

# **WHY YOU SHOULD START BY LEARNING DATA VISUALIZATION AND MANIPULATION**

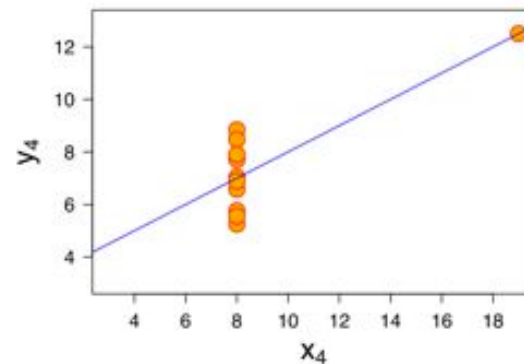
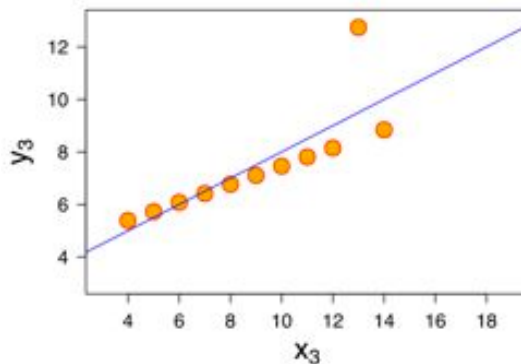
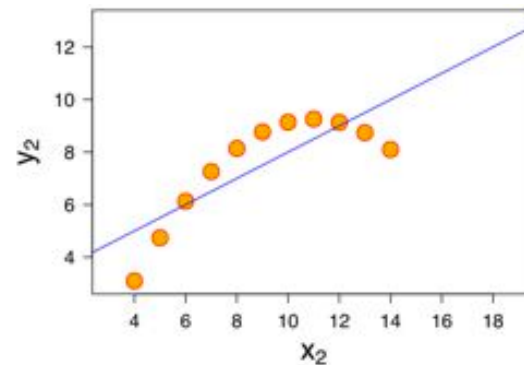
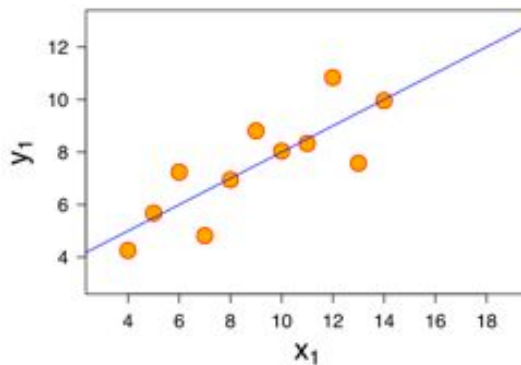
# Why visualize data?

- Four groups
- 11 observations (x, y) per group

Property	Value
Mean of $x$ in each case	9 (exact)
Sample variance of $x$ in each case	11 (exact)
Mean of $y$ in each case	7.50 (to 2 decimal places)
Sample variance of $y$ in each case	4.122 or 4.127 (to 3 decimal places)
Correlation between $x$ and $y$ in each case	0.816 (to 3 decimal places)
Linear regression line in each case	$y = 3.00 + 0.500x$ (to 2 and 3 decimal places, respectively)

