



EDA III: Visualizations with ggplot2

Yvonne Phillips
Bevera Solutions

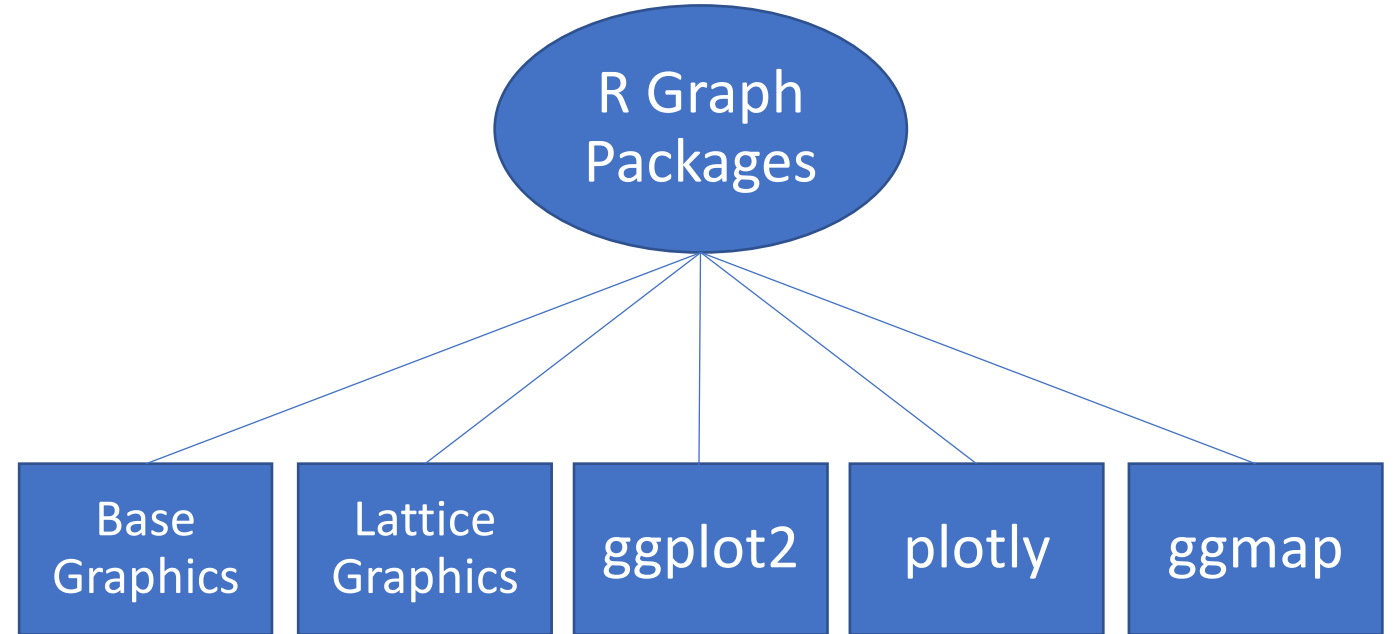
yvonne.Phillips@beverasolutions.com

- Review EDA II Wrangling
- Foundation of many graphic applications
 - Grammar of Graphics
- Focus on teaching the underlying theory of ggplot2
- How is Grammar of Graphics reflected in the API
- Chart types
- Dealing with colors



