  5 #F8766D 1             1.54  0.350    0.350
##  6 #F8766D 1             1.55  0.351    0.351
##  7 #F8766D 1             1.55  0.352    0.352
```

scroll ↓

Note: this is a rough illustration and some actual processes may differ

Data transformation 10 *One more visit to stat*





- Retrain data one more time with stat

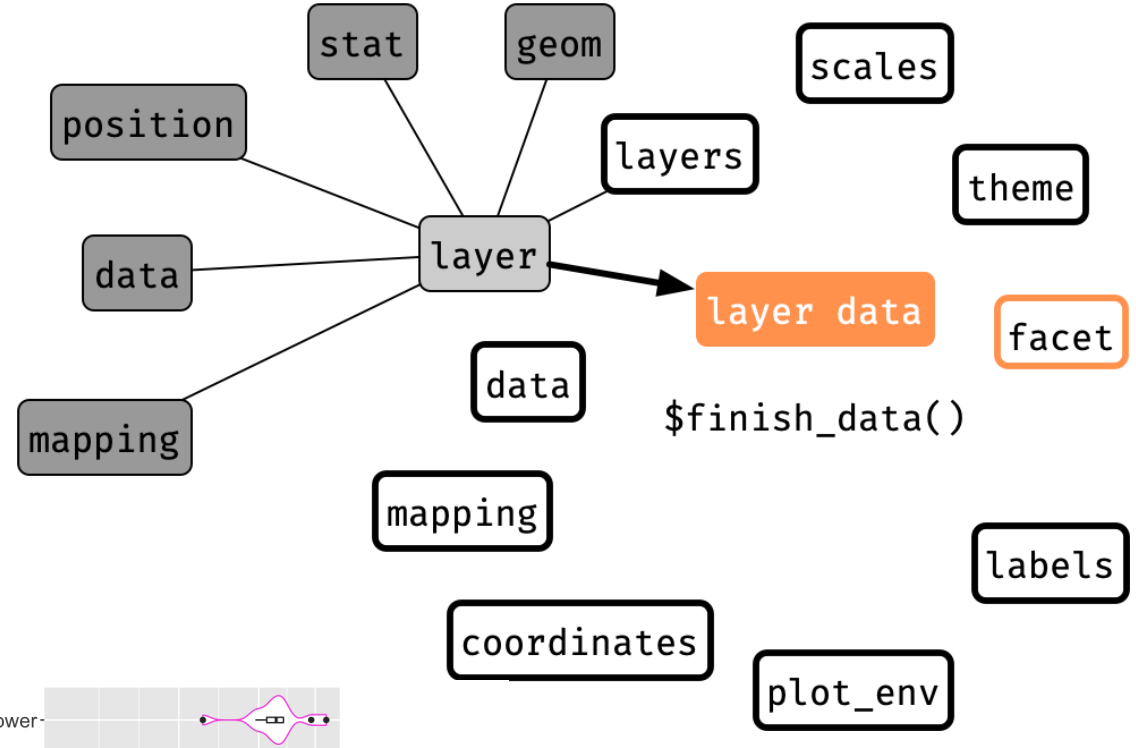
▶ Click to see difference to the previous data

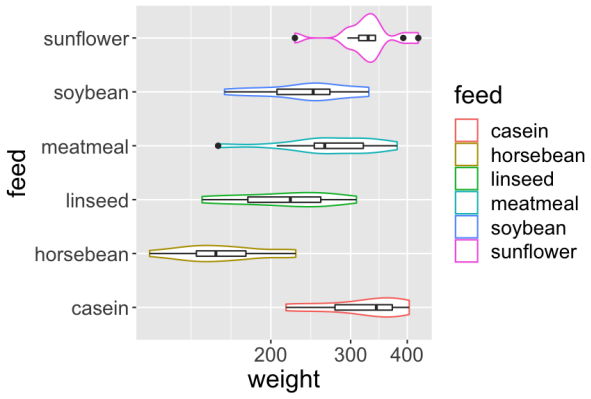
```
## [[1]]
## # A tibble: 3,072 × 21
##    colour  y          density scaled ndensity
##    <chr>   <mppd_dsc>   <dbl>  <dbl>    <dbl>
##  1 #F8766D 1             1.52  0.345    0.345
##  2 #F8766D 1             1.53  0.346    0.346
##  3 #F8766D 1             1.53  0.347    0.347
##  4 #F8766D 1             1.54  0.348    0.348
##  5 #F8766D 1             1.54  0.350    0.350
##  6 #F8766D 1             1.55  0.351    0.351
##  7 #F8766D 1             1.55  0.352    0.352
```

scroll ↓

Note: this is a rough illustration and some actual processes may differ

Data transformation 11 *Finale*





- Let Layout modify data before rendering

▶ Click to see difference to the previous data

