Hands–On Data Visualization with ggplot2

Segment 5: Create Multipanel Figures

Dr. Cédric Scherer

Pearson Live Training Session for O'Reilly

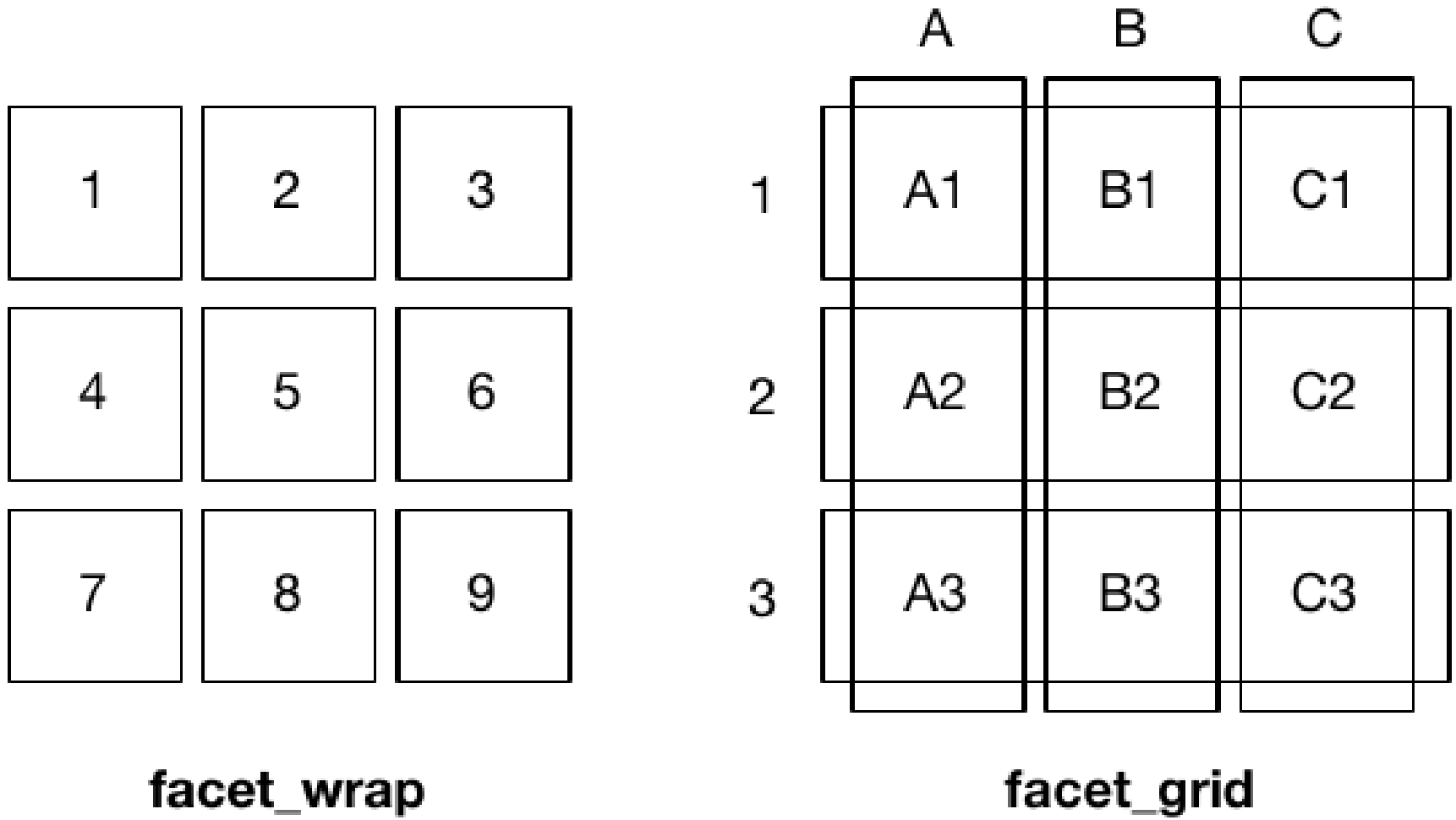
September 3, 2021

Photo by Richard Strozynski

Facets

Small Multiples

facet_*()



Adapted from "ggplot2: Elegant Graphics for Data Analysis" by Hadley Wickham

Our Small-Multiple Candidate

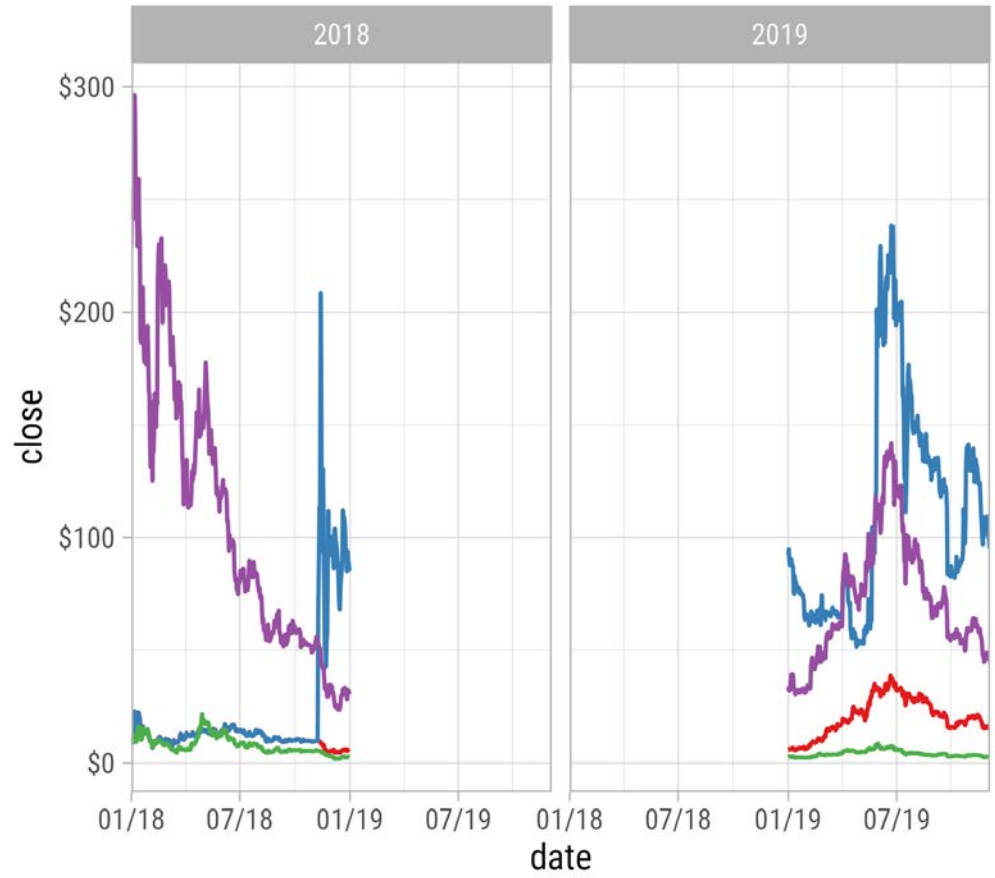
The well-known plot we are going to use for our small multiples, saved as g.

```
g <-
  ggplot(data, aes(date, close)) +
    geom_line(
      aes(color = currency),
      size = 1
    ) +
    scale_x_date(
      date_labels = "%m/%y",
      expand = c(0, 0)
    ) +
    scale_y_continuous(
      labels = scales::dollar_format()
    ) +
    scale_color_brewer(
      palette = "Set1",
      guide = "none"
    )

g
```

facet_wrap()

`facet_wrap()` splits the data into small multiples based on **one grouping variable**:

```
g +
  facet_wrap(vars(year))
```



facet_wrap()

`facet_wrap()` splits the data into small multiples based on **one grouping variable**:

```
g +
  facet_wrap(vars(currency))
```

facet_wrap()

`facet_wrap()` splits the data into small multiples based on **one grouping variable**:

```
g +
  facet_wrap(~ currency)
```

facet_wrap()

`facet_wrap()` splits the data into small multiples based on **one grouping variable**:

```
g +
  facet_wrap(
    vars(currency),
    scales = "free"
  )
```

facet_wrap()

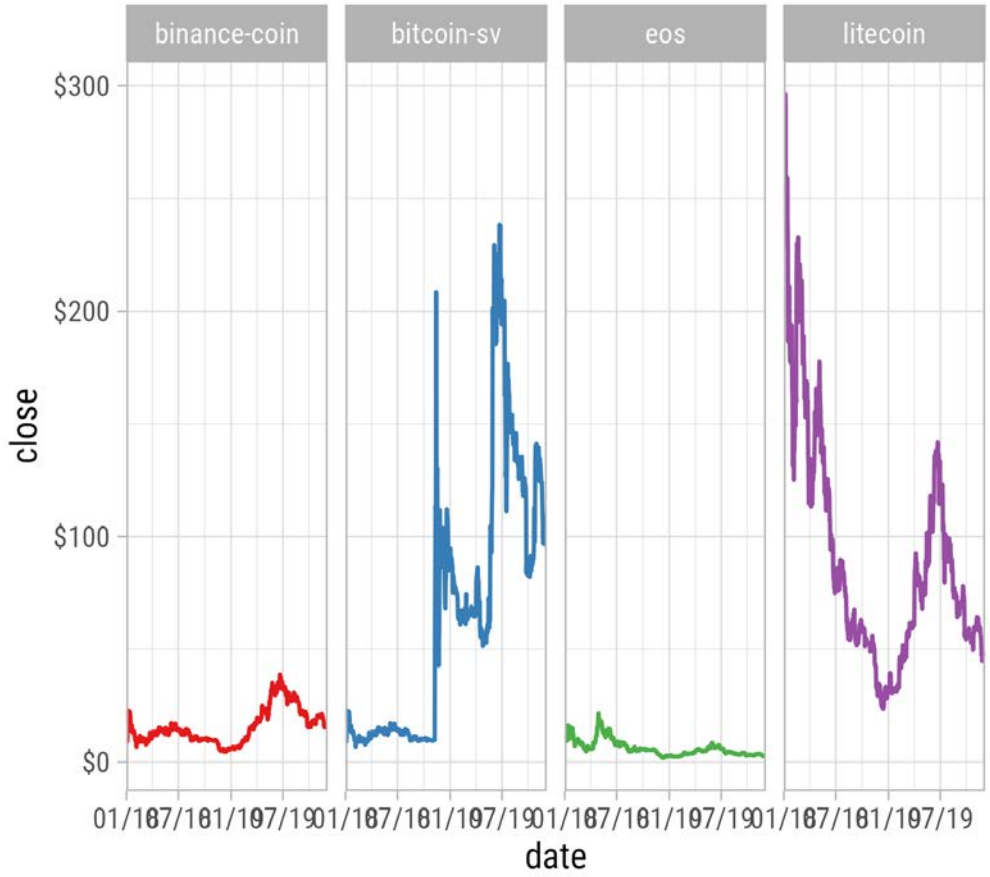
`facet_wrap()` splits the data into small multiples based on **one grouping variable**:

```
g +
  facet_wrap(
    vars(currency),
    scales = "free_y"
  )
```

facet_wrap()

`facet_wrap()` splits the data into small multiples based on **one grouping variable**:

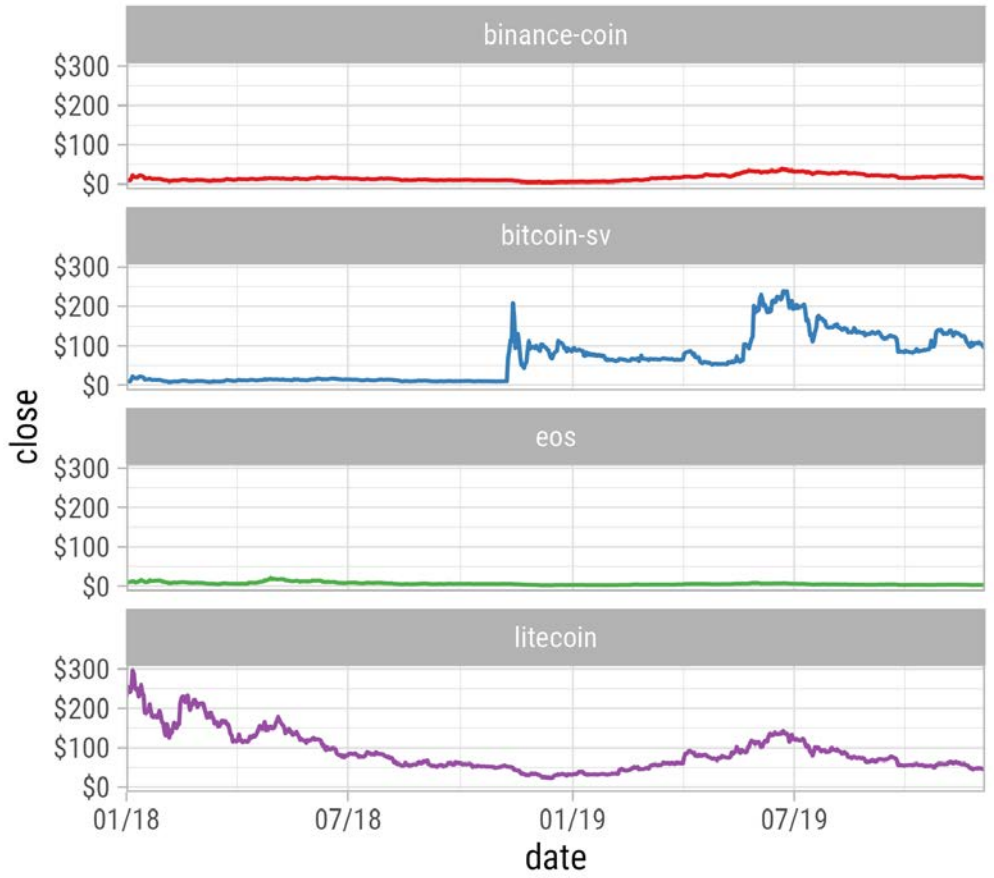
```
g +
  facet_wrap(
    vars(currency),
    nrow = 1
  )
```



facet_wrap()