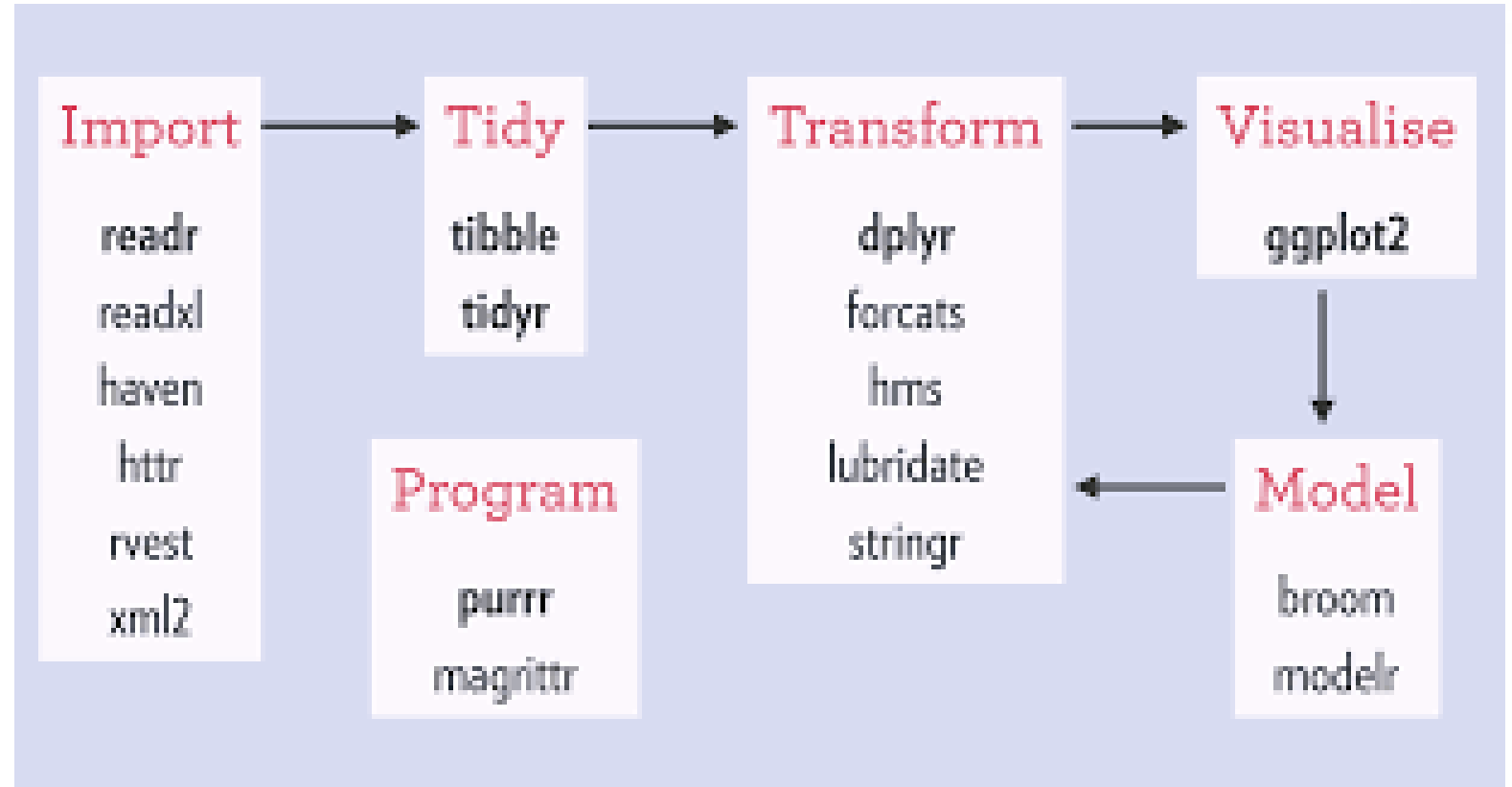


**Data
visualization
in R can be
performed in
the following
ways:**





<https://www.tidyverse.org/>



```
install.packages("tidyverse")
```

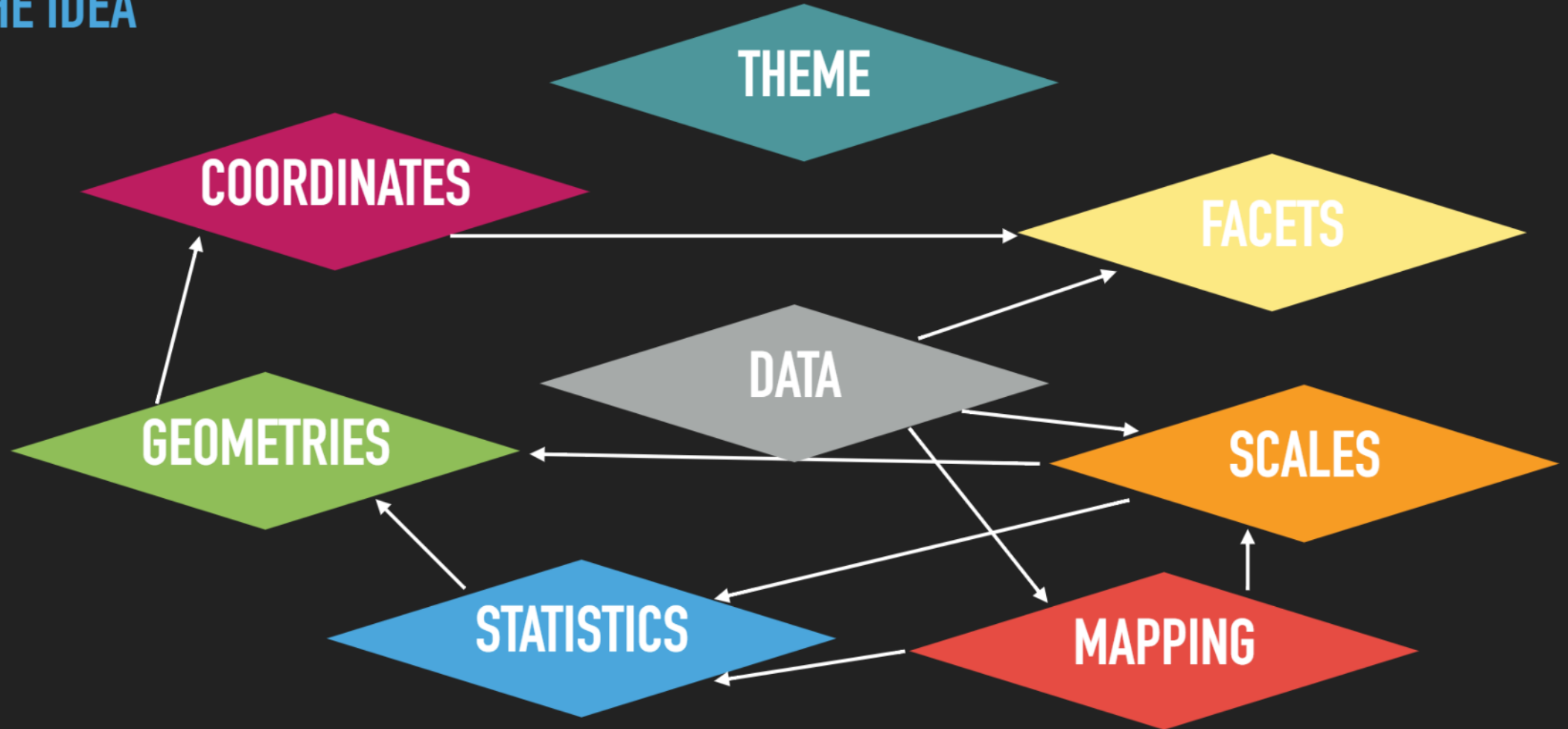
```
library("tidyverse")
```

<https://github.com/rstudio/master-the-tidyverse/archive/master.zip>



ggplot2 is one of the most elegant and most versatile. ggplot2 implements the grammar of graphics, a coherent system for describing and building graphs. With ggplot2, you can do more faster by learning one system and applying it in many places.

THE IDEA

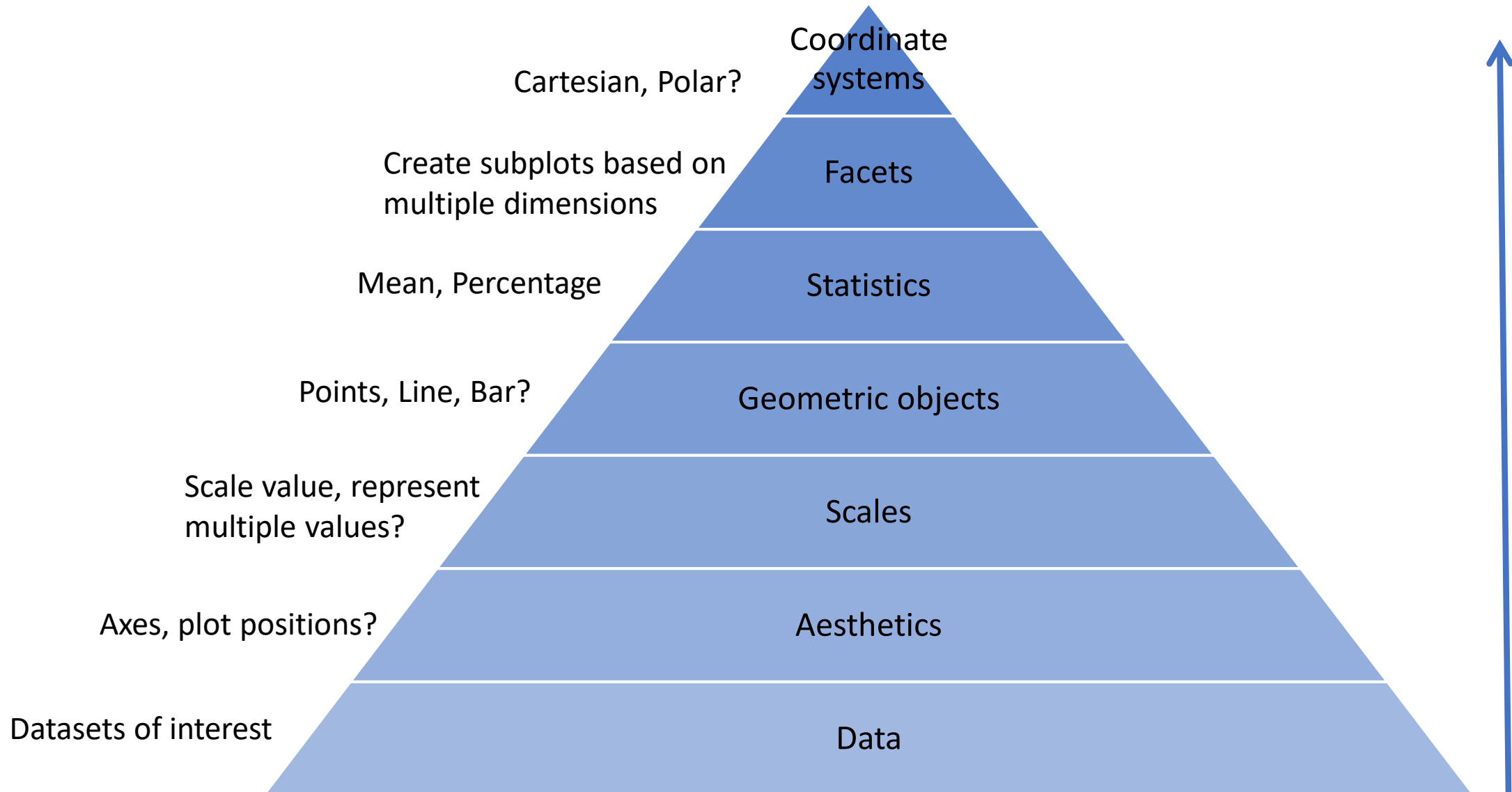




THEME
COORDINATES
FACETS
GEOMETRIES
SCALES
STATISTICS
MAPPING
DATA



Components of the Grammar of Graphics



1. **Data** are the values represented in the visualization.
2. **Aesthetic mappings** are directions for how data are mapped in a plot in a way that we can perceive. Aesthetic mappings include linking variables to the x-position, y-position, color, shape, and size.
3. **Geometric objects** are representations of the data, including points, lines, and polygons.
4. **Scales** turn data values, which are quantitative or categorical, into aesthetic values.
5. **Coordinate systems** map scaled geometric objects to the position of objects on the plane of a plot. The two most popular coordinate systems are the Cartesian coordinate system and the polar coordinate system.
6. **Facets (optional)** break data into meaningful subsets. Split data into multiple panels.
7. **Statistical transformations (optional)** transform the data, typically through summary statistics and functions, before aesthetic mapping.
8. **Theme** controls the visual style of plot with font types, font sizes, background colors, margins, and positioning.

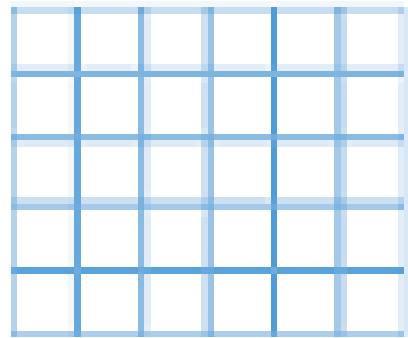
Plot Basics

```
p <- ggplot(data= <data>,  
           mapping= aes(<aesthetic> = <variable>,  
                        <aesthetic> = <variable>,  
                        <...> = <...>)
```

```
p + geom_<type>(<...>)+  
    scale_<mapping>_<type>(<...>)+  
    coord_<type>(<...>)+  
    labs(<...>)
```

| | |
|----------------------------|---|
| ggplot() | Create a new ggplot |
| aes() | Construct aesthetic mappings |
| `+`(<gg>) ` %+% ` | Add components to a plot |
| <u>ggsave()</u> | Save a ggplot (or other grid object) with sensible defaults |
| qplot() quickplot() () | Quick plot |

data



MAPPED
TO

aesthetic
properties

OF

geometric
objects

$x + y$

color

fill

alpha

geom_point

geom_histogram

geom_bar

geom_line

statistical
transformations
(sometimes)

