Stats with geoms

INTERMEDIATE DATA VISUALIZATION WITH GGPLOT2

Rick Scavetta
Founder, Scavetta Academy

datacamp

ggplot2, course 2

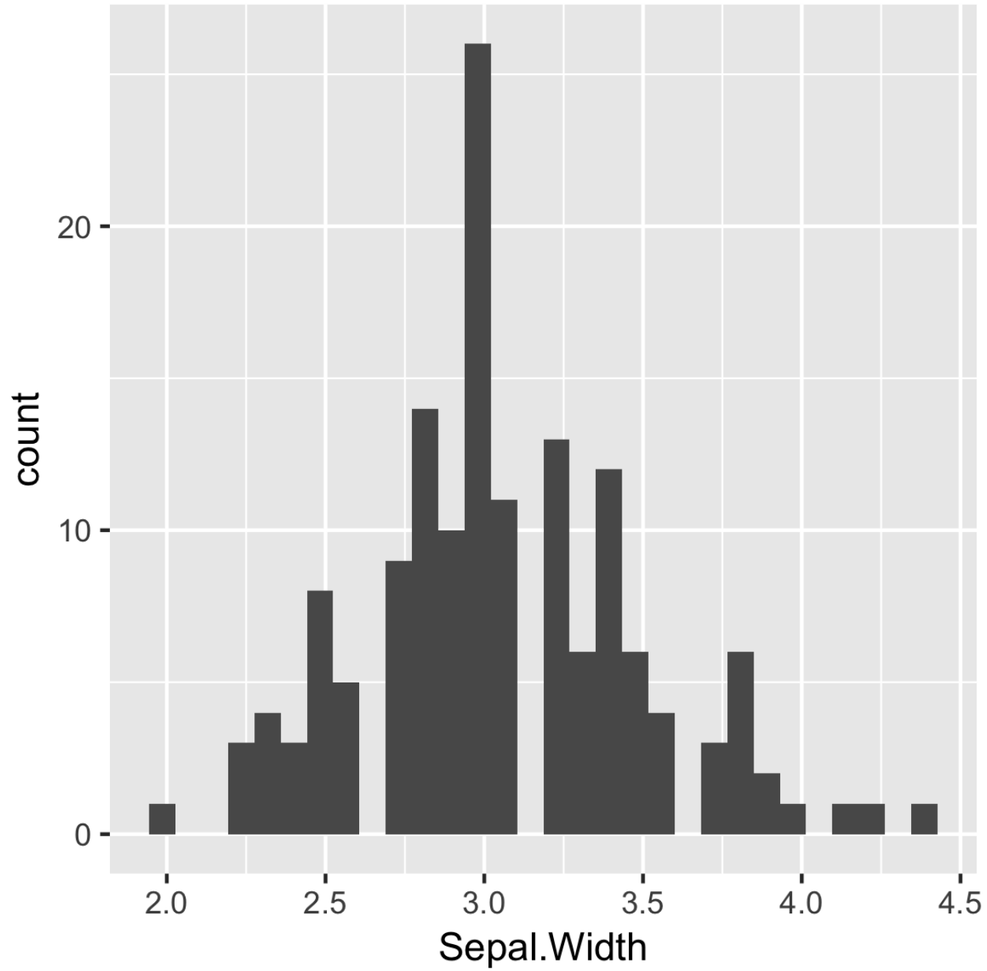
- Statistics
- Coordinates
- Facets
- Data Visualization Best Practices

Statistics layer

- Two categories of functions
  - Called from within a geom
  - Called independently
- `stats_`