```
## [[1]]
## # A tibble: 3,072 × 21
##    colour  y          density scaled ndensity
##    <chr>   <mppd_dsc>   <dbl>  <dbl>    <dbl>
##  1 #F8766D 1             1.52  0.345    0.345
##  2 #F8766D 1             1.53  0.346    0.346
##  3 #F8766D 1             1.53  0.347    0.347
##  4 #F8766D 1             1.54  0.348    0.348
##  5 #F8766D 1             1.54  0.350    0.350
##  6 #F8766D 1             1.55  0.351    0.351
##  7 #F8766D 1             1.55  0.352    0.352
```

scroll ↓

Note: this is a rough illustration and some actual processes may differ

Output from `ggplot2` build step

```
ggplot_build(g)
```

```
## $data
## $data[[1]]
##       colour y      density       scaled     ndensity        count  n        x
## 1    #F8766D 1  1.521016562 0.3445258223 0.3445258223  18.25219874 12 2.334454
## 2    #F8766D 1  1.526908684 0.3458604483 0.3458604483  18.32290421 12 2.334986
## 3    #F8766D 1  1.532605352 0.3471508019 0.3471508019  18.39126423 12 2.335518
## 4    #F8766D 1  1.538302020 0.3484411555 0.3484411555  18.45962424 12 2.336050
## 5    #F8766D 1  1.543900913 0.3497093619 0.3497093619  18.52681095 12 2.336582
## 6    #F8766D 1  1.549400375 0.3509550465 0.3509550465  18.59280450 12 2.337114
## 7    #F8766D 1  1.554899838 0.3522007311 0.3522007311  18.65879806 12 2.337647
## 8    #F8766D 1  1.560203789 0.3534021304 0.3534021304  18.72244547 12 2.338179
```

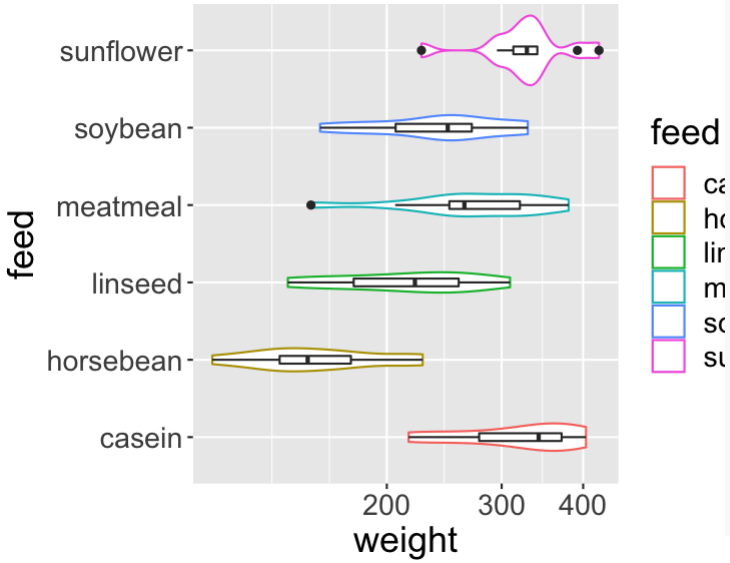
scroll ↓

- Prepares the **data** for each layer
- `Layout` **object** holding information about the trained coordinate system and facetting
- Copy of the **plot object** g with trained scales

#2> ggplot_gtable()

List of grobs in a ggplot object