This call fully specifies the five components to the layer:

1. **mapping**: A set of aesthetic mappings, specified using the `aes()` function and combined with the plot defaults as described in aesthetic mappings. If `NULL`, uses the default mapping set in `ggplot()`.
2. **data**: A dataset which overrides the default plot dataset. It is usually omitted (set to `NULL`), in which case the layer will use the default data specified in `ggplot()`. The requirements for data are explained in more detail in `data`. `ggplot` needs a data frame object for the input parameter, `data`, and not a table. If you specify a table as an input parameter, `data` then it will result in an error.
3. **geom**: The name of the geometric object to use to draw each observation. Geoms can have additional arguments. All geoms take aesthetics as parameters.

geom_bar

geom_bin

geom_boxplot

geom_density

geom_error

geom_hex

geom_hist

geom_hline

geom_jitter

geom_label

geom_line

geom_point

geom_polygon

geom_rect

geom_ribbon

geom_rug

geom_segment

geom_smooth

geom_text

geom_tile

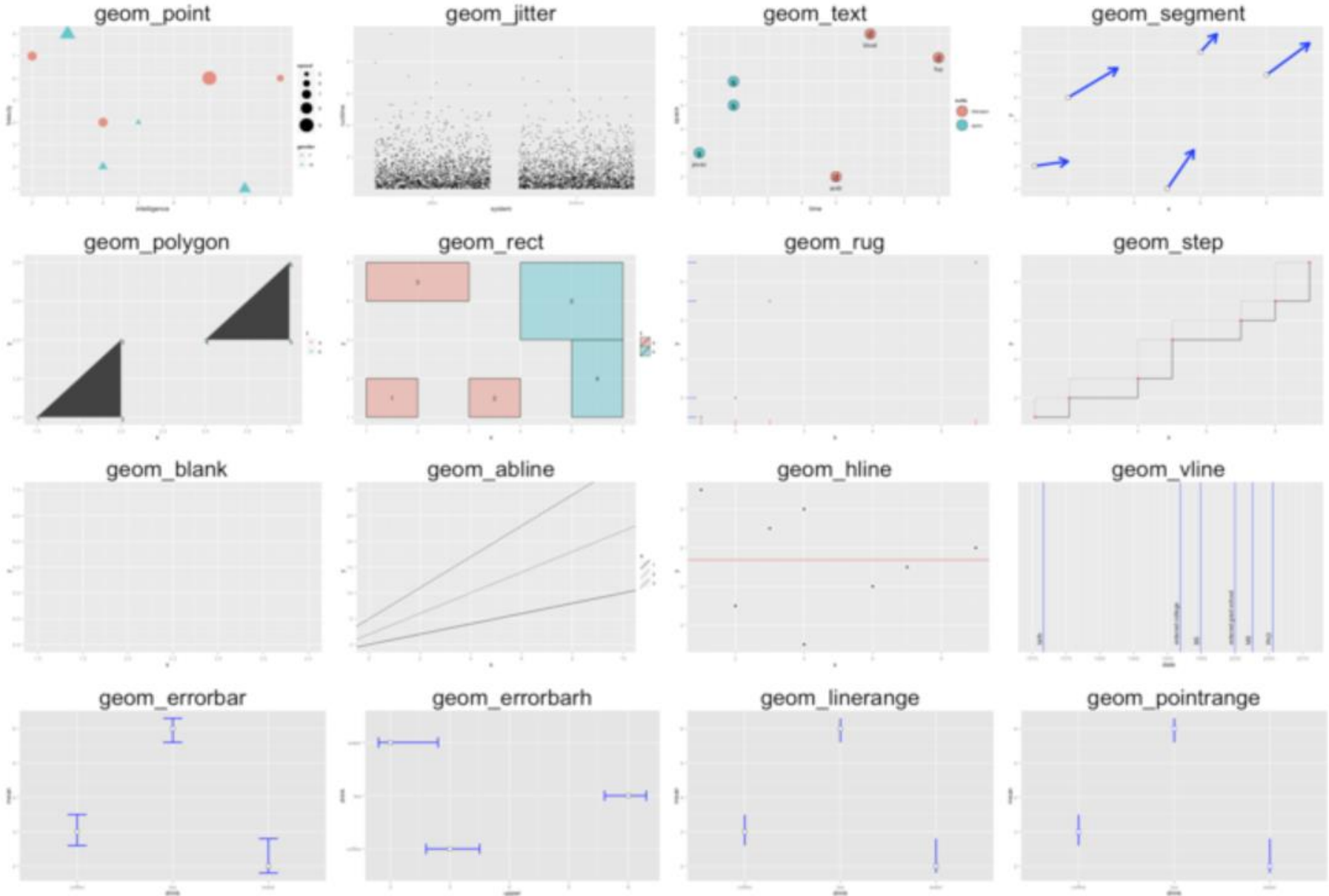
geom_violin

geom_vline

ggplot2 builds charts through layers using geom_ functions. Here is a list of the different available geoms.

Find a complete list of all Geoms [Here](#)

Geometric objects (geoms) are the visual representations of (subsets of) observations. Their requirements differ.



This call fully specifies the five components to the layer:

4. **stat(istic)**: The name of the statistical transformation to use. A statistical transformation performs some useful statistical summary is key to histograms and smoothes. To keep the data as is, use the “identity” stat.

You only need to set one of stat and geom: every geom has a default stat, and every stat a default geom.

Most stats take additional parameters to specify the details of statistical transformation. You can supply params either in ... (in which case stat and geom parameters are automatically teased apart), or in a list called `stat_params`.

5. **position**: The method used to adjust overlapping objects, like jittering, stacking or dodging. More details in position.

Using `stat_identity`: the identity statistic leaves the data unchanged.

| Geom | Description | Default Stat |
|---|--|------------------------------|
| <code>geom_bar()</code> # not pre-counted data where each observation contributes one unit to the height of the bar | Bar chart | <code>stat_bin()</code> |
| <code>geom_col()</code> # pre-counted data. y-position aesthetic mapped to the variable that has the counts. | | |
| <code>geom_point()</code> | Scatterplot | <code>stat_identity()</code> |
| <code>geom_line()</code> | Line diagram, connecting observations in order by x-value | <code>stat_identity()</code> |
| <code>geom_boxplot</code> | Box-and-whisker plot | <code>stat_boxplot()</code> |
| <code>geom_path</code> | Line diagram, connecting observations in original order | <code>stat_identity()</code> |
| <code>geom_smooth</code> | Add a smoothed conditioned mean | <code>stat_smooth()</code> |
| <code>geom_histogram</code> | An alias for <code>geom_bar()</code> and <code>stat_bin()</code> | <code>stat_bin()</code> |

Colour and fill

Lines

Polygons

Point

Text

ggplot2: Layers: Aesthetics

[aes colour fill alpha](#)

Colour related aesthetics: colour, fill, and alpha

[aes group order](#)

Aesthetics: grouping

[aes linetype size shape](#)

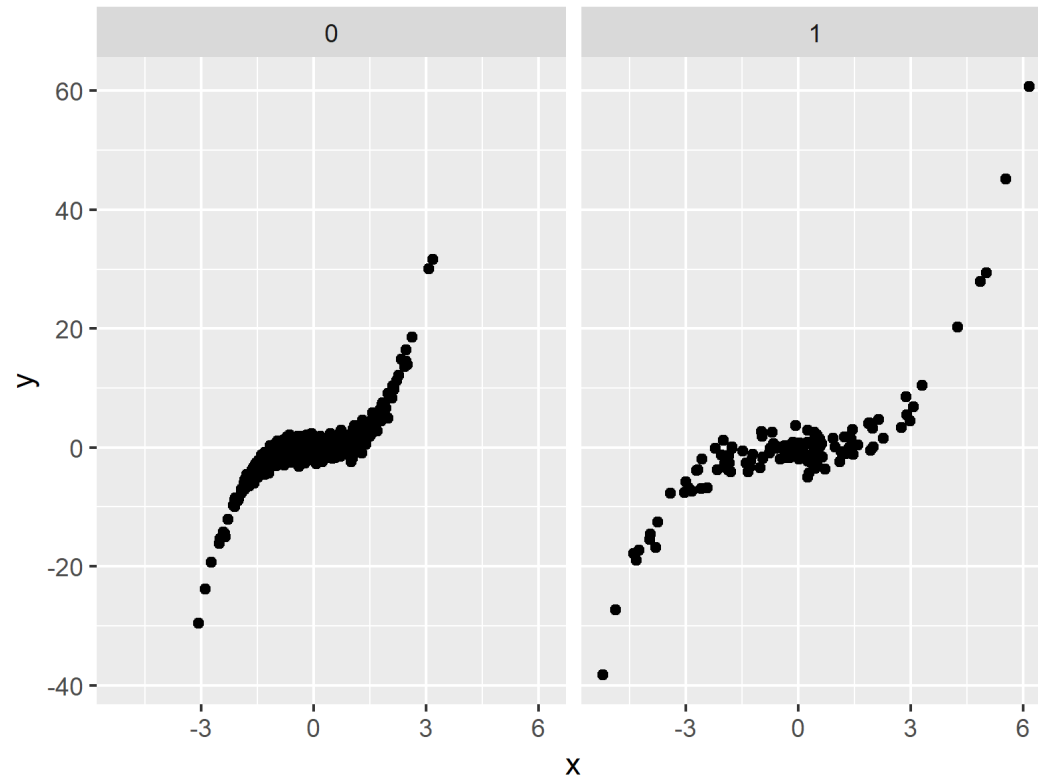
Differentiation related aesthetics: linetype, size, shape

[aes position](#)

Position related aesthetics: x, y, xmin, xmax, ymin, ymax, xend, yend

ggplot2: Layers: Facetting

Facetting generates small multiples, each displaying a different subset of the data. Facets are an alternative to aesthetics for displaying additional discrete variables.



ggplot2: themes

This theme creates a set of rules for styling the components of the grammar of graphics. The first part of the theme creates the function `theme_urban()`, which handles sizes, spacing, font families, orientation, and the placement of elements.

