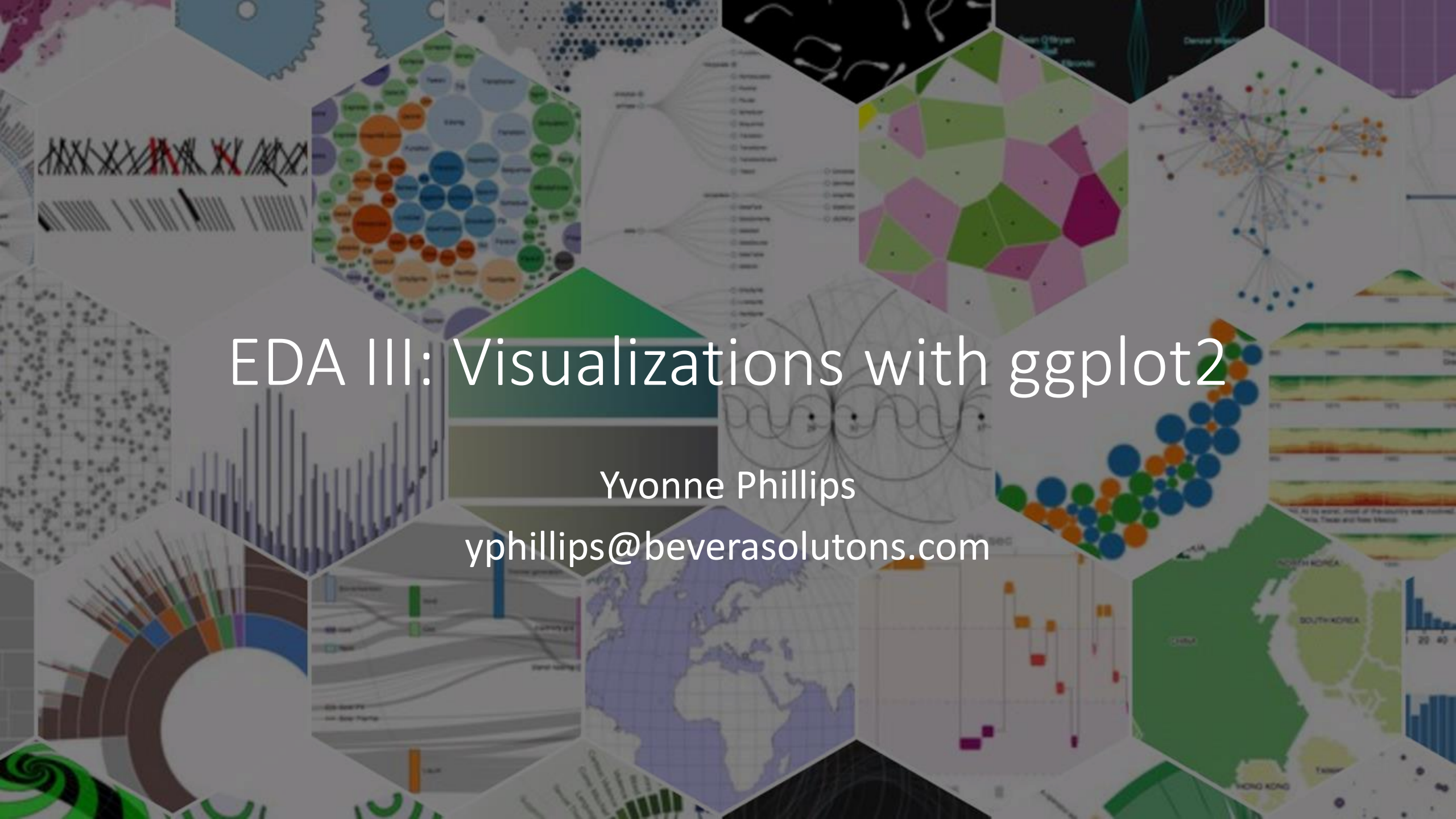


EDA III: Visualizations with ggplot2

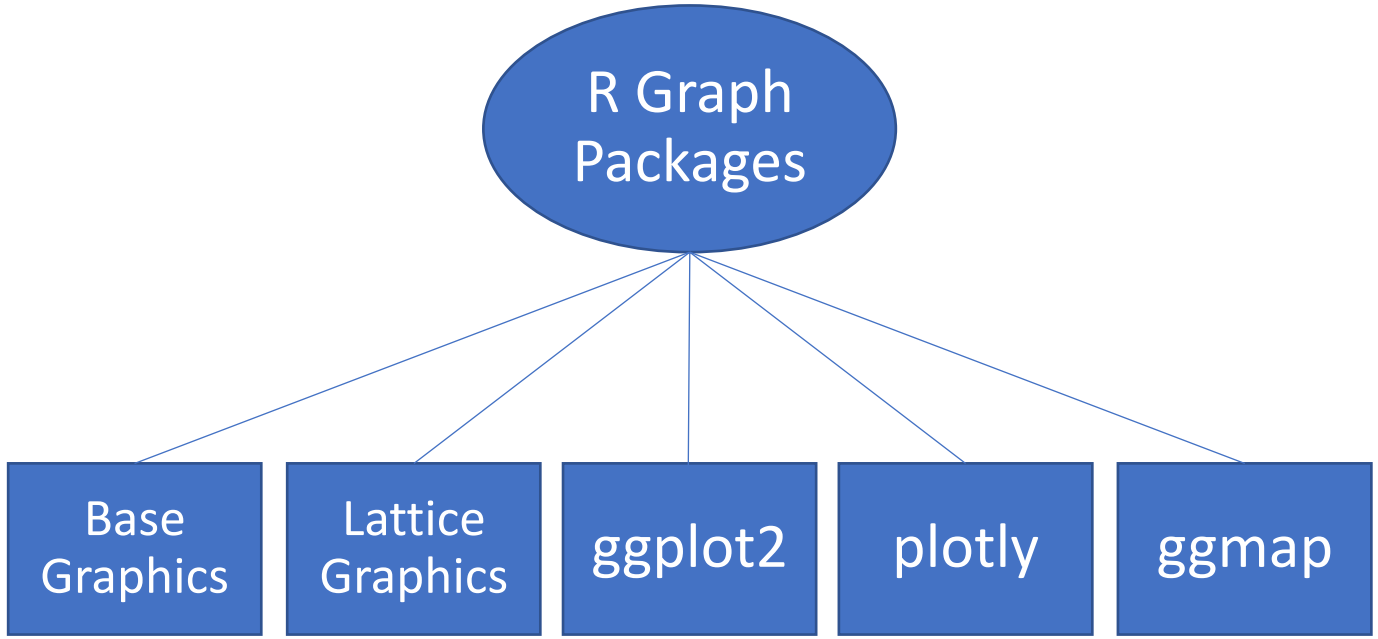
Yvonne Phillips

yphillips@beverasolutons.com

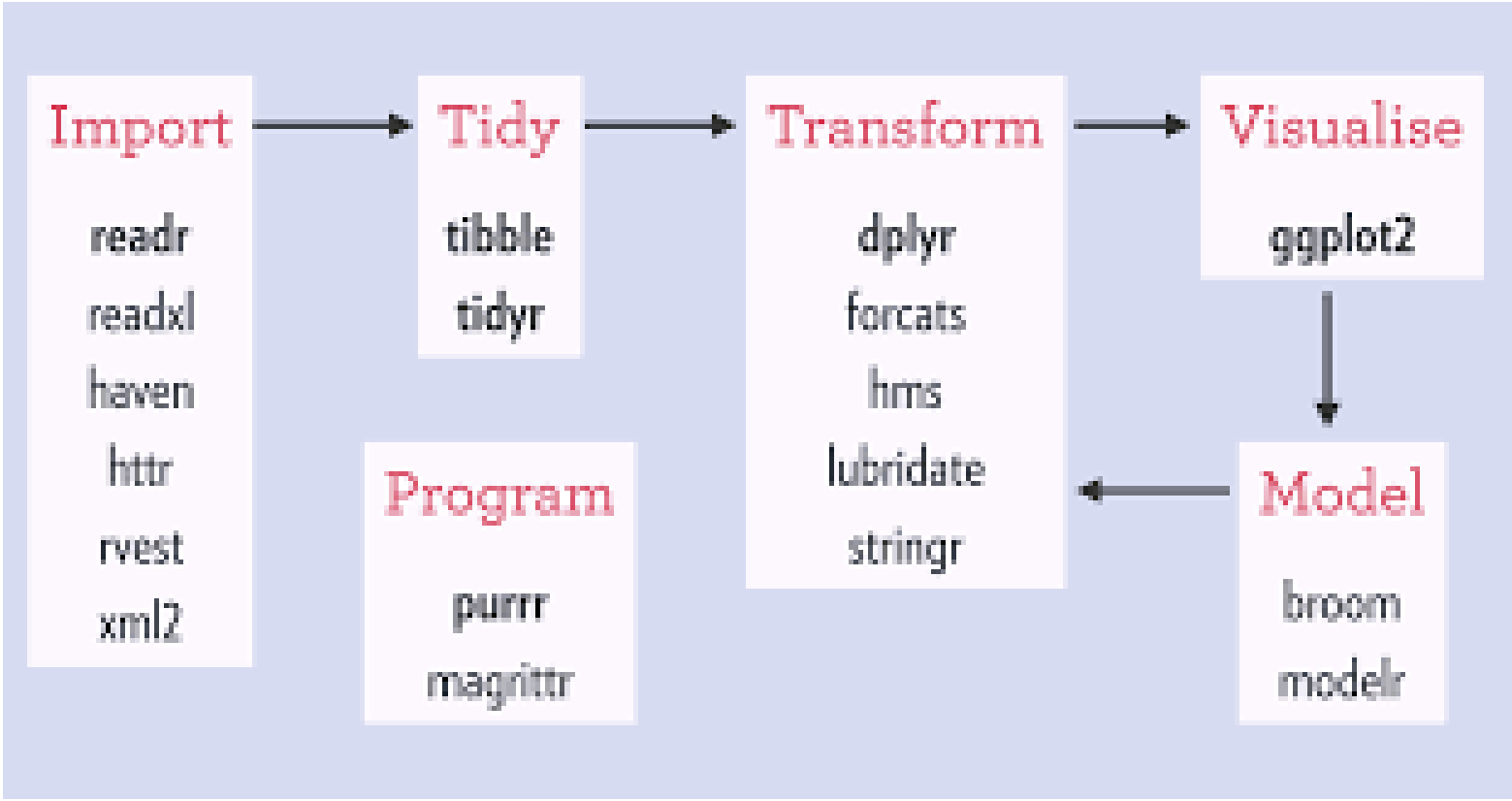
- Foundation of graphic applications
  - Grammar of Graphics
- Focus on teaching the underlying theory of ggplot2
- How is Grammar of Graphics reflected in the API
- Chart types
- Dealing with colors