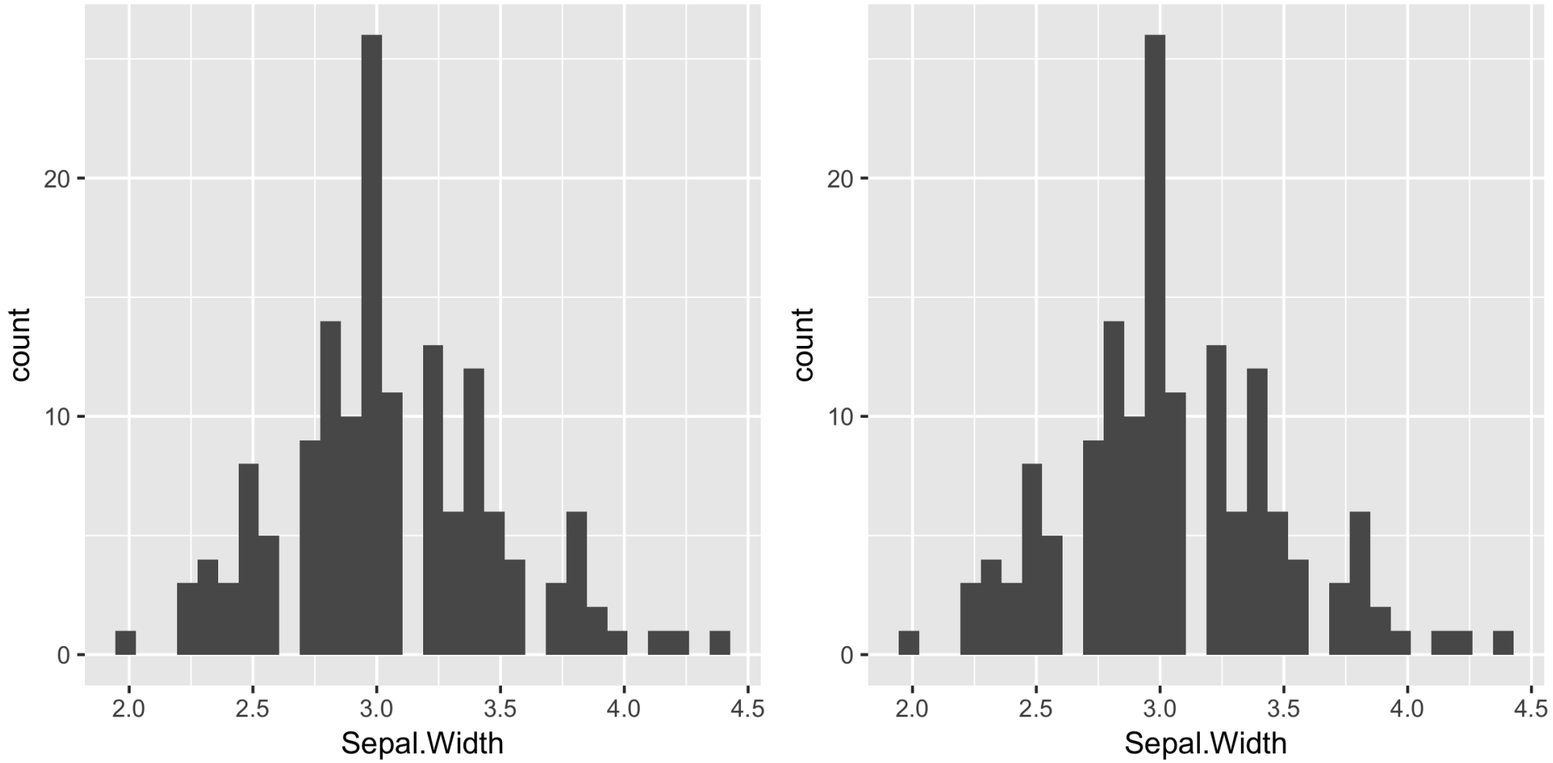
geom_ <-> stat_

```
p <- ggplot(iris, aes(x = Sepal.Width))
p + geom_histogram()
```



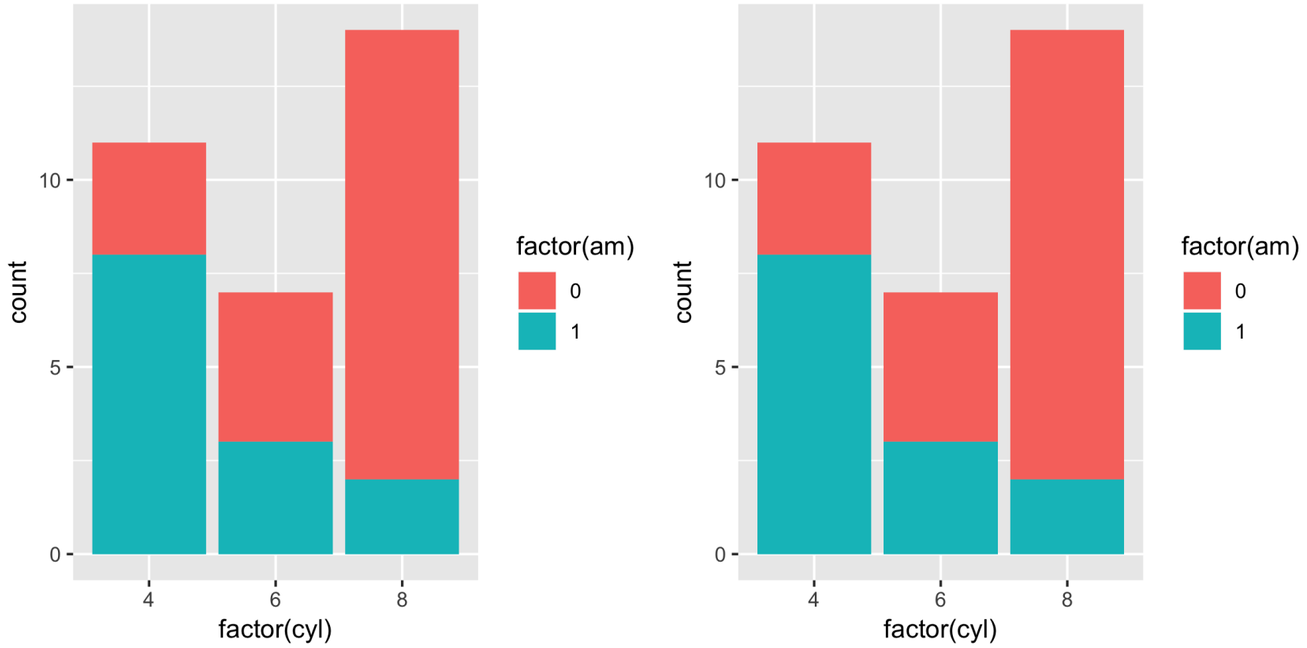
geom_ <-> stat_

```
p <- ggplot(iris, aes(x = Sepal.Width))
p + geom_histogram()
p + geom_bar()
```