`facet_wrap()` splits the data into small multiples based on **one grouping variable**:

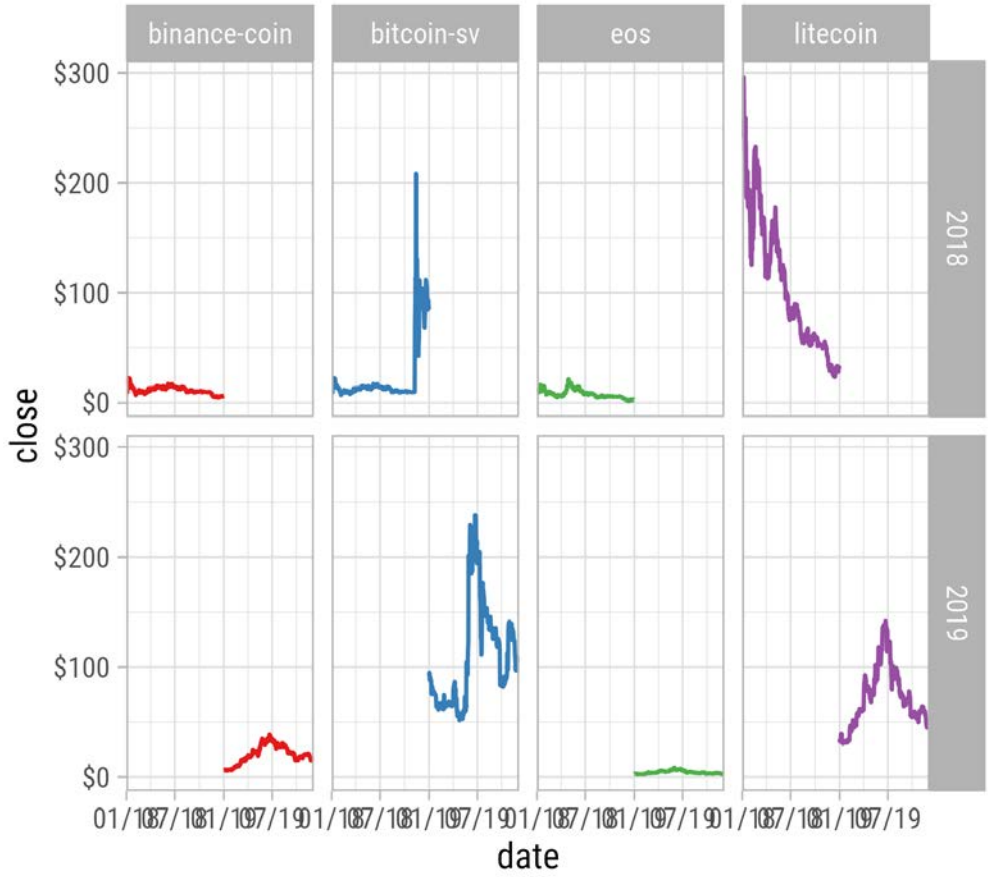
```
g +
  facet_wrap(
    vars(currency),
    ncol = 1
  )
```



facet_grid()

`facet_grid()` spans a grid of each combination of **two grouping variables**:

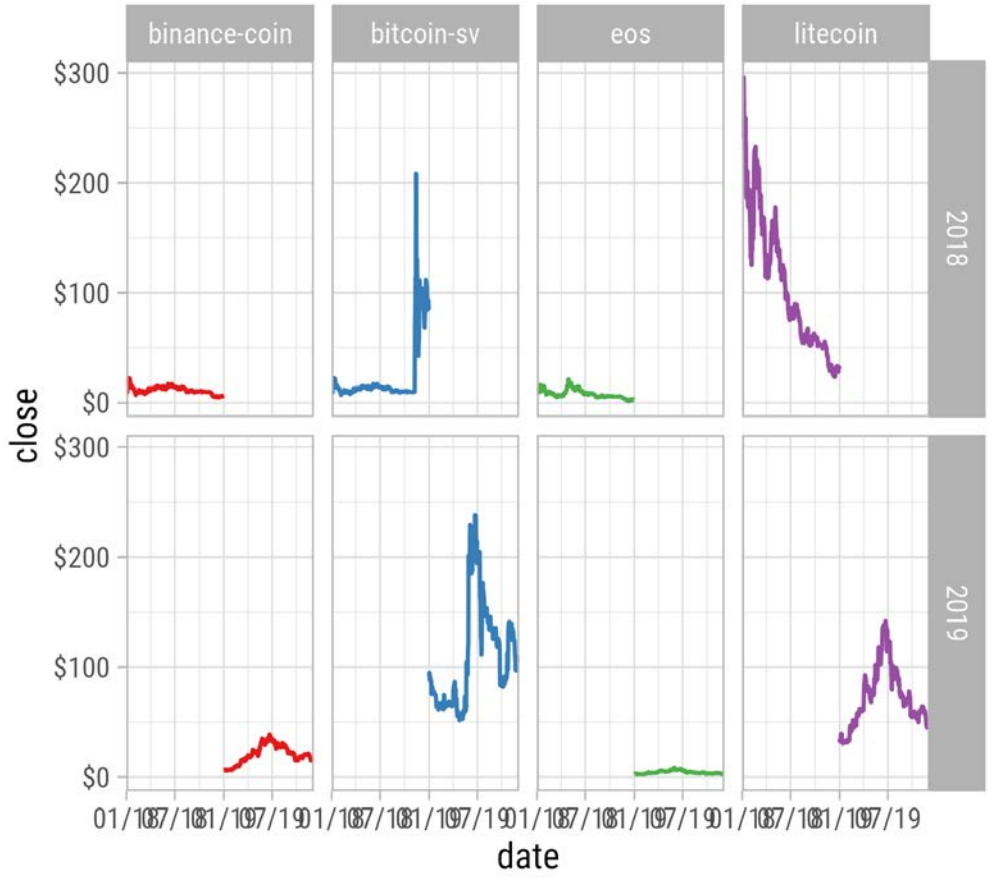
```
g +
  facet_grid(
    cols = vars(currency),
    rows = vars(year)
  )
```



facet_grid()

`facet_grid()` spans a grid of each combination of **two grouping variables**:

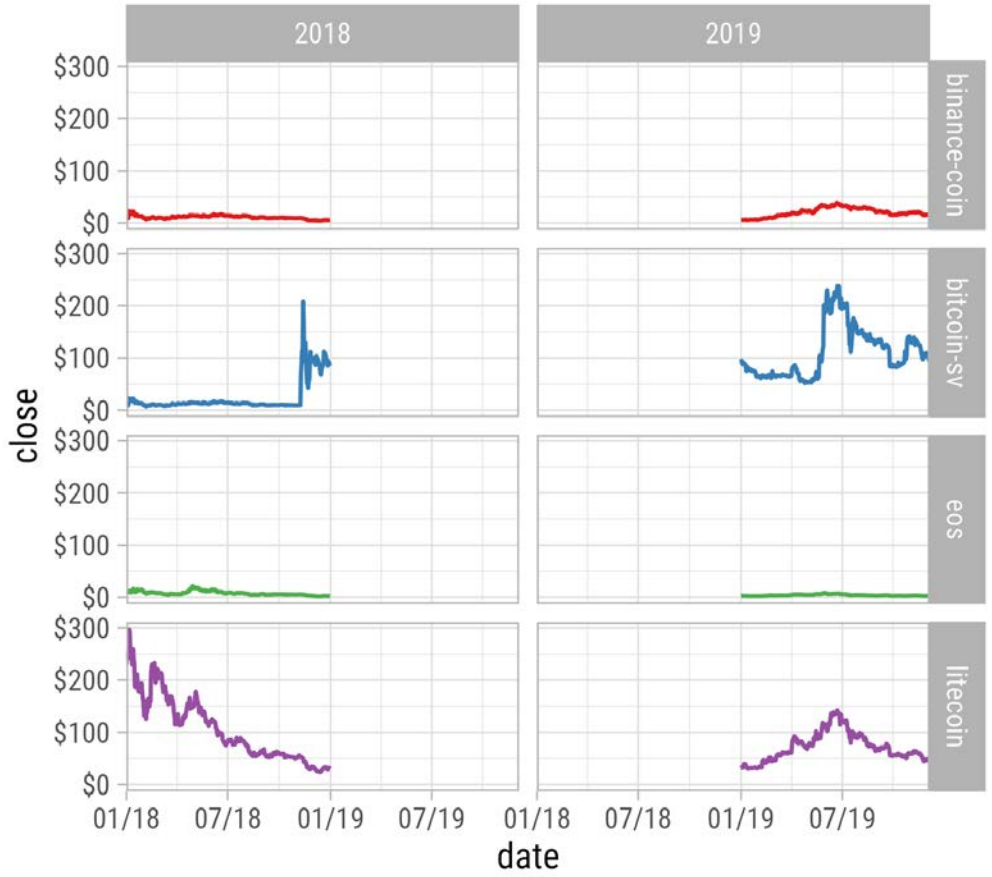
```
g +
  facet_grid(
    year ~ currency
  )
```



facet_grid()

`facet_grid()` spans a grid of each combination of **two grouping variables**:

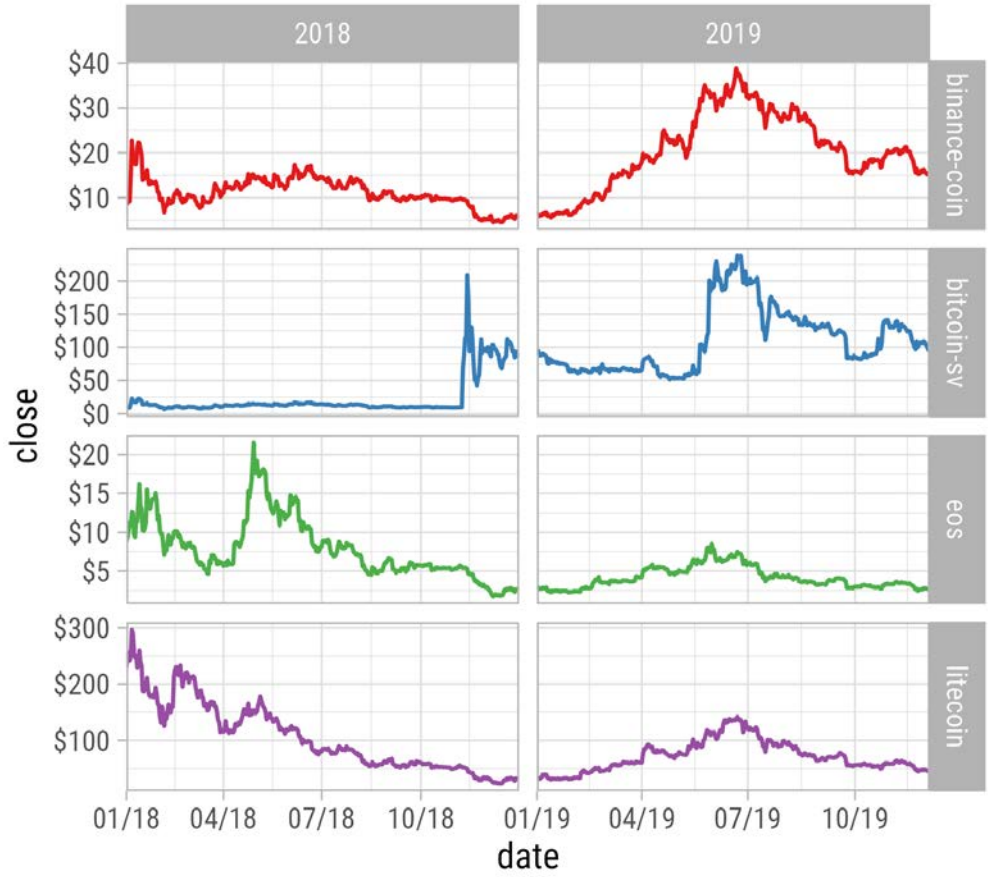
```
g +
  facet_grid(
    currency ~ year
  )
```



facet_grid()

`facet_grid()` spans a grid of each combination of **two grouping variables**:

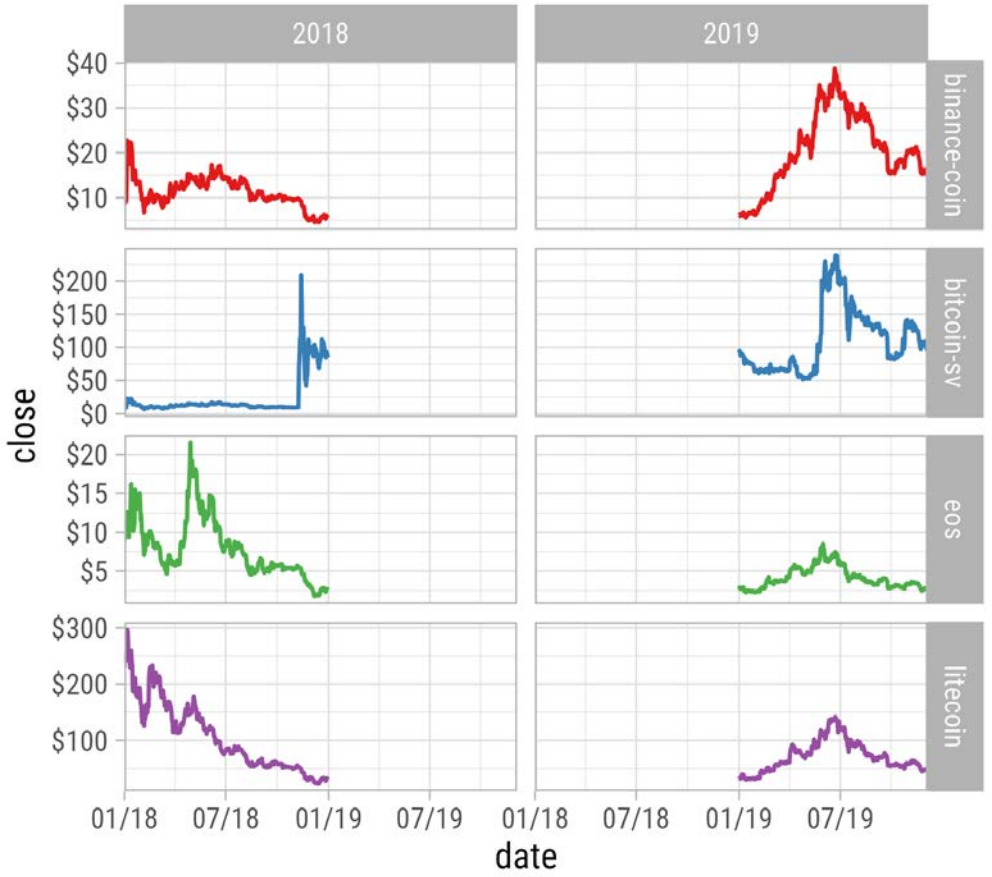
```
g +
  facet_grid(
    currency ~ year,
    scales = "free"
  )
```



facet_grid()

`facet_grid()` spans a grid of each combination of **two grouping variables**:

```
g +
  facet_grid(
    currency ~ year,
    scales = "free_y"
  )
```



Exercise 1:

- A famous statistical data set is the `“Datasaurus Dozen”`, which is based on `“Anscome's Quartet”`.