1
Collect the data and gain domain knowledge
2
Confirm data types and their probabilities
3
Measures of Central Tendency
a. Mean
b. Median
C. Mode
4
Measures of Dispersion
a. Variance
b. Std. Dev.
c. Range
5
Skewness right & left
Kurtosis
Thinner peck
Wider peck
6
Graphical Representation
a. Histogram
b. Boxplot
c. Dotplot
d. Stem and Leaf
EXPLORATORY DATA ANALYSIS

Data visualization in R can be performed in several ways:



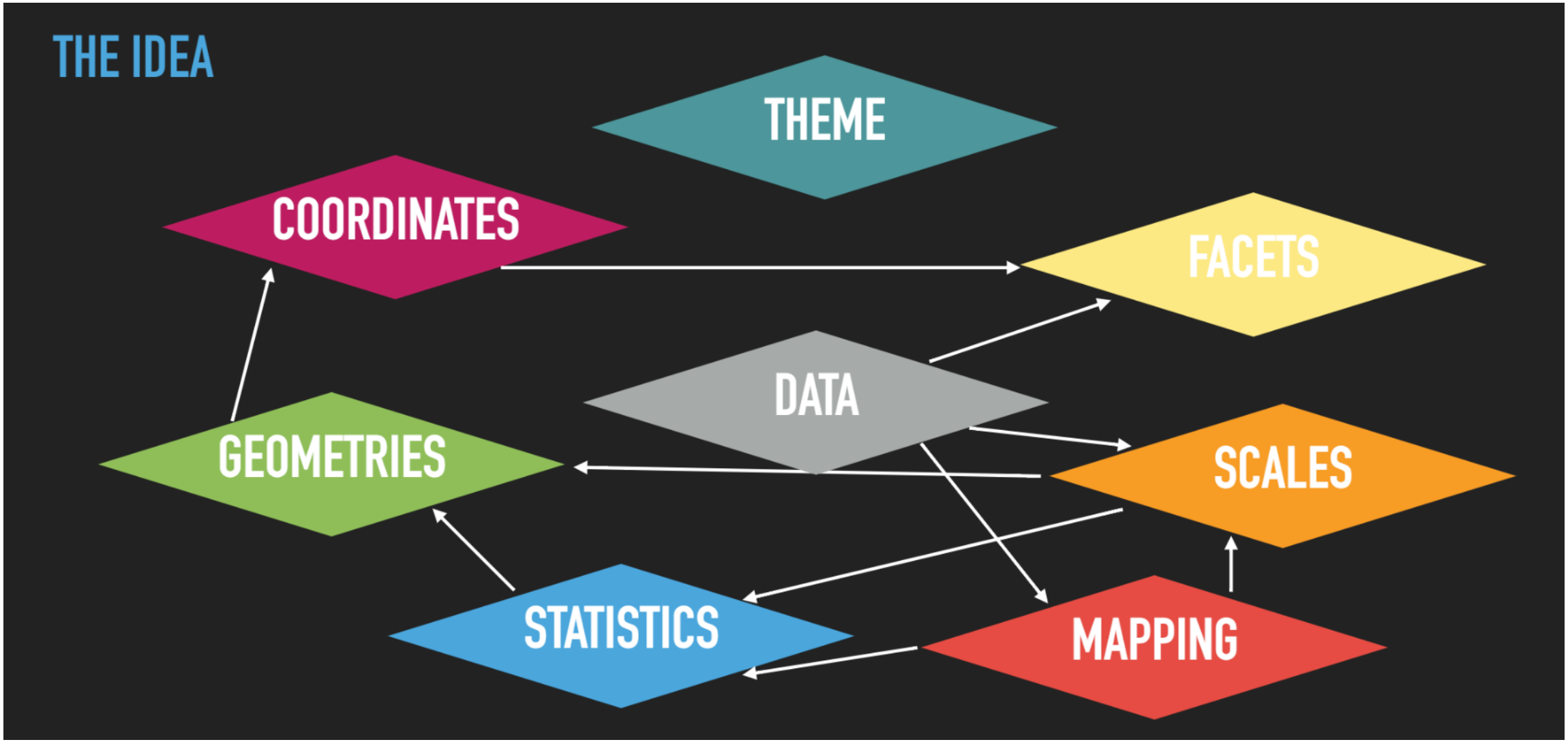
https://www.tidyverse.org/




```
install.packages("tidyverse")

library("tidyverse")
```

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

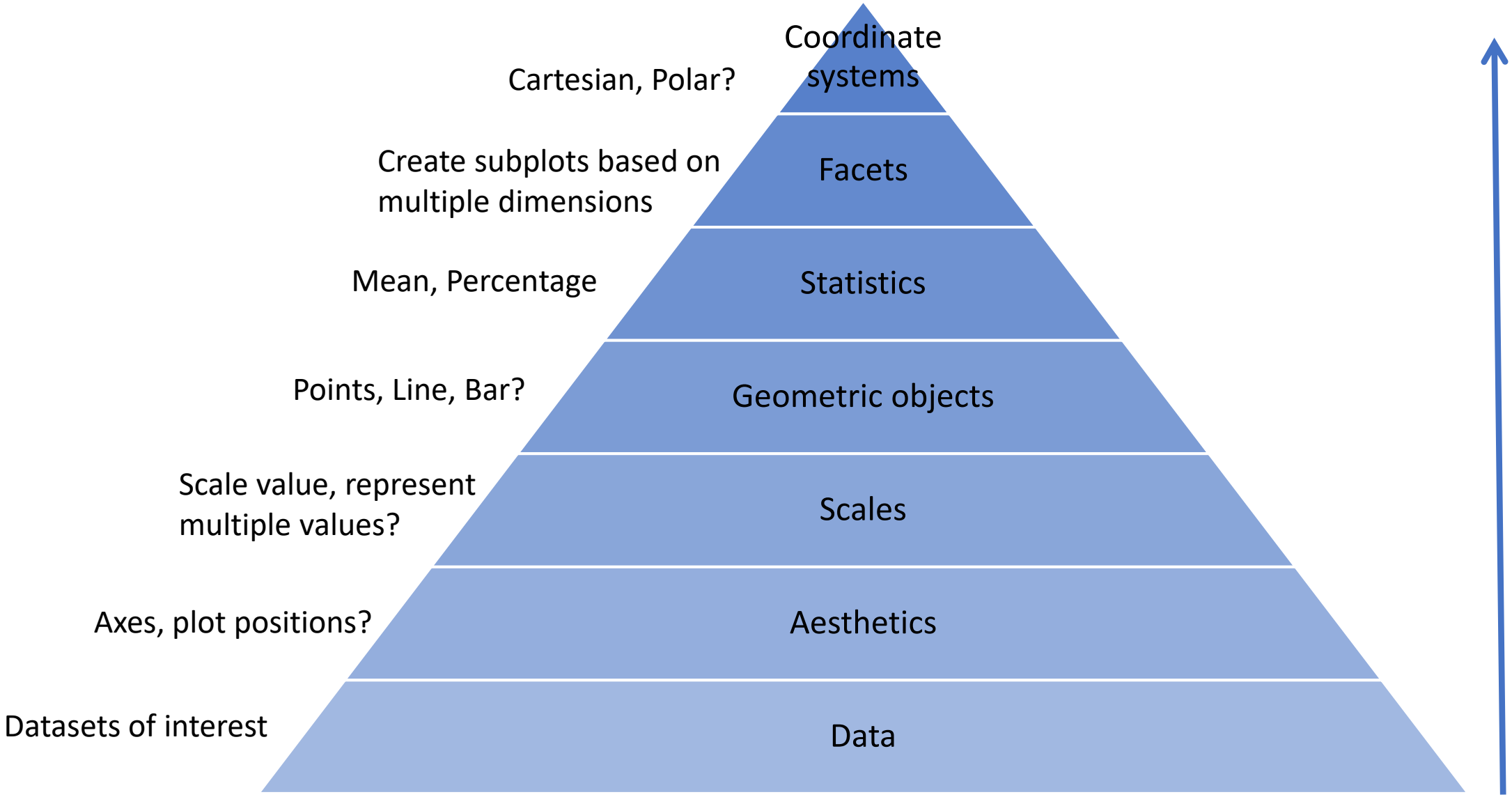

THE IDEA
THEME
COORDINATES
FACETS
DATA
GEOMETRIES
SCALES
STATISTICS
MAPPING

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.

https://ggplot2.tidyverse.org/reference/

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.

