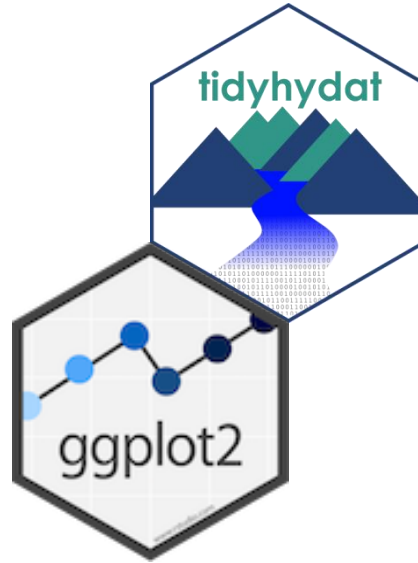


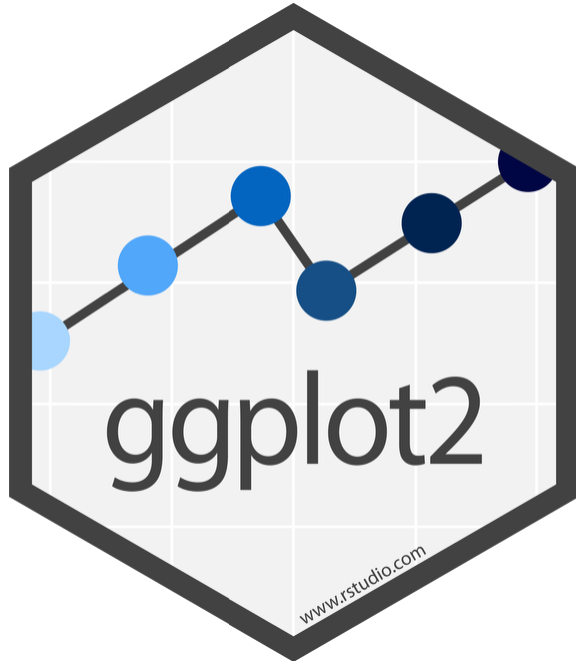
Visualize Data with



"The simple graph has brought more information to the data analyst's mind than any other device. "

- John Tukey

ggplot2



One of the earliest members of the tidyverse.

Complicated plots, come from combining simple components.



Data for this section

The number of stations and average length of record by real time and activity status

```
data_range <- hy_stn_data_range() %>%  
  group_by(STATION_NUMBER) %>%  
  summarise(total_record_length = sum(RECORD_LENGTH, na.rm =  
TRUE))  
  
cdn_prov <- c("NB", "PE", "NS", "ON", "QC", "NL", "MB", "AB",  
"SK", "NU", "NT", "BC", "YT")  
station_meta <- hy_stations() %>%  
  filter(PROV_TERR_STATE_LOC %in% cdn_prov) %>%  
  group_by(PROV_TERR_STATE_LOC, HYD_STATUS, REAL_TIME) %>%  
  left_join(data_range, by = c("STATION_NUMBER")) %>%  
  summarise(number_stns = n(),  
            mean_record_length = mean(total_record_length,  
                                      na.rm = TRUE))
```



station_meta

```
# A tibble: 41 x 5
```

```
# Groups:   PROV_TERR_STATE_LOC, HYD_STATUS [?]
```

	PROV_TERR_STATE_LOC	HYD_STATUS	REAL_TIME	number_stns	mean_record_length
	<chr>	<chr>	<lgl>	<int>	<dbl>
1	AB	ACTIVE	FALSE	123	43.9
2	AB	ACTIVE	TRUE	331	57
3	AB	DISCONTINUED	FALSE	609	15.8
4	AB	NA	FALSE	2	3
5	BC	ACTIVE	FALSE	78	46.6
6	BC	ACTIVE	TRUE	360	53.6
7	BC	DISCONTINUED	FALSE	1842	10.9
8	BC	NA	FALSE	1	8
9	MB	ACTIVE	FALSE	91	34.8
10	MB	ACTIVE	TRUE	221	63.2

```
# ... with 31 more rows
```

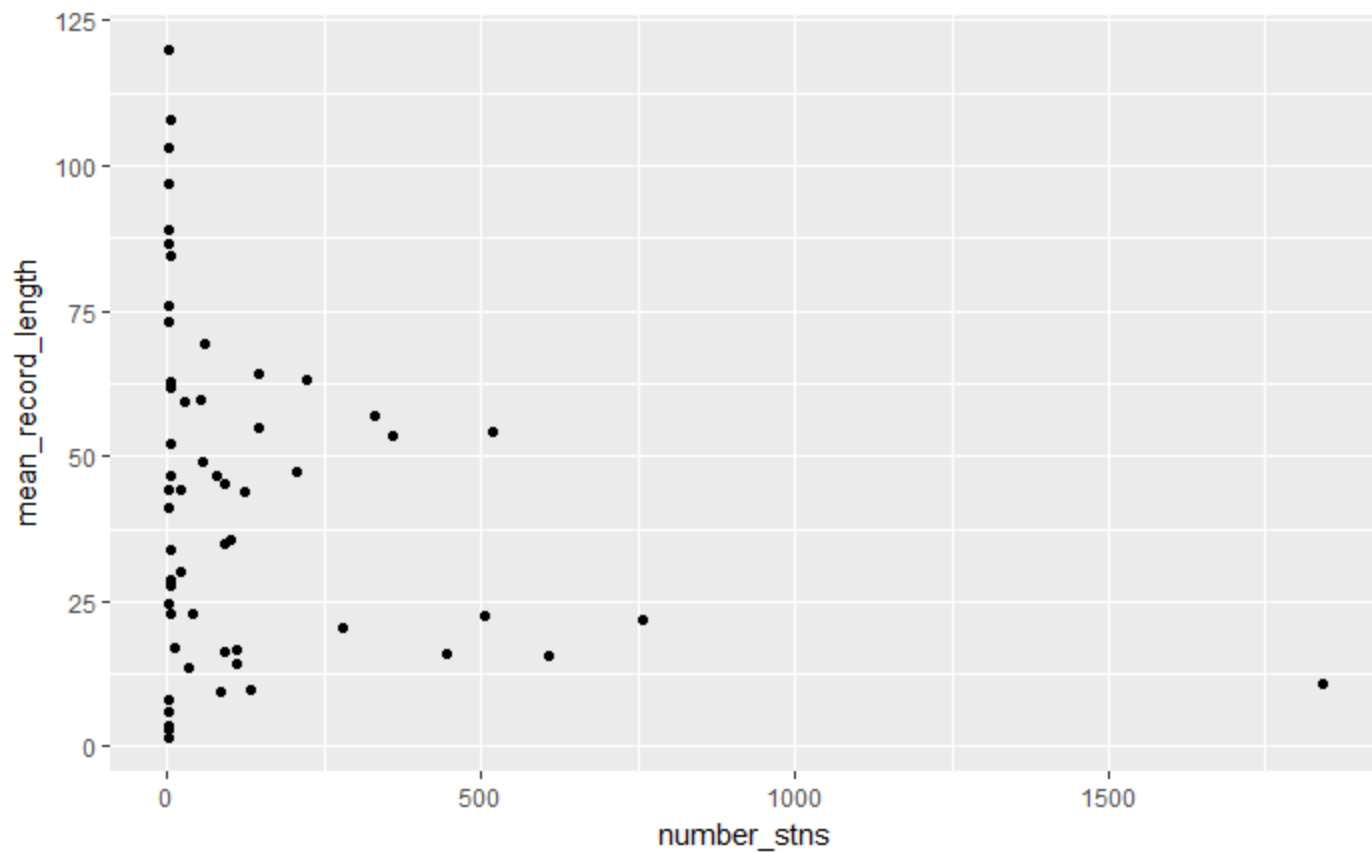
Your turn

Confer with your neighbours.

What relationships might interest you with this data?

Let's plot it:

```
ggplot(data = station_meta, mapping = aes(x =  
number_stns, y = mean_record_length)) +  
  geom_point()
```



1. "Initialize" a plot with `ggplot()`
2. Add layers with `geom_functions`

```
ggplot(data = station_meta) +  
  geom_point(mapping = aes(x = number_stns,  
                           y = mean_record_length))
```


data

+ before new line

```
ggplot(data = station_meta) +  
  geom_point(mapping = aes(x = number_stns,  
                           y = mean_record_length))
```

type of layer

aes()

variables



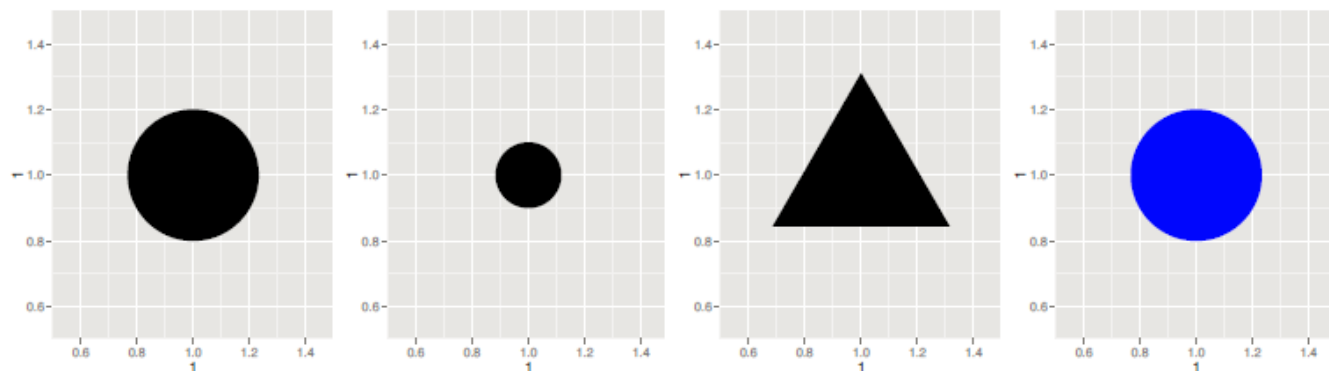
A Template

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Mappings

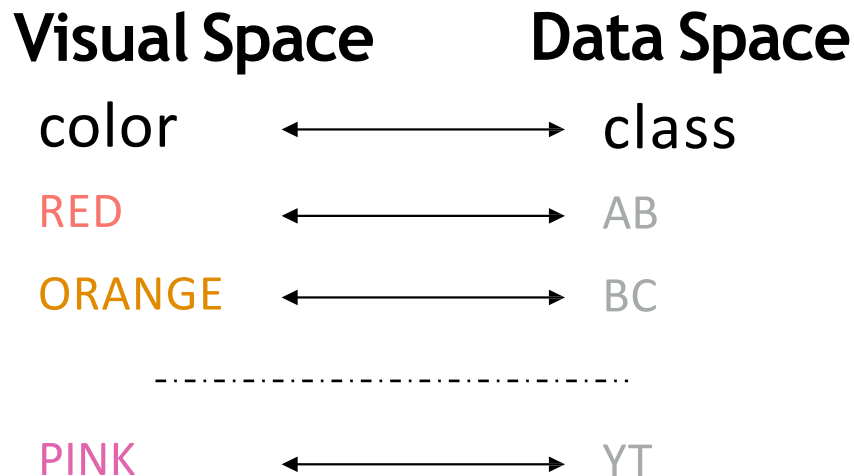
Aesthetics

Visual properties of a geometric object



How do the appearance of these points vary?

Mappings describe how aesthetics should relate to variables in the data.