→ ggplot2

`default`

`theme_bw()`

`theme_minimal()`

`theme_classic()`

`theme_gray()`

→ ggthemes

`theme_excel()`

`theme_economist()`

`theme_fivethirtyeight()`

`theme_tufte()`

`theme_gdocs()`

`theme_wsj()`

`theme_calc()`

`theme_hc()`

→ other

`theme_article()`

`theme_pubclean()`

`theme_bigstatsr()`

`theme_ipsum()`

The theme is based on ggplot2's web of inheritances. At the top, the theme sets default line, rectangle, and text styles. These three attributes are then passed as defaults to the next layer of arguments.



Practice

Practice

Practice

Example: Air Quality

- <https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/airquality.html>
- Daily air quality measurements in New York, May to September 1973.
- Daily readings of the following air quality values for May 1, 1973 (a Tuesday) to September 30, 1973.
 - Ozone: Mean ozone in parts per billion from 1300 to 1500 hours at Roosevelt Island
 - Solar.R: Solar radiation in Langleys in the frequency band 4000–7700 Angstroms from 0800 to 1200 hours at Central Park
 - Wind: Average wind speed in miles per hour at 0700 and 1000 hours at LaGuardia Airport
 - Temp: Maximum daily temperature in degrees Fahrenheit at La Guardia Airport.
 - Month: Numeric value between 1-12
 - Day: Numeric value between 1-31

Upload Numeric Data

```
14 ## Download data from R using data() and see what the set is composed of
15 ## Make sure you download data in the working directory
16
17 data("airquality")
18 str(airquality)
```

```
> str(airquality)
'data.frame':  153 obs. of  6 variables:
 $ Ozone   : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ wind    : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp    : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month   : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day     : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
> |
```

Data Cleaning

```
20 ## To remove NA values, we use complete.cases() which will assign all NA as False,
21 ## else, True.
22 complete.cases(airquality)
23
24 ## To drop values option 1:
25 x <- airquality[complete.cases(airquality), ]
26 str(x)
27
28 ## To drop values option 2:
29 y <- na.omit(airquality)
30 str(y)
31
```

Output



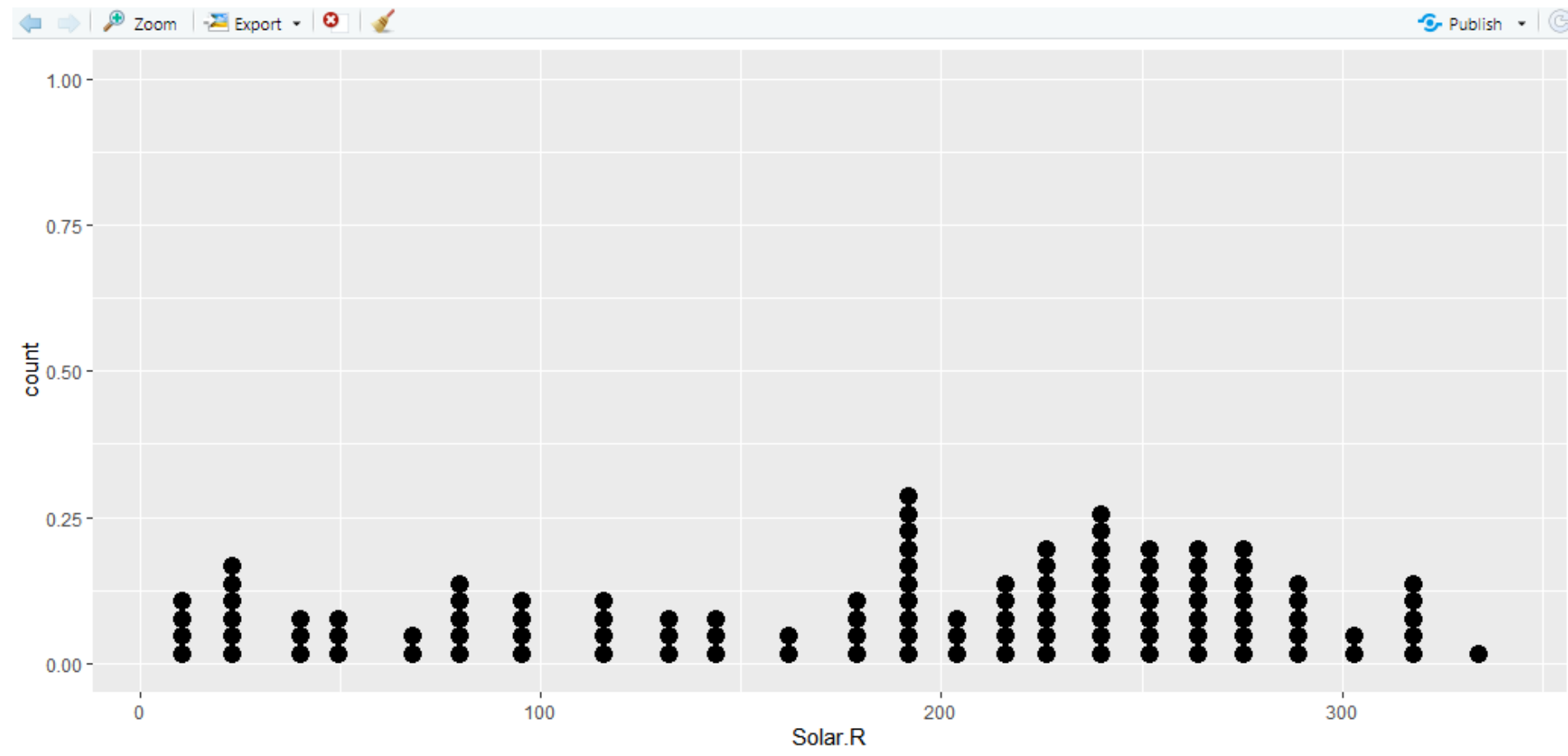
```
> str(x)
'data.frame':  111 obs. of  6 variables:
 $ Ozone   : int  41 36 12 18 23 19 8 16 11 14 ...
 $ Solar.R : int  190 118 149 313 299 99 19 256 290 274 ...
 $ wind    : num  7.4 8 12.6 11.5 8.6 13.8 20.1 9.7 9.2 10.9 ...
 $ Temp    : int  67 72 74 62 65 59 61 69 66 68 ...
 $ Month   : int   5 5 5 5 5 5 5 5 5 5 ...
 $ Day     : int   1 2 3 4 7 8 9 12 13 14 ...

> ## To drop values option 2:
> y <- na.omit(airquality)
> str(y)
'data.frame':  111 obs. of  6 variables:
 $ Ozone   : int  41 36 12 18 23 19 8 16 11 14 ...
 $ Solar.R : int  190 118 149 313 299 99 19 256 290 274 ...
 $ wind    : num  7.4 8 12.6 11.5 8.6 13.8 20.1 9.7 9.2 10.9 ...
 $ Temp    : int  67 72 74 62 65 59 61 69 66 68 ...
 $ Month   : int   5 5 5 5 5 5 5 5 5 5 ...
 $ Day     : int   1 2 3 4 7 8 9 12 13 14 ...
- attr(*, "na.action")= 'omit' Named int [1:42] 5 6 10 11 25 26 27 32 33 34 ...
..- attr(*, "names")= chr [1:42] "5" "6" "10" "11" ...

>
```