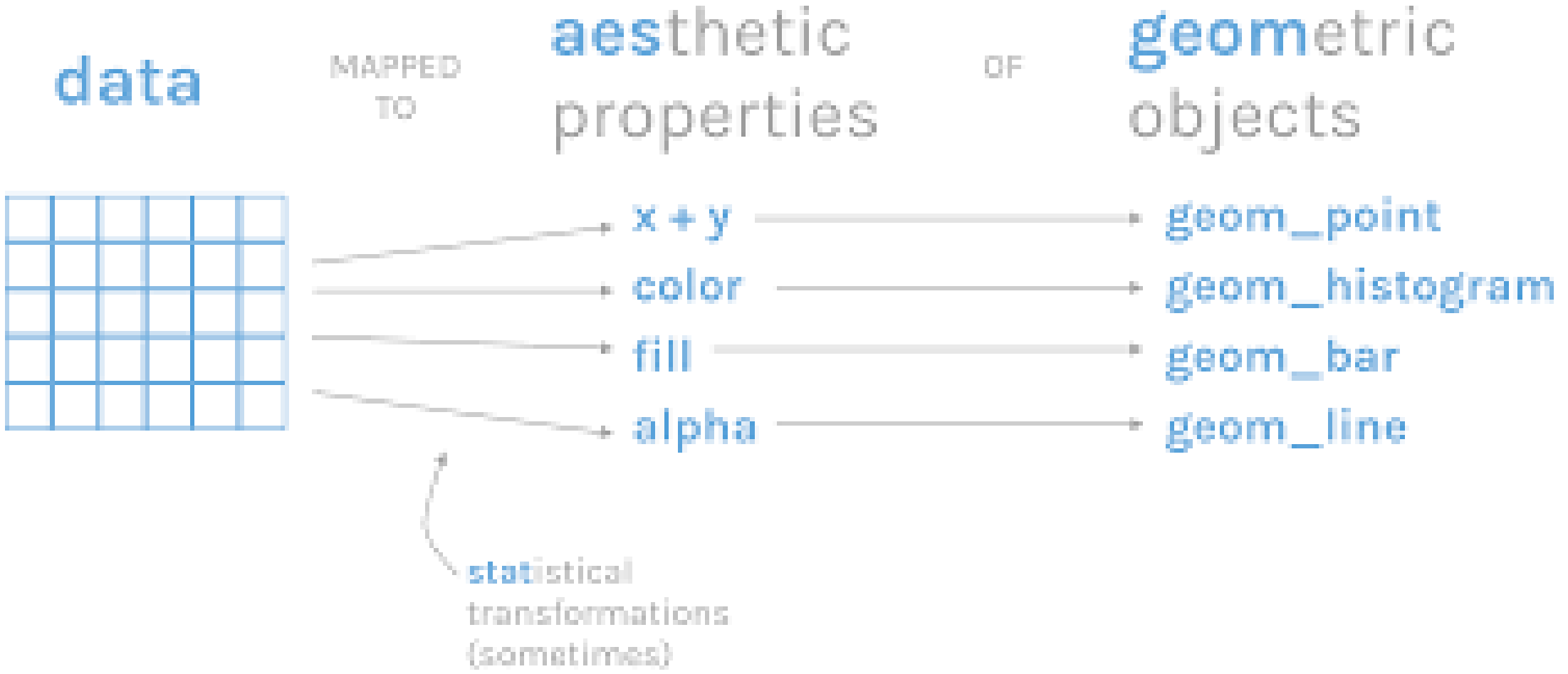
8. **Theme** controls the visual style of plot with font types, font sizes, background colors, margins, and positioning.

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
ggsave()	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)

Data visualization with ggplot2 : : CHEAT SHEET

Basics

ggplot2 is based on the grammar of graphics, the idea that you can build every graph from the same components: a data set, a coordinate system, and geoms—visual marks that represent data points.

To display values, map variables in the data to visual properties of the geom (aesthetics) like size, color, and x and y locations.

Complete the template below to build a graph.

```
ggplot(data = <DATA>) +                                  required
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
  stat = <STAT>, position = <POSITION>) +                not required, sensible defaults supplied
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION> +
  <SCALE_FUNCTION> +
  <THEME_FUNCTION>
```

ggplot(data = mpg, aes(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

last_plot() Returns the last plot.

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Aes

Common aesthetic values.

color and fill - string ("red", "#RRGGBB")

linetype - integer or string (0 = "blank", 1 = "solid", 2 = "dashed", 3 = "dotted", 4 = "dotdash", 5 = "longdash", 6 = "twodash")

linewidth - string ("round", "butt", or "square")

linejoin - string ("round", "mitre", or "bevel")

size - integer (line width in mm)

shape - integer/shape name or a single character ("a")

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(seals, aes(x = long, y = lat))
```

a + geom_blank() and **a + expand_limits()** Ensure limits include values across all plots.

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

a + geom_path(lineend = "butt", linejoin = "round", linemitre = 1**)** x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(alpha = 50)**)** - x, y, alpha, color, fill, group, subgroup, linetype, size

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

a + geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900)**)** - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

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

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

b + geom_segment(aes(yend = lat + 1, xend = long + 1)**)**
b + geom_spoke(aes(angle = 1:1155, radius = 1)**)**

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```

c + geom_area(stat = "bin"**)** x, y, alpha, color, fill, linetype, size

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

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

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5**)** x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)**)** x, y, alpha, color, fill, linetype, size, weight

discrete

```
d <- ggplot(mpg, aes(fl))
```

d + geom_bar() x, alpha, color, fill, linetype, size, weight

TWO VARIABLES both continuous

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

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1**)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_point() x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile() x, y, alpha, color, group, linetype, size, weight

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

e + geom_smooth(method = lm**)** x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1**)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

one discrete, one continuous

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

f + geom_col() x, y, alpha, color, fill, group, linetype, size

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

f + geom_dotplot(binaxis = "y", stackdir = "center"**)** x, y, alpha, color, fill, group

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

both discrete

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

g + geom_count() x, y, alpha, color, fill, shape, size, stroke

e + geom_jitter(height = 2, width = 2**)** x, y, alpha, color, fill, shape, size

continuous bivariate distribution

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

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

h + geom_density_2d() x, y, alpha, color, group, linetype, size

h + geom_hex() x, y, alpha, color, fill, size

continuous function

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

i + geom_area() x, y, alpha, color, fill, linetype, size

i + geom_line() x, y, alpha, color, group, linetype, size

i + 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))
```

j + geom_crossbar(fatten = 2**)** - x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar() - x, ymax, ymin, alpha, color, group, linetype, size, width
Also **geom_errorbarh()**.

j + geom_linerange() x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange() - x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

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

k + geom_map(aes(map_id = state), map = map**)**
+ expand_limits(x = map$long, y = map$lat**)**
map_id, alpha, color, fill, linetype, size

THREE VARIABLES

```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))
```

l + geom_contour(aes(z = z)**)** x, y, z, alpha, color, group, linetype, size, weight

l + geom_contour_filled(aes(fill = z)**)** x, y, alpha, color, fill, group, linetype, size, subgroup

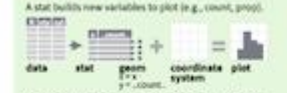
l + geom_raster(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE**)** x, y, alpha, fill

l + geom_tile(aes(fill = z)**)** x, y, alpha, color, fill, linetype, size, width

Stats

An alternative way to build a layer.

A stat builds new variables to plot (e.g., count, prop).



Visualize a stat by changing the default stat of a geom function, **geom_bar(stat="count")** or by using a stat function, **stat_count(geom="bar")**, which calls a default geom to make a layer (equivalent to a geom function). Use **..name..** syntax to map stat variables to aesthetics.

c + stat_bin(binwidth = 1, boundary = 10**)**
x, y | ..count.., ..ncount.., ..density.., ..ndensity..

c + stat_count(width = 1**)** **x, y** | ..count.., ..prop..

c + stat_density(adjust = 1, kernel = "gaussian"**)**
x, y | ..count.., ..density.., ..scaled..

e + stat_bin_2d(bins = 30, drop = T**)**
x, y, fill | ..count.., ..density..

e + stat_bin_hex(bins = 30**)** **x, y, fill** | ..count.., ..density..

e + stat_density_2d(contour = TRUE, n = 100**)**
x, y, color, size | ..level..

e + stat_ellipse(level = 0.95, segments = 51, type = "t"**)**

l + stat_contour(aes(z = z)**)** **x, y, z, order** | ..level..

l + stat_summary_hex(aes(z = z), bins = 30, fun = max**)**
x, y, z, fill | ..value..

l + stat_summary_2d(aes(z = z), bins = 30, fun = mean**)**
x, y, z, fill | ..value..

f + stat_boxplot(coef = 1.5**)**
x, y | ..lower.., ..middle.., ..upper.., ..width.., ..ymin.., ..ymax..

f + stat_ydensity(kernel = "gaussian", scale = "area"**)** **x, y**
| ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

e + stat_ecdf(n = 40**)** **x, y** | ..x.., ..y..

e + stat_quantile(quantiles = c(0.1, 0.9),
formula = y ~ log(x), method = "rq"**)** **x, y** | ..quantile..

e + stat_smooth(method = "lm", formula = y ~ x, se = T,
level = 0.95**)** **x, y** | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

ggplot() + xlim(-5, 5) + stat_function(fun = dnorm, n = 20, geom = "point"**)** ..x.., ..y..

ggplot() + stat_qq(aes(sample = 1:100)**)**
x, y, sample | ..sample.., ..theoretical..

e + stat_sum() **x, y, size** | ..n.., ..prop..

e + stat_summary(fun.data = "mean_cl_boot"**)**

h + stat_summary_bin(fun = "mean", geom = "bar"**)**

e + stat_identity()

e + stat_unique()

Scales

Override defaults with **scales** package.