# Why visualize data?

- Four groups
- 11 observations  $(x, y)$  per group



# R base graphics

`plot()`

generic x-y plotting

`barplot()`

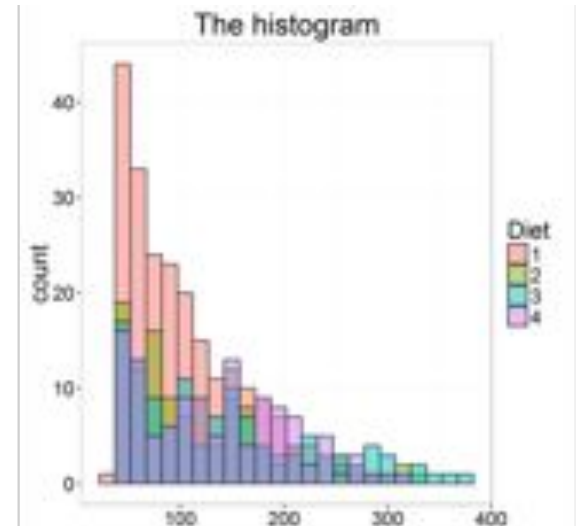
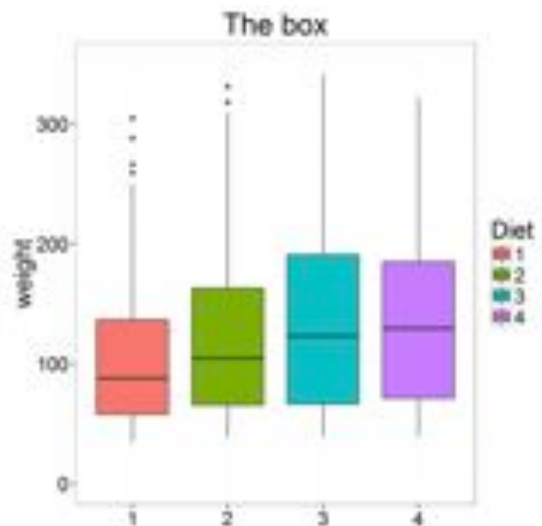
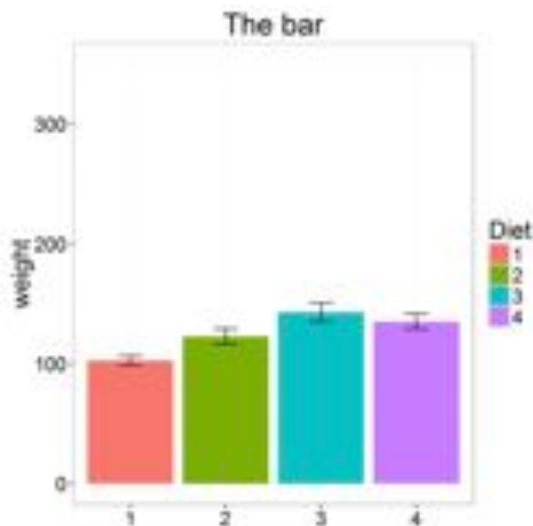
bar plots