Aesthetic

```
ggplot(data = station_meta) +  
  geom_point(mapping = aes(x = number_stns,  
                           y = mean_record_length,  
                           colour = PROV_TERR_STATE_LOC))
```

aesthetic
property

Variable to
map it to



Aesthetic

```
ggplot(data = station_meta) +  
  geom_point(mapping = aes(x = number_stns,  
                           y = mean_record_length,  
                           shape = HYD_STATUS))
```

aesthetic
property

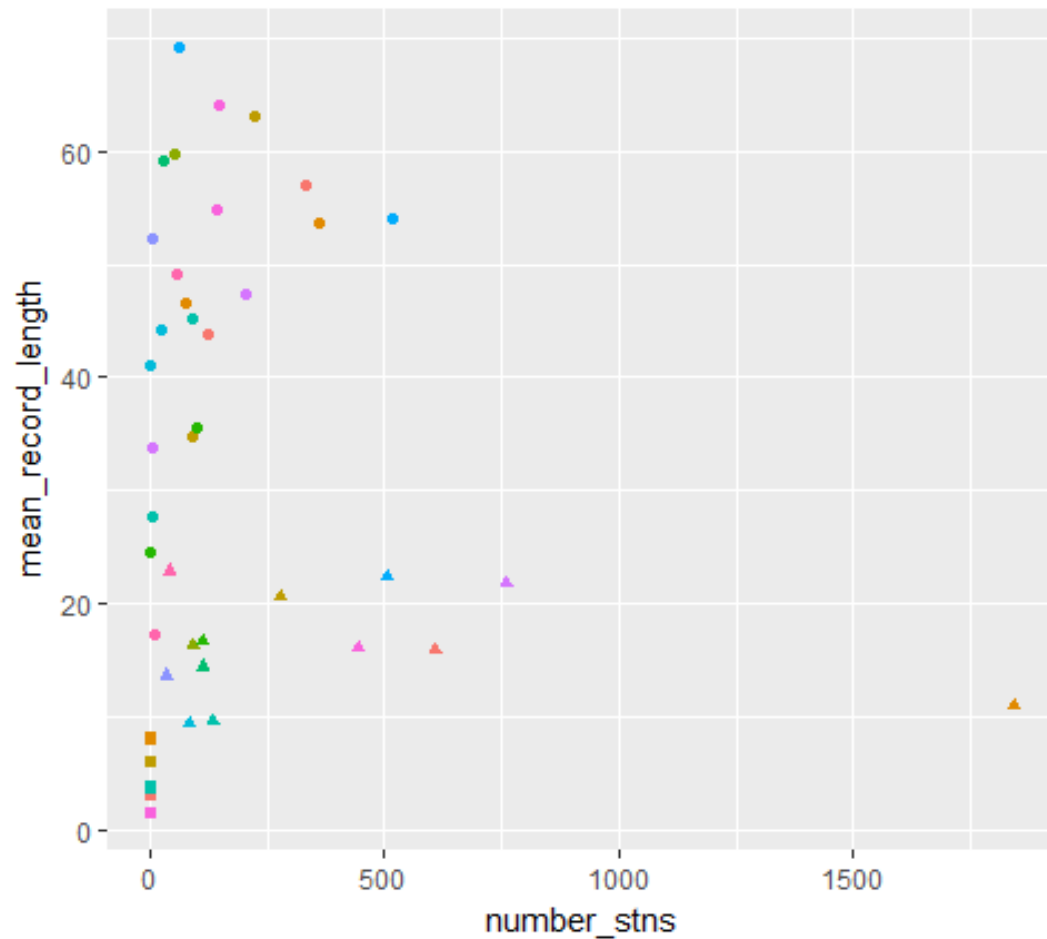
Variable to
map it to



Aesthetic

```
ggplot(data = station_meta) +  
  geom_point(mapping = aes(x = number_stns,  
                           y = mean_record_length,  
                           colour = PROV_TERR_STATE_LOC,  
                           shape = HYD_STATUS))
```





- ACTIVE
- ▲ DISCONTINUED
- NA

PROV_TERR_STATE_LOC

- AB
- BC
- MB
- NB
- NL
- NS
- NT
- NU
- ON
- PE
- QC
- SK
- YT

Legends
added
automatically



Your turn

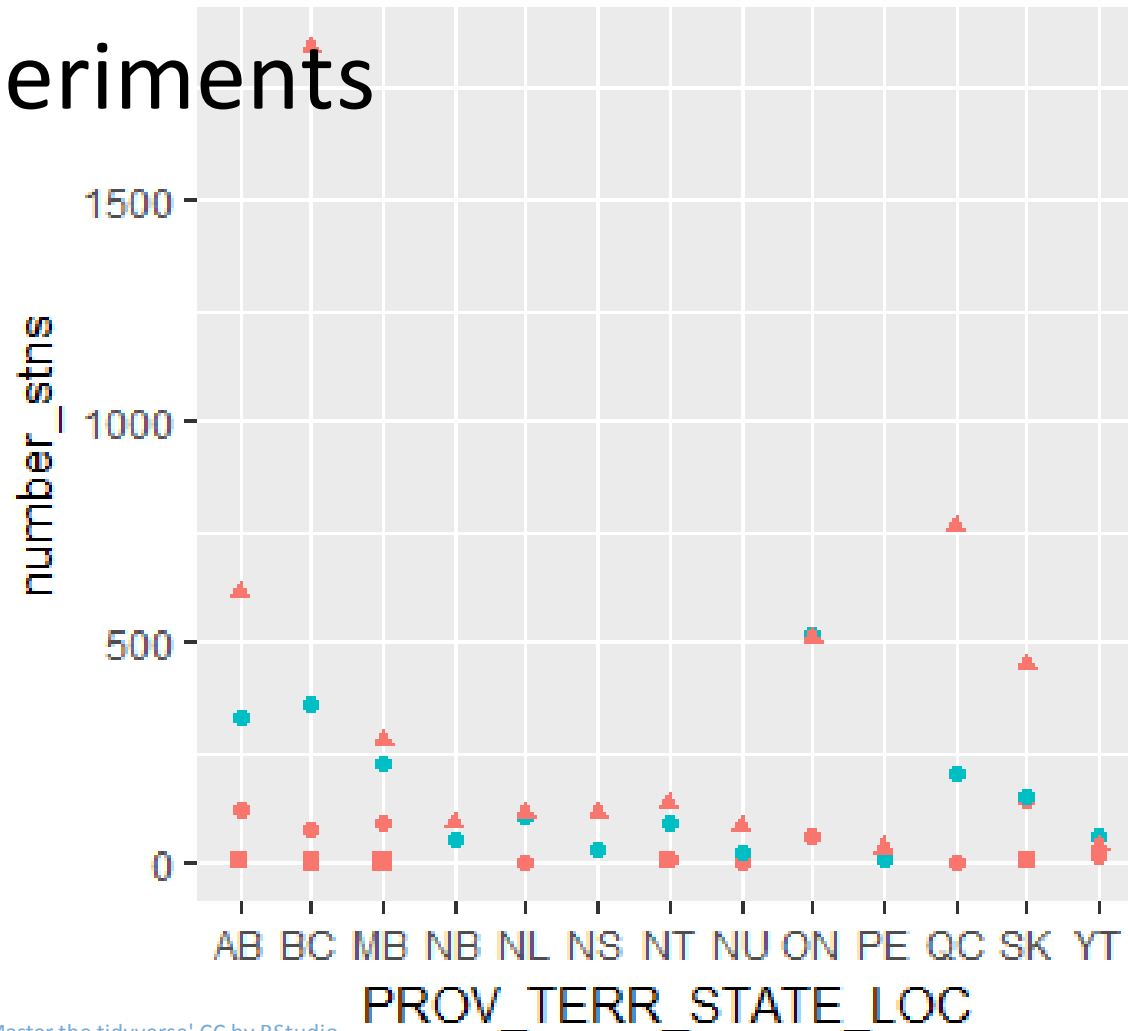
In the next chunk, modify which data is mapped to x, y, color, size, alpha, and shape aesthetics to your graph. Experiment. Generate two different examples.

Experiments

```
ggplot(data = station_meta, mapping = aes(x = PROV_TERR_STATE_LOC,  
                                           y = number_stns,  
                                           shape = HYD_STATUS,  
                                           colour = REAL_TIME)) +  
  geom_point()
```



Experiments



HYD_STATUS

- ACTIVE
- ▲ DISCONTINUED
- NA

REAL_TIME

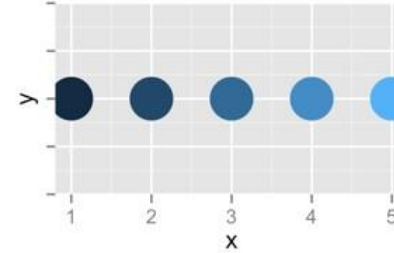
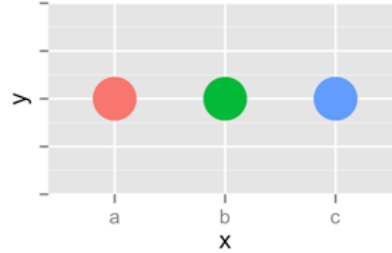
- FALSE
- TRUE



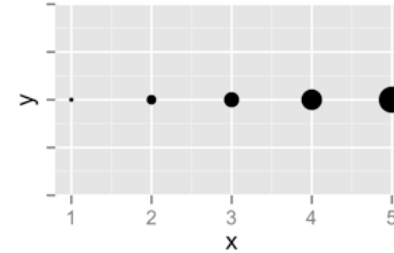
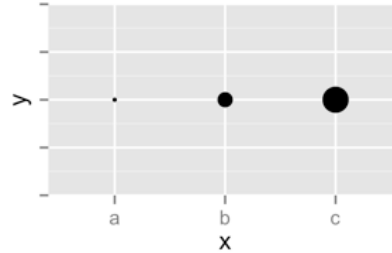
Discrete

Continuous

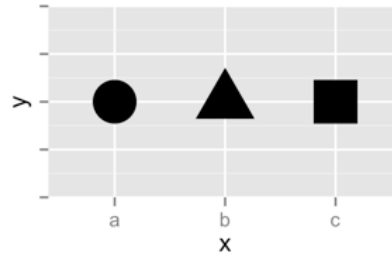
Color



Size



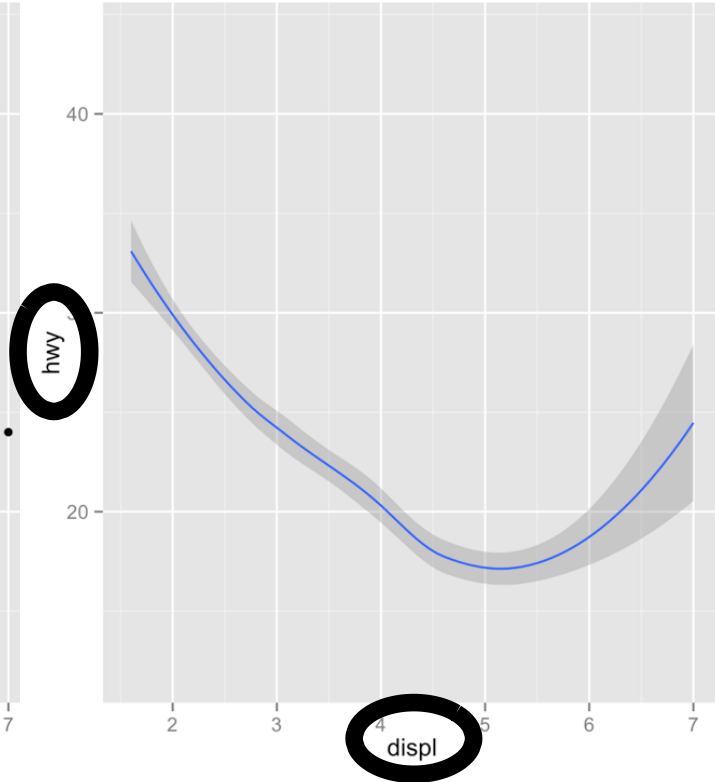
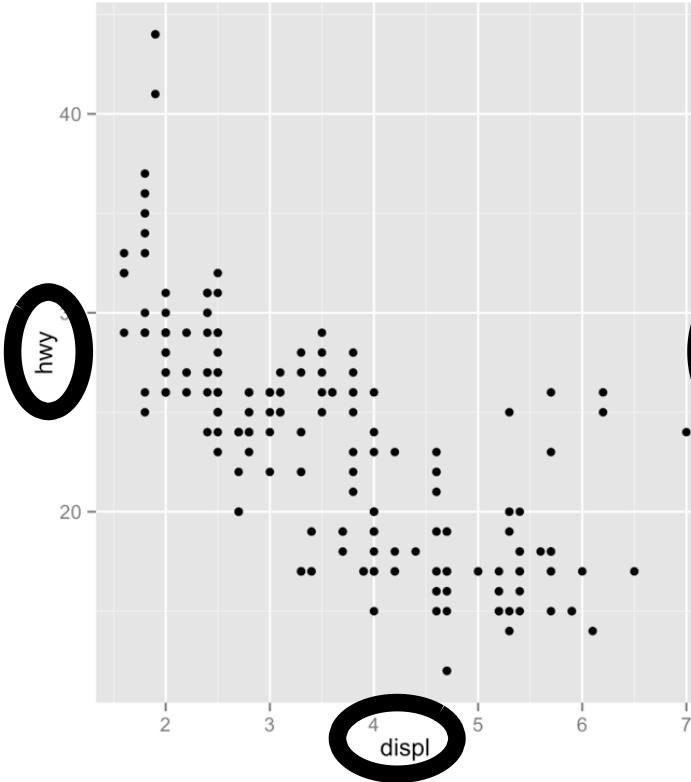
Shape