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

GENERAL PURPOSE SCALES

Use with most aesthetics

scale_*_continuous() - Map cont' values to visual ones.
scale_*_discrete() - Map discrete values to visual ones.
scale_*_binned() - Map continuous values to discrete bins.
scale_*_identity() - Use data values as visual ones.
scale_*_manual(values = c()**)** - Map discrete values to manually chosen visual ones.
scale_*_date(date_labels = "%m/%d"),
date_breaks = "2 weeks"**)** - Treat data values as dates.
scale_*_datetime() - Treat data values as date times. Same as scale_*_date(). See ?strptime for label formats.

X & Y LOCATION SCALES

Use with x or y aesthetics (x shown here)

scale_x_log10() - Plot x on log10 scale.
scale_x_reverse() - Reverse the direction of the x axis.
scale_x_sqrt() - Plot x on square root scale.

COLOR AND FILL SCALES (DISCRETE)

n + scale_fill_brewer(palette = "Blues"**)**
For palette choices:
RColorBrewer::display.brewer.all()

n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red"**)**

COLOR AND FILL SCALES (CONTINUOUS)

```
c <- c + geom_dotplot(aes(fill = ..x..))
```

n + scale_fill_distiller(palette = "Blues"**)**

n + scale_fill_gradient(low="red", high="yellow"**)**

n + scale_fill_gradient2(low = "red", high = "blue", mid = "white", midpoint = 25**)**

n + scale_fill_gradientn(colors = topo.colors(6)**)**
Also: rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal()

SHAPE AND SIZE SCALES

```
p <- e + geom_point(aes(shape = fl, size = cyl))
```

p + scale_shape() + **scale_size()**
p + scale_shape_manual(values = c(3:7)**)**

p + scale_radius(range = c(1,6)**)**
p + scale_size_area(max_size = 6**)**

Coordinate Systems

```
r <- d + geom_bar()
```

r + coord_cartesian(xlim = c(0, 5)**)** - xlim, ylim
The default cartesian coordinate system.

r + coord_fixed(ratio = 1/2**)**
ratio, xlim, ylim - Cartesian coordinates with fixed aspect ratio between x and y units.

ggplot(mpg, aes(y = fl)**)** + **geom_bar()**
Flip cartesian coordinates by switching x and y aesthetic mappings.

r + coord_polar(theta = "x", direction=1**)**
theta, start, direction - Polar coordinates.

r + coord_trans(y = "sqrt"**)** - x, y, xlim, ylim
Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.

π + coord_quickmap()
π + coord_map(projection = "ortho", orientation = c(41, -74, 0)**)** - projection, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.).

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

```
s <- ggplot(mpg, aes(fl, fill = drv))
```

s + geom_bar(position = "dodge")
Arrange elements side by side.

s + geom_bar(position = "fill")
Stack elements on top of one another, normalize height.

e + geom_point(position = "jitter")
Add random noise to X and Y position of each element to avoid overplotting.

e + geom_label(position = "nudge")
Nudge labels away from points.

s + geom_bar(position = "stack")
Stack elements on top of one another.

Each position adjustment can be recast as a function with manual **width** and **height** arguments.
s + geom_bar(position = position_dodge(width = 1))

Themes

r + theme_bw()
White background with grid lines

r + theme_gray()
Grey background (default theme).

r + theme_dark()
Dark for contrast.

r + theme_classic()

r + theme_light()

r + theme_linedraw()

r + theme_minimal()
Minimal theme.

r + theme_void()
Empty theme.

r + theme() Customize aspects of the theme such as axis, legend, panel, and facet properties.
r + ggtitle("Title") + theme(plot.title.position = "plot")
r + theme(panel.background = element_rect(fill = "blue"))

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

```
t <- ggplot(mpg, aes(cty, hwy)) + geom_point()
```

t + facet_grid(cols = vars(fl)**)**
Facet into columns based on fl.

t + facet_grid(rows = vars(year)**)**
Facet into rows based on year.

t + facet_grid(rows = vars(year), cols = vars(fl)**)**
Facet into both rows and columns.

t + facet_wrap(vars(fl)**)**
Wrap facets into a rectangular layout.

Set **scales** to let axis limits vary across facets.

t + facet_grid(rows = vars(drv), cols = vars(fl), scales = "free"**)**
x and y axis limits adjust to individual facets:
"free_x" - x axis limits adjust
"free_y" - y axis limits adjust

Set **labeller** to adjust facet label.

t + facet_grid(cols = vars(fl), labeller = label_both**)**

t + facet_grid(rows = vars(fl), labeller = label_bquote(alpha ^ .(fl))**)**

Labels and Legends

Use **labs()** to label the elements of your plot.

t + labs(x = "New x axis label", y = "New y axis label", title ="Add a title above the plot", subtitle = "Add a subtitle below title", caption = "Add a caption below plot", alt = "Add alt text to the plot", <AES> = "New <AES> legend title"**)**

t + annotate(geom = "text", x = 8, y = 9, label = "A"**)**
Places a geom with manually selected aesthetics.

p + guides(x = guide_axis(n.dodge = 2)**)** Avoid crowded or overlapping labels with guide_axis(n.dodge or angle).

n + guides(fill = "none"**)** Set legend type for each aesthetic: colorbar, legend, or none (no legend).

n + theme(legend.position = "bottom"**)**
Place legend at "bottom", "top", "left", or "right".

n + scale_fill_discrete(name = "Title", labels = c("A", "B", "C", "D", "E")**)**
Set legend title and labels with a scale function.

Zooming

Without clipping (preferred):

t + coord_cartesian(xlim = c(0, 100), ylim = c(10, 20)**)**

With clipping (removes unseen data points):

t + xlim(0, 100**)** + **ylim(**10, 20**)**

t + scale_x_continuous(limits = c(0, 100)**)** + **scale_y_continuous(**limits = c(0, 100)**)**

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().

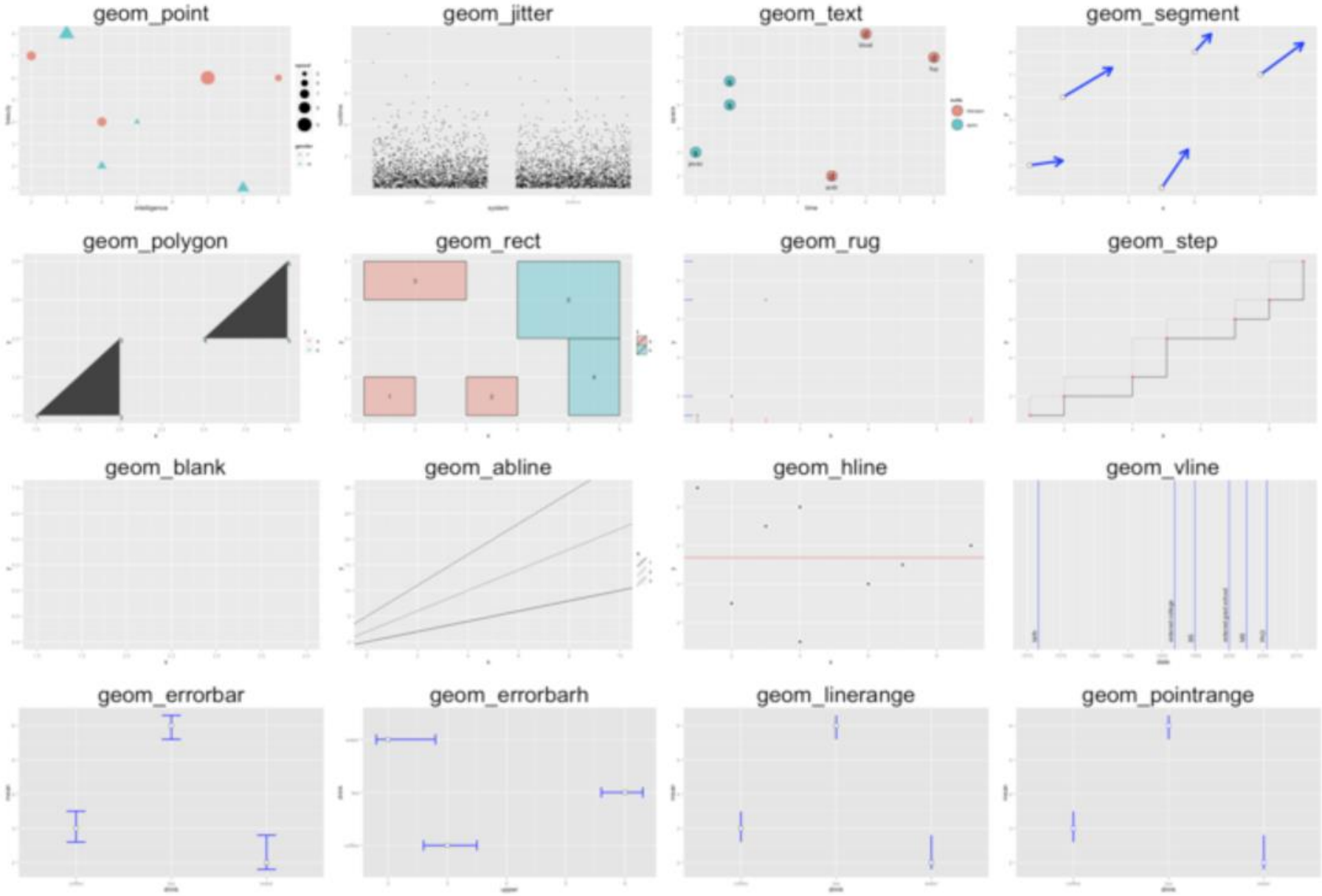
2. **data**: A dataset which overrides the default plot dataset. It is usually omitted (set to NULL), in which case the layer will use the default data specified in ggplot(). The requirements for data are explained in more detail in data. ggplot needs a data frame object for the input parameter, data, 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.