`boxplot()`

box-and-whisker plot

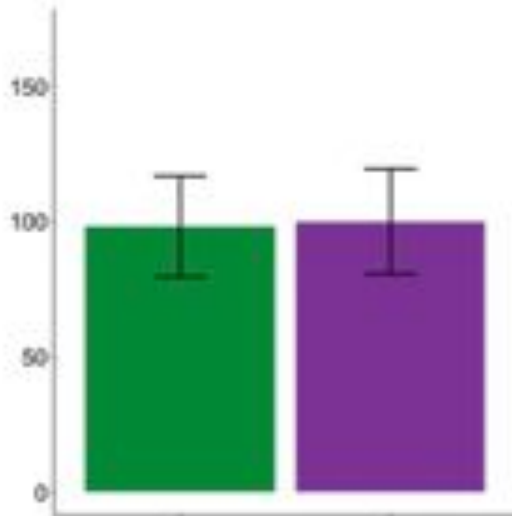
`hist()`

histograms

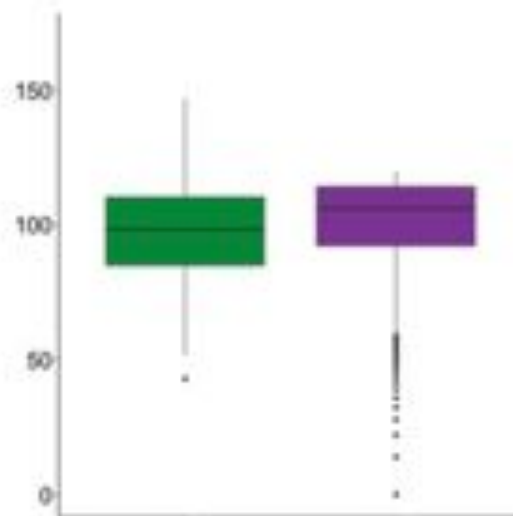


# Don't use Barplots

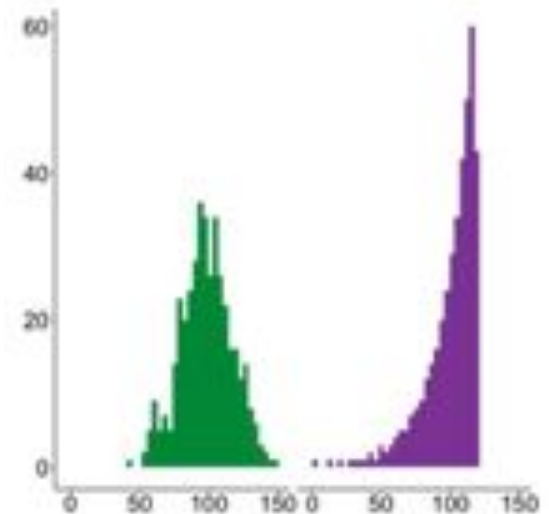
These look the same!



Wait a minute...



Oooh!



# R base graphics

`heatmap()` heatmap. Alternatives:

`qplots::heatmap.2()`

`pheatmap::pheatmap()`

`NMF::aheatmap()`

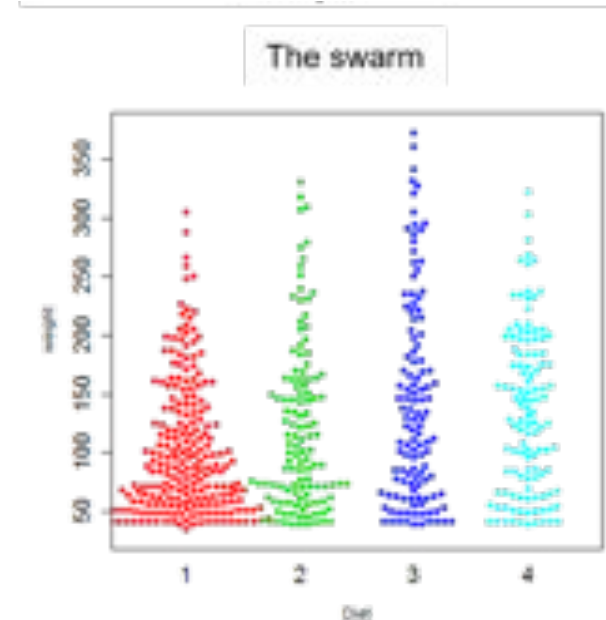
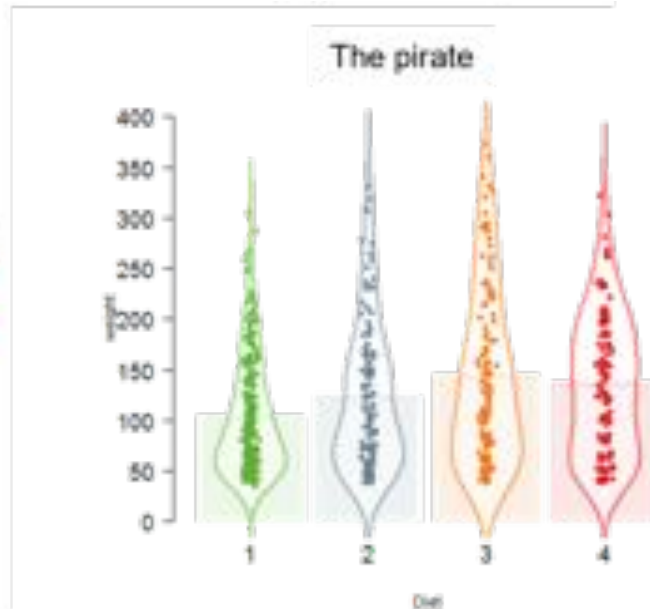
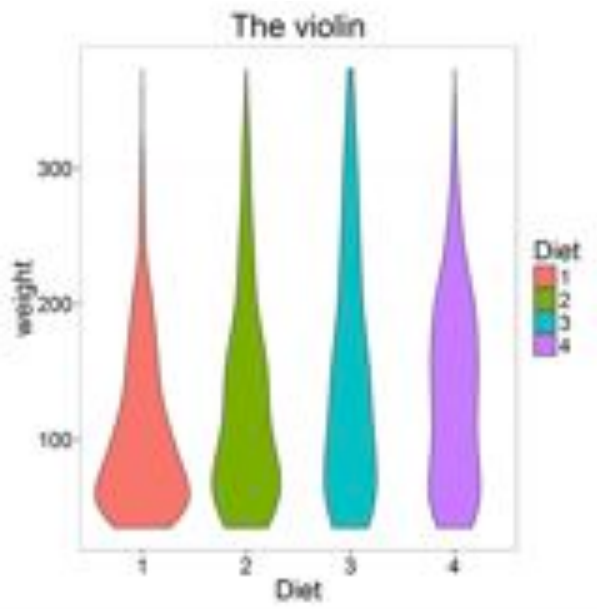
`qqnorm()`, `qqline()`, `qqplot()`  
distribution comparison plots