geoms

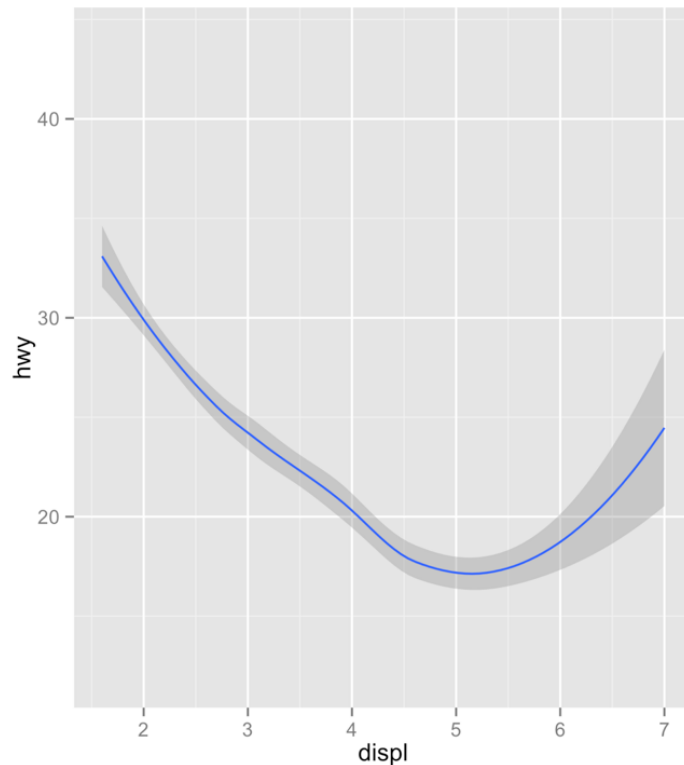
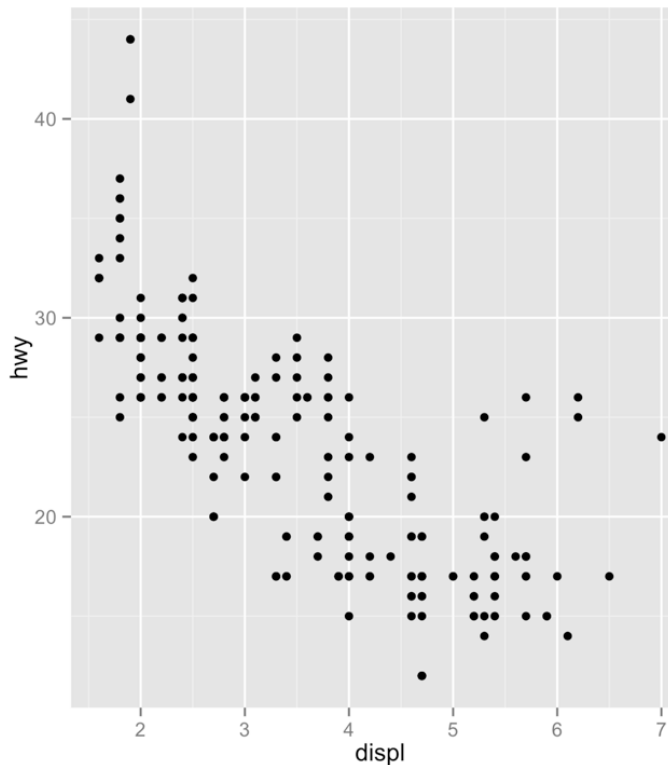
How are these
plots similar?

Same X and Y



How are these
plots different?

Different: geometric object (geom)
e.g. the visual object used to represent the data



geoms

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

geom_functions

Every geom requires a mapping argument.

Data Visualization with ggplot2 :: CHEAT SHEET



R Studio



Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables.
Each function returns a layer.

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))  
b <- ggplot(economics, aes(x = long, y = lat))
```

a = geom_blank()
(Useful for expanding limits)

**b = geom_curve(aes(yend = lat + 1,
xend = long + 1, curvature = 2), x = x, y = yend,
alpha, angle, color, curvature, linetype, size)**

**c = geom_path(linetype = "solid", linjoin = "round",
linelimit = 1)**
x, y, alpha, color, group, linetype, size

d = geom_polygon(aes(group = group))
x, y, alpha, color, fill, group, linetype, size

**e = geom_rect(aes(xmin = long, ymin = lat, xmax =
long + 1, ymax = lat + 1), x = x, y = y, alpha, color,
fill, linetype, size)**

**f = geom_ribbon(aes(ymin = unemploy - 500,
ymax = unemploy + 500))** x, y, alpha, color, fill, group, linetype, size

LINE SEGMENTS

Common aesthetics: x, y, alpha, color, linetype, size

a = geom_abline(aes(intercept = 0, slope = 1))
b = geom_hline(aes(yintercept = lat))
c = geom_vline(aes(xintercept = long))

d = geom_segment(aes(yend = lat + 1, xend = long + 1))
e = geom_spoke(aes(angle = 12.155, radius = 1))

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy))  
d <- ggplot(mpg)
```

a = geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

b = geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c = geom_dotplot()
x, y, alpha, color, fill

**d = geom_freqpoly(x, y, alpha, color, group,
linetype, size)**

e = geom_histogram(binwidth = 5) x, y, alpha,
color, fill, linetype, size, weight

f = geom_qq(aes(sample = hwy)) x, y, alpha,
color, fill, linetype, size, weight

TWO VARIABLES

continuous x, continuous y

```
e <- ggplot(mpg, aes(cty, hwy))
```

**a = geom_label(aes(label = cty), nudges = 1,
nudge_x = 1, check_overlap = TRUE)** x, y, label,
alpha, angle, color, family, fontface, hjust,
lheight, size, vjust

b = geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size

c = geom_point() x, y, alpha, color, fill, shape,
size, stroke

d = geom_quantile() x, y, alpha, color, group,
linetype, size, weight

e = geom_rug(sides = "bl") x, y, alpha, color,
linetype, size

f = geom_smooth(method = "lm") x, y, alpha,
color, fill, group, linetype, size, weight

**g = geom_text(aes(label = cty), nudges = 1,
nudge_x = 1, check_overlap = TRUE)** x, y, label,
alpha, angle, color, family, fontface, hjust,
lheight, size, vjust

discrete x, continuous y

```
f <- ggplot(mpg, aes(class, hwy))
```

a = geom_col() x, y, alpha, color, fill, group,
linetype, size

b = geom_boxplot() x, y, lower, middle, upper,
ymin, ymax, alpha, color, fill, group, linetype,
shape, size, weight

**c = geom_dotplot(binaxis = "y", stackorder =
"center")** x, y, alpha, color, fill, group

d = geom_violin(scale = "area") x, y, alpha, color,
fill, group, linetype, size, weight

discrete x, discrete y

```
g <- ggplot(diamonds, aes(cut, color))
```

a = geom_count() x, y, alpha, color, fill, shape,
size, stroke

THREE VARIABLES

```
seals2 <- with(seals, sqrt(pelt_area * delta_year * 2))  
ggplot(seals, aes(long, lat))
```

continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))
```

a = geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

b = geom_density2d()
x, y, alpha, color, group, linetype, size

c = geom_hex()
x, y, alpha, color, fill, size

continuous function

```
i <- ggplot(economics, aes(date, unemploy))
```

a = geom_area()
x, y, alpha, color, fill, linetype, size

b = geom_line()
x, y, alpha, color, group, linetype, size

c = geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4.5, se = 1.2)  
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))
```

a = geom_crossbar(latten = 2)
x, y, ymin, ymax, alpha, color, fill, group, linetype,
size

b = geom_errorbar() x, y, ymax, ymin, alpha, color,
group, linetype, size, width (also
geom_errorbarh())

c = geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

d = geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, group, linetype,
shape, size

maps

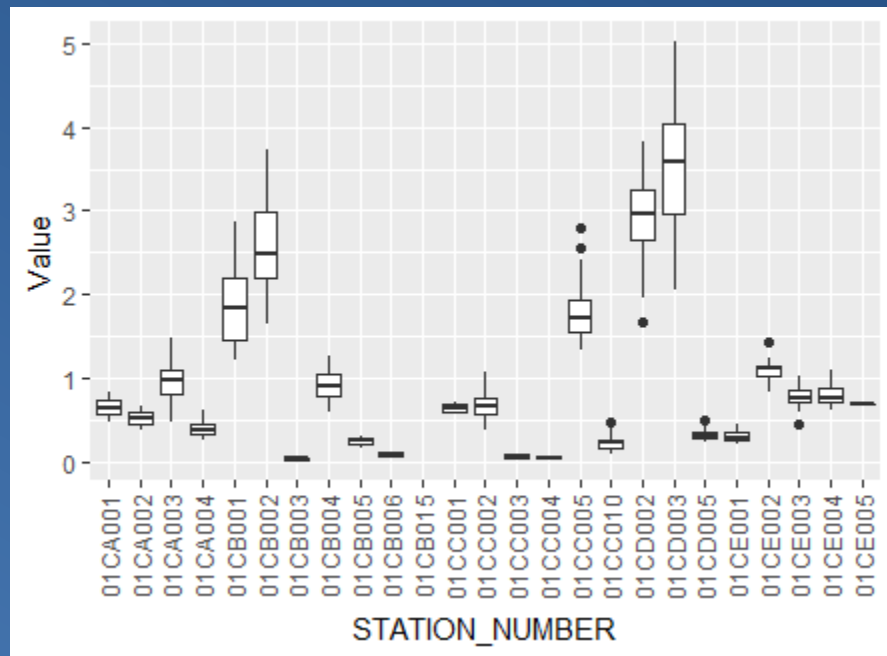
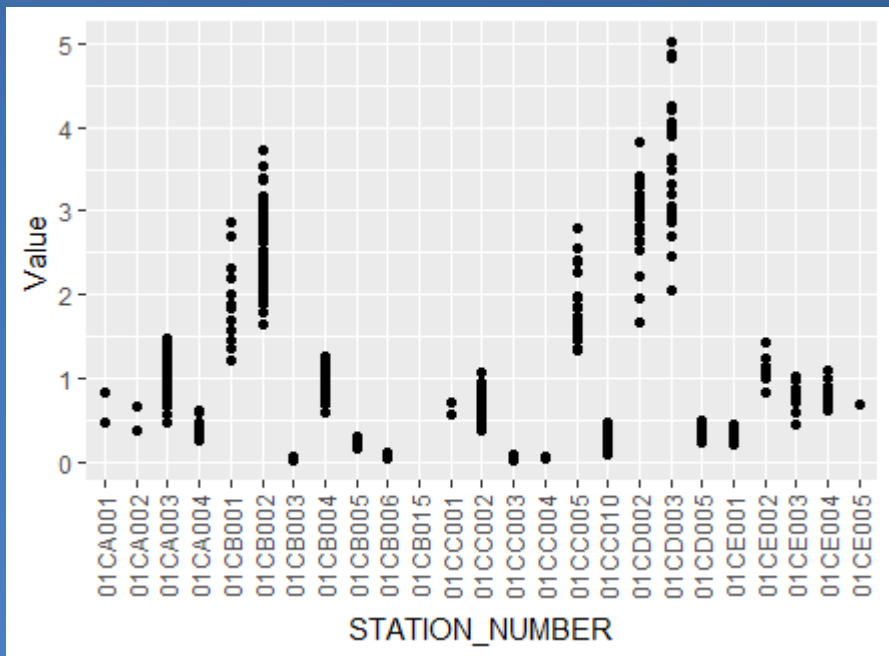
```
data <- data.frame(murder = USArrests$Murder,  
state = tolower(row.names(USArrests)))  
map <- map_data("state")  
k <- ggplot(data, aes(fill = murder))
```