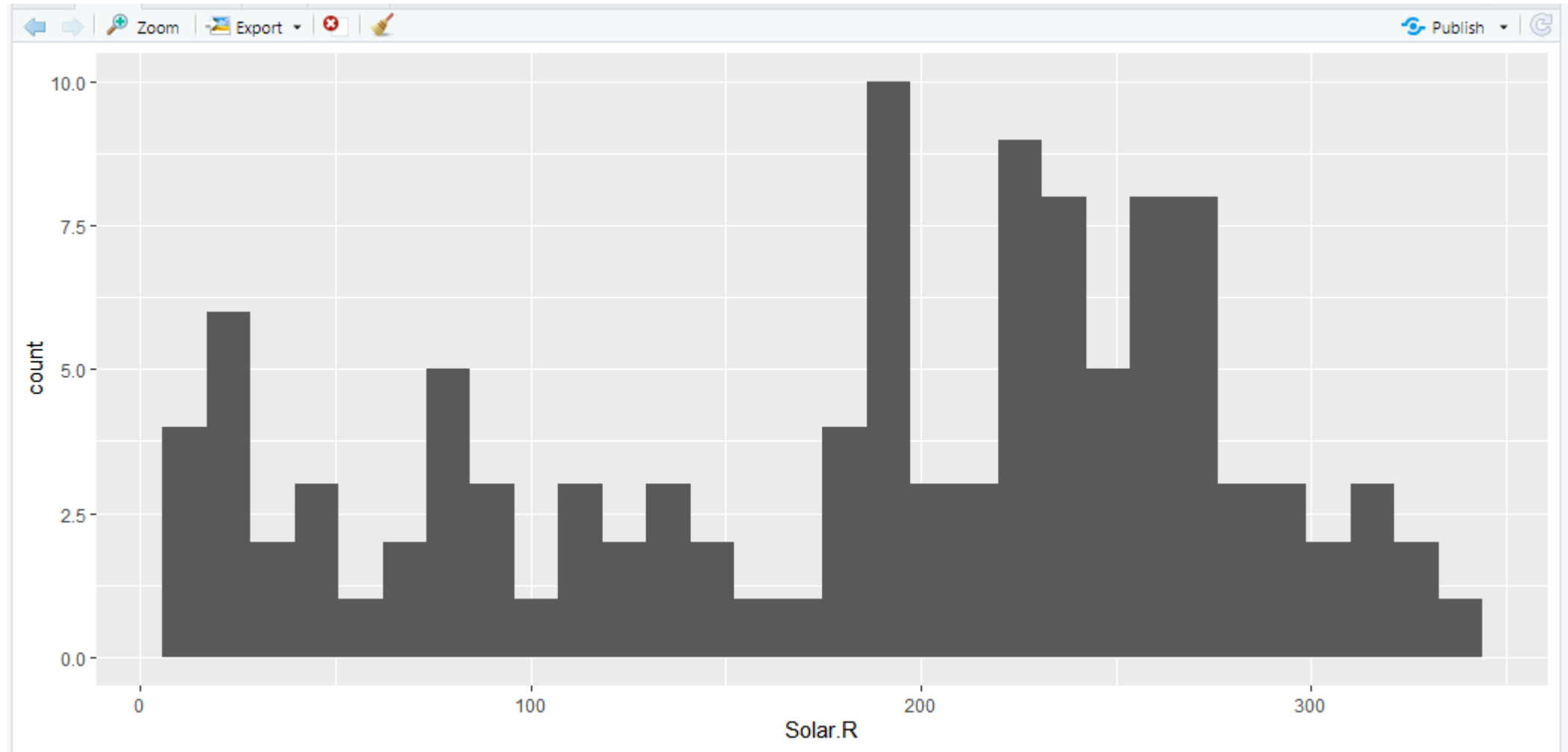
Dotplot

```
32 ## Making a dotplot to show numerical data. It's like a bar chart,  
33 ## but with points stacked on top of each other  
34 ggplot(y, aes(x=Solar.R)) + geom_dotplot(dotsize=0.4)  
35
```



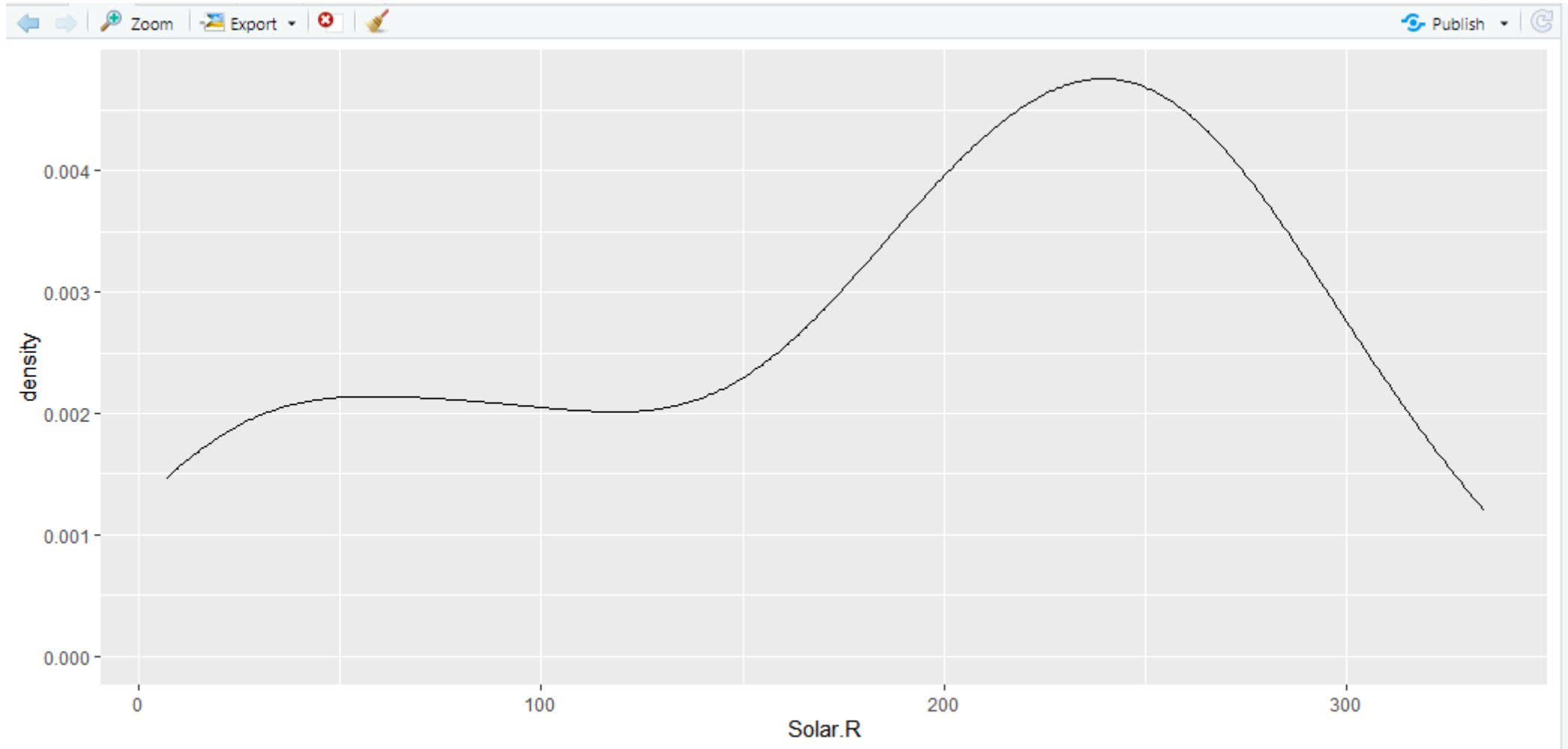
Histogram

```
36 ## Histogram combines the dots, and the y axis now shows the actual count  
37 ggplot(y,aes(x=Solar.R)) + geom_histogram()
```