  Import the according data into R and inspect it: `datasaurus.csv`
- Visualize all 12 sets as small multiples of scatter plots.
- Also, add to each facet a linear fitting in the back.
- Use a built-in theme and add a title and explanation of the visual.

Exercise 1: Import and Inspect Data

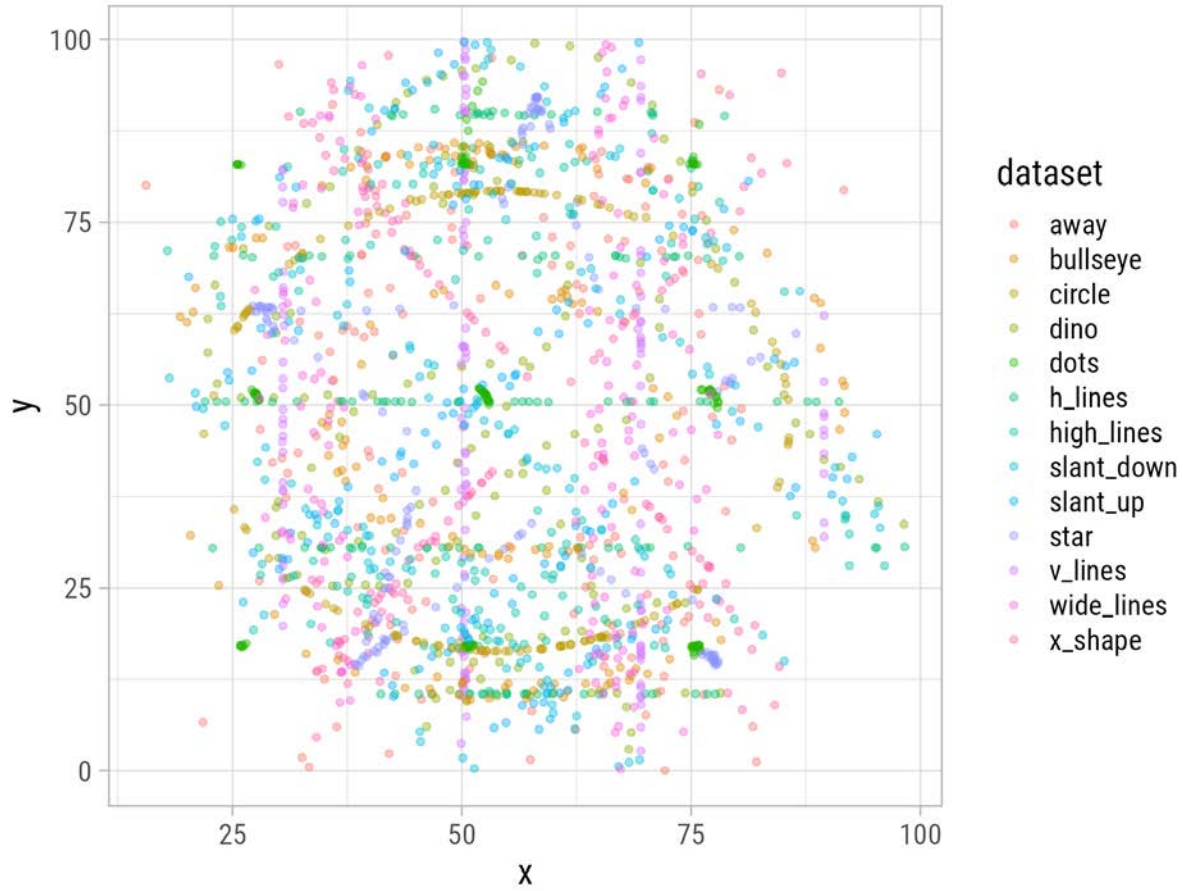
```
saurus <- readr::read_csv(here::here("data", "datasaurus.csv"))

unique(saurus$dataset)
##  [1] "dino"       "away"       "h_lines"    "v_lines"    "x_shape"
##  [6] "star"       "high_lines" "dots"       "circle"     "bullseye"
## [11] "slant_up"   "slant_down" "wide_lines"
```

Exercise 1: Import and Inspect Data

```
saurus <- readr::read_csv(here::here("data", "datasaurus.csv"))

saurus %>%
  group_by(dataset) %>%
  summarize(mean_x = mean(x), sd_x = sd(x), mean_y = mean(y), sd_y = sd(y))
## # A tibble: 13 x 5
##    dataset    mean_x  sd_x mean_y  sd_y
##    <chr>       <dbl> <dbl>  <dbl> <dbl>
##  1 away         54.3  16.8   47.8  26.9
##  2 bullseye     54.3  16.8   47.8  26.9
##  3 circle       54.3  16.8   47.8  26.9
##  4 dino         54.3  16.8   47.8  26.9
##  5 dots         54.3  16.8   47.8  26.9
##  6 h_lines      54.3  16.8   47.8  26.9
##  7 high_lines   54.3  16.8   47.8  26.9
##  8 slant_down   54.3  16.8   47.8  26.9
##  9 slant_up     54.3  16.8   47.8  26.9
## 10 star         54.3  16.8   47.8  26.9
## 11 v_lines      54.3  16.8   47.8  26.9
## 12 wide_lines   54.3  16.8   47.8  26.9
## 13 x_shape      54.3  16.8   47.8  26.9
```

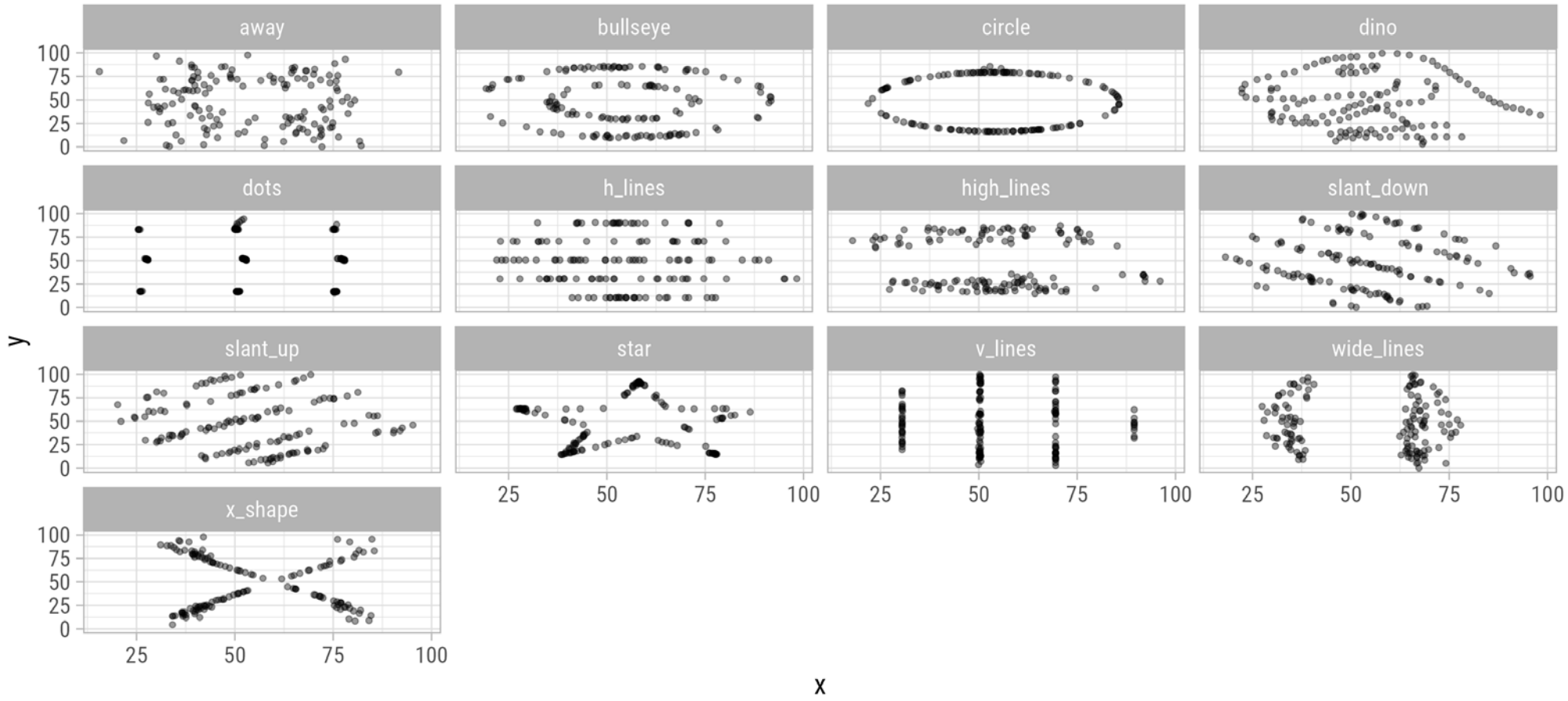
Exercise 1: Plot the Data

```
ggplot(saurus, aes(x, y, color = dataset)) +
  geom_point(alpha = .4)
```



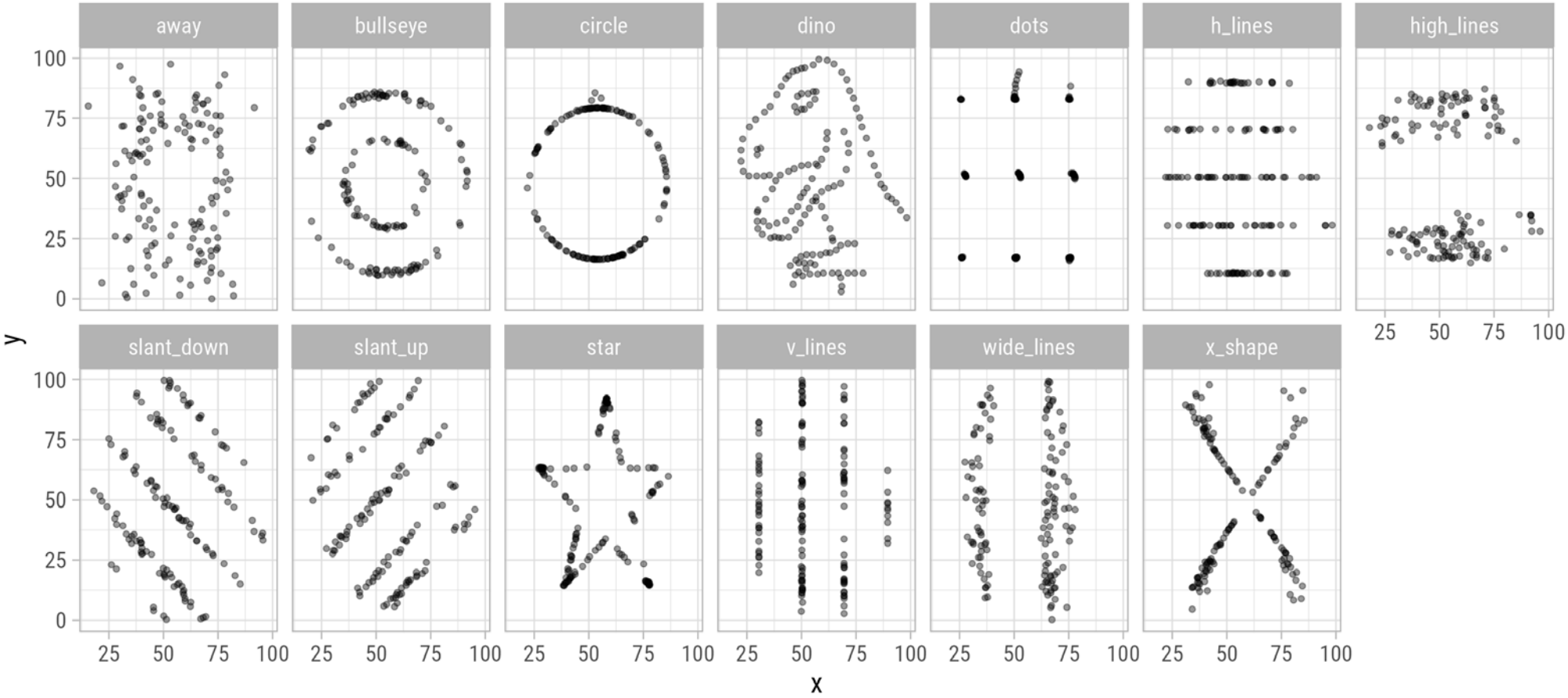
Exercise 1: Turn into Small Multiples

```
ggplot(saurus, aes(x, y)) +
  geom_point(alpha = .4) +
  facet_wrap(~ dataset)
```