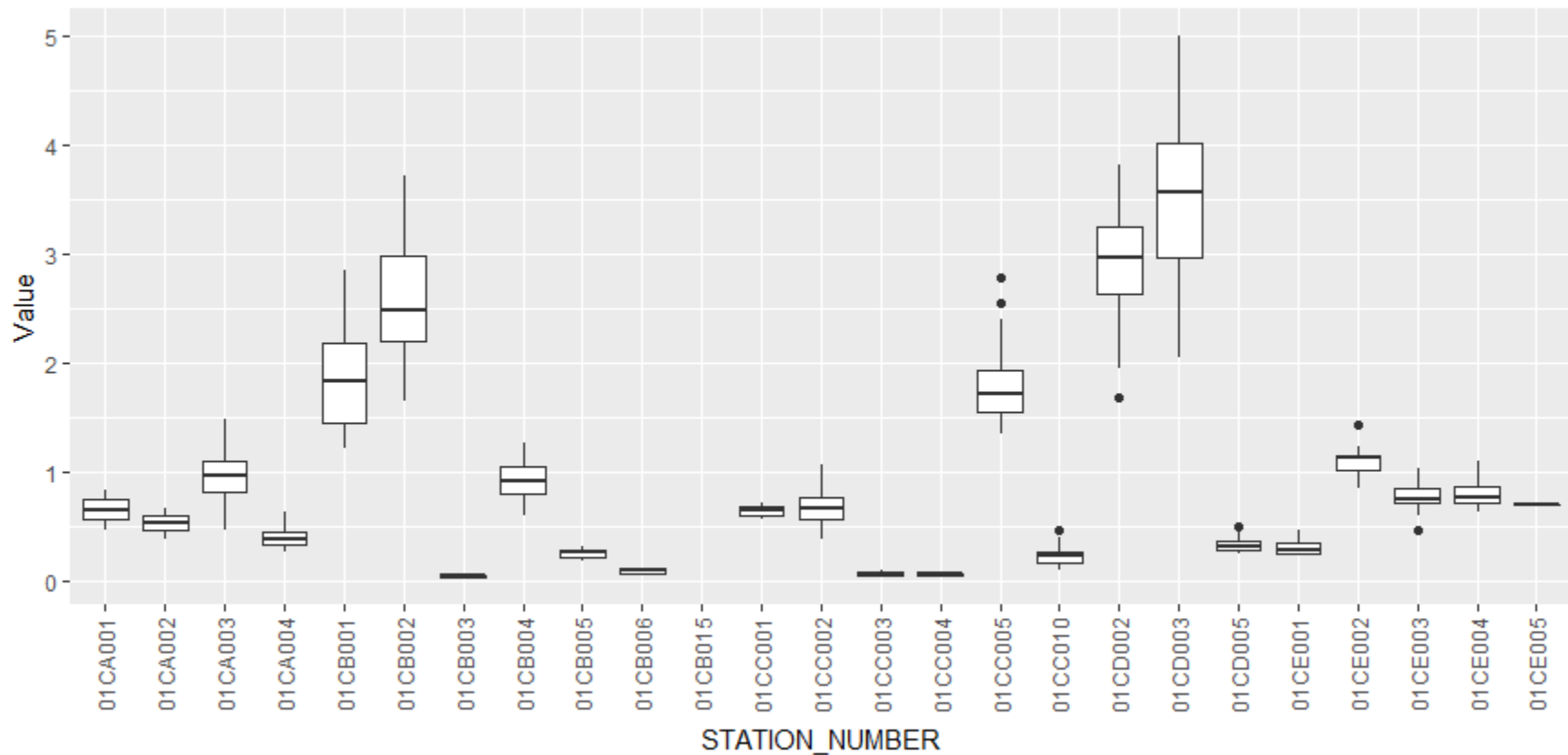
a = geom_map(aes(map_id = state), map = map)
+ expand_limits("map_longitude", map_longitude)

b = geom_map(aes(map_id = state), map = map)
+ expand_limits("map_longitude", map_longitude)

Your turn

With your neighbour, discuss how to convert the plot on left to the plot on the right. (Use the cheatsheet and `pei_mean`)





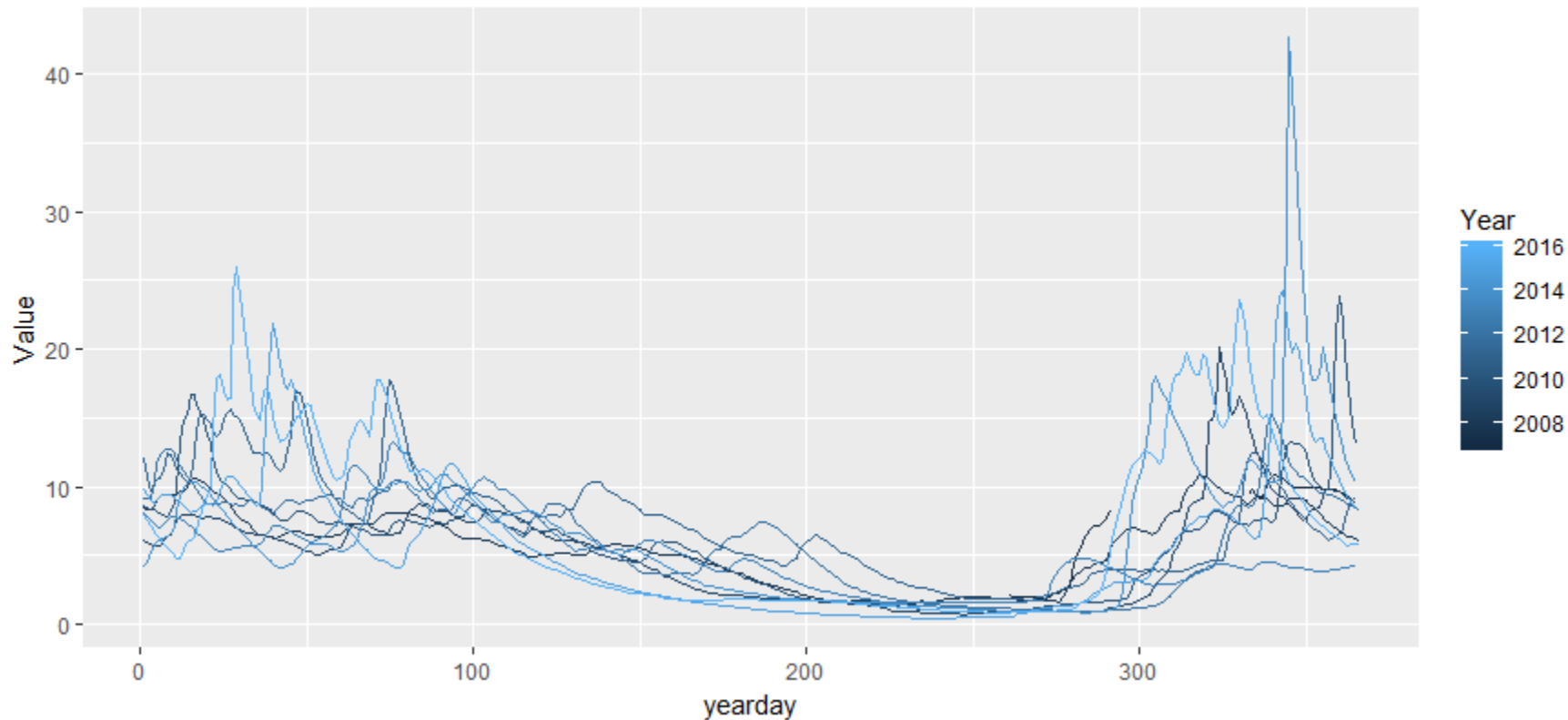
```
ggplot(data = pei_mean) +  
  geom_boxplot(mapping = aes(x = STATION_NUMBER, y = Value)) +  
  theme(axis.text.x = element_text(angle=90, vjust=0.5))
```



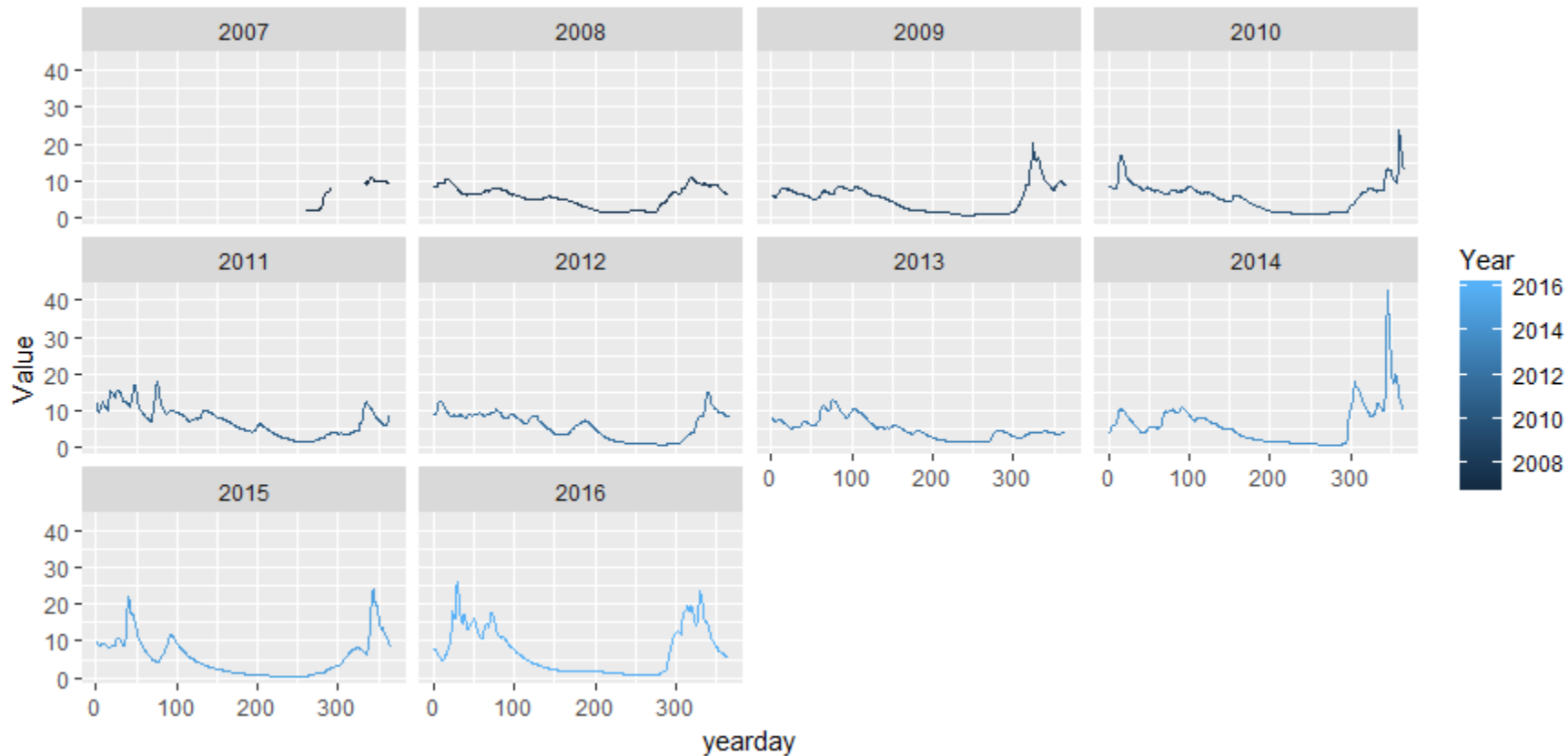
Your turn

Make a hydrograph of the 08GB014 HORSESHOE RIVER ABOVE LOIS LAKE BC station data for each year using the Year and yearday columns created.