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

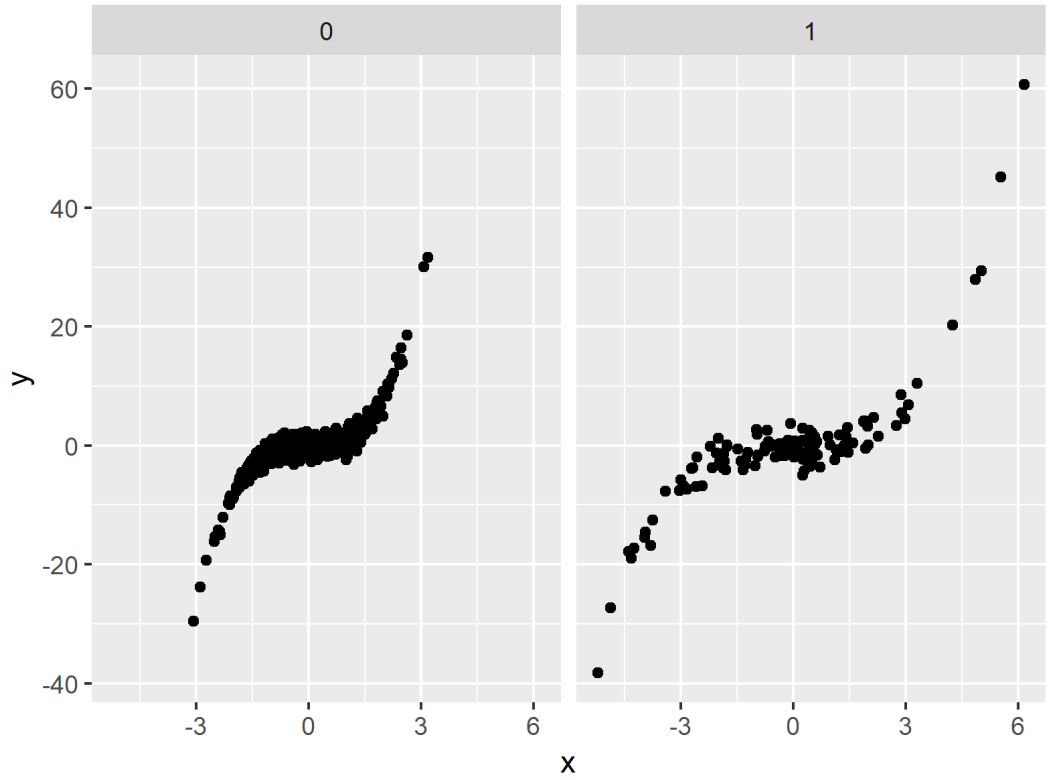
ggplot2: Layers: Aesthetics

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

https://ggplot2.tidyverse.org/reference/

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.



https://ggplot2.tidyverse.org/reference/

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.

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

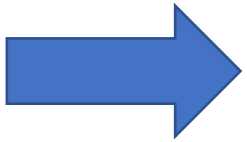
```
14  ## Download data from R using data() and see what the set is composed of
15  ## Make sure you download data in the working directory
16
17  data("airquality")
18  str(airquality)
```

```
> 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
```

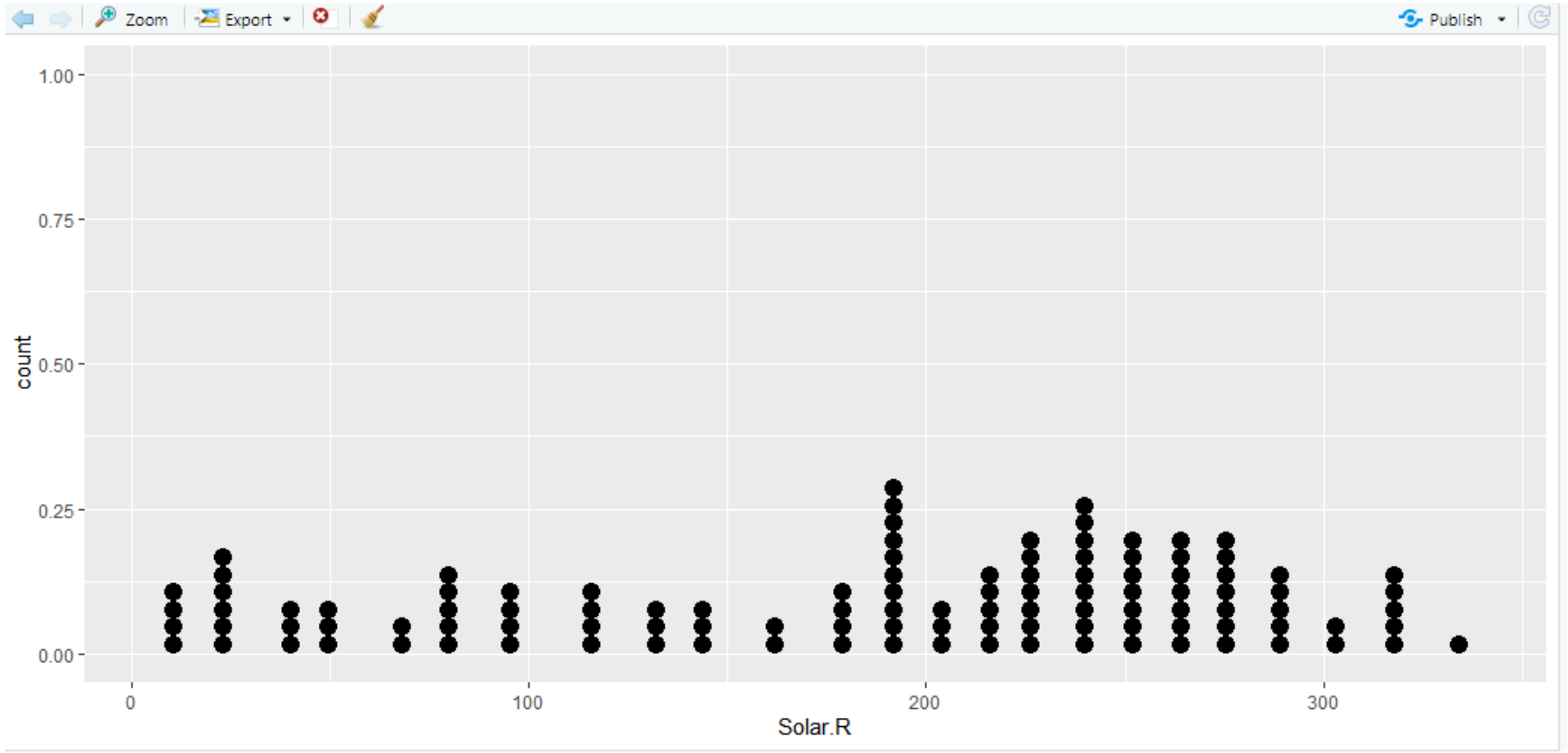
Output



```
> 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" ...
>
```

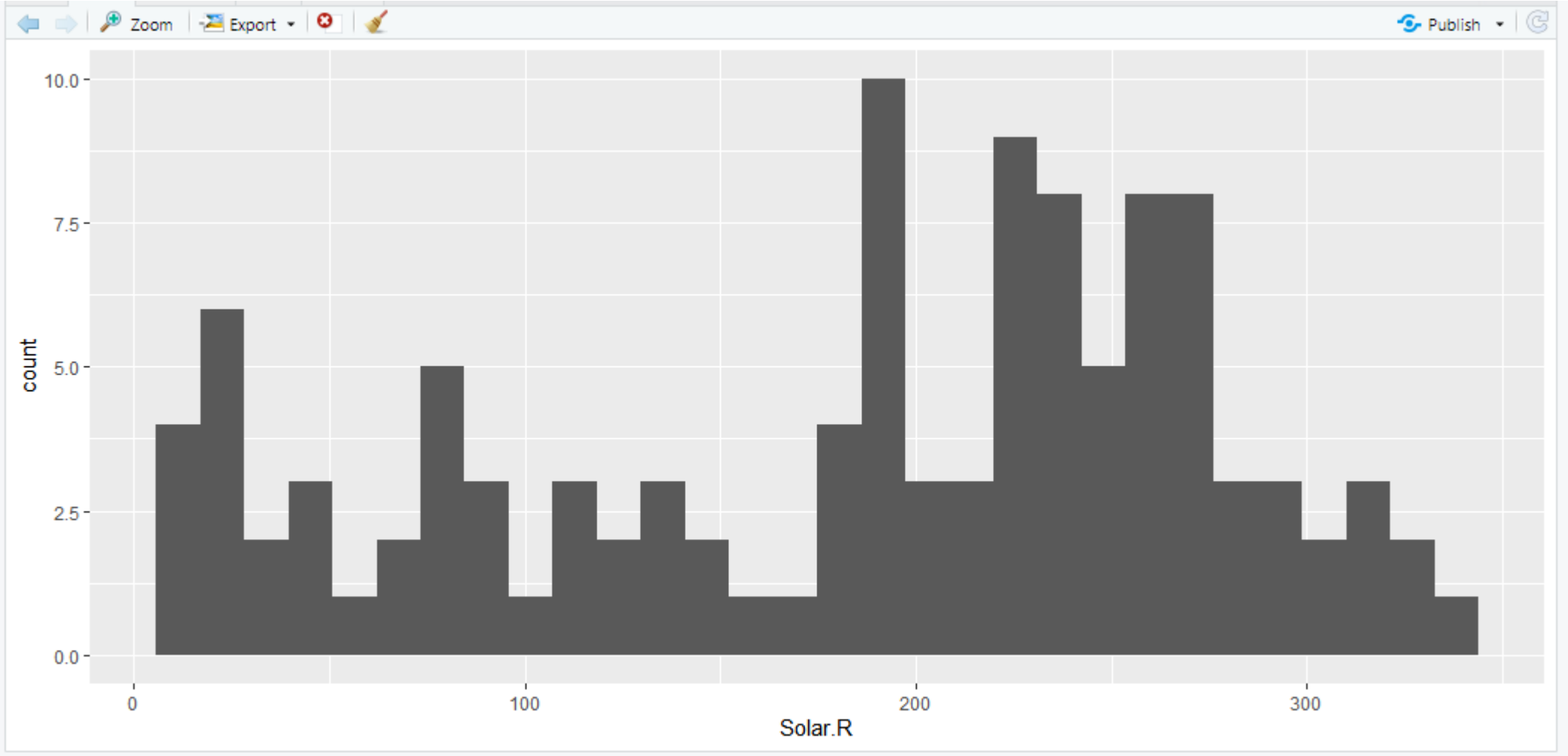
Dotplot

```
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)
```