`pairs()` pair-wise plot of multivariate data



# Special plots

- **vioplot**: Violin plot, <https://cran.r-project.org/web/packages/vioplot/>
- **PiratePlot**: violin plot enhanced. `install_github("ndphillips/yarrr")`, <http://nathanielphillips.com/>
- **beeswarm**: The Bee Swarm Plot, an Alternative to Stripchart, <https://cran.r-project.org/web/packages/beeswarm/index.html>



# Saving plots

- Exercise: Save to PDF

```
pdf("filename.pdf", width = 7, height = 5)  
plot(1:10, 1:10)  
dev.off()
```

Other formats: `bmp()`, `jpg()`, `pdf()`,  
`png()`, or `tiff()`



# R Base Graphics Cheatsheet

## SET GRAPHICAL PARAMETERS

*the following can only be set with par()*

**par (...)**

<i>multiple plots</i>	<code>mfc</code> = <code>c(nrow, ncol)</code>	<i>plot margins (outer)</i>	<code>oma</code> = <code>c(bottom, left, top, right)</code> default: <code>c(0, 0, 0, 0)</code> lines
<i>plot margins</i>	<code>mar</code> = <code>c(bottom, left, top, right)</code> default: <code>c(5.1, 4.1, 4.1, 2.1)</code> lines	<i>query x &amp; y limits</i>	<code>par("usr")</code>

## CREATE A NEW PLOT

<b>Bar charts</b>	<b>barplot</b> ( <i>height, ...</i> )	<b>Histograms</b>	<b>hist</b> ( <i>x, ...</i> )
<i>bar labels</i>	<code>names.arg =</code>	<i>breakpts</i>	<code>breaks =</code>
<i>border</i>	<code>border =</code>		
<i>fill color</i>	<code>col =</code>		
<i>horizontal</i>	<code>horiz = TRUE</code>		
<b>Box plots</b>	<b>boxplot</b> ( <i>x, ...</i> )		
<i>horizontal</i>	<code>horizontal = TRUE</code>		
<i>box labels</i>	<code>names =</code>		
<b>Dot plots</b>	<b>dotchart</b> ( <i>x, ...</i> )		
<i>dot labels</i>	<code>labels =</code>		
		<b>Line charts</b>	<b>plot</b> ( <i>x, type = "l"</i> )
		<i>line type</i>	<code>lty =</code> "blank" 0 "solid" 1 "dashed" 2 "dotted" 3
		<i>line width</i>	<code>lwd =</code>
		<b>Scatterplots</b>	<b>plot</b> ( <i>x, ...</i> )
		<i>symbol</i>	<code>pch =</code>

## REMOVE

<i>axis labels</i>	<code>ann = FALSE</code>
<i>axis, tickmarks, and labels</i>	<code>xaxt = "n"</code> <code>yaxt = "n"</code>
<i>plot box</i>	<code>bty = "n"</code>

NOTE: Many of the parameters here can be also be set in par(). See R help for more options.

## ADJUST

<i>allow plotting out of plot region</i>	<code>xpd = TRUE</code>
<i>aspect ratio</i>	<code>asp =</code>
<i>axis limits</i>	<code>xlim =, ylim =</code>
<i>axis lines to match axis limits</i>	<code>xaxs = "i", yaxs = "i"</code> (internal axis calculation)

## ADD TEXT

### location

<i>axis labels</i>	<code>xlab =, ylab =</code>
<i>subtitle</i>	<code>sub =</code>
<i>title</i>	<code>main =</code>

### style

<i>font face</i>	<code>font = 1 (plain)</code> <code>2 (bold) 3 (italic)</code> <code>4 (bold italic)</code>
------------------	---

<i>font family</i>	<code>family = "serif"</code> "sans" "mono"
--------------------	--