Hint checkout ?yday for an explanation of what it does



```
ggplot(data = horseshoe, mapping = aes(x = yearday, y = Value,  
  colour = Year)) +  
  geom_line(mapping = aes(group = Year))
```

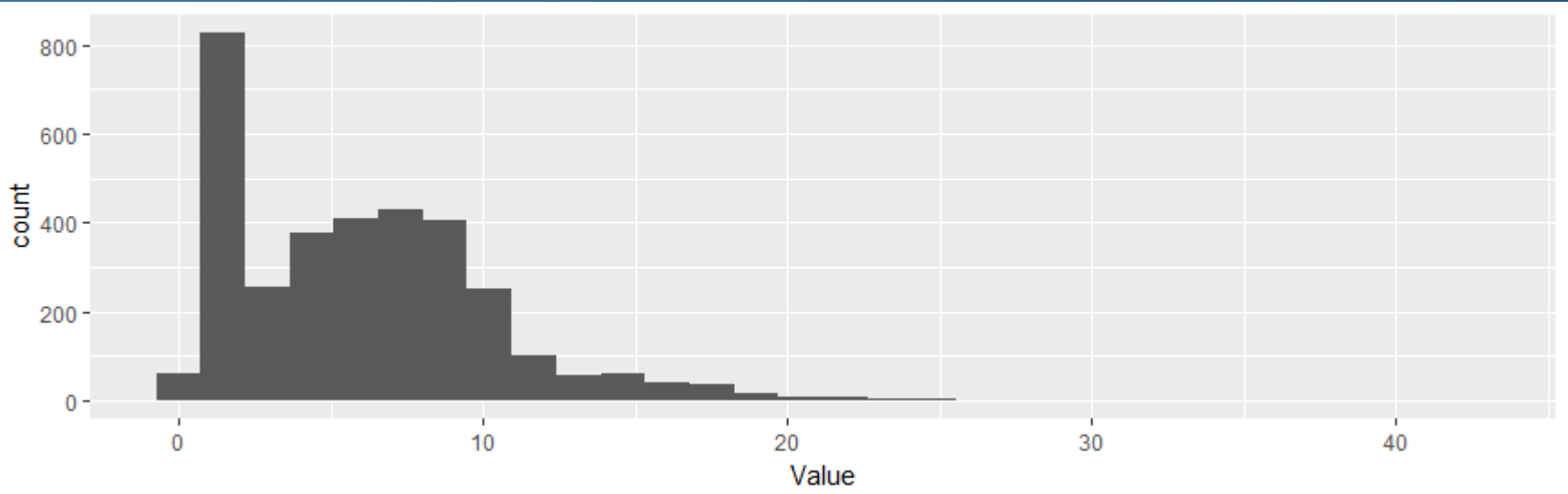


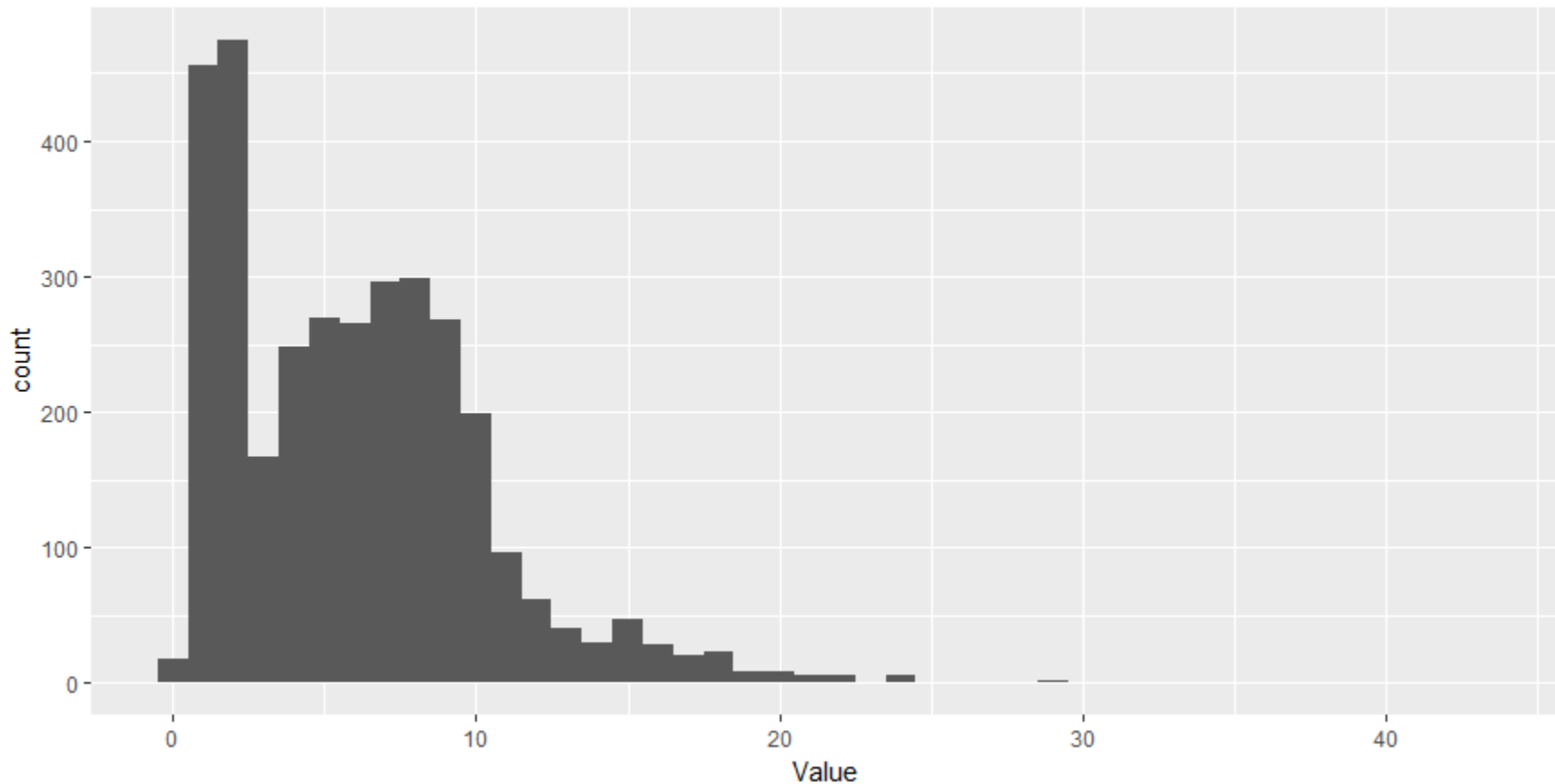
```
ggplot(data = horseshoe, mapping = aes(x = yearday, y = Value, colour =  
Year)) +  
  geom_line(mapping = aes(group = Year)) +  
  facet_wrap(~Year)
```



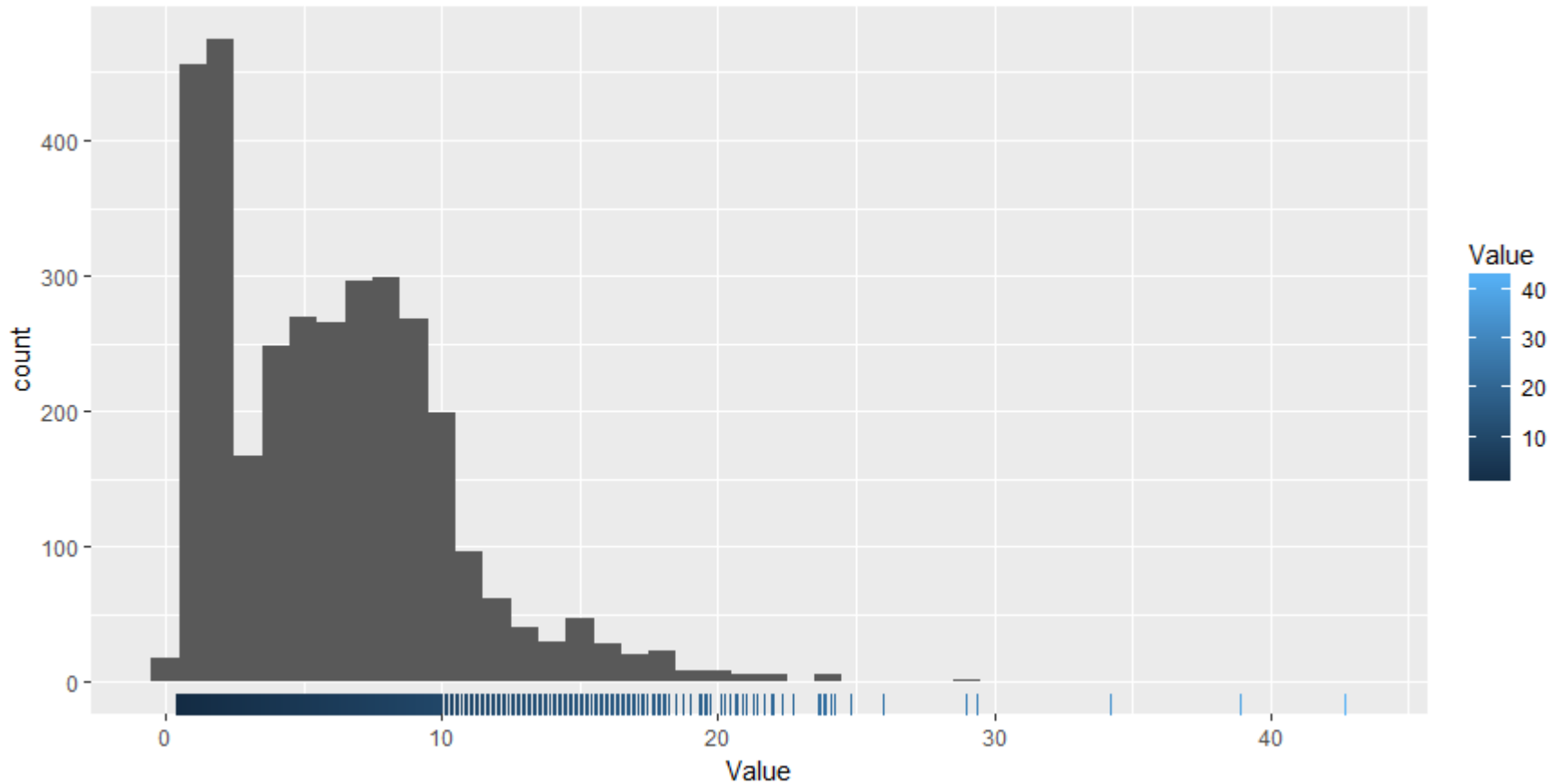
Your turn

With your partner, make the histogram of flow (i.e. Value) from the horseshoe data. Use the cheatsheet. Hint: do not supply a y variable.





```
ggplot(data = horseshoe, mapping = aes(x = Value)) +  
  geom_histogram(binwidth = 1)
```



```
ggplot(data = horseshoe, mapping = aes(x = Value)) +  
  geom_histogram(binwidth = 1) +  
  geom_rug(mapping = aes(colour = Value))
```



Saving graphs

Your turn

What does this command return?

`getwd()`

Working Directory

R associates itself with a folder (i.e. directory) on your computer.