```
print(g)
```




```
data <- ggplot_build(g)
gtable <- ggplot_gtable(data)
gtable

## TableGrob (12 x 11) "layout": 19 grobs
##     z         cells       name                                                grob
## 1   0 ( 1-12, 1-11) background              rect[plot.background..rect.1080]
## 2   5 ( 6- 6, 4- 4)     spacer                                      zeroGrob[NULL]
## 3   7 ( 7- 7, 4- 4)     axis-l           absoluteGrob[GRID.absoluteGrob.1020]
## 4   3 ( 8- 8, 4- 4)     spacer                                      zeroGrob[NULL]
## 5   6 ( 6- 6, 5- 5)     axis-t                                      zeroGrob[NULL]
## 6   1 ( 7- 7, 5- 5)      panel                          gTree[panel-1.gTree.1012]
## 7   9 ( 8- 8, 5- 5)     axis-b           absoluteGrob[GRID.absoluteGrob.1016]
## 8   4 ( 6- 6, 6- 6)     spacer                                      zeroGrob[NULL]
## 9   8 ( 7- 7, 6- 6)     axis-r                                      zeroGrob[NULL]
```

scroll ↓

- `gtable` is actually a special type of `grid::gTree`

```
class(gtable)

## [1] "gtable" "gTree"  "grob"   "gDesc"
```

Inspecting `gtable`

```
grid::grid.ls(gtable)

## layout

gftable <- grid::grid.force(gtable)
grid::grid.ls(gftable)

## layout
##   background.1-11-12-1
##   panel.7-5-7-5
##     grill.gTree.1010
##       panel.background..rect.1003
##       panel.grid.minor.x..polyline.1005
##       panel.grid.major.y..polyline.1007
##       panel.grid.major.x..polyline.1009
##     NULL
##     geom_violin.gTree.945
##       geom_violin.polygon.928
##       geom_violin.polygon.931
##       geom_violin.polygon.934
##       geom_violin.polygon.937
##       geom_violin.polygon.940
##       geom_violin.polygon.943
##     geom_boxplot.gTree.999
##       geom_boxplot.gTree.953
##         GRID.segments.947
##         geom_crossbar.gTree.952
```

Manipulating the gtable

- As gtable is a gTree, you can use the same approach to manipulate it via grid
- The exact grob names are long (and not easy to predict) in ggplot2 so we'll use regular expression to find grob paths (via grep = TRUE)

```
grid::grid.grep("panel.background", gftable, grep = TRUE)

## layout::panel.7-5-7-5::grill.gTree.1010::panel.background..rect.1003
```

- We'll edit the panel background so it has a red border and tilted 30 degrees

```
getable <- grid::editGrob(gftable, gPath = "panel.background", grep = TRUE,
                          gp = grid::gpar(col = "red"),
                          vp = grid::viewport(angle = 30))
```

- In another modification, we'll remove the panel background

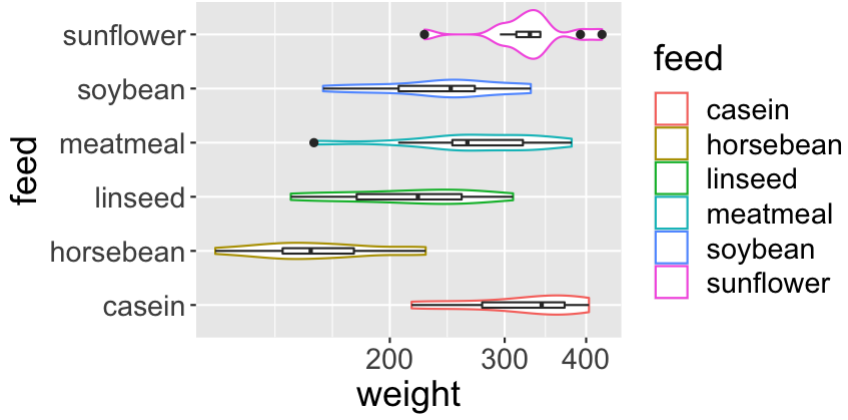
```
grtable <- grid::removeGrob(gftable, gPath = "panel.background", grep = TRUE)
```

#3> grid.draw()

Resulting edit using `grid` on `gtable`

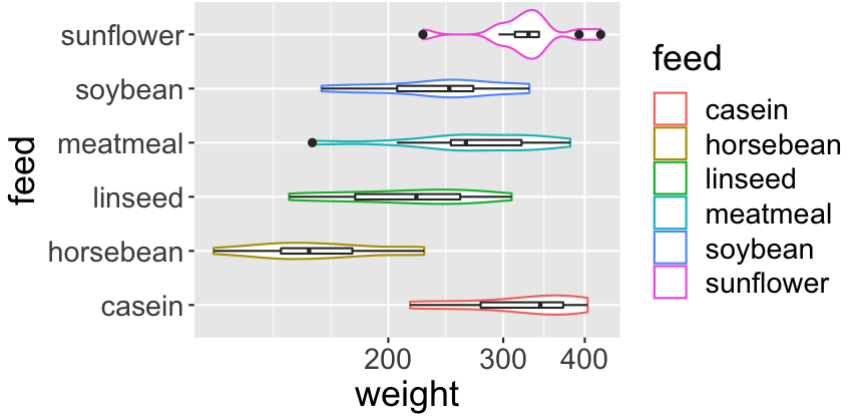
Original