### size

*(magnification factor)*

<i>all elements</i>	<code>cex =</code>
<i>axis labels</i>	<code>cex.lab =</code>
<i>subtitle</i>	<code>cex.sub =</code>
<i>tick mark labels</i>	<code>cex.axis =</code>
<i>title</i>	<code>cex.main =</code>

### position

<i>text direction</i>	<code>las = 1 (horizontal)</code>
<i>justification</i>	<code>adj = 0 .5 1</code> (left, center, right)

## ADD TO AN EXISTING PLOT

<b>Add new plot</b>	<b>[any plot function]</b>
	<code>(..., add = TRUE)</code>
	ex. <code>barplot(x, add = TRUE)</code>


<b>Axes</b>	<b>axis</b> ( <i>side, ...</i> )
<i>location</i>	<code>side = 1 2 3 4</code> (bottom, left, top, right)

<i>tick mark: labels</i>	<code>labels =</code>
<i>location</i>	<code>at =</code>
<i>remove</i>	<code>tick = FALSE</code>
<i>rotate text</i>	<code>las = 1 (horizontal)</code>

<b>Axis labels</b>	<b>mtext</b> ( <i>text, ...</i> )
<i>location</i>	<code>side = 1 2 3 4</code> (bottom, left, top, right)
<i>lines to skip</i>	<code>line = (from plot region, default = 0)</code>

<i>position</i>	<code>at = x or y-coord</code> (depending on side)
<i>justification</i>	<code>adj = 0 .5 1</code> (left, center, right)

<b>Lines</b>	<b>lines</b> ( <i>x, ...</i> )
<i>line style</i>	<code>lty =</code>
<i>line width</i>	<code>lwd =</code>
<i>color</i>	<code>col =</code>

<b>Points</b>	<b>points</b> ( <i>x, ...</i> )
<i>symbol</i>	<code>pch =</code>
	
<i>color</i>	<code>col =</code>
<i>fill color</i>	<code>bg = (pch: 21-25 only)</code>

<b>Text</b>	<b>text</b> ( <i>x, y, text, ...</i> )
<i>position (rel. to x,y)</i>	<code>pos = 1 2 3 4</code> (below, left, above, right) (default=center)

<b>Title</b>	<b>title</b> ( <i>main, ...</i> )
<i>axis labels</i>	<code>xlab =, ylab =</code>
<i>subtitle</i>	<code>sub =</code>
<i>title</i>	<code>main =</code>

# **DATA MANIPULATION**

# dplyr: data manipulation with R

- 80% of your work will be data preparation
  - getting data (from databases, spreadsheets, flat-files)
  - performing exploratory/diagnostic data analysis
  - reshaping data
  - visualizing data

# dplyr: data manipulation with R

- 80% of your work will be data preparation
  - Filtering rows (to create a subset)
  - Selecting columns of data (i.e., selecting variables)
  - Adding new variables
  - Sorting
  - Aggregating
  - Joining

# Dplyr: A grammar of data manipulation

- <https://github.com/hadley/dplyr>
- `install.packages("dplyr")`



# The pipe %>% operator

- Pipe output of one command into an input of another command - chain commands together
- Think about the “|” operator in Linux
- Read as “then”. Take the dataset, *then* do ...

```
library(dplyr)
```

```
head(diamonds)  
diamonds %>% head
```

```
summary(diamonds$price)  
diamonds$price %>% summary(object = .)
```



# dplyr::filter()

- Filter (select) rows based on the condition of a column

```
diamonds %>% head
```

```
df.diamonds_ideal <- filter(diamonds, cut == "Ideal")
```

```
df.diamonds_ideal <- diamonds %>% filter(cut ==  
"Ideal")
```



# dplyr::select()

- Select columns from the dataset by names

```
df.diamonds_ideal %>% head
```

```
select(df.diamonds_ideal, carat, cut, color,  
price, clarity)
```

```
df.diamonds_ideal <- df.diamonds_ideal %>%  
select(., carat, cut, color, price, clarity)
```





# dplyr::mutate()

- Add columns to your dataset

```
df.diamonds_ideal %>% head
```

```
mutate(df.diamonds_ideal, price_per_carat =  
price/carat)
```

```
df.diamonds_ideal <- df.diamonds_ideal %>%  
mutate(price_per_carat = price/carat)
```