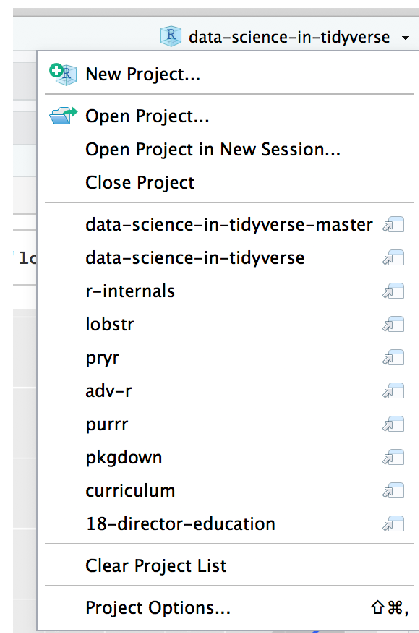
- This folder is known as your "**working directory**"
- When you save files, R will save them here
- When you load files, R will look for them here



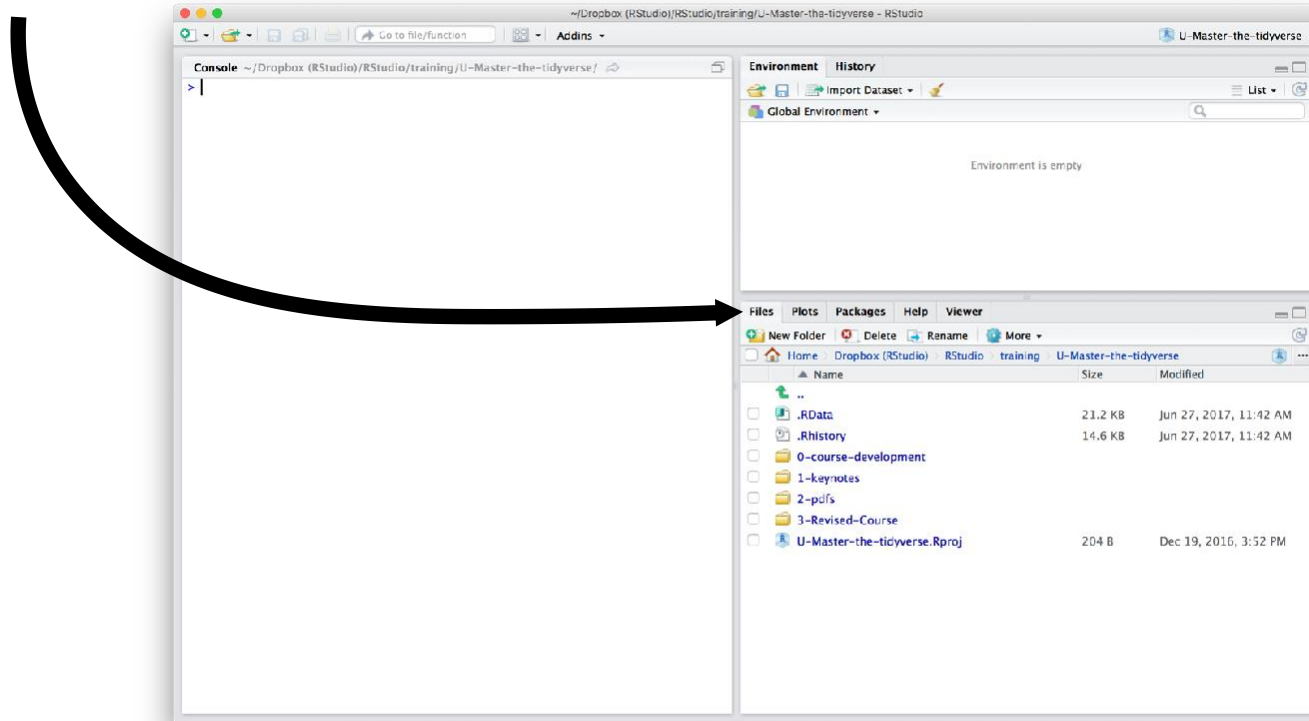
Projects

The best way of managing your working directory is with RStudio Projects.

One RStudio project = one real life project
One RStudio project = one directory



The files pane of the IDE displays the contents of your working directory



Saving plots

`ggsave()` saves the last plot.

Uses size on screen:

```
ggsave("my-plot.pdf")  
ggsave("my-plot.png")
```

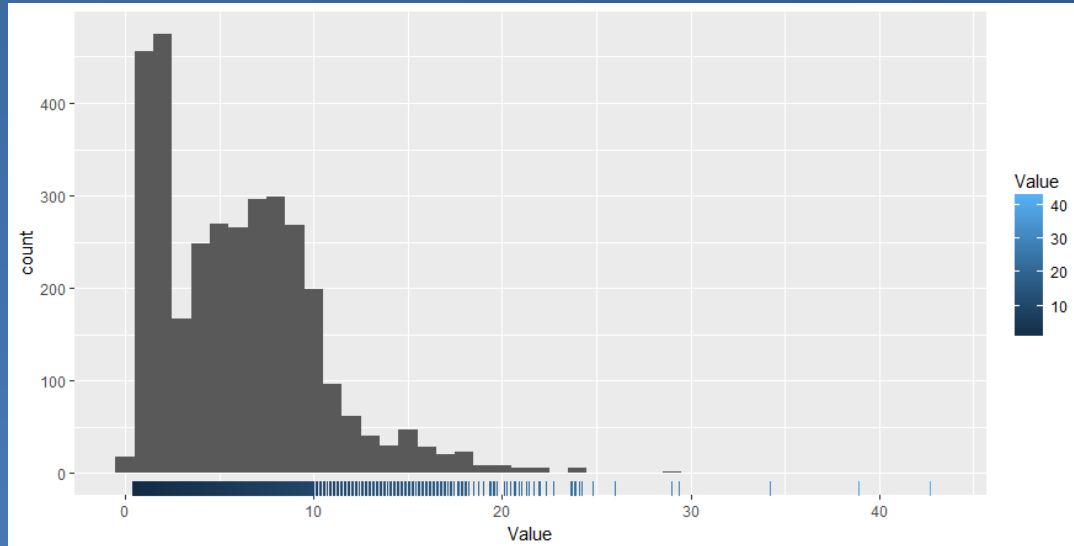
Specify size in inches

```
ggsave("my-plot.pdf", width = 6, height = 6)
```



Your turn

Save your last plot and then locate it in your files pane. (You may have to refresh the files list).



Grammar of Graphics

To make a graph

[template]

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

To make a graph

1. Pick a **dataset**

|

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

data

To make a graph

mpg	cyl	displacement	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

data

geom

1. Pick a **dataset**

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

2. Choose a **geom**
to display cases



To make a graph

mappings

mpg	cyl	disp	hp	fill
21.0	6	160.0	2	●
21.0	6	160.0	2	●
22.8	4	108.0	1	●
21.4	6	258.0	2	●
18.7	8	360.0	3	●
18.1	6	225.0	2	●
14.3	8	360.0	5	●
24.4	4	146.7	1	●
22.8	4	140.8	1	●
19.2	6	167.6	2	●
17.8	6	167.6	2	●
16.4	8	275.8	3	●
17.3	8	275.8	3	●
15.2	8	275.8	3	●
10.4	8	472.0	4	●
10.4	8	460.0	4	●
14.7	8	440.0	4	●
32.4	4	78.7	1	●
30.4	4	75.7	1	●
33.9	4	71.1	1	●