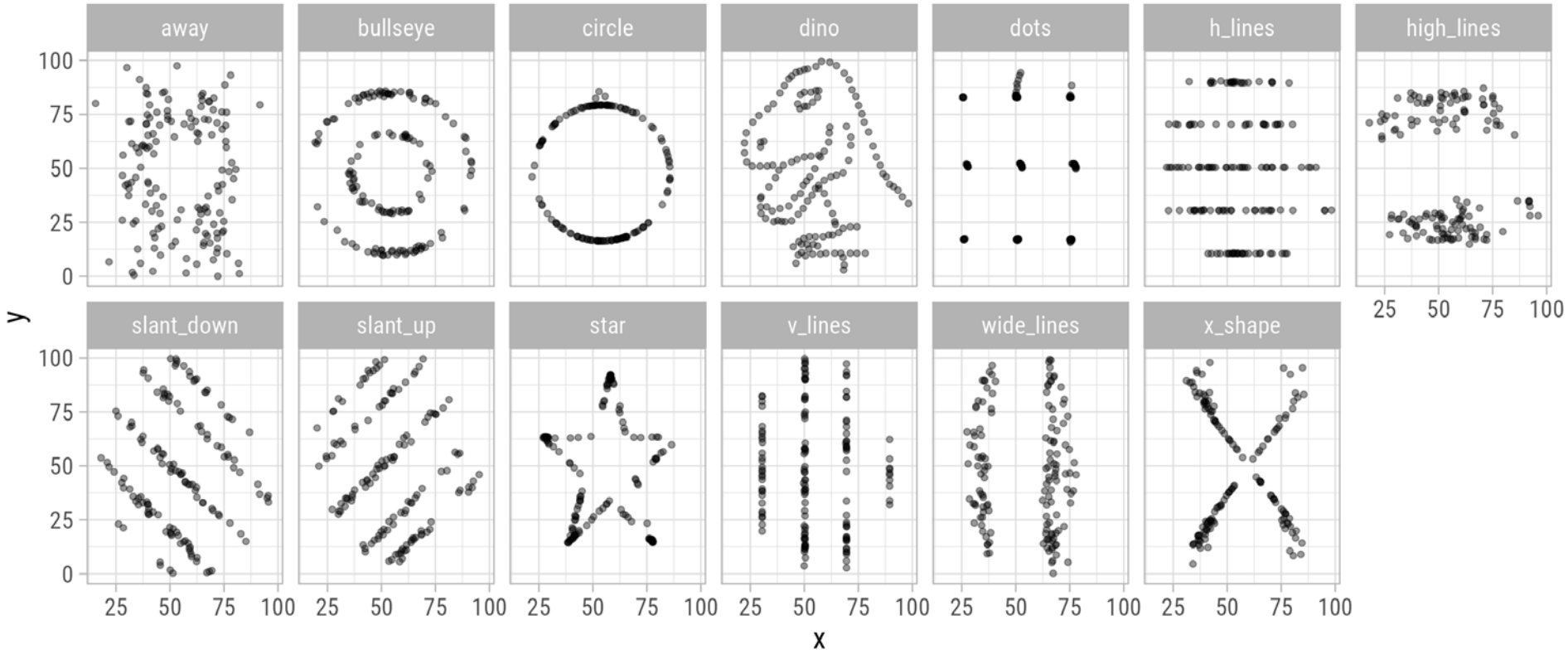
Exercise 1: Turn into Small Multiples

```
ggplot(saurus, aes(x, y))  +
  geom_point(alpha = .4) +
  facet_wrap(~ dataset, nrow = 2)
```



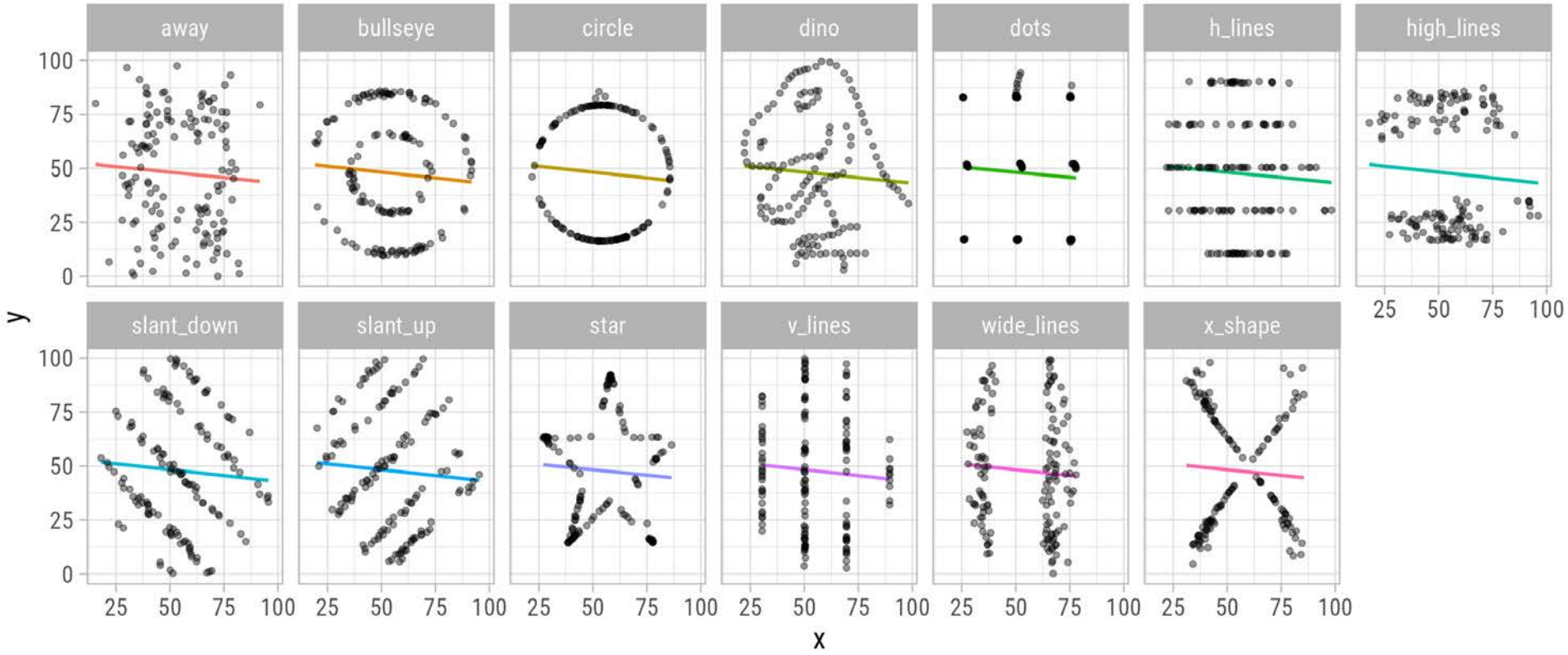
Exercise 1: Fix Aspect Ratio

```
ggplot(saurus, aes(x, y))  +
  geom_point(alpha = .4) +
  facet_wrap(~ dataset, nrow = 2) +
  coord_fixed()
```



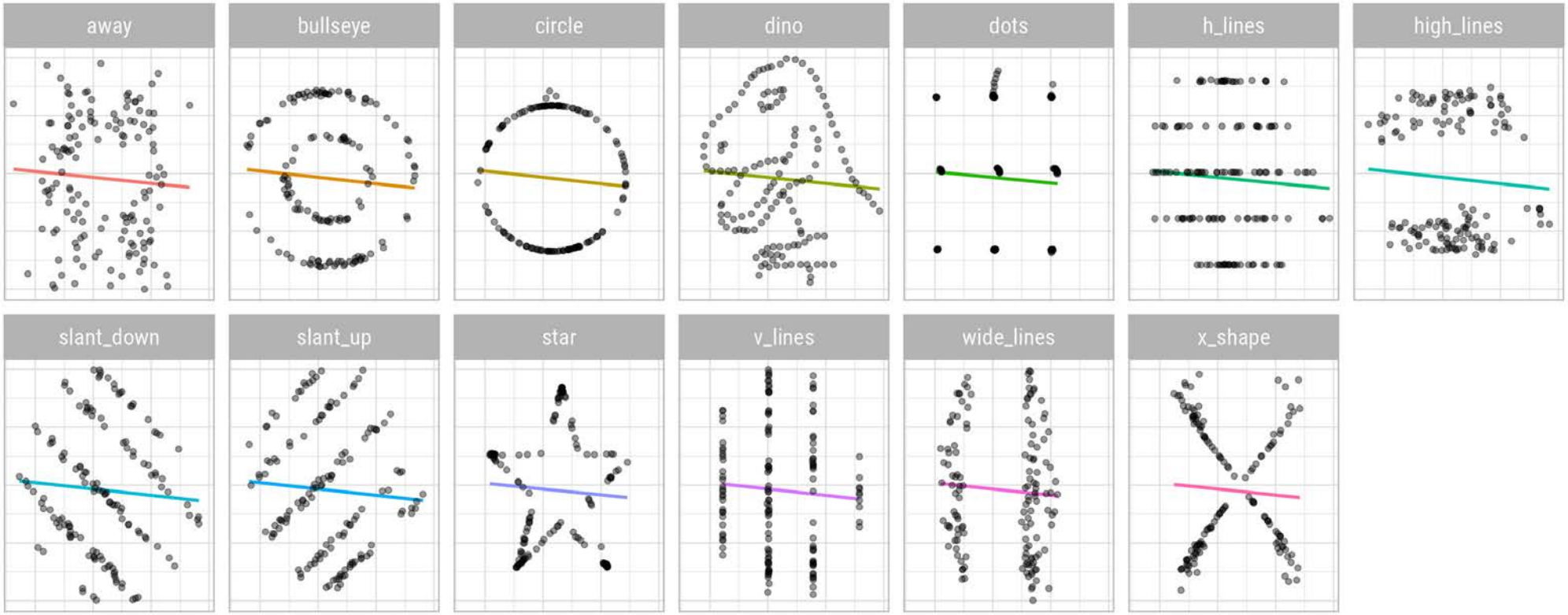
Exercise 1: Add Linear Regression Line

```
(g <- ggplot(saurus, aes(x, y)) +
  geom_smooth(method = "lm", aes(color = dataset), se = FALSE, show.legend = FALSE) +
  geom_point(alpha = .4) +
  facet_wrap(~ dataset, nrow = 2) +
  coord_fixed())
```



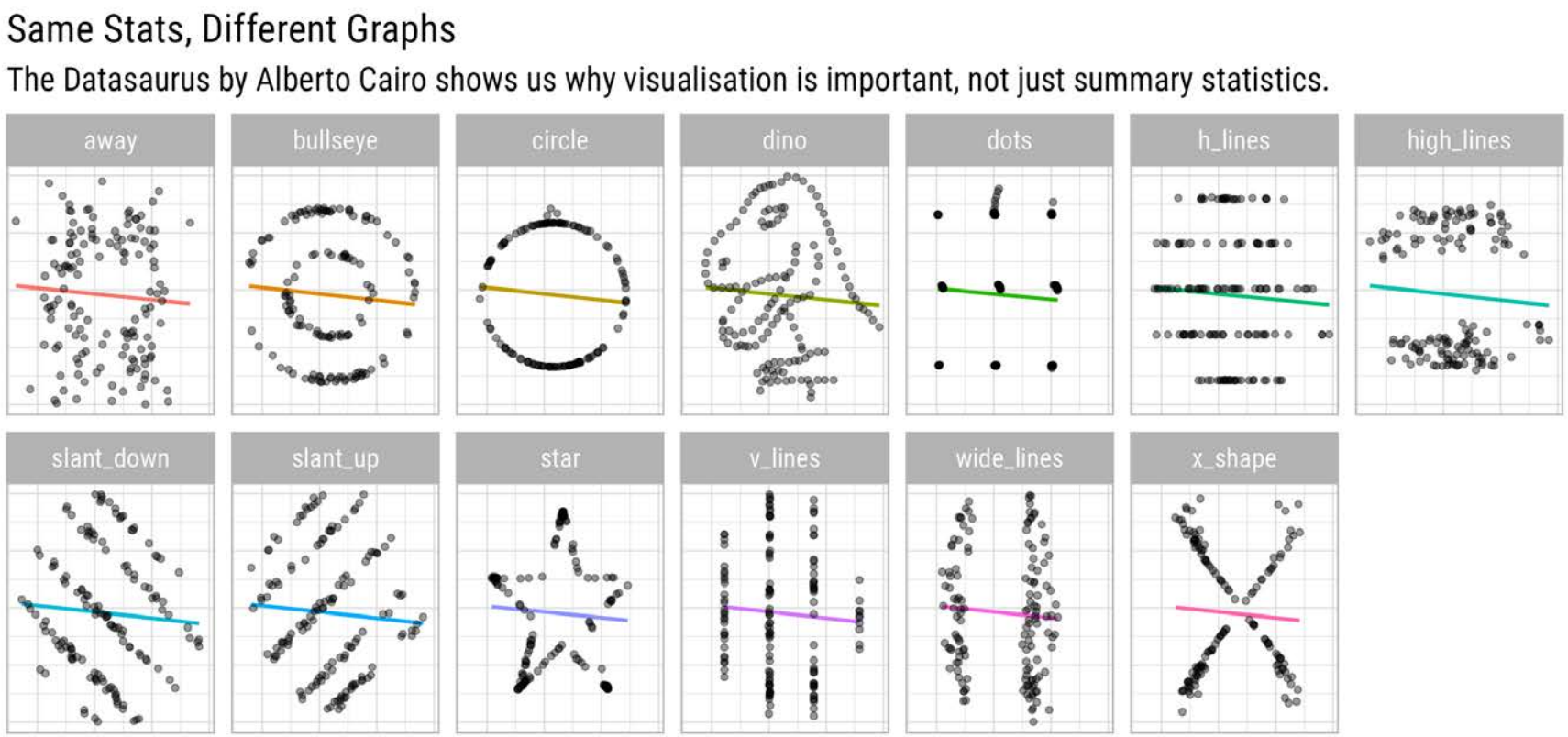
Exercise 1: Remove Chart Junk

```
g +
  scale_x_continuous(guide = "none", name = NULL) +
  scale_y_continuous(guide = "none", name = NULL)
```



Exercise 1: Add Title and Caption

```
g +
  scale_x_continuous(guide = "none", name = NULL) +
  scale_y_continuous(guide = "none", name = NULL) +
  labs(title = "Same Stats, Different Graphs", subtitle = "The Datasaurus by Alberto Cairo shows us why vi
```