# dplyr::arrange()

- Sort your data by columns

```
df.diamonds_ideal %>% head
```

```
arrange(df.diamonds_ideal, price)
```

```
df.diamonds_ideal %>% arrange(price,  
price_per_carat)
```



# dplyr::summarize()

- Summarize columns by custom summary statistics

```
summarize(df.diamonds_ideal, length = n(),  
avg_price = mean(price))
```

```
df.diamonds_ideal %>% summarize(length =  
n(), avg_price = mean(price))
```



# dplyr::group\_by()

- Summarize *subsets of* columns by custom summary statistics

```
group_by(diamonds, cut) %>%  
summarize(mean(price))
```

```
group_by(diamonds, cut, color) %>%  
summarize(mean(price))
```



# The power of pipe %>%

```
arrange(mutate(arrange(filter(tbl_df(diamonds), cut == "Ideal"), price), price_per_carat = price/carat), price_per_carat)
```

```
arrange(  
  mutate(  
    arrange(  
      filter(tbl_df(diamonds),  
             cut == "Ideal"),  
      price),  
    price_per_carat = price/carat),  
price_per_carat)
```

```
diamonds %>% filter(cut == "Ideal") %>%  
arrange(price) %>% mutate(price_per_carat = price/  
carat) %>% arrange(price_per_carat)
```



# **GGPLOT2 - THE GRAMMAR OF GRAPHICS**

# ggplot2 package

- <http://ggplot2.org/>
- `install.packages("ggplot2")`

## ggplot2

ggplot2 is a plotting system for R, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and none of the bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.

### Documentation

ggplot2 documentation is now available at [docs.ggplot2.org](https://docs.ggplot2.org).

# The basics of ggplot2 graphics

- Data mapped to graphical elements
- Add graphical layers and transformations
- Commands are chained with “+” sign

<b>Data</b>		<b>The raw data that you want to plot</b>
Aesthetics	<b>aes ()</b>	How to map your data on x, y axis, color, size, shape (aesthetics)
Geometries	<b>geom_</b>	The geometric shapes that will represent the data

**data +**

**aesthetic mappings of data to plot coordinates +**

**geometry to represent the data**



# Examples of ggplot2 graphics

```
diamonds %>% filter(cut == "Good",  
color == "E") %>%  
  ggplot(aes(x = price, y = carat)) +  
  geom_point() # aes(size = price) +  
  geom_smooth() # method = lm  
  
  geom_line()  
  geom_boxplot()  
  geom_bar(stat="identity")  
  geom_histogram()
```



# Fine tuning ggplot2 graphics

Facets	<b>facet_</b>	Split one plot into multiple plots based on a grouping variable
Scales	<b>scale_</b>	Maps between the data ranges and the dimensions of the plot
Visual Themes	<b>theme</b>	The overall visual defaults of a plot: background, grids, axes, default typeface, sizes, colors, etc.
Statistical transformations	<b>stat_</b>	Statistical summaries of the data that can be plotted, such as quantiles, fitted curves (loess, linear models, etc.), sums etc.
Coordinate systems	<b>coord_</b>	Expressing coordinates in a system other than Cartesian

# Putting it all together

```
diamonds %>% # Start with the 'diamonds' dataset
  filter(cut == "Ideal") %>% # Then, filter rows where cut == Ideal
  ggplot(aes(price)) + # Then, plot using ggplot
  geom_histogram() + # and plot histograms
  facet_wrap(~ color) + # in a 'small multiple' plot, broken
out by 'color'
  ggtitle("Diamond price distribution per color") +
  labs(x="Price", y="Count") +
  theme(panel.background = element_rect(fill="lightblue")) +
  theme(plot.title = element_text(family="Trebuchet MS", size=28,
face="bold", hjust=0, color="#777777")) +
  theme(axis.title.y = element_text(angle=0)) +
  theme(panel.grid.minor = element_blank())
```



# Other resources

- **Plotly** for R, <https://plot.ly/r/>
- **GoogleVis** for R,  
[https://cran.r-project.org/web/packages/googleVis/vignettes/googleVis\\_examples.html](https://cran.r-project.org/web/packages/googleVis/vignettes/googleVis_examples.html)
- **ggbio** – grammar of graphics for genomic data,  
<http://www.tengfei.name/ggbio/>