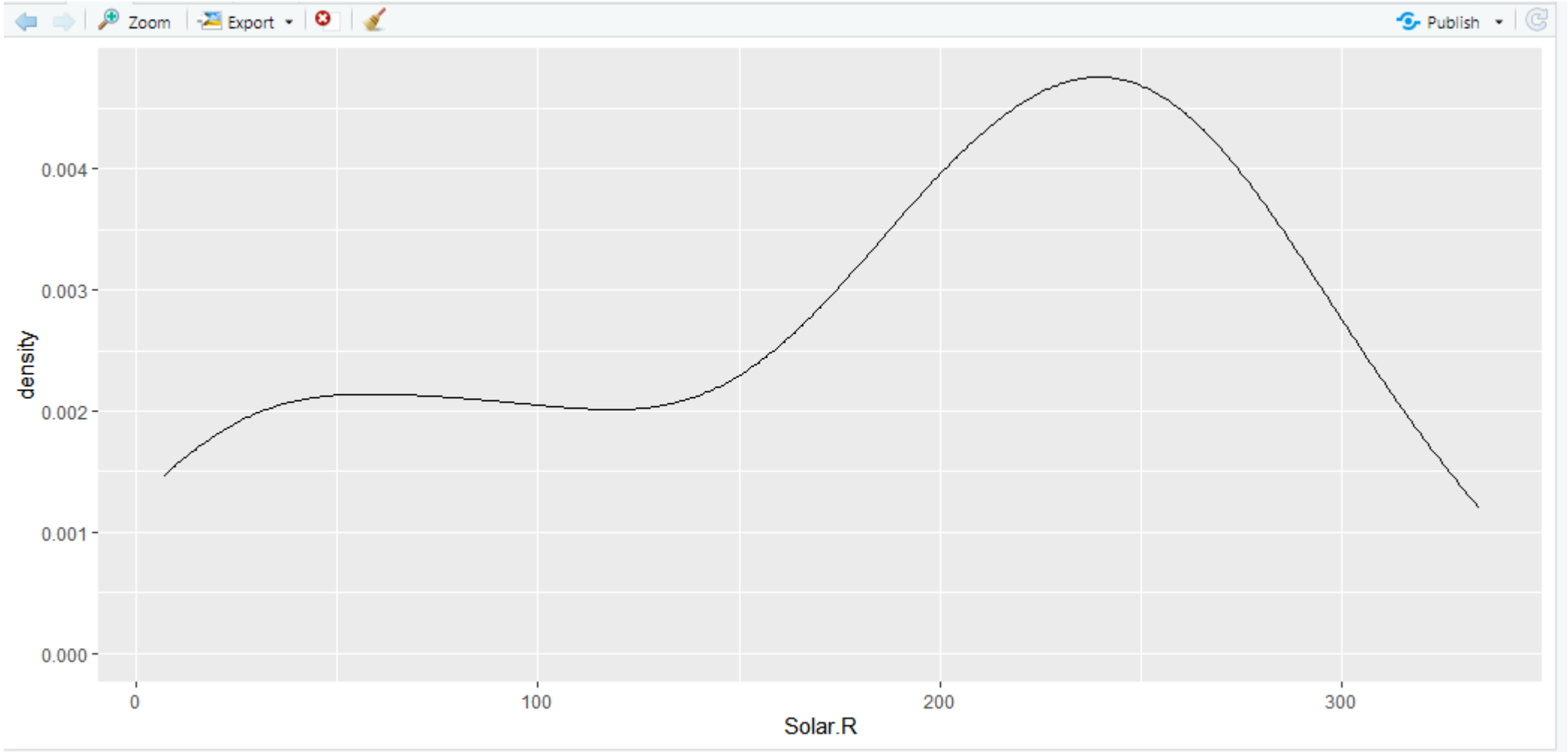
Histogram

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



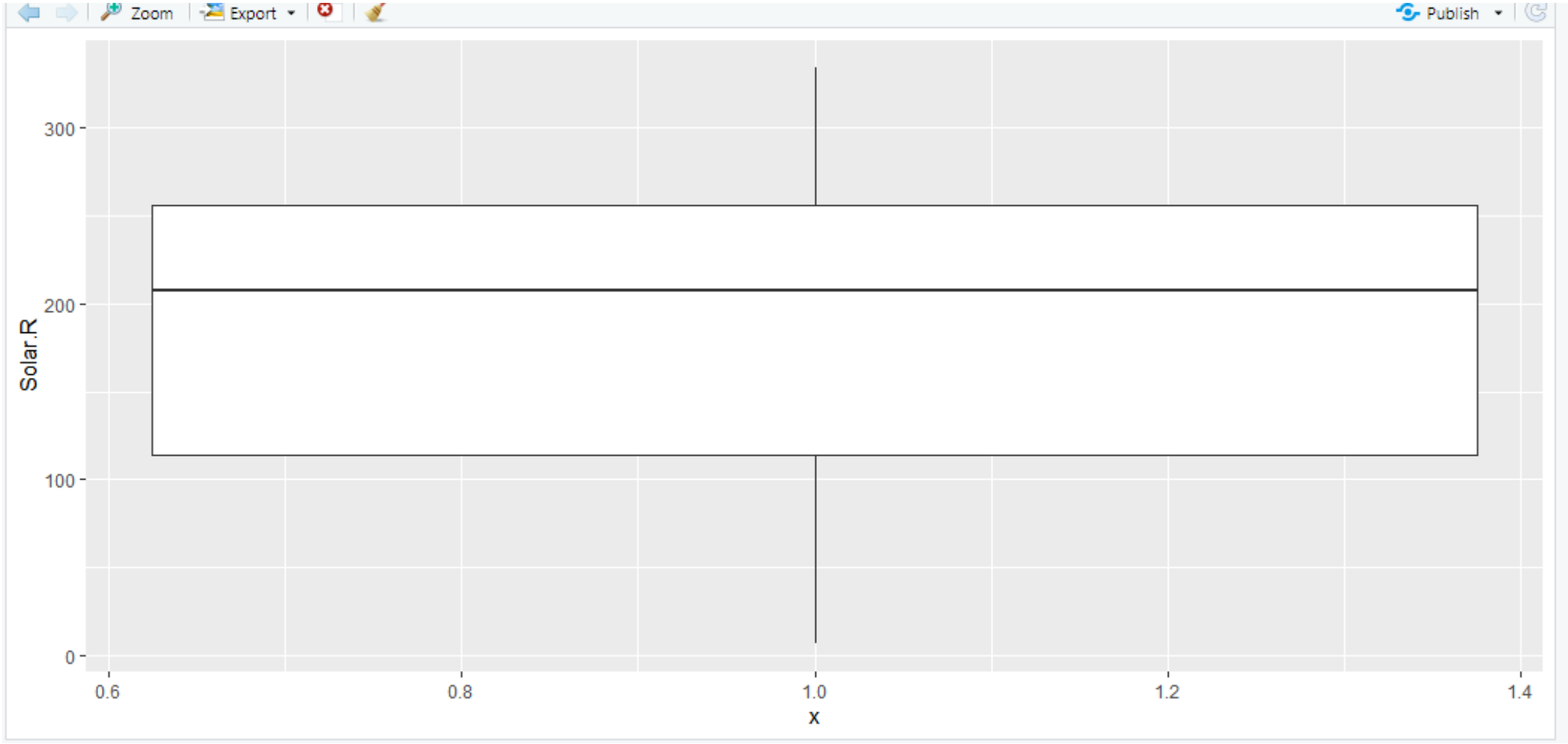
Density plot

```
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()
```