The patchwork Package

The Composer of Plots

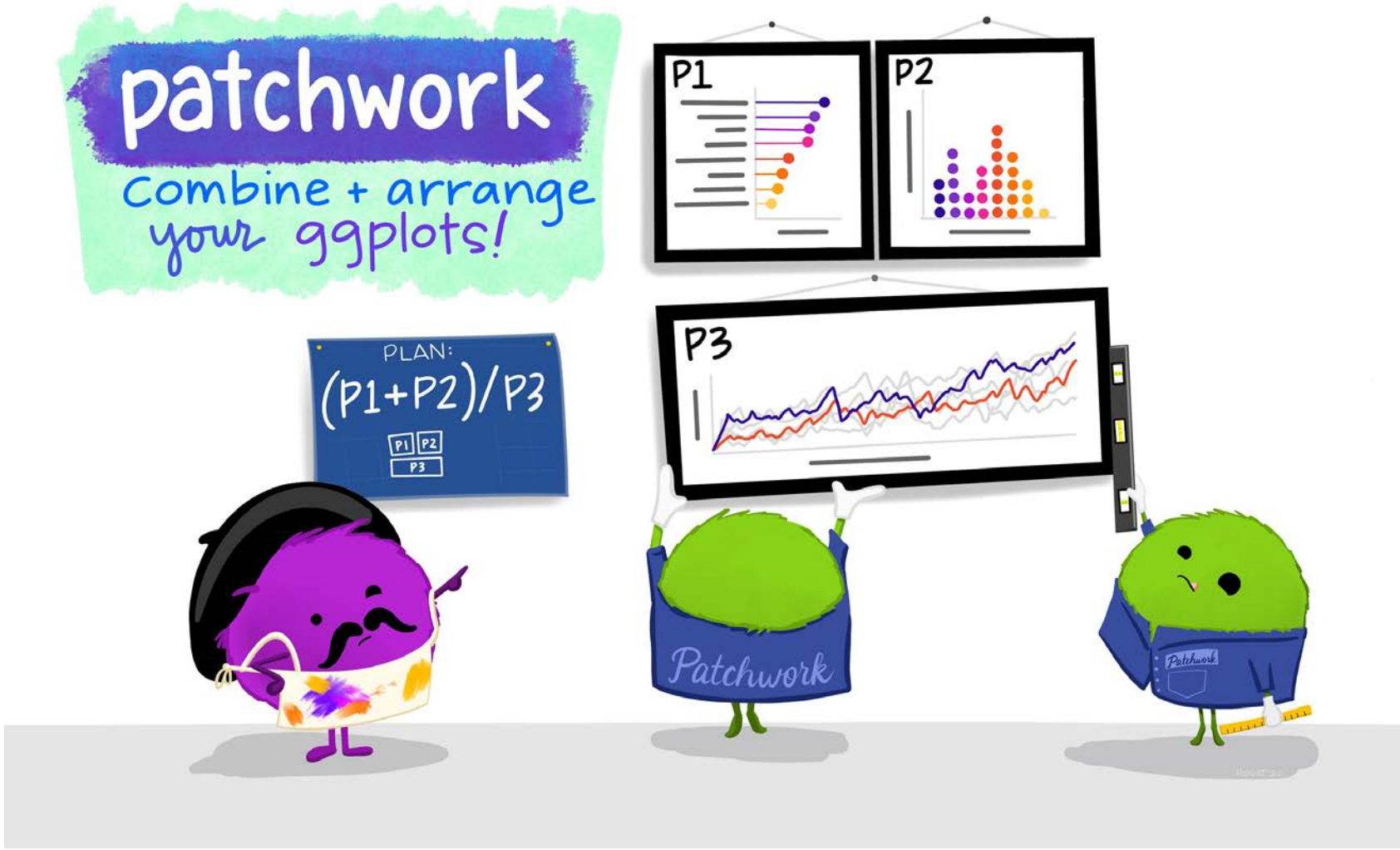


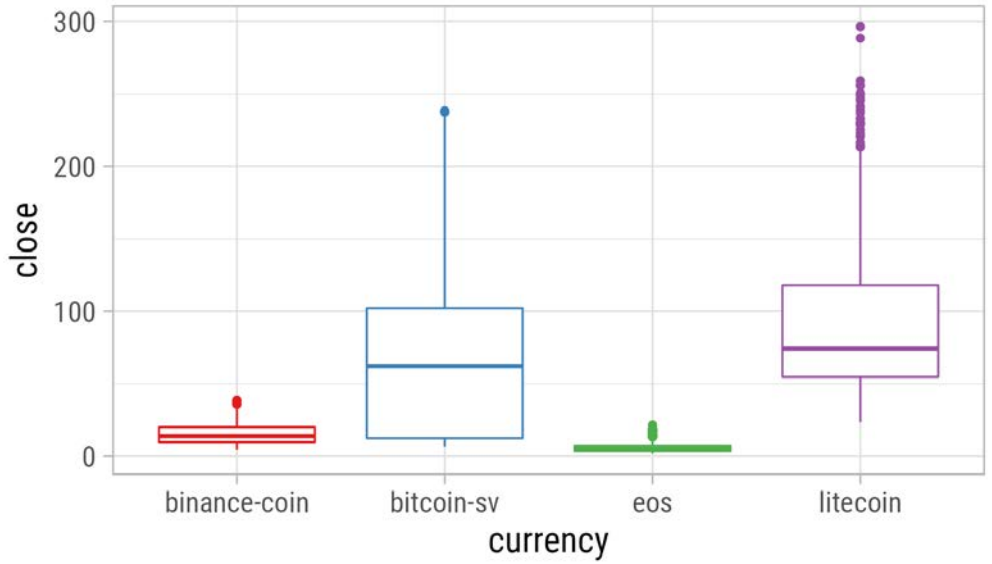
Illustration by Allison Horst

The patchwork package

Build up your multipanel plot sequentially using **The Composer of Plots**:

```
(time <- ggplot(data, aes(date, close)) +
  geom_line(aes(color = currency)) +
  scale_color_brewer(palette = "Set1",
                     guide = "none"))
```

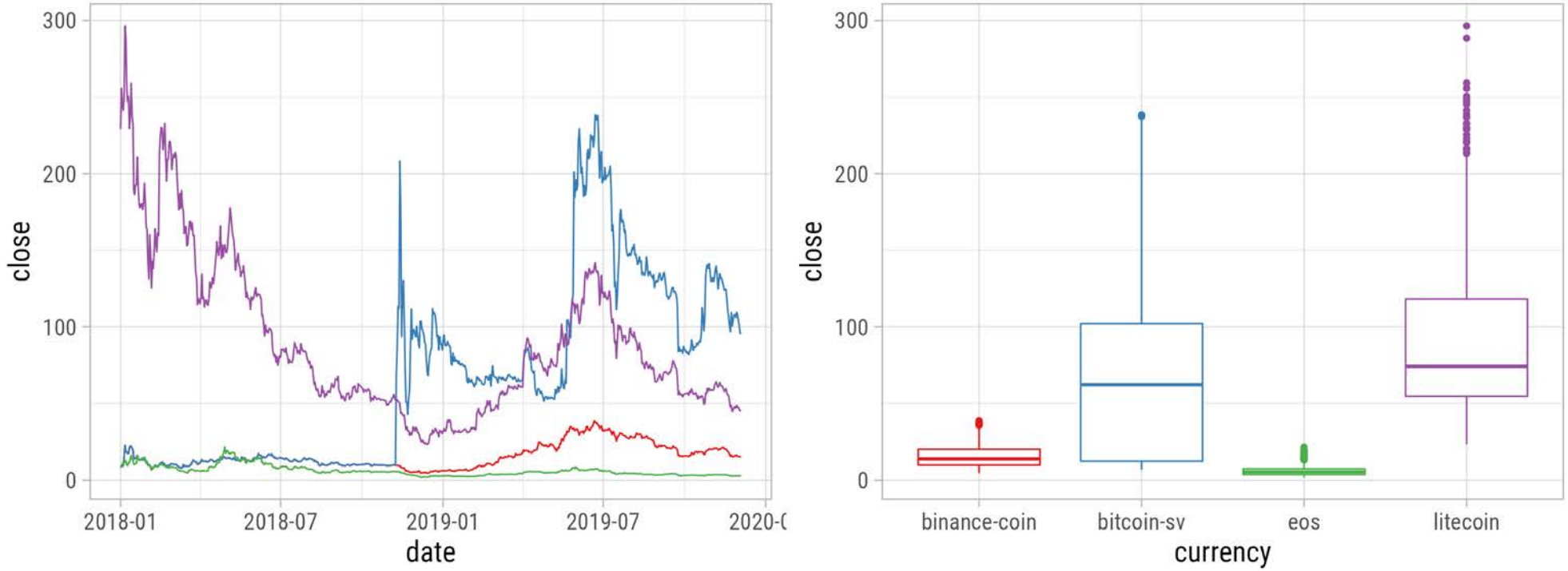
```
(box <- ggplot(data, aes(currency, close)) +
  geom_boxplot(aes(color = currency)) +
  scale_color_brewer(palette = "Set1",
                     guide = "none"))
```



The patchwork package

Build up your multipanel plot sequentially using **The Composer of Plots**:

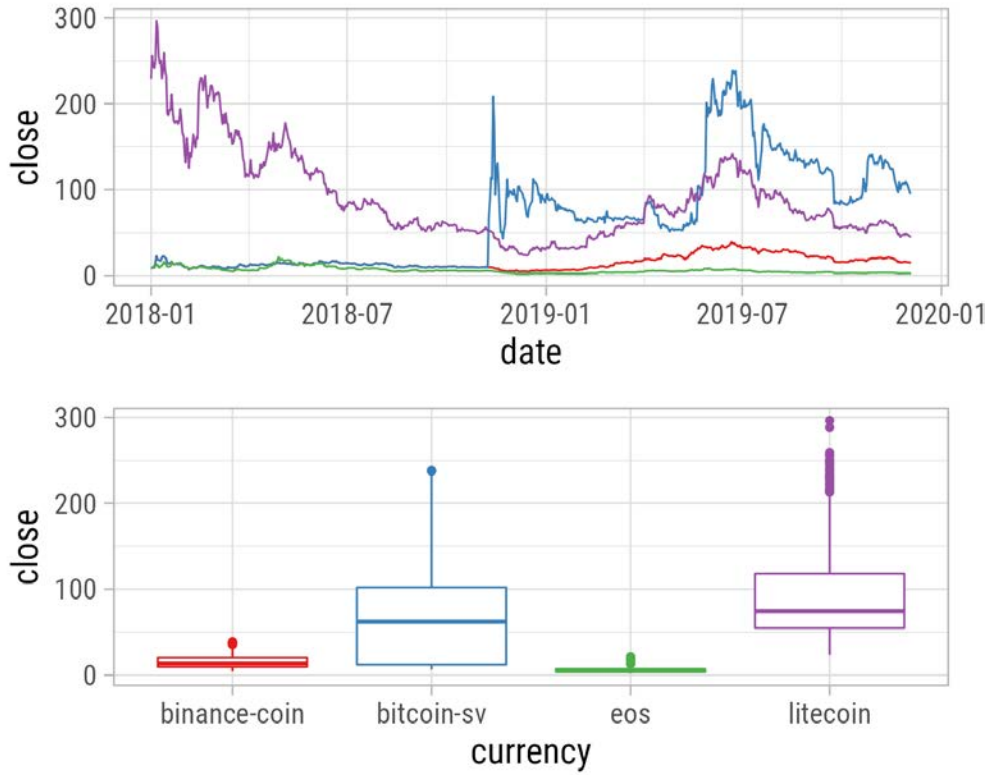
```
#install.packages("patchwork")
library(patchwork)
time + box
```



The patchwork package

Build up your multipanel plot sequentially using **The Composer of Plots**:

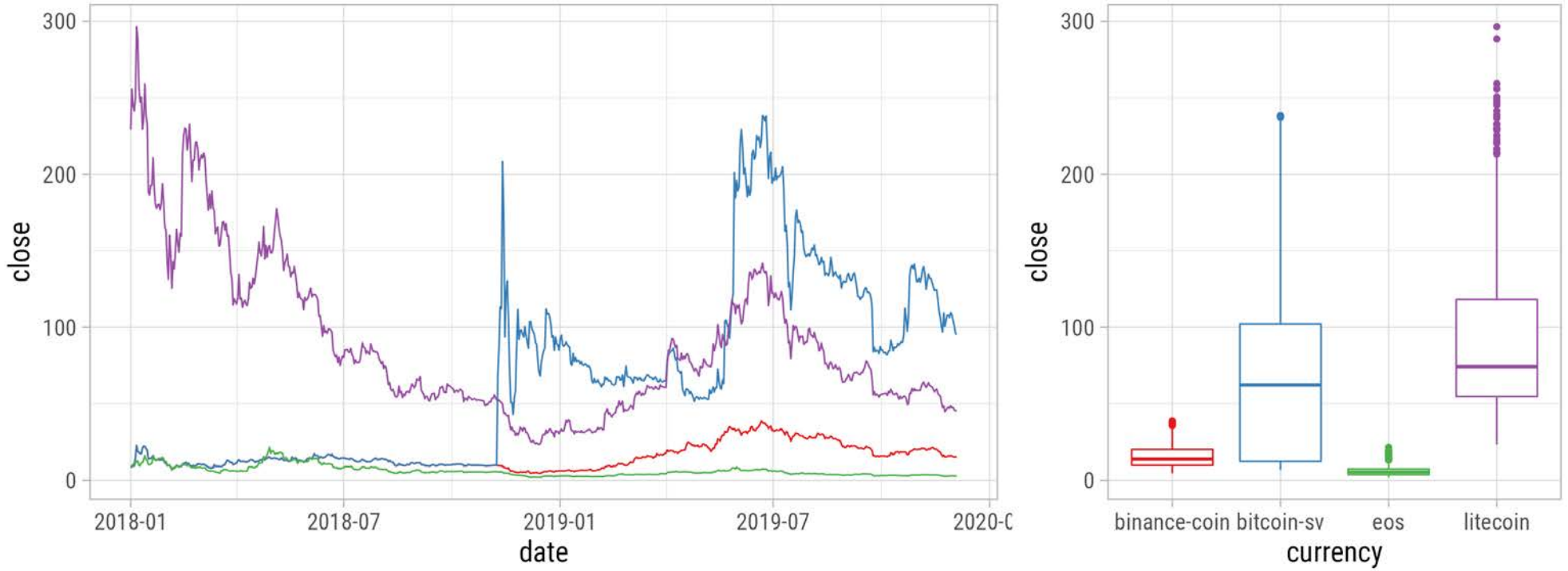
```
time / box
```



The patchwork package

With the help of `plot_layout()` you can adjust widths and/or heights:

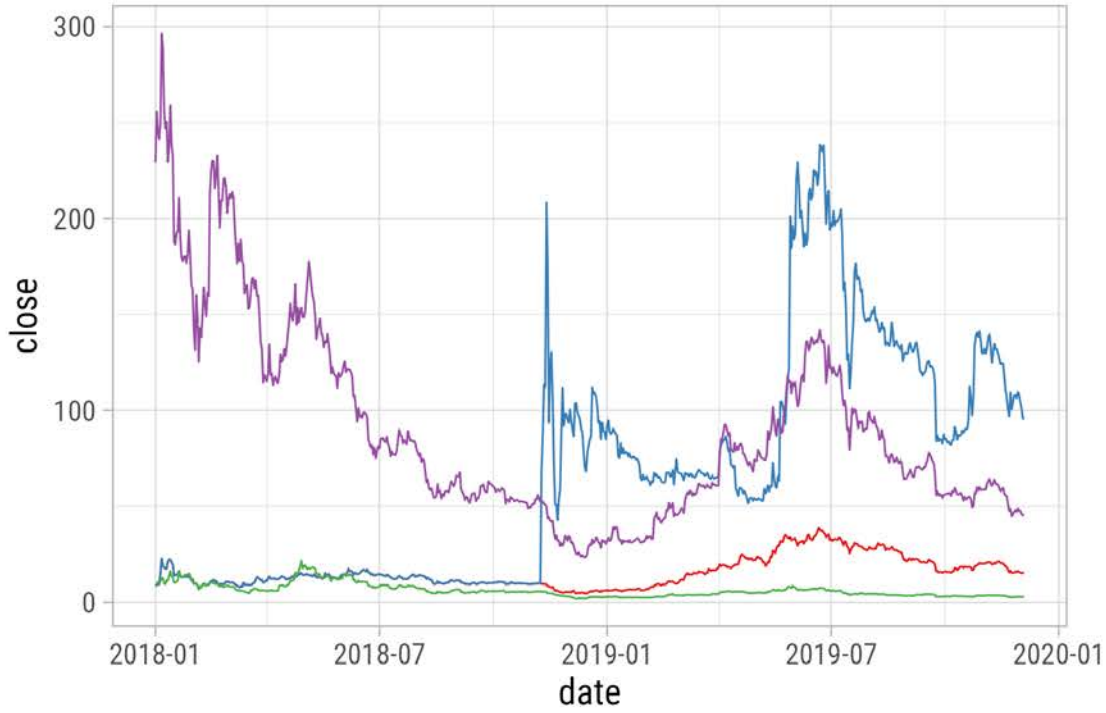
```
time + box +
  plot_layout(widths = c(2, 1))
```

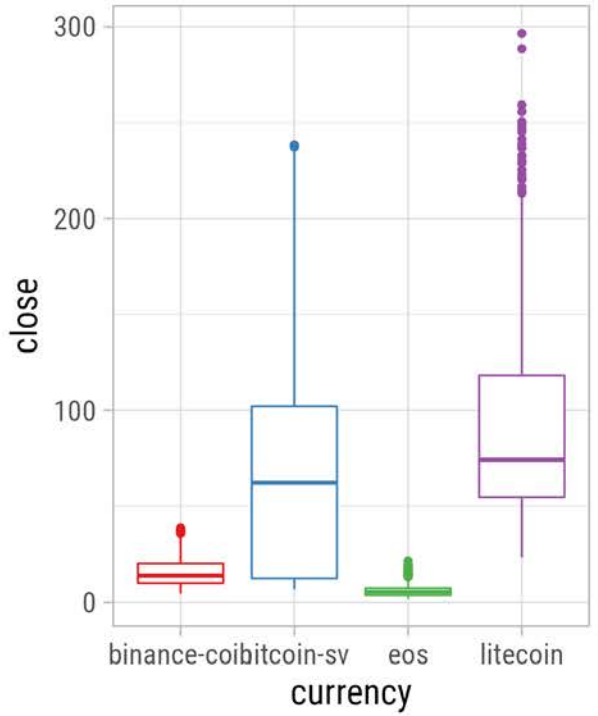


The patchwork package

The plot_spacer() function allows to add empty panels:

```
time + plot_spacer() + box +
  plot_layout(widths = c(2, .5, 1))
```

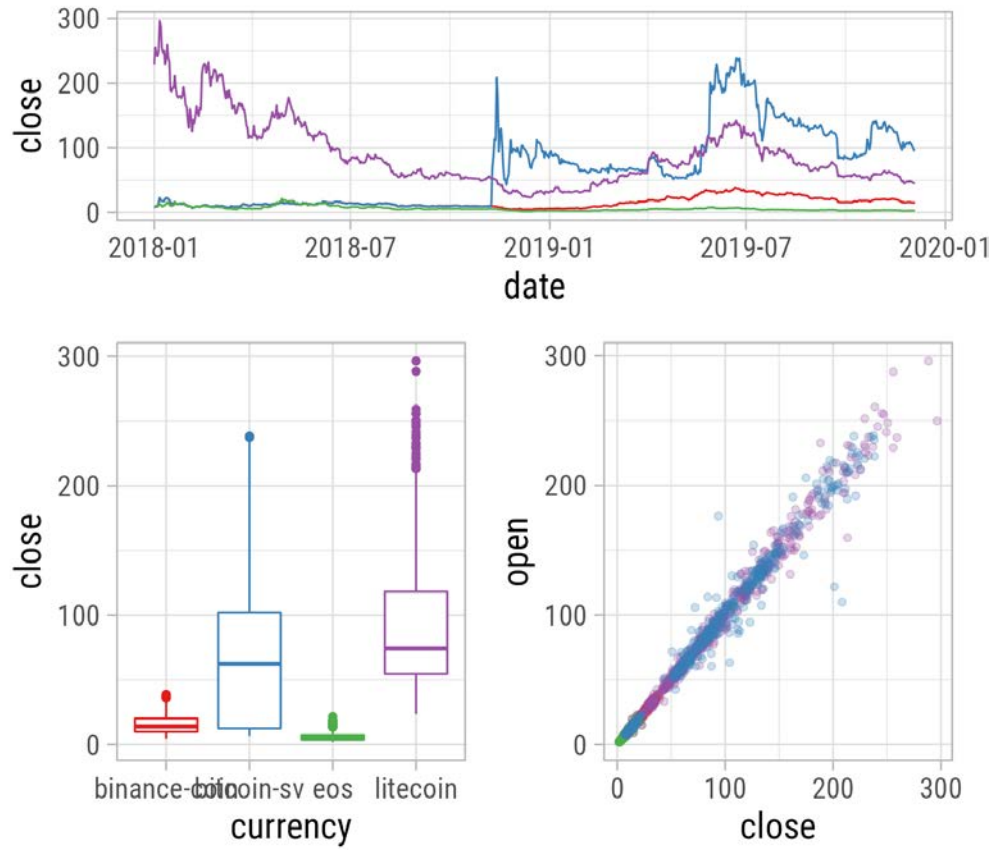




The patchwork package

Let's add another plot — with parentheses you can nest plots:

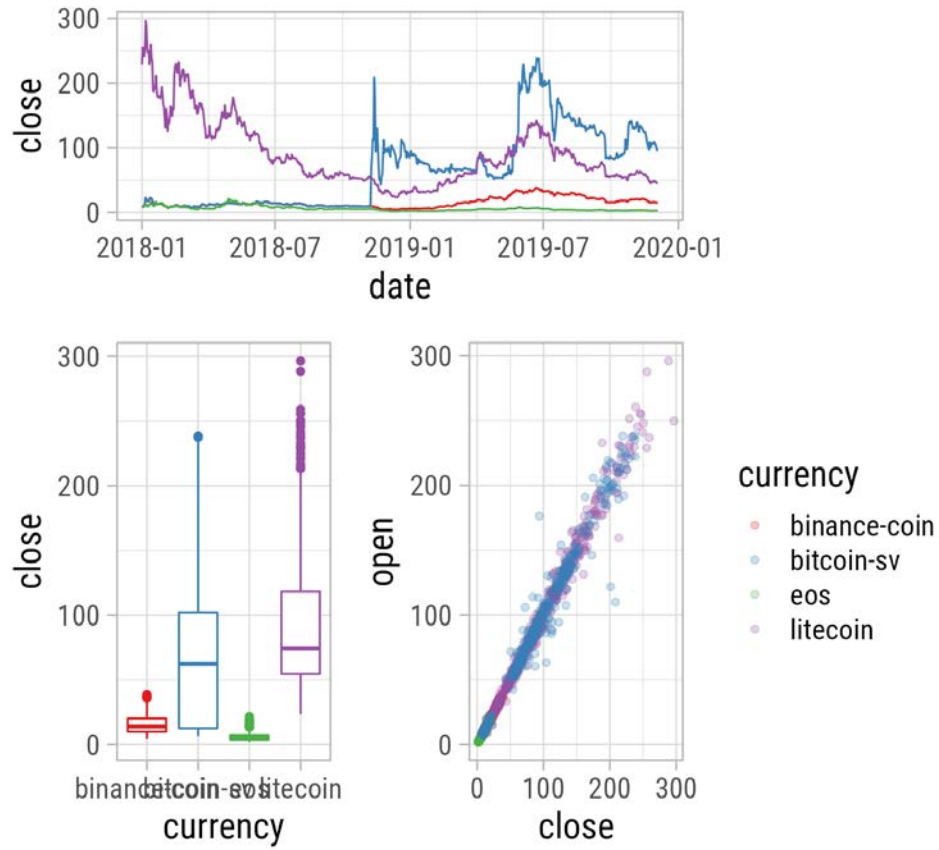
```
scatter <-
  ggplot(data,
         aes(close, open)) +
  geom_point(
    aes(color = currency),
    size = 1.5,
    alpha = .25
  ) +
  scale_color_brewer(
    palette = "Set1",
    guide = "none"
  )

time / (box + scatter) +
  plot_layout(heights = c(1, 2))
```



The patchwork package

Note that even with legends the plot panels align:

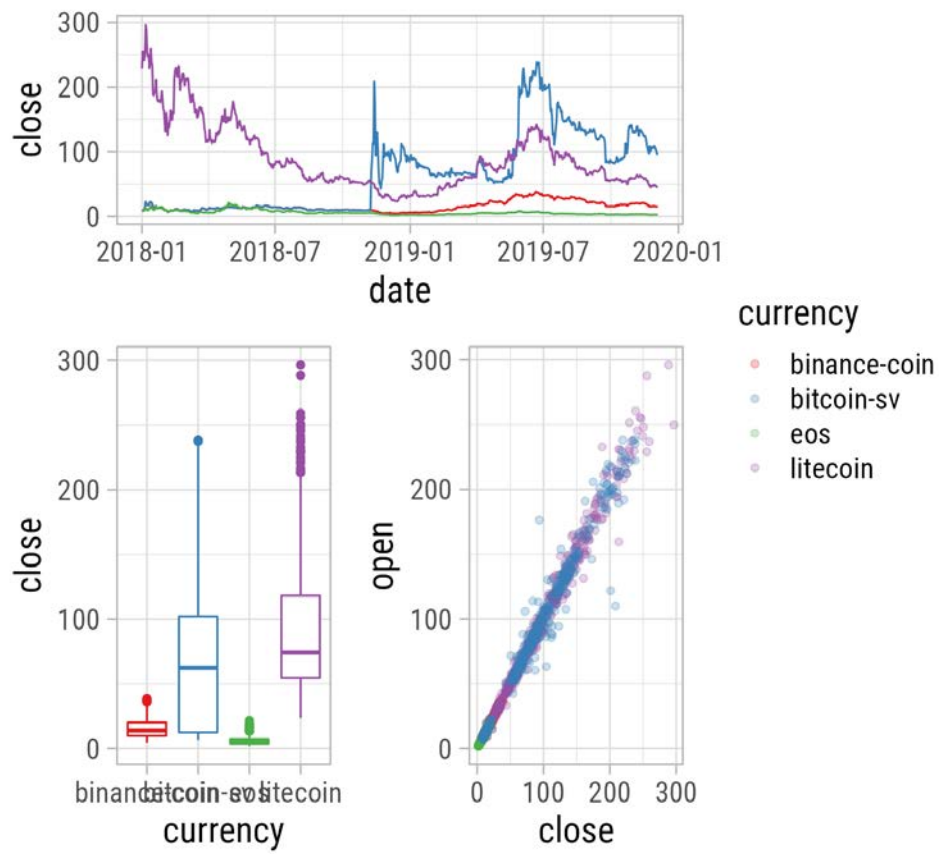
```
scatter_guide <-
  ggplot(data,
         aes(close, open)) +
  geom_point(
    aes(color = currency),
    size = 1.5,
    alpha = .25
  ) +
  scale_color_brewer(
    palette = "Set1",
    #guide = "none"
  )

time / (box + scatter_guide) +
  plot_layout(heights = c(1, 2))
```



The `patchwork` package

The argument `guides = "collect"` groups legends and places them relative to the composition:

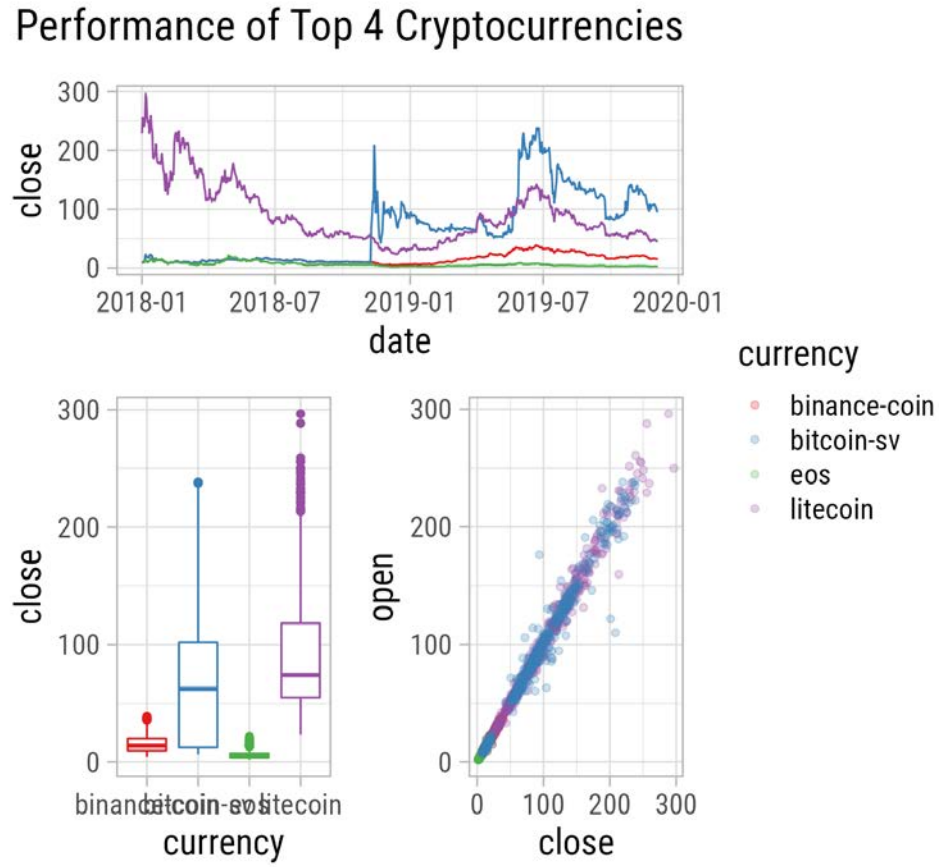
```
time / (box + scatter_guide) +
  plot_layout(
    heights = c(1, 2),
    guides = "collect"
  )
```



The patchwork package

Furthermore, `plot_annotation()` allows to add labels on the compositional level:

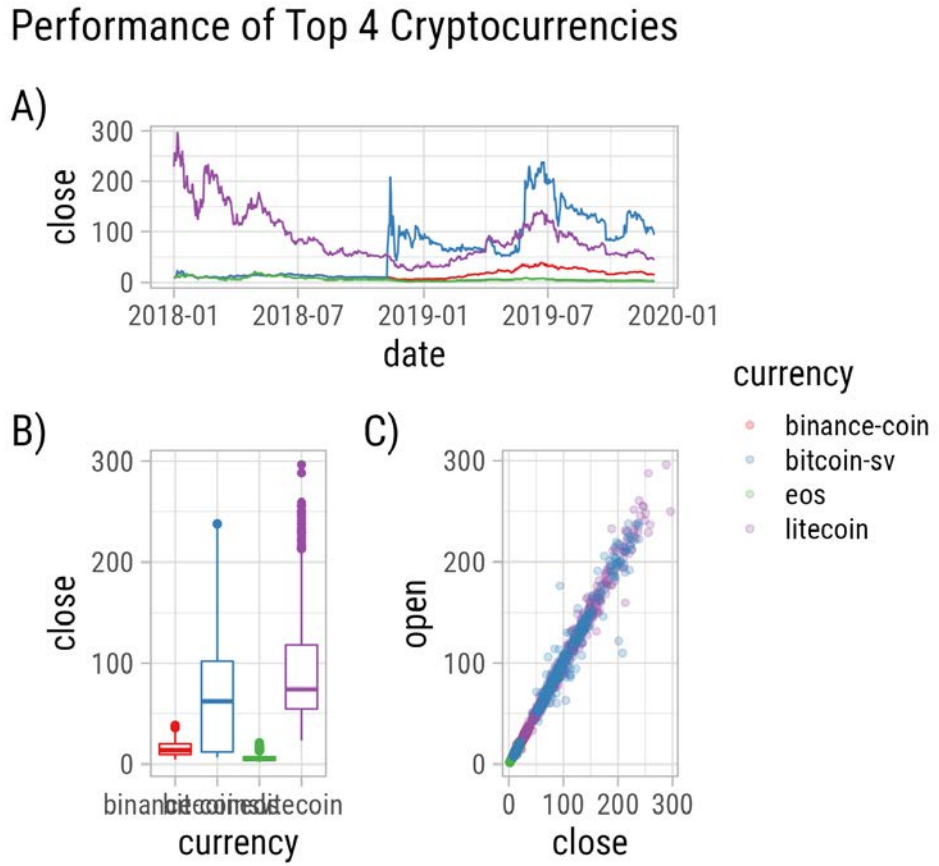
```
time / (box + scatter_guide) +
  plot_layout(
    heights = c(1, 2),
    guides = "collect"
  ) +
  plot_annotation(
    title = "Performance of Top 4 Cryptocurrencies"
  )
```



The patchwork package

plot_annotation() also comes with the functionality to tag panels in one step:

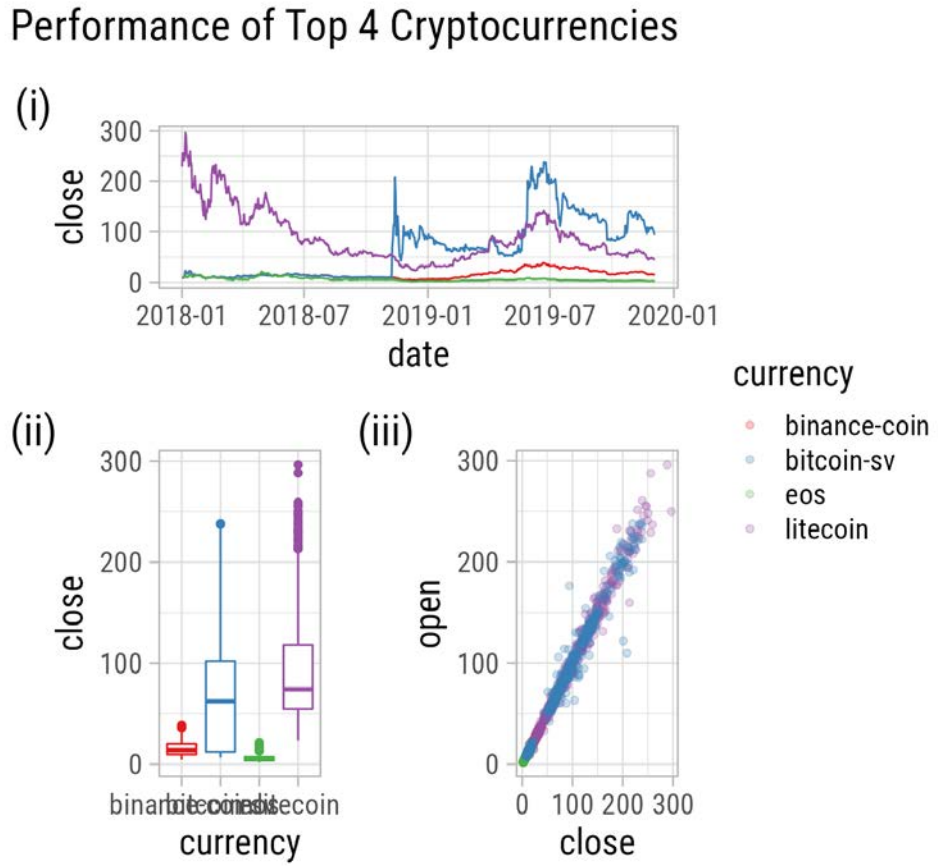
```
time / (box + scatter_guide) +
  plot_layout(
    heights = c(1, 2),
    guides = "collect"
  ) +
  plot_annotation(
    title = "Performance of Top 4 Cryptocurrencies",
    tag_levels = "A",
    tag_suffix = ")"
  )
```



The patchwork package

`plot_annotation()` also comes with the functionality to tag panels in one step:

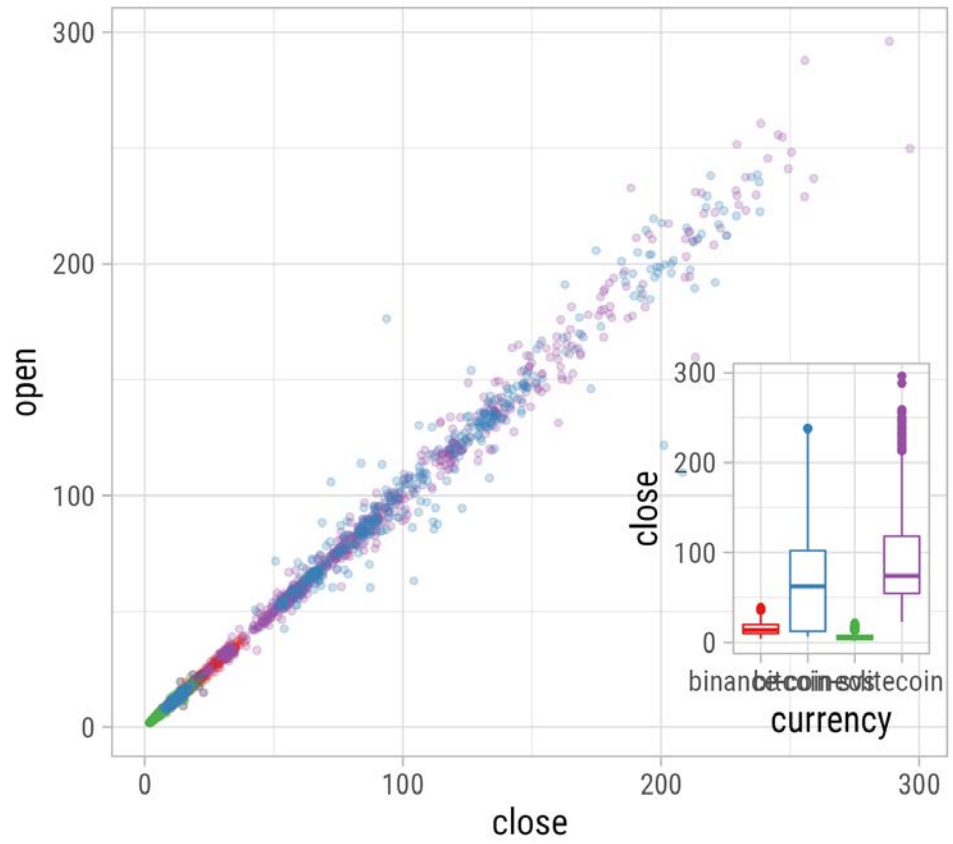
```
time / (box + scatter_guide) +
  plot_layout(
    heights = c(1, 2),
    guides = "collect"
  ) +
  plot_annotation(
    title = "Performance of Top 4 Cryptocurrencies",
    tag_levels = "i",
    tag_prefix = "(",
    tag_suffix = ")"
  )
```



The patchwork package

{patchwork} also allows to place plots as insets:

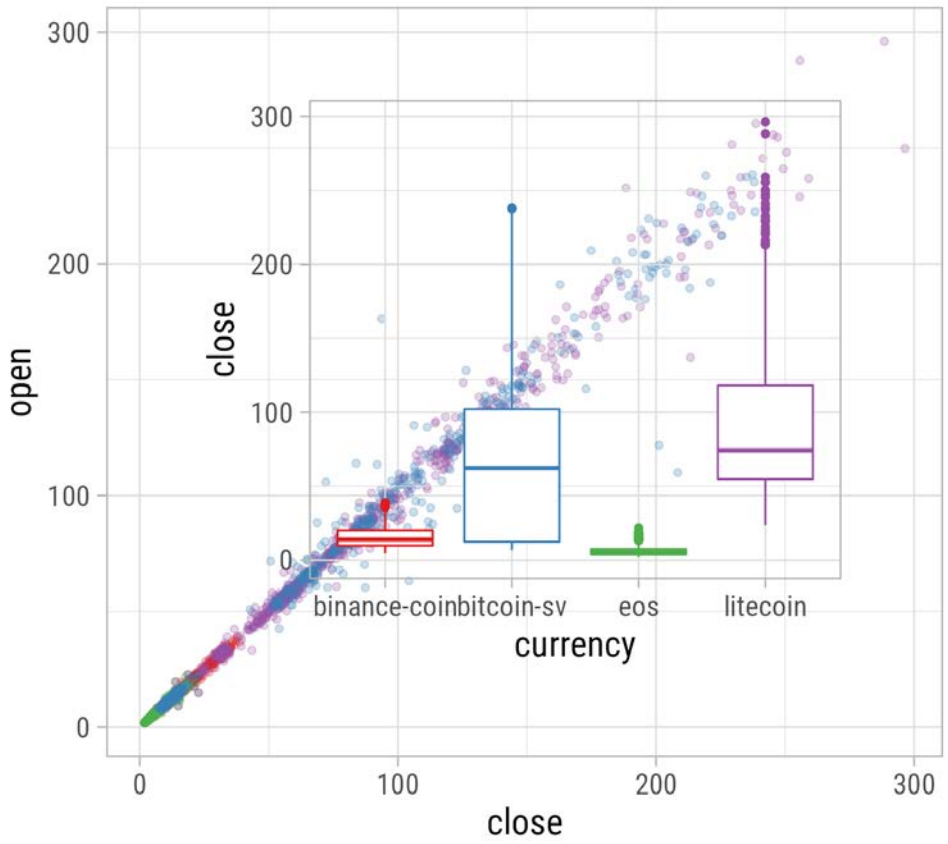
```
box <- box +
  theme(plot.background = element_blank())

scatter +
  inset_element(
    box,
    top = .55,
    right = 1,
    bottom = 0,
    left = .6
  )
```