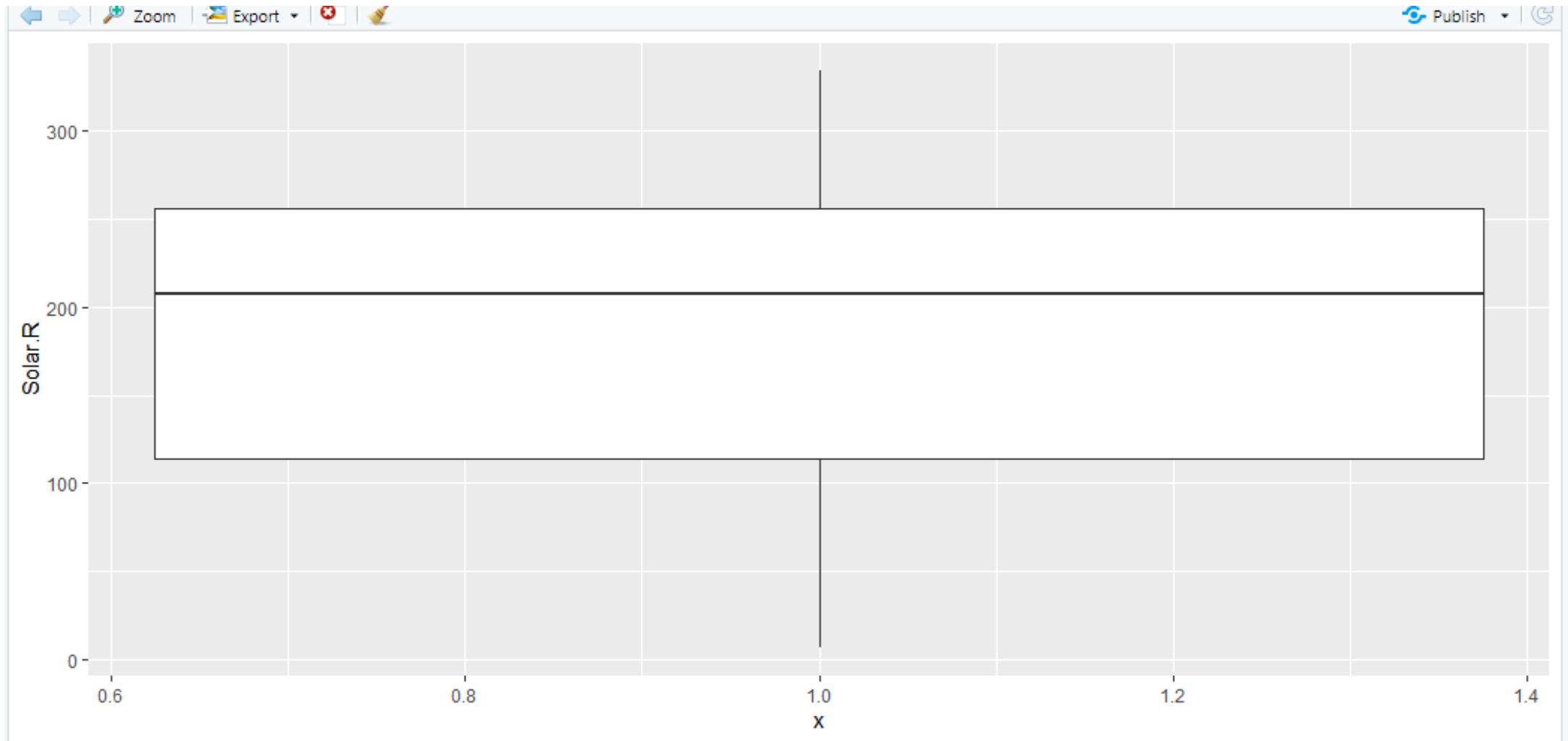
Density plot

```
39 ## The shape of the distribution can be better represented with a density plot,  
40 ## without the stepwise nature of a histogram  
41 ggplot(y,aes(x=Solar.R)) + geom_density()  
42
```



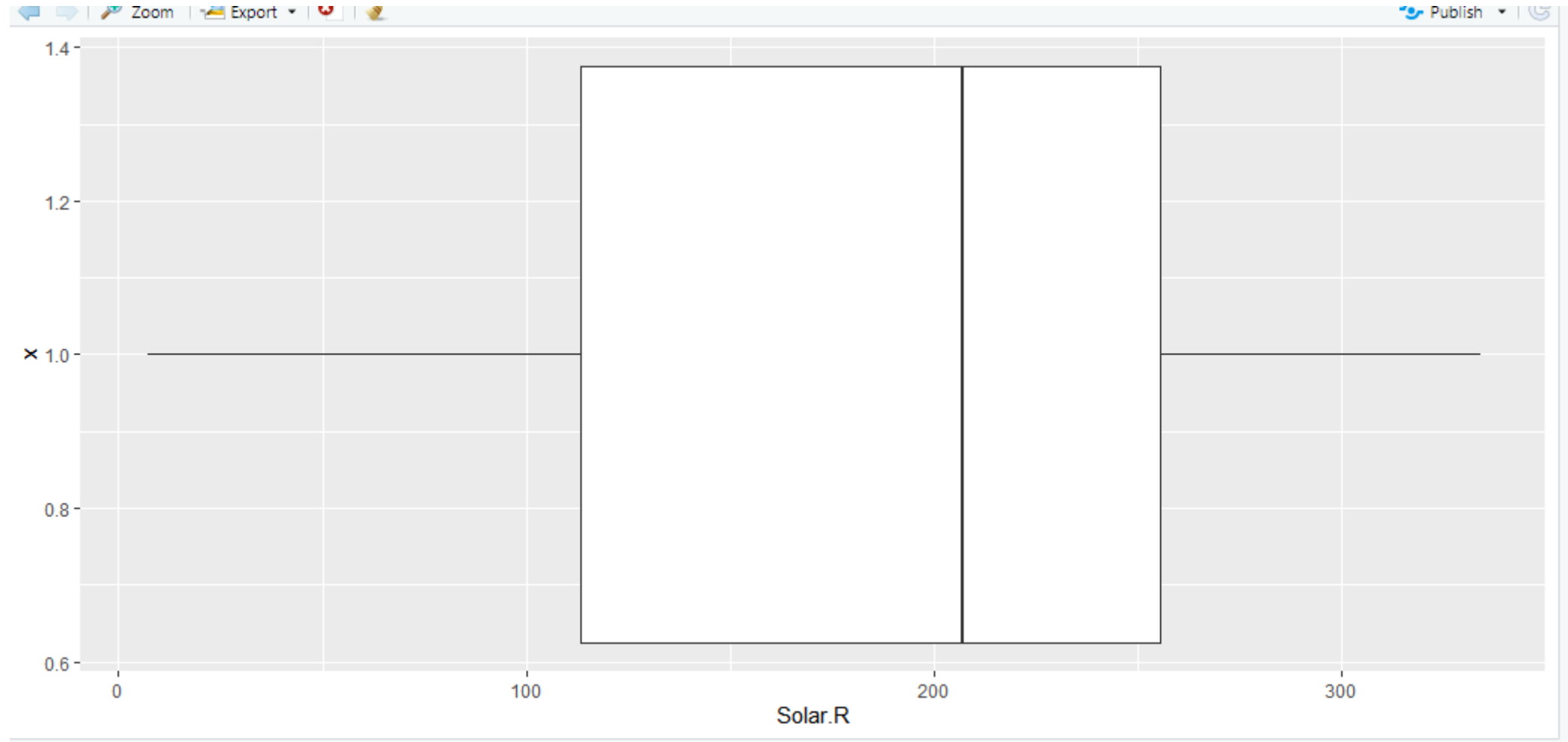
Boxplot

```
43 ## Another view of distribution where you use a boxplot  
44 ggplot(y,aes(x=1,y=Solar.R)) + geom_boxplot()  
45
```