data

geom

1. Pick a **dataset**

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

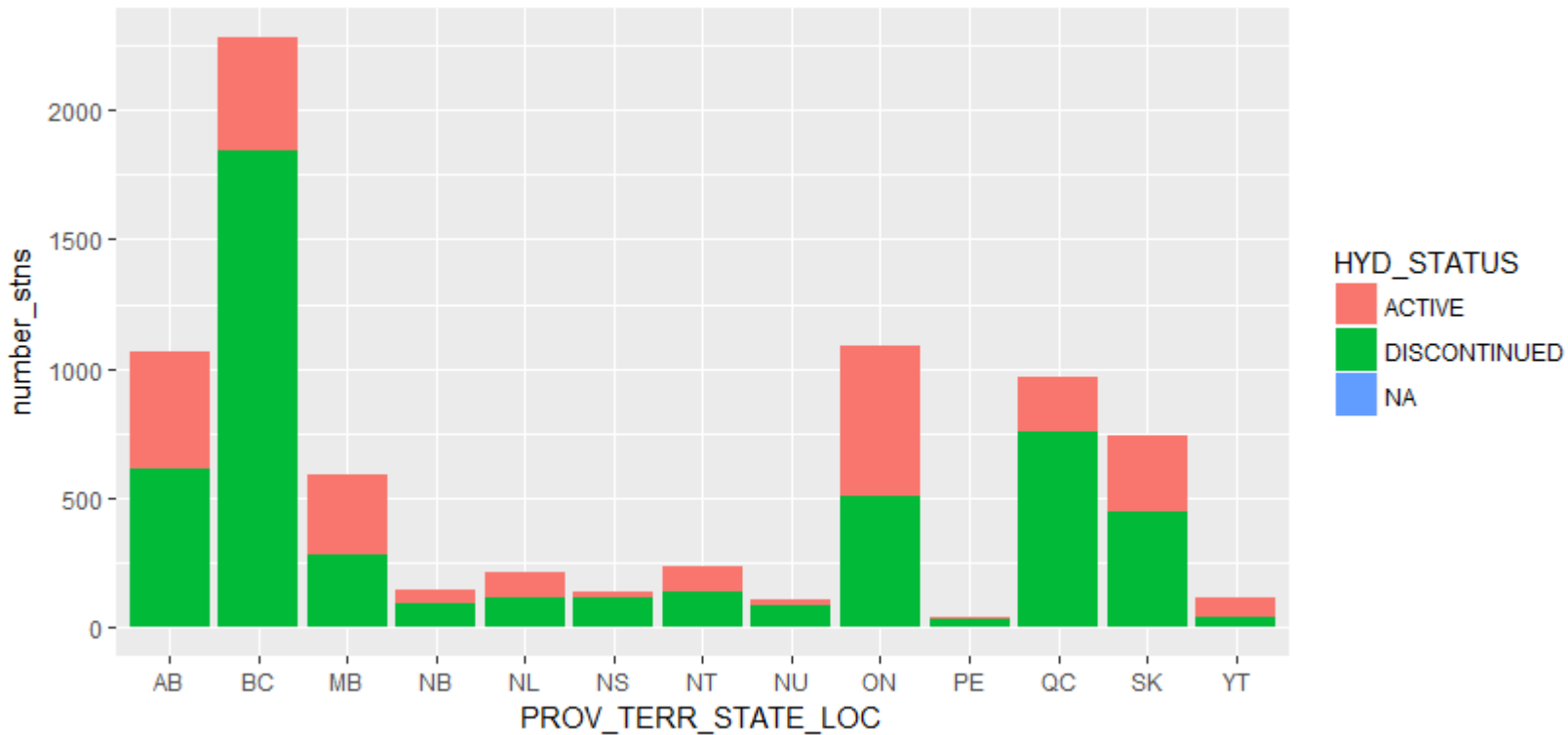
2. Choose a **geom**
to display cases

3. **Map** aesthetic
properties to
variables

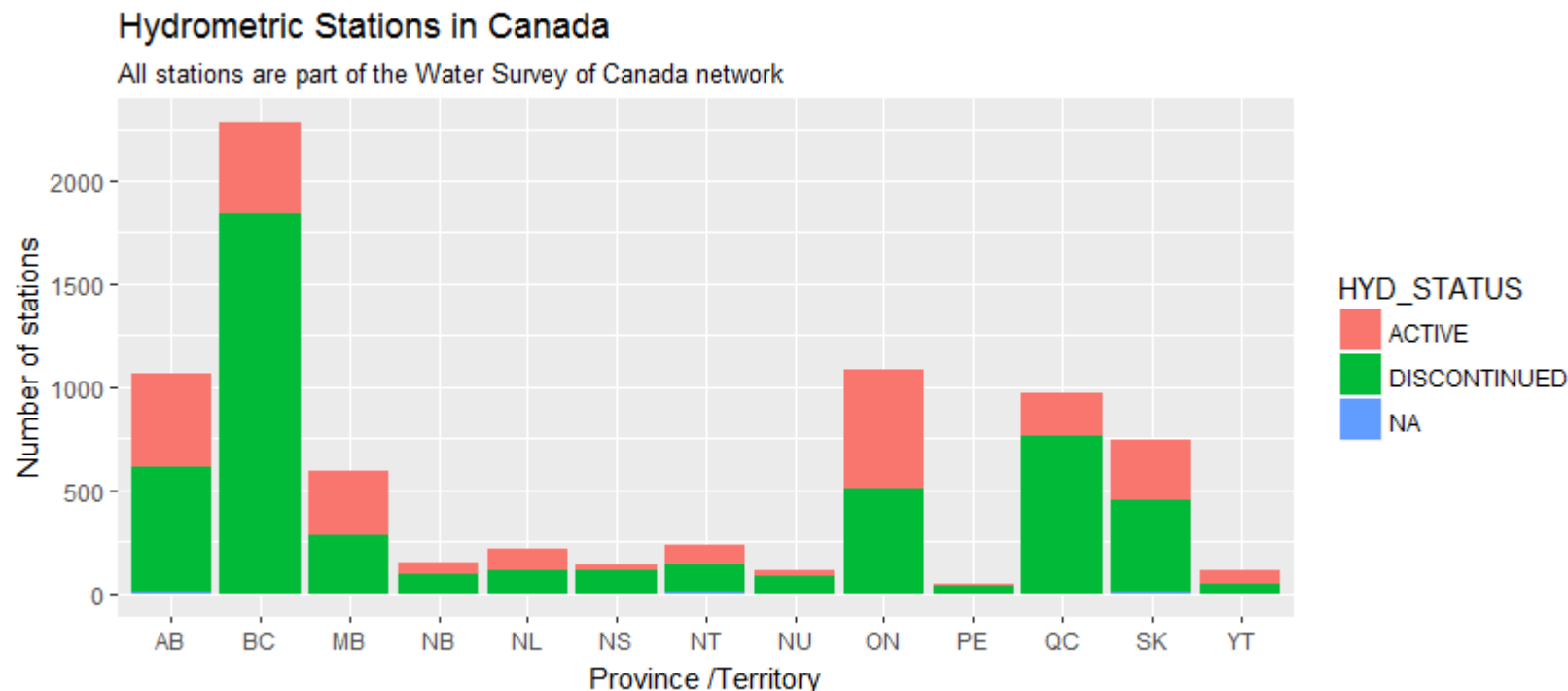


What else?

Basic Plot

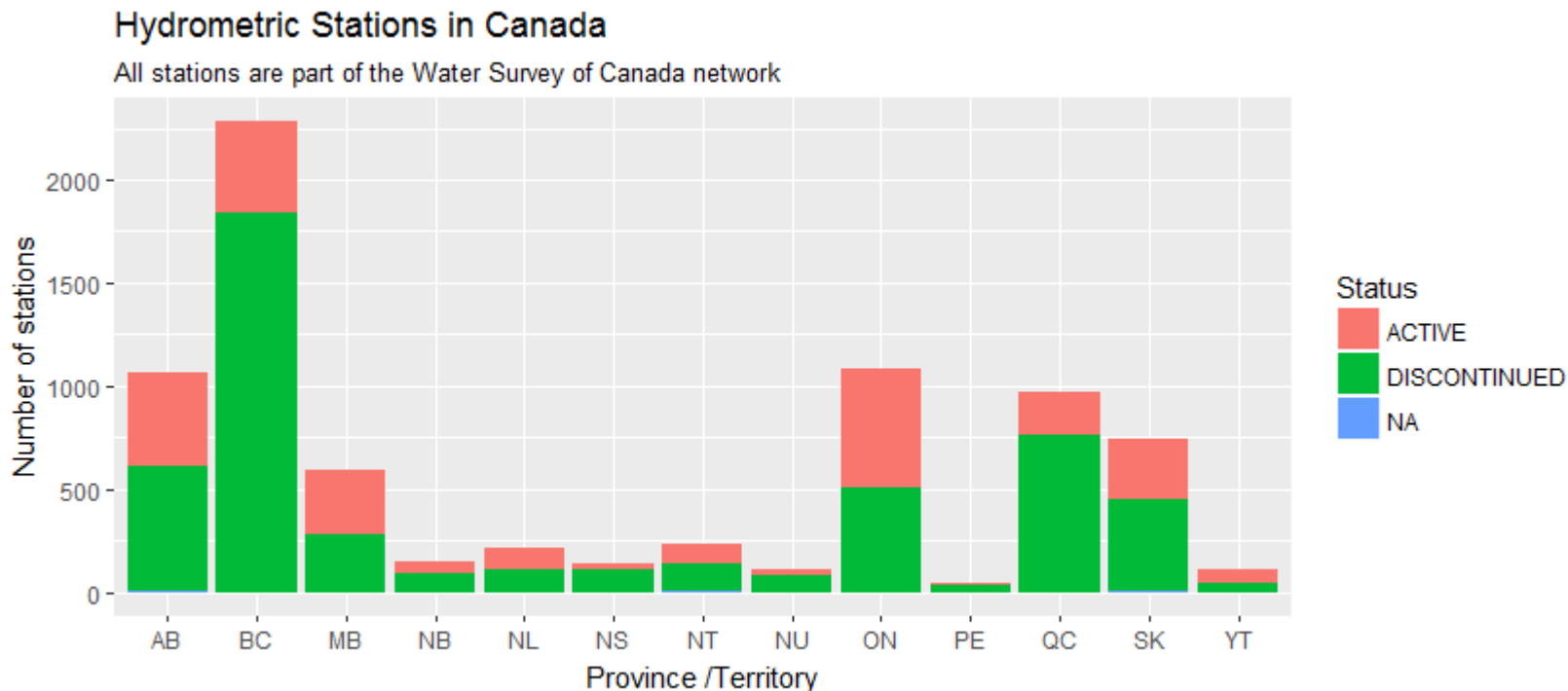


Titles and captions+ labs()