The patchwork package

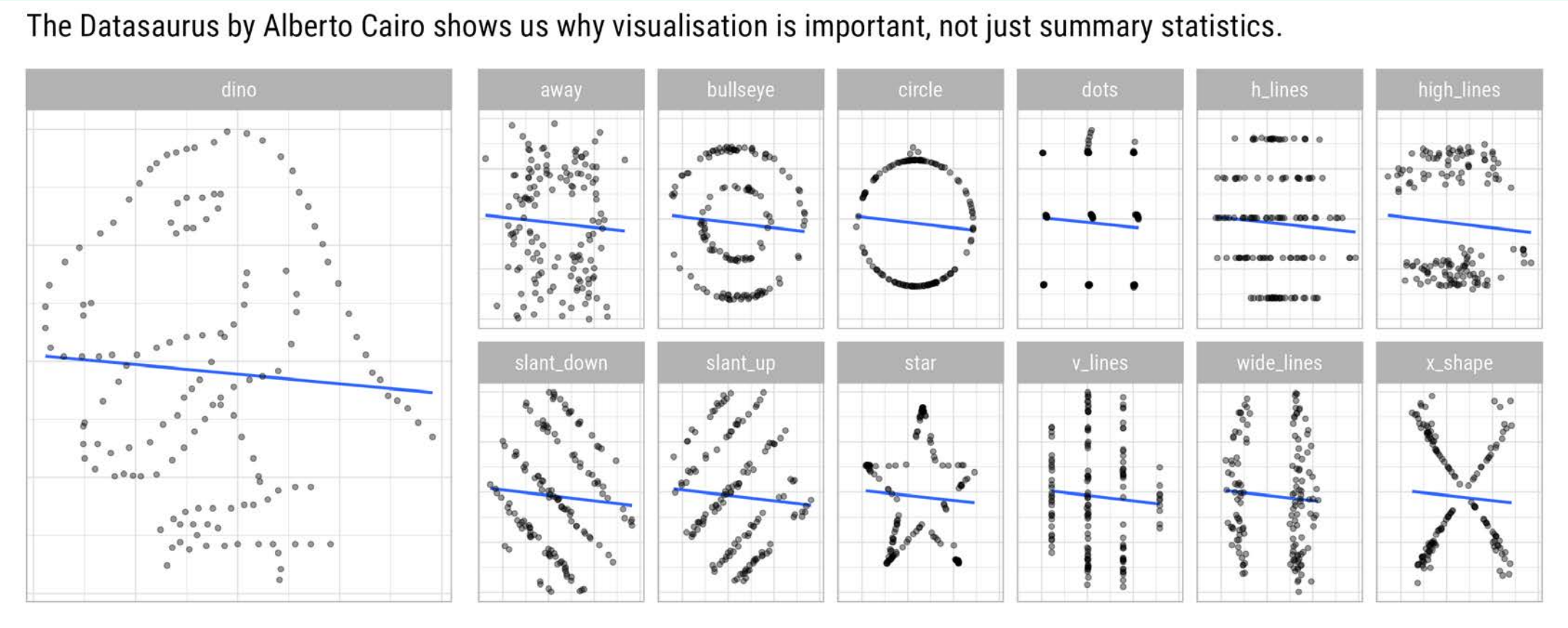
{patchwork} also allows to place plots as insets:

```
box <- box +
  theme(panel.background = element_blank())

scatter +
  inset_element(
    box,
    top = .9,
    right = .9,
    bottom = .1,
    left = .1
  )
```



Exercise 2:

- Using the `"Datasaurus Dozen"` data set and the previous codes, create this multipanel visualization with the help of `facet_wrap()` and the `{patchwork}` package.



Exercise 2: Create Plots

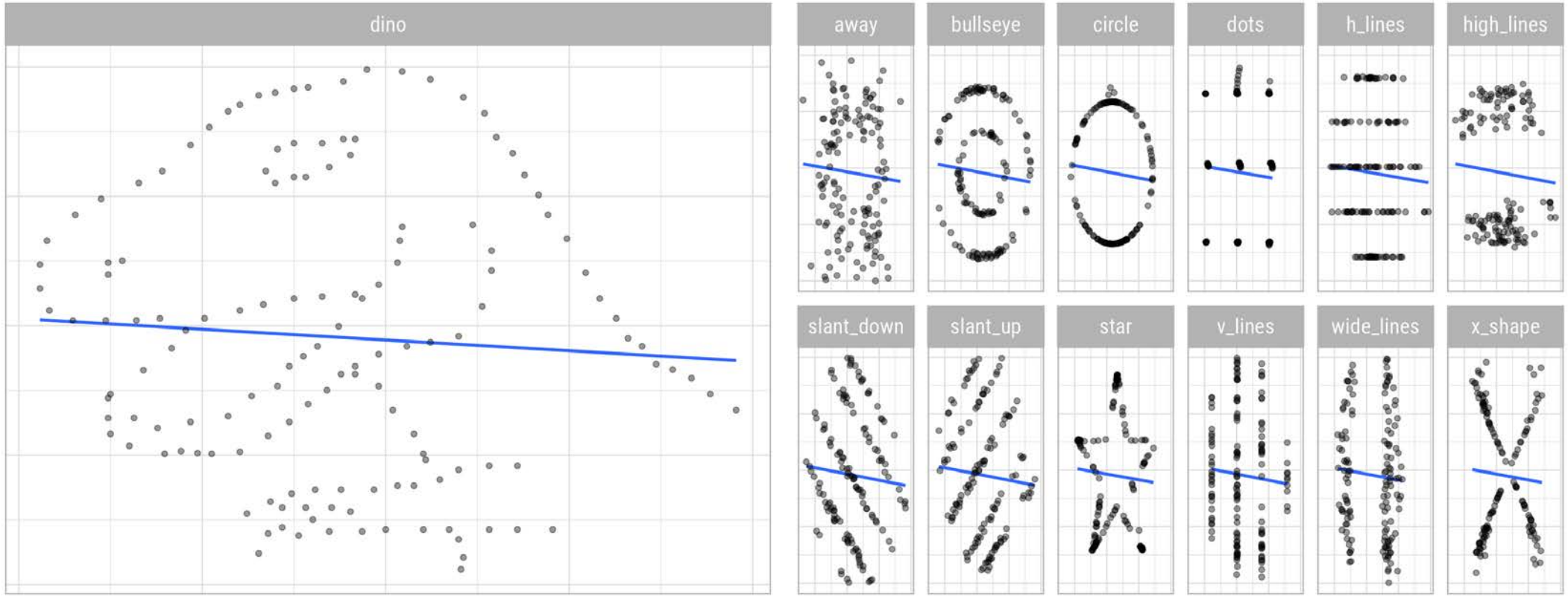
```
facet <-
  ggplot(filter(saurus, dataset != "dino"), aes(x, y)) +
    geom_smooth(method = "lm", se = FALSE) +
    geom_point(alpha = .4) +
    facet_wrap(~ dataset, nrow = 2) +
    scale_x_continuous(guide = "none", name = NULL) +
    scale_y_continuous(guide = "none", name = NULL)
```

Exercise 2: Create Plots

```
facet <-
  ggplot(filter(saurus, dataset != "dino"), aes(x, y)) +
    geom_smooth(method = "lm", se = FALSE) +
    geom_point(alpha = .4) +
    facet_wrap(~ dataset, nrow = 2) +
    scale_x_continuous(guide = "none", name = NULL) +
    scale_y_continuous(guide = "none", name = NULL)

dino <-
  ggplot(filter(saurus, dataset == "dino"), aes(x, y)) +
    geom_smooth(method = "lm", se = FALSE) +
    geom_point(alpha = .4) +
    facet_wrap(~ dataset) +
    scale_x_continuous(guide = "none", name = NULL) +
    scale_y_continuous(guide = "none", name = NULL)
```

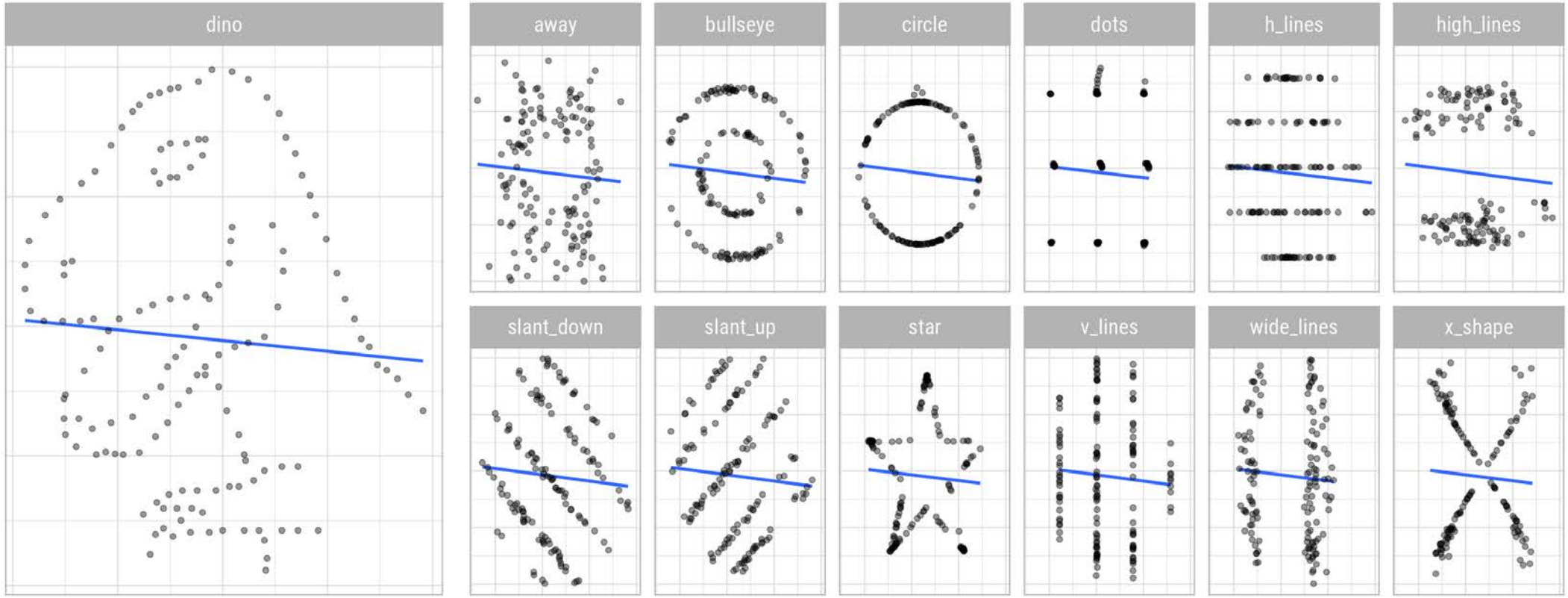
Exercise 2: Create multipanel Plot

```
dino + facet
```



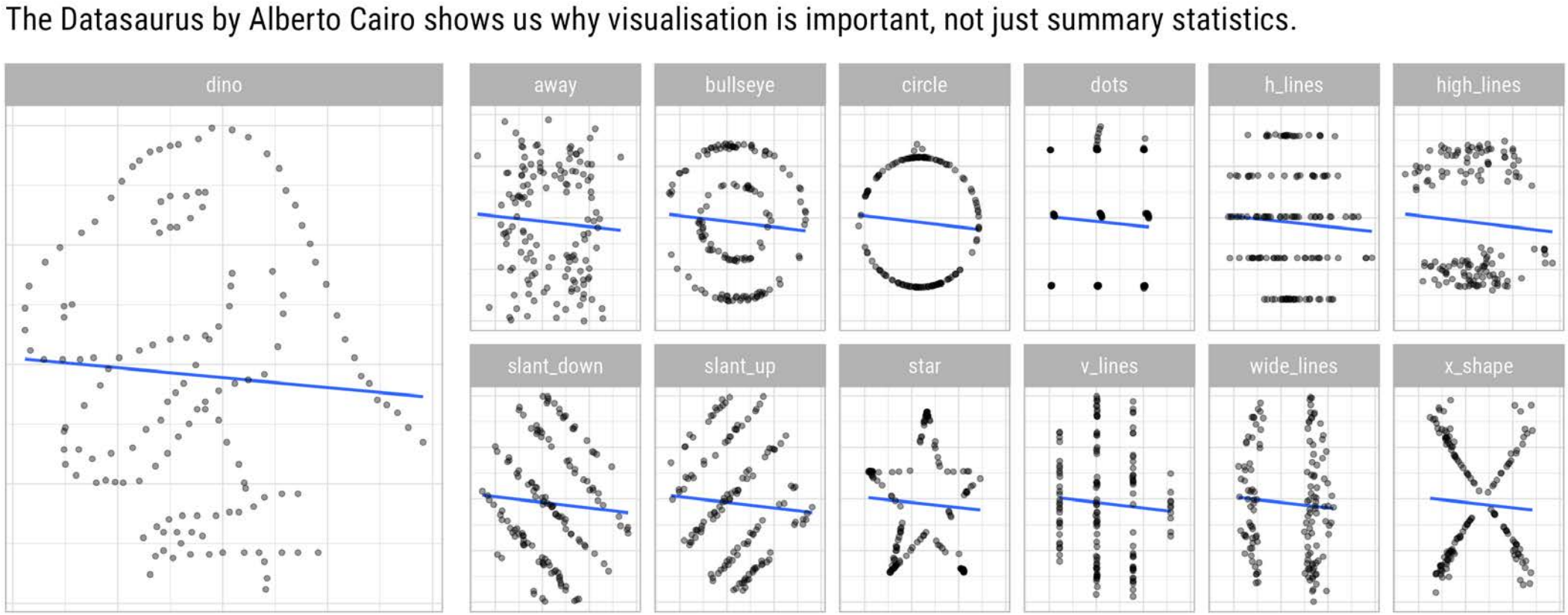
Exercise 2: Adjust Widths

```
dino + facet + plot_layout(widths = c(1, 2.5))
```



Exercise 2: Add the Title

```
dino + facet + plot_layout(widths = c(1, 2.5)) +
  plot_annotation(title = "The Datasaurus by Alberto Cairo shows us why visualisation is important, not ju
```



Resources

- Chapters on `faceting` and `arranging plots` of the “ggplot2” book by Hadley Wickham et al.
- `{patchwork} package reference` with lots of articles on ho to create (more complex) layouts
- `{cowplot} package reference`, another package to arrange multiple plots in a grid
  (and to add insets and images)
- `How to add annotations to individual facets`, part of the “R Graphics Cookbook” book by Winston Chang
- `“A {ggplot2} Tutorial for Beautiful Plotting in R”`, my extensive "how to"-tutorial