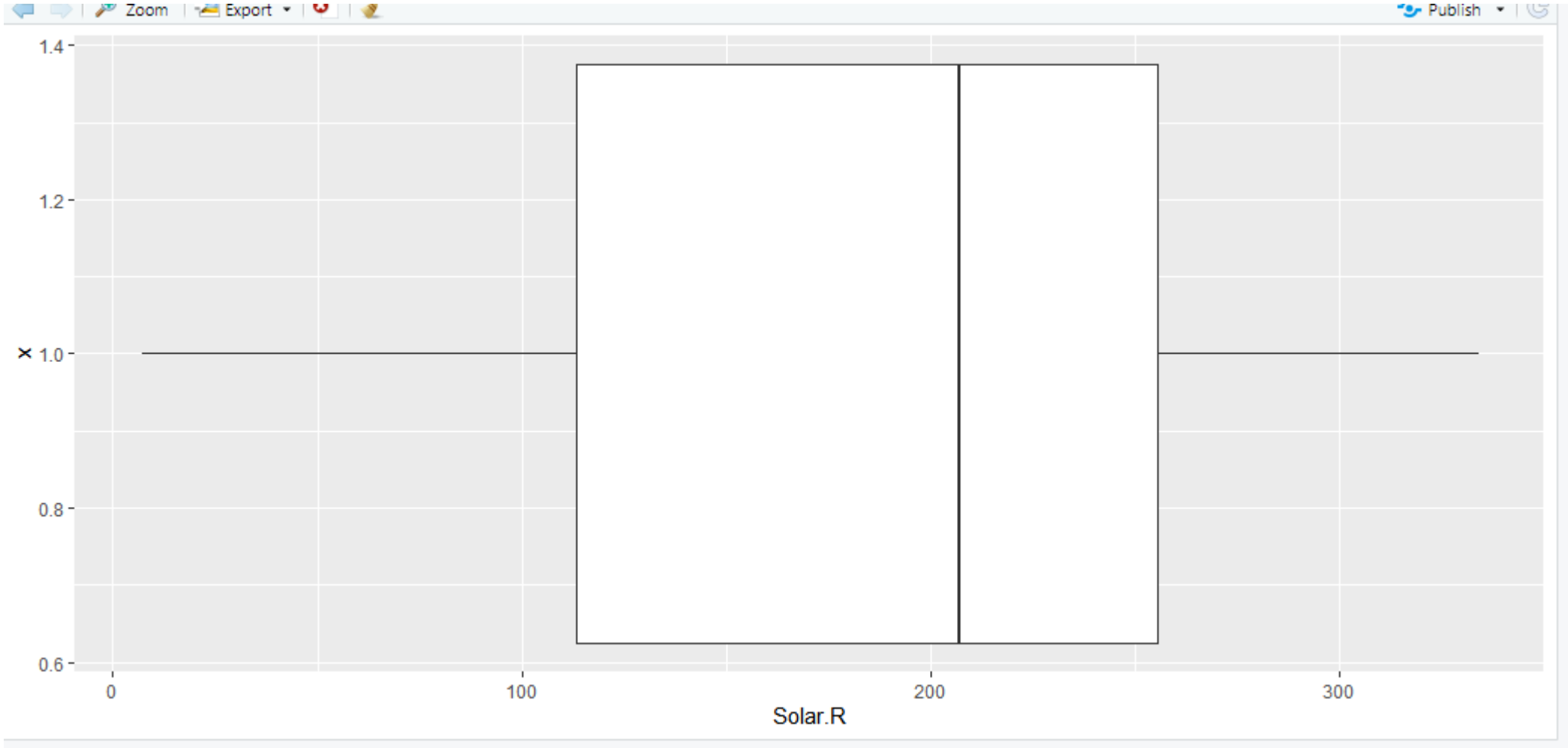
Boxplot

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



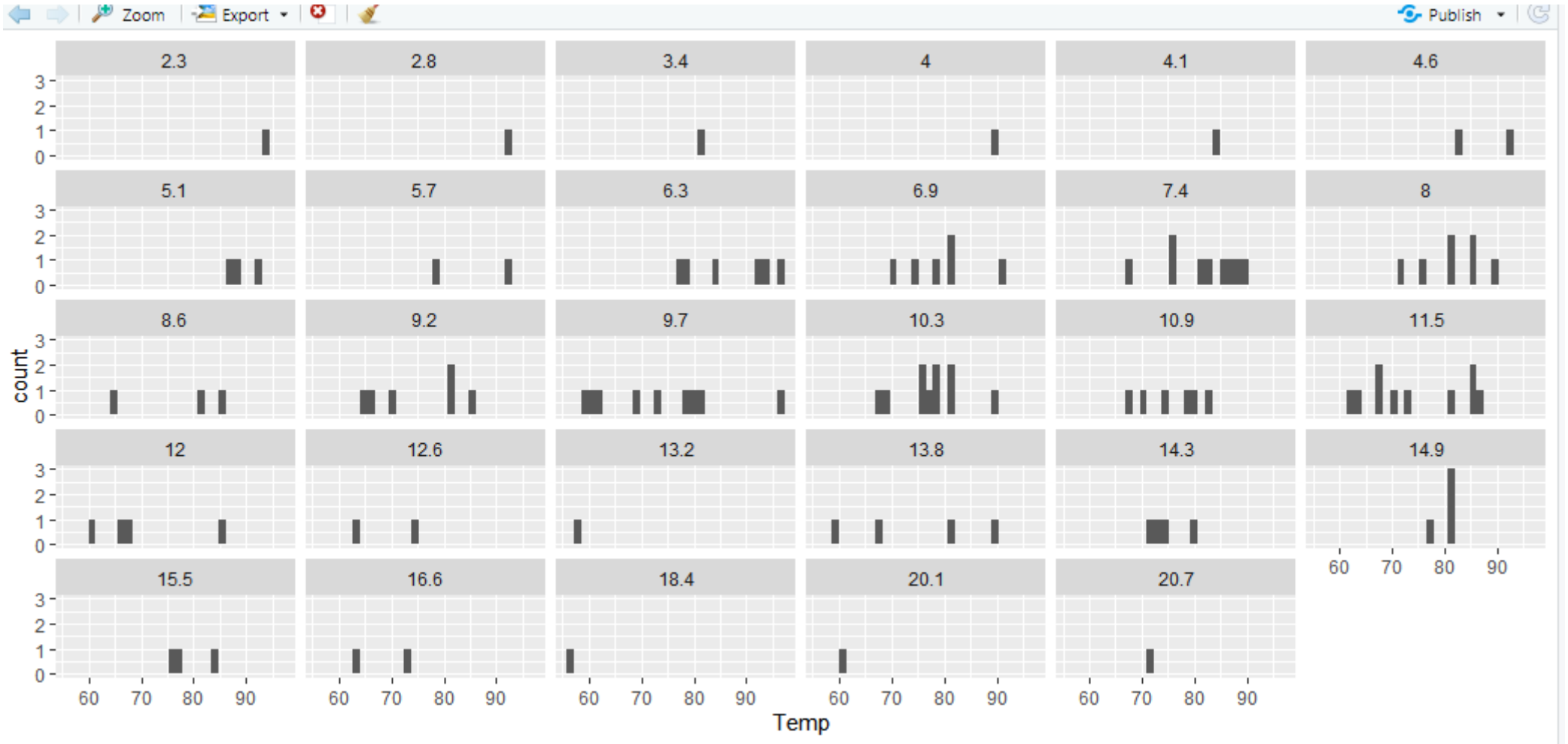
Boxplot (coord_flipped)

```
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", # outlines the shape in the color*

  fill="69b3a2", …) # fills the shape
- *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 additionnal 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
  - Paletteer

R Color Brewer

Set1 (qualitative)

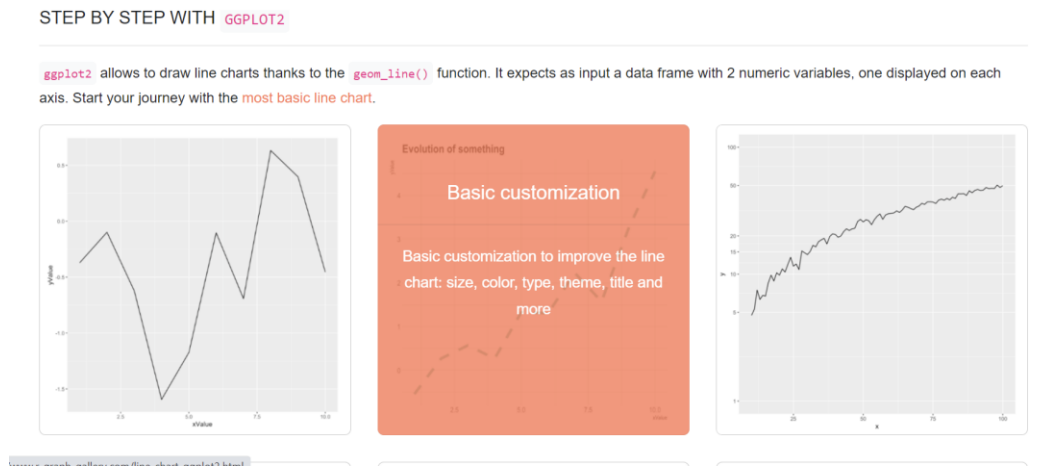
R Color Brewer's palettes

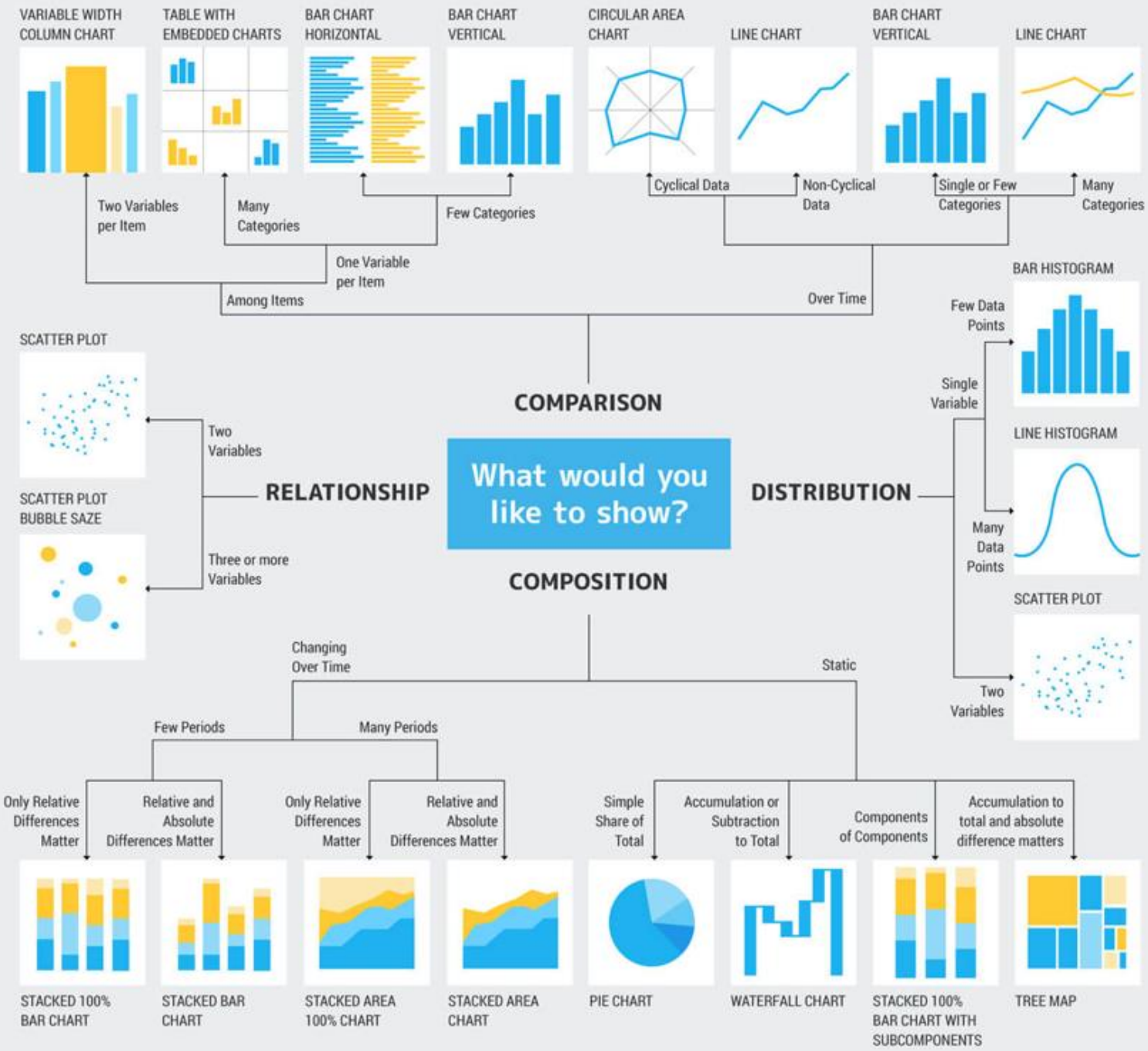
The `RColorBrewer` package offers several color palette for R. This post displays all of them to help you pick the right one.

COLOR SECTION | ABOUT LINE CHART


Practice
Practice
Practice

Chart types: examples here




What would you like to show?
COMPARISON
RELATIONSHIP
DISTRIBUTION
COMPOSITION
VARIABLE WIDTH COLUMN CHART
TABLE WITH EMBEDDED CHARTS
BAR CHART HORIZONTAL
BAR CHART VERTICAL
CIRCULAR AREA CHART
LINE CHART
BAR CHART VERTICAL
LINE CHART
Two Variables per Item
Many Categories
Few Categories
Cyclical Data
Non-Cyclical Data
Single or Few Categories
Many Categories
One Variable per Item
Among Items
Over Time
SCATTER PLOT
Two Variables