Figures generated with tidyhydat and ggplot2

Modifying the legend title

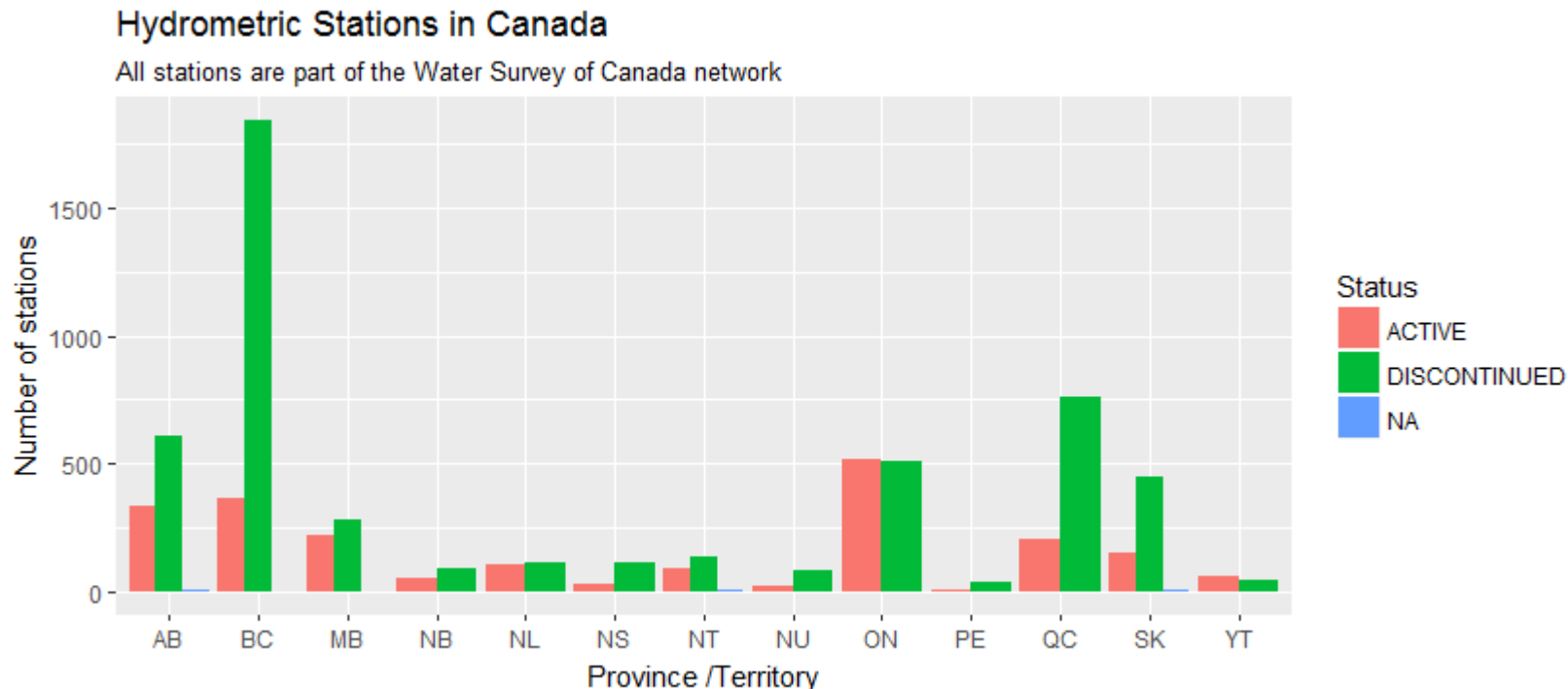


Figures generated with tidyhydat and ggplot2



Position Adjustments

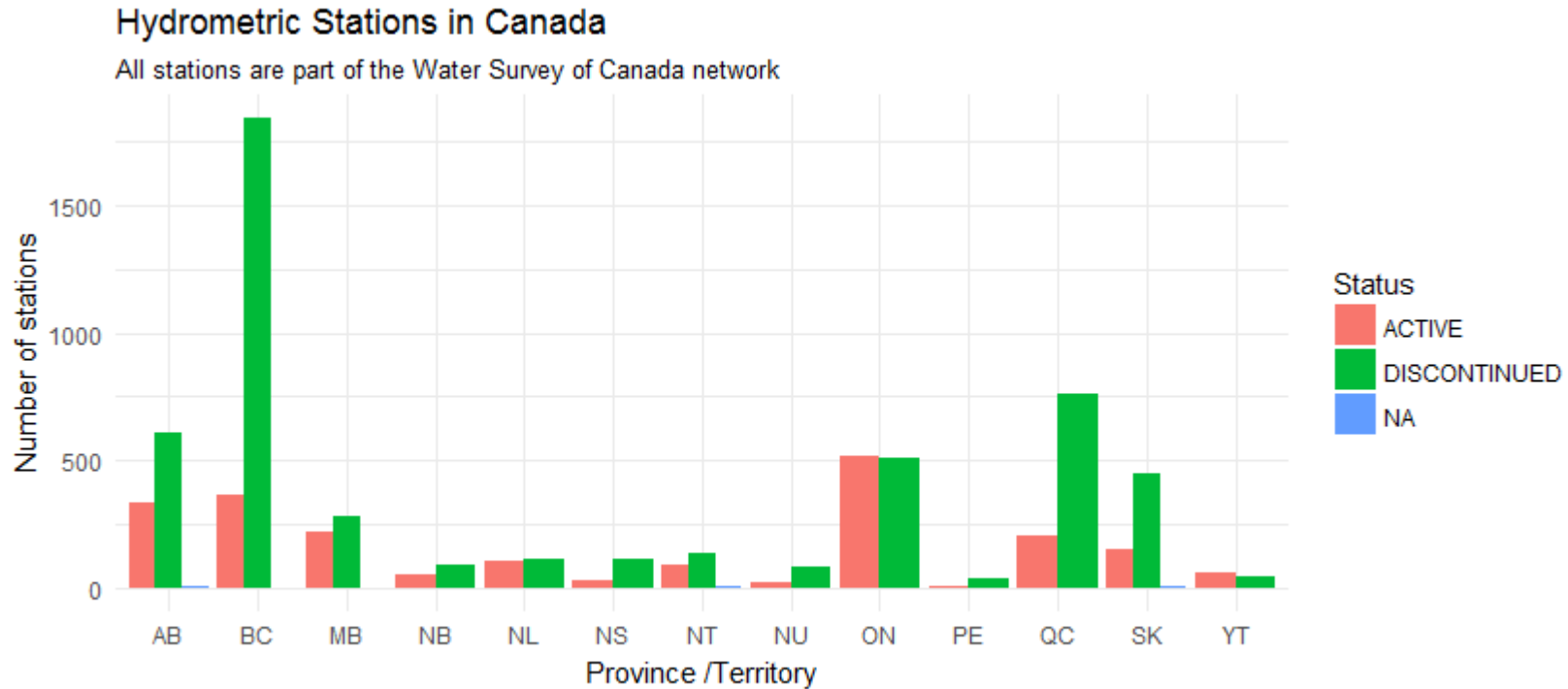
How overlapping objects are arranged



Figures generated with tidyhydat and ggplot2



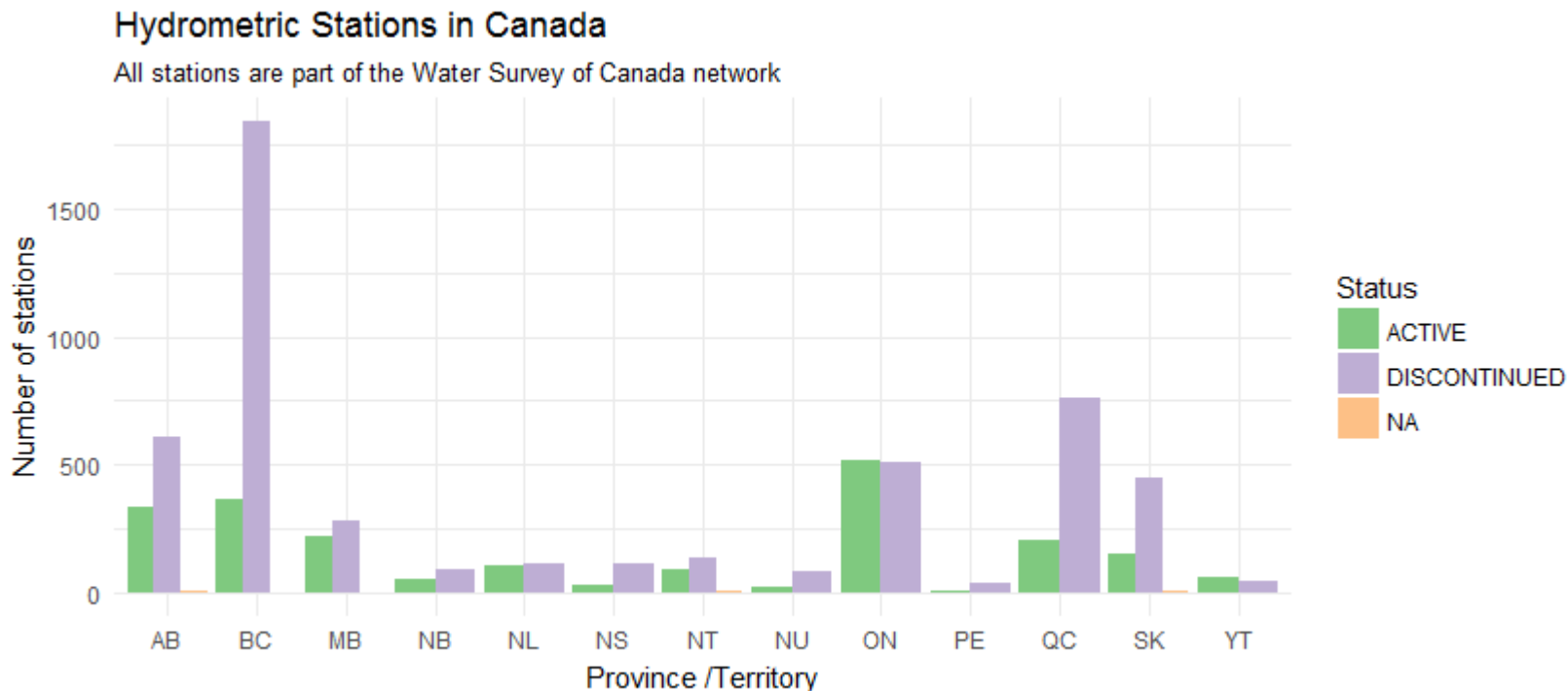
Theme - Visual appearance of non-data elements



Figures generated with tidyhydat and ggplot2



Scales - Customize color scales, other mappings



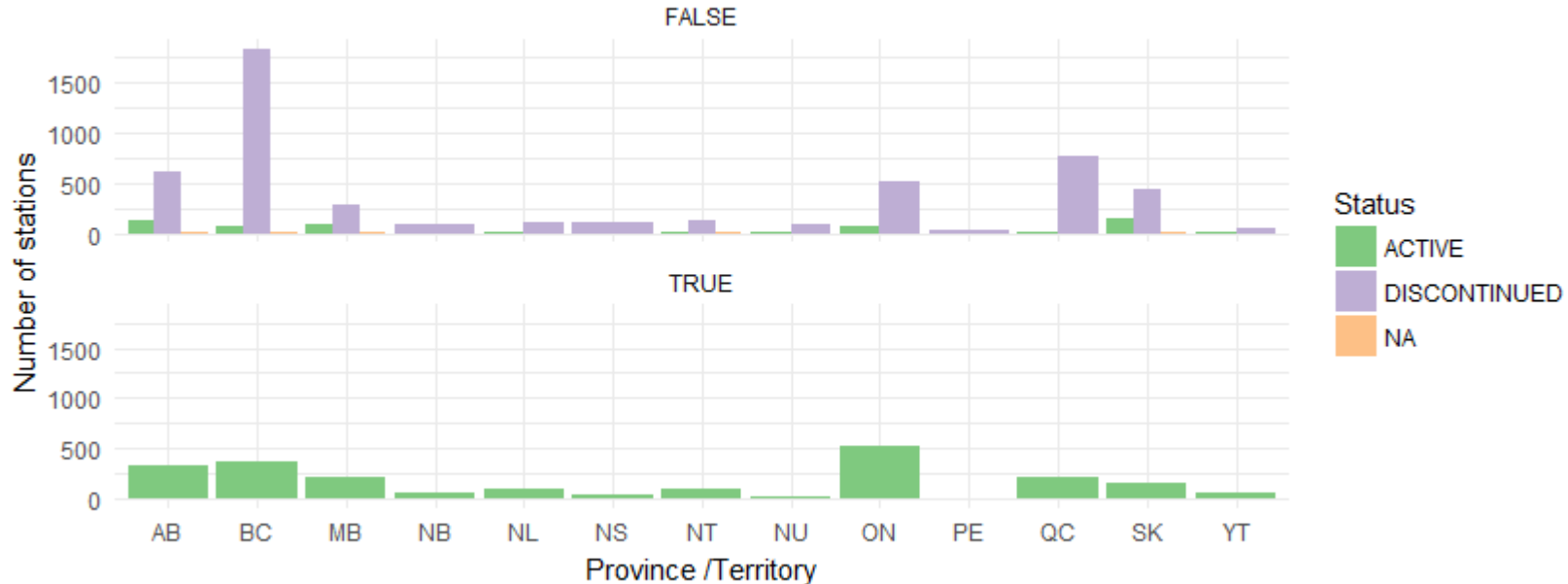
Figures generated with tidyhydat and ggplot2



Facets - Subplots that display subsets of the data.

Hydrometric Stations in Canada

All stations are part of the Water Survey of Canada network



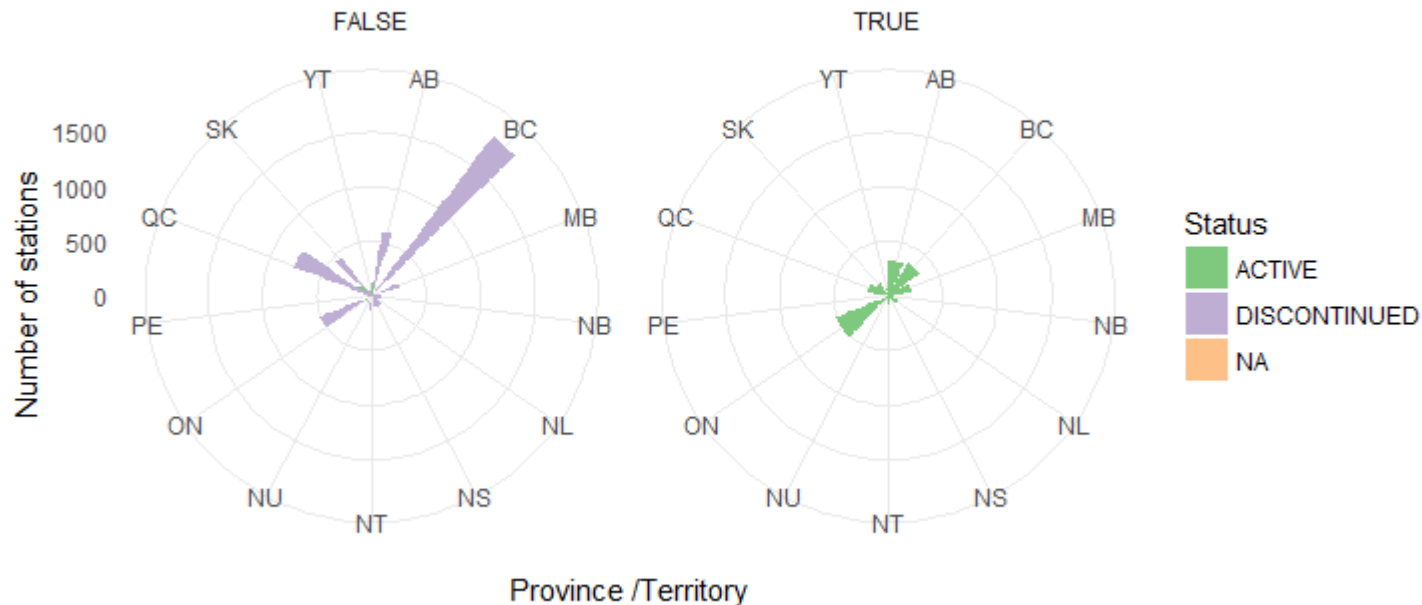
Figures generated with tidyhydat and ggplot2



Coordinate systems

Hydrometric Stations in Canada

All stations are part of the Water Survey of Canada network



Figures generated with tidyhydat and ggplot2

ggplot2.tidyverse.org

The screenshot shows the ggplot2 website. The browser tab is titled 'Create Elegant Data Visualisations' and the address bar shows 'ggplot2.tidyverse.org'. The page header includes the ggplot2 logo, the text 'part of the tidyverse', and navigation links for 'Reference', 'Articles', and 'News'. The main content area is titled 'Usage' and contains a paragraph explaining the philosophy of ggplot2. Below the text is a code block showing the library and plotting functions. To the right, there is a 'Links' section with various URLs. At the bottom right, there is a 'License' section and a 'Developers' section. A scatter plot is visible at the bottom left of the page.

Usage

It's hard to succinctly describe how ggplot2 works because it embodies a deep philosophy of visualisation. However, in most cases you start with `ggplot()`, supply a dataset and aesthetic mapping (with `aes()`). You then add on layers (like `geom_point()` or `geom_histogram()`), scales (like `scale_colour_brewer()`), faceting specifications (like `facet_wrap()`) and coordinate systems (like `coord_flip()`).

```
library(ggplot2)

ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point()
```

Links

- Download from CRAN at <https://cran.r-project.org/package=ggplot2>
- Browse source code at <https://github.com/tidyverse/ggplot2>
- Report a bug at <https://github.com/tidyverse/ggplot2/issues>
- Learn more at <http://r4ds.had.co.nz/data-visualisation.html>

License

GPL-2 | [file LICENSE](#)

Developers

Hadley Wickham
Author, maintainer

class

- 2seater



Visualize Data with