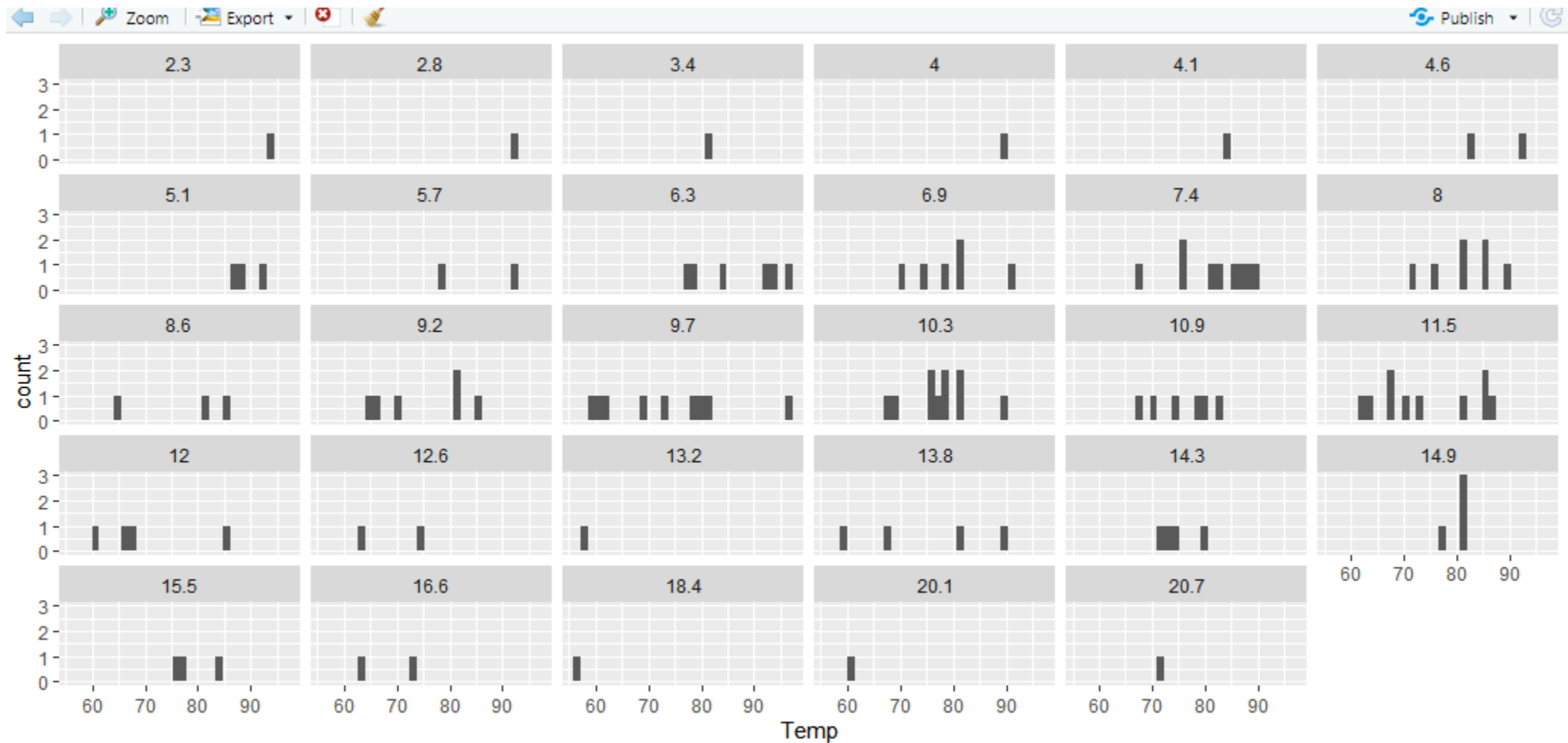
Boxplot (coord_flipped)

```
46 ggplot(y, aes(x=1, y=Solar.R)) + geom_boxplot() + coord_flip()  
47 |
```



Faceted plots

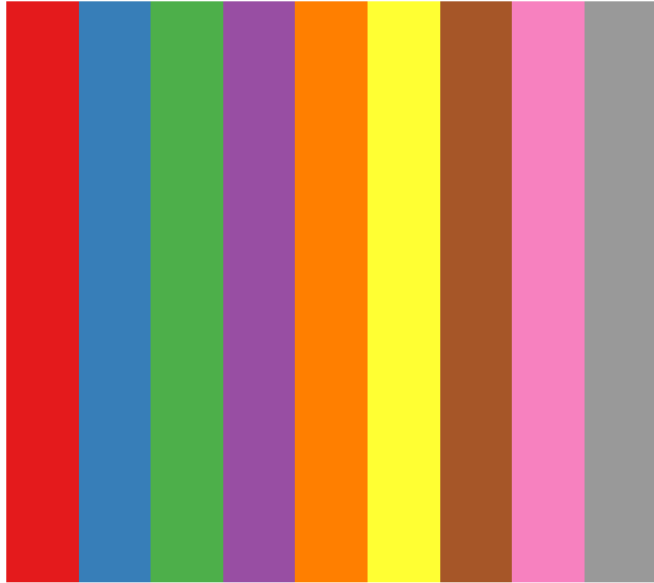
```
48 ## Temperature faceted by wind speeds|
49 ggplot(y,aes(x=Temp)) + geom_histogram() + facet_wrap(~wind)
50
```



Dealing with colors in ggplot2

- Setting a color with *fill* and *color*
- *geom_density(color="purple", fill="69b3a2", ...)* # outlines the shape in the color
- *Picking a color with R:*
 - *Name:* call a color by its name *# colors()*
 - *rgb():* function builds a color using a quantity of red, green and blue. An additional parameter is available to set the transparency. All parameters ranged from 0 to 1.
 - *Number:* call a function by its number. (i.e. *colors()[450]*)
 - *Hex code:* All colors can be defined by their hex code. A hex code looks like this: *#69b3a2*
- *Change the color scale:*
 - default (included in ggplot2)
 - R Color Brewer
 - Viridis
 - Paletteeer

R Color Brewer



Set1 (qualitative)

R Color Brewer's palettes



The `RColorBrewer` package offers several color palette for R. This post displays all of them to help you pick the right one.

COLOR SECTION

ABOUT LINE CHART

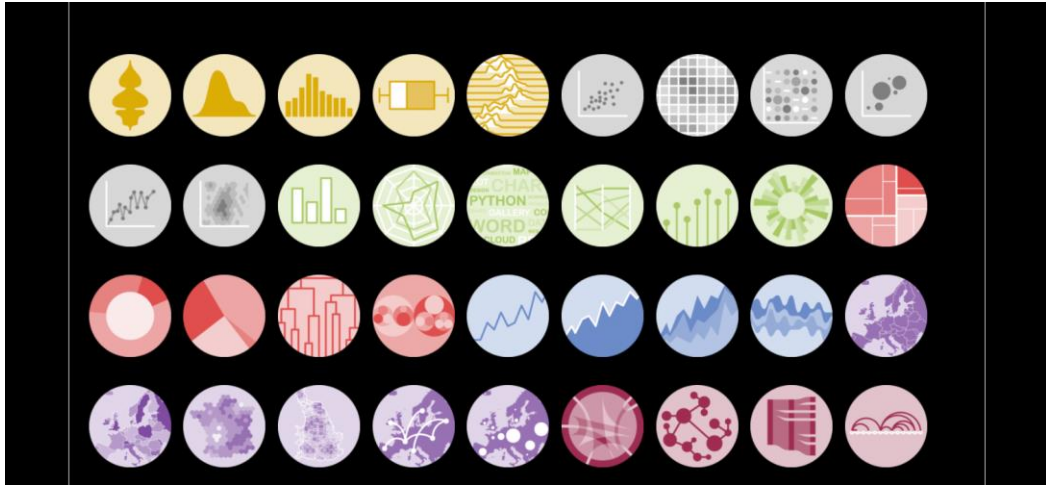


Practice

Practice

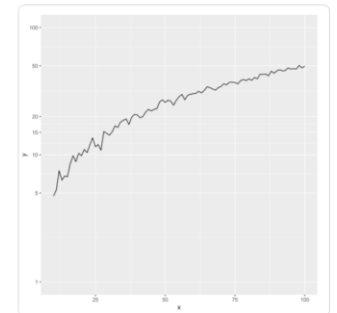
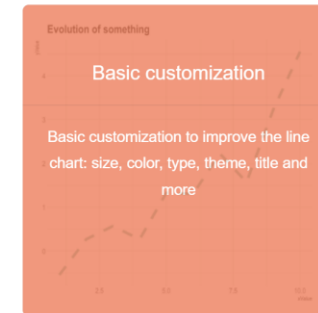
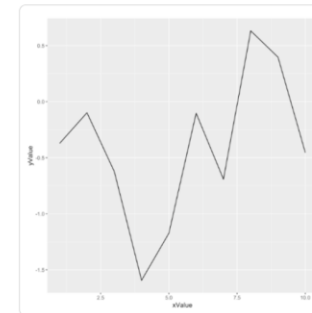
Practice