SCATTER PLOT BUBBLE SAZE
Three or more Variables
BAR HISTOGRAM
Few Data Points
Single Variable
LINE HISTOGRAM
Many Data Points
SCATTER PLOT
Two Variables
Changing Over Time
Static
Few Periods
Many Periods
Only Relative Differences Matter
Relative and Absolute Differences Matter
Only Relative Differences Matter
Relative and Absolute Differences Matter
Simple Share of Total
Accumulation or Subtraction to Total
Components of Components
Accumulation to total and absolute difference matters
STACKED 100% BAR CHART
STACKED BAR CHART
STACKED AREA 100% CHART
STACKED AREA CHART
PIE CHART
WATERFALL CHART
STACKED 100% BAR CHART WITH SUBCOMPONENTS
TREE MAP

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.

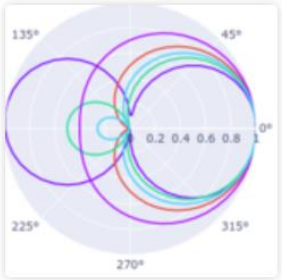
Plotly.R is free and open source and you can view the source, report issues or contribute on GitHub.

Deploy R AI Dash apps on private Kubernetes clusters: Pricing | Demo | Overview | AI App Services

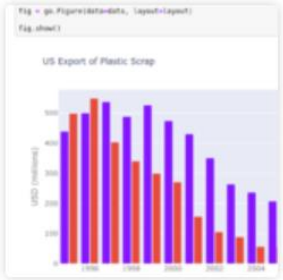
Fundamentals

More Fundamentals »

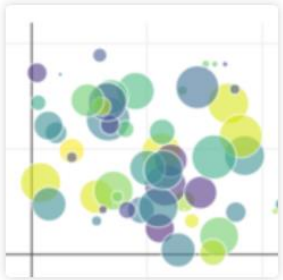
The Figure Data Structure



Creating and Updating Figures

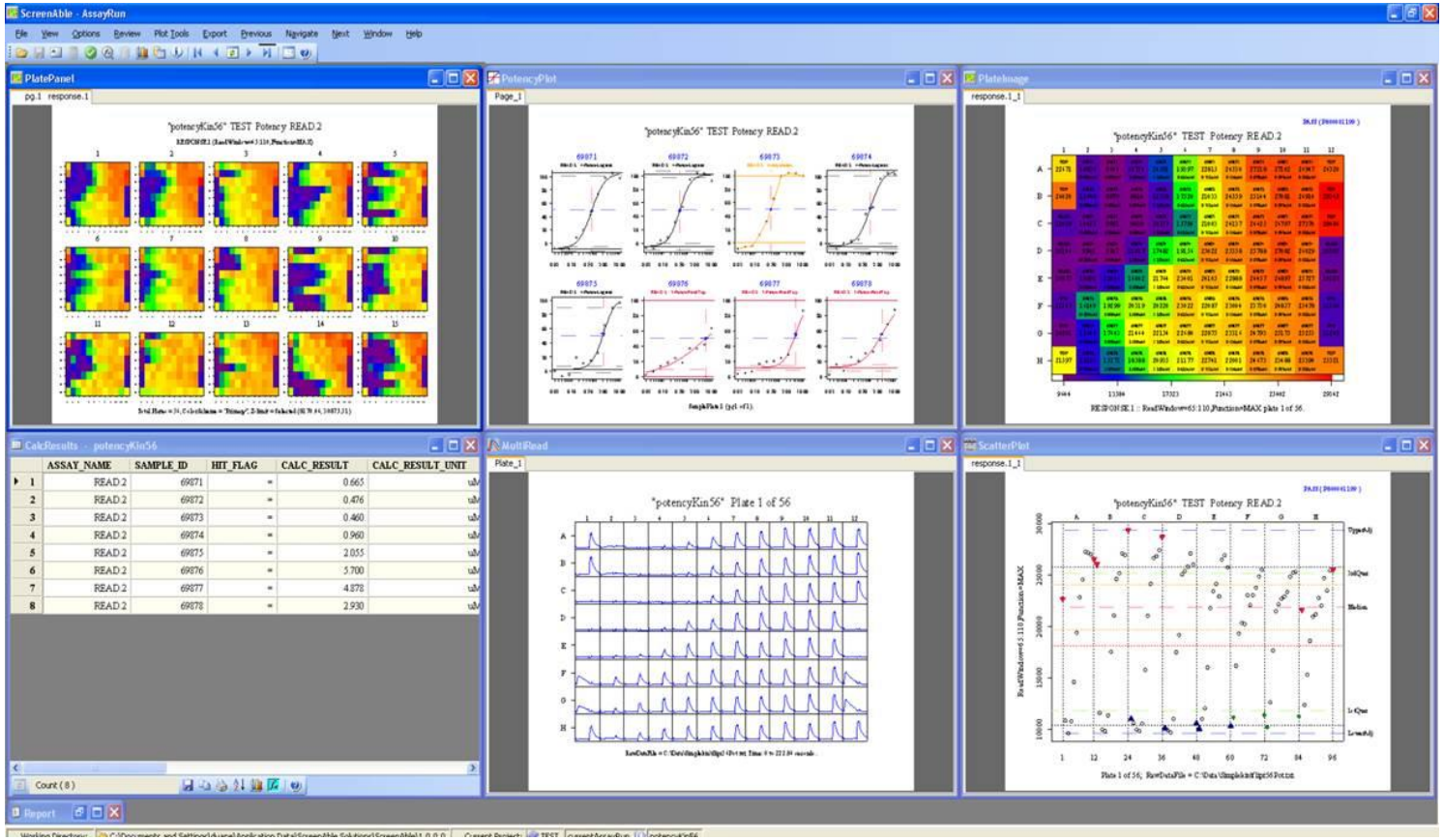


Displaying Figures



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/