```
grid::grid.draw(gtable)
```



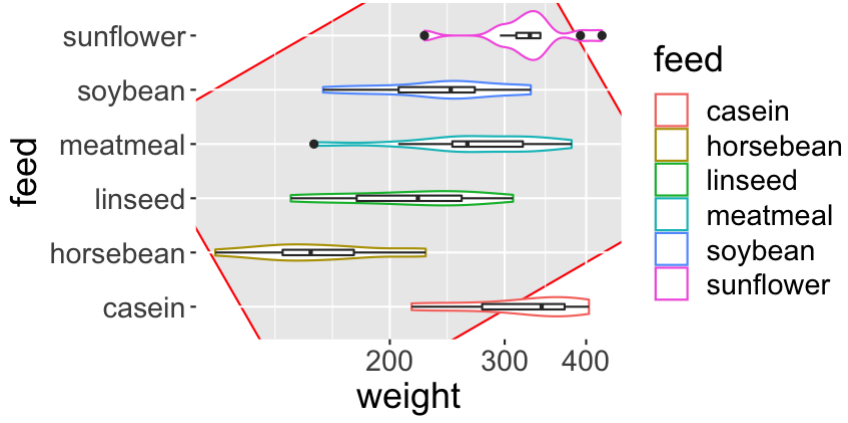
Forced

```
grid::grid.draw(gftable)
```



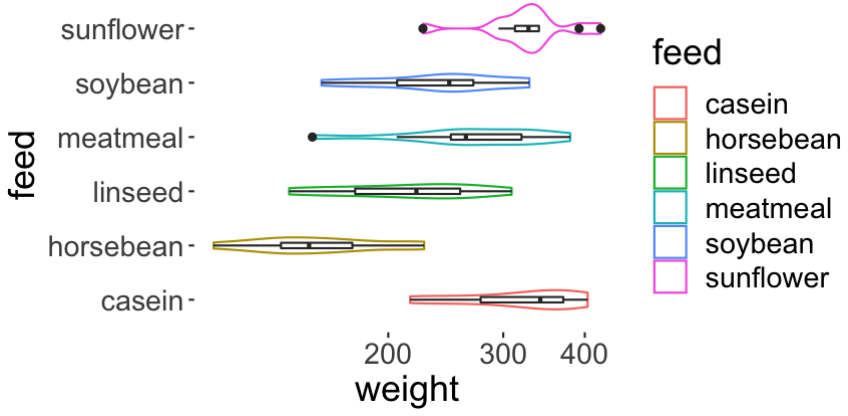
Editted

```
grid::grid.draw(getable)
```



Removed

```
grid::grid.draw(grtable)
```



Resources

Check out also the 3rd edition of the ggplot2 book

https://ggplot2-book.org/internals.html

</> Open day1-exercise-02.Rmd

15:00

Session Information

```
devtools::session_info()
```

```
## ─ Session info  🚚  🃟  ☝  ──────────────────────────────────────────────
##  hash: delivery truck, joker, index pointing up: light skin tone
##
##  setting  value
##  version  R version 4.1.2 (2021-11-01)
##  os       macOS Catalina 10.15.7
##  system   x86_64, darwin17.0
##  ui       X11
##  language (EN)
##  collate  en_AU.UTF-8
##  ctype    en_AU.UTF-8
##  tz       Australia/Melbourne
##  date     2021-12-07
```