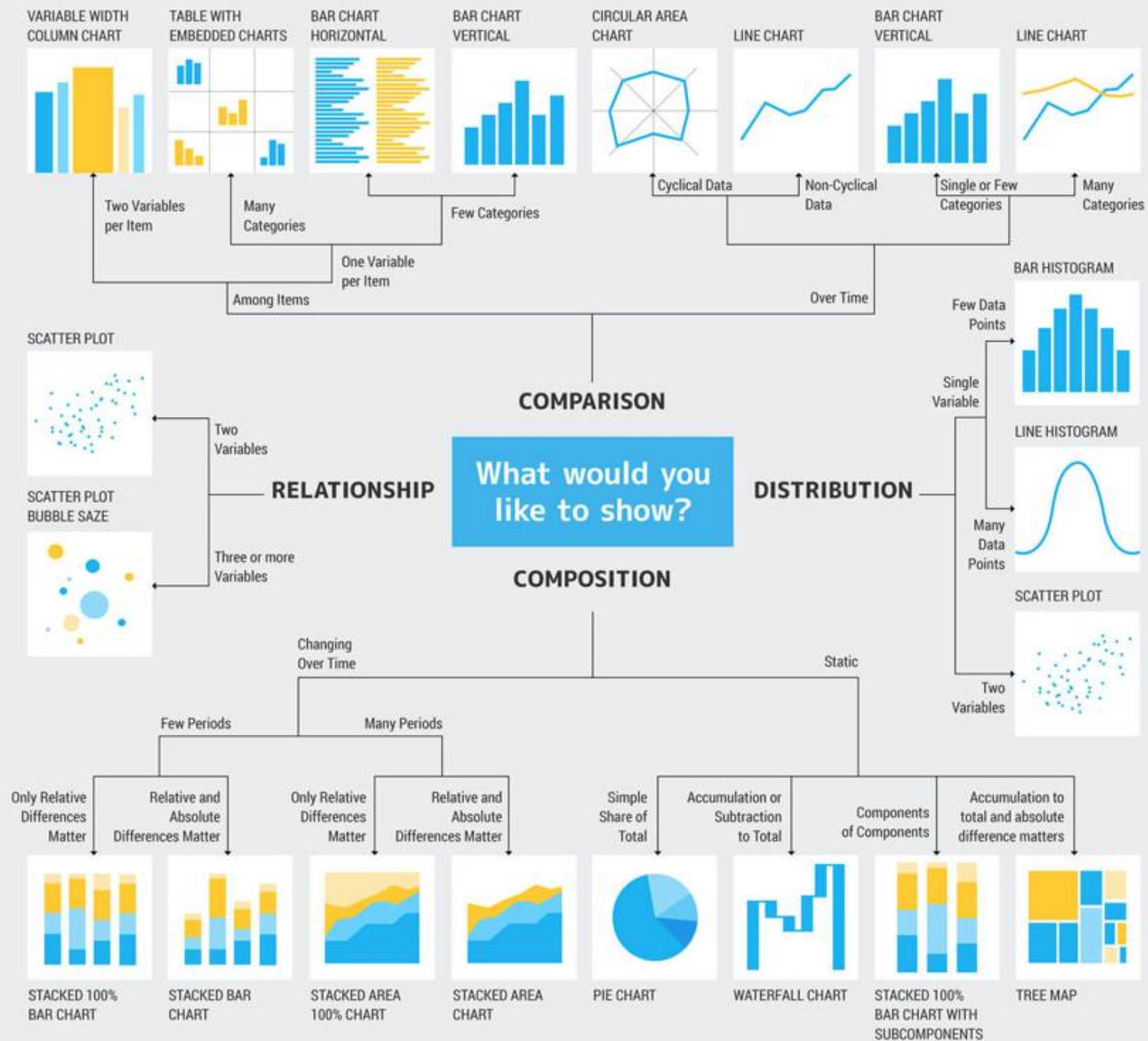
Chart types: [examples here](#)



STEP BY STEP WITH **GGPLOT2**

ggplot2 allows to draw line charts thanks to the `geom_line()` function. It expects as input a data frame with 2 numeric variables, one displayed on each axis. Start your journey with the **most basic line chart**.





Make your ggplot2 chart interactive with plotly()

<https://plotly.com/r/>



Plotly R Open Source Graphing Library

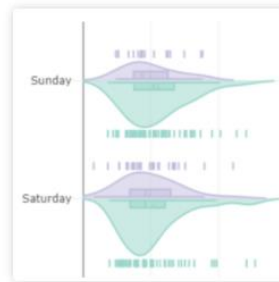
Plotly's R graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, and 3D (WebGL based) charts.

Plotly.R is [free and open source](#) and you can [view the source](#), [report issues](#) or [contribute on GitHub](#).

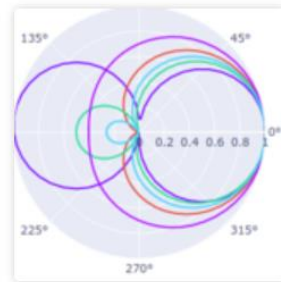
Deploy R AI Dash apps on private Kubernetes clusters: [Pricing](#) | [Demo](#) | [Overview](#) | [AI App Services](#)

Fundamentals

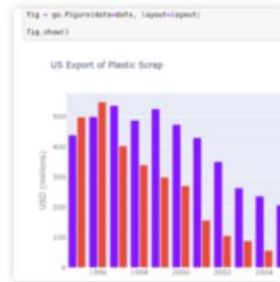
[More Fundamentals »](#)



The Figure Data Structure



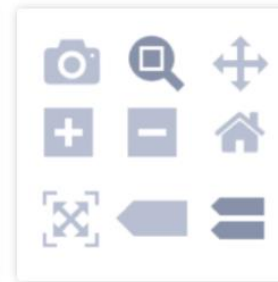
Creating and Updating Figures



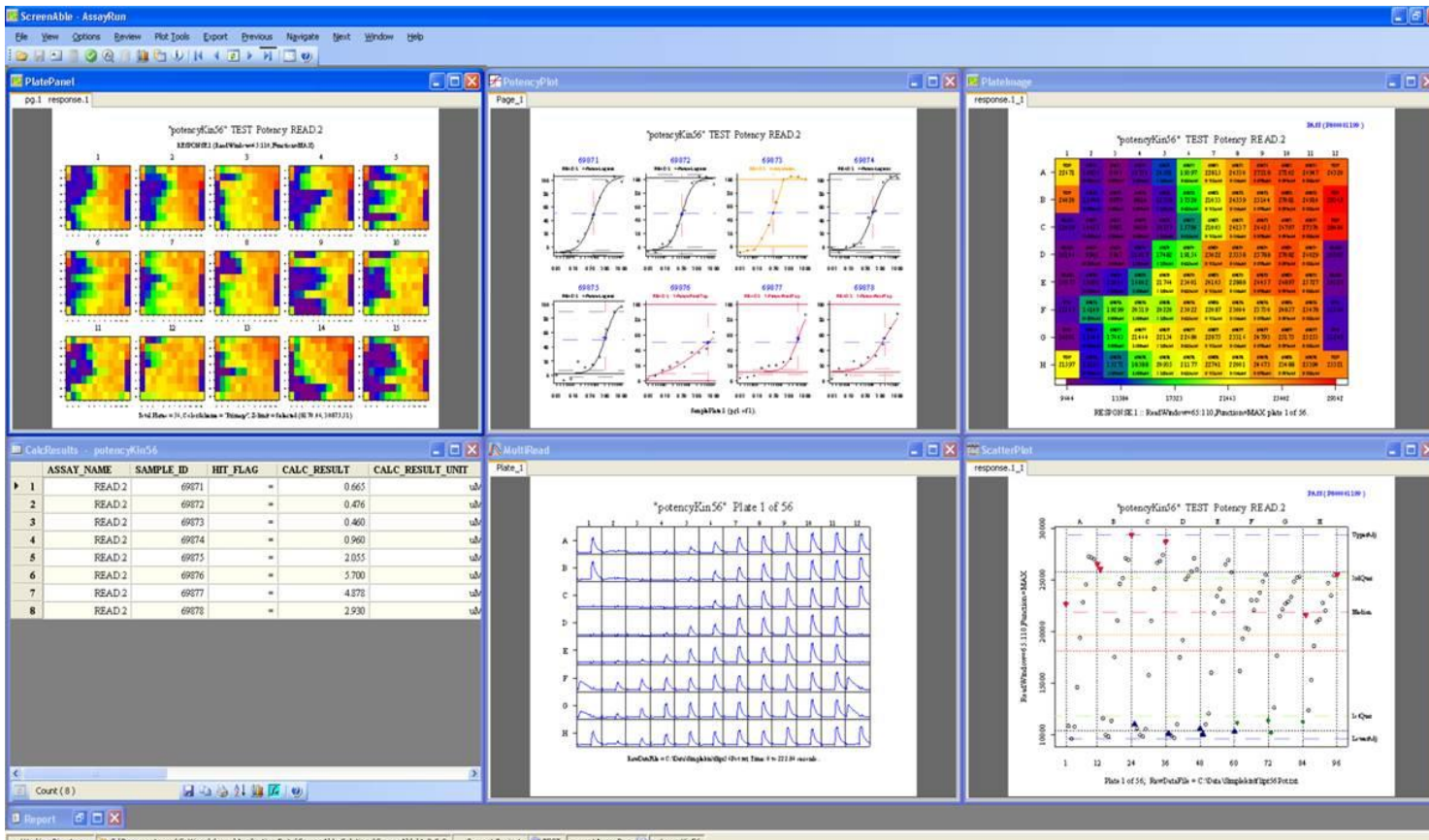
Displaying Figures



Exporting Graphs as Static Images



Configuration



Packages for more visualization options

- ggforce: a collection of features providing missing functionality with the only commonality being their tie to the ggplot2 API.
- ggthemes: extra themes, geoms, and scales for ggplot2
- ggalluvial: mapping survey data
- esquisse: a Shiny gadget to create ggplot charts interactively with drag-and-drop to map your variables, explore and visualize your data
- dichromat: accommodating color blindness

Additional references:

- <https://ggplot2.tidyverse.org/index.html>
- <https://towardsdatascience.com/ten-random-but-useful-things-to-know-about-ggplot2-197dc4439d10>
- <https://datavizpyr.com/category/r/ggplot2/>