geom_ <-> stat_

```
p <- ggplot(mtcars, aes(x = factor(cyl),  fill = factor(am)))
p + geom_bar()
p + stat_count()
```



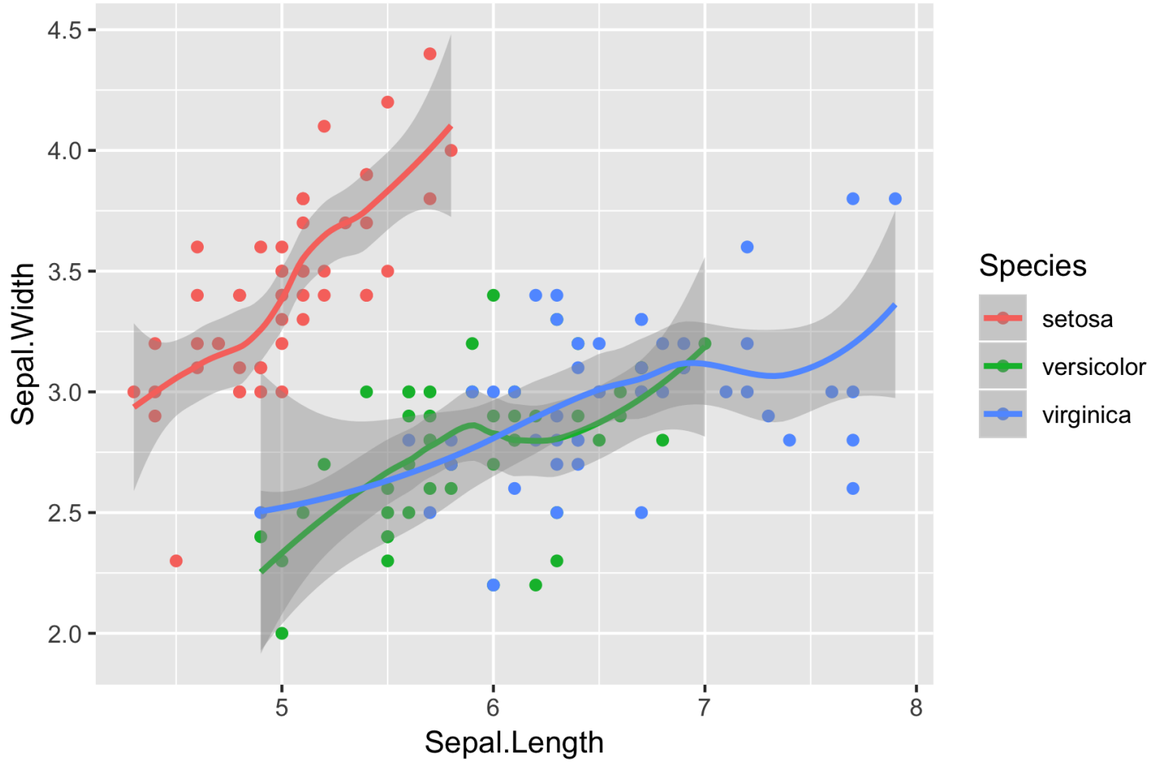
The geom_/stat_ connection

stat_	geom_
`stat_bin()`	`geom_histogram()`, `geom_freqpoly()`
`stat_count()`	`geom_bar()`

stat_smooth()

```
ggplot(iris, aes(x = Sepal.Length,
                 y = Sepal.Width,
                 color = Species)) +
  geom_point() +
  geom_smooth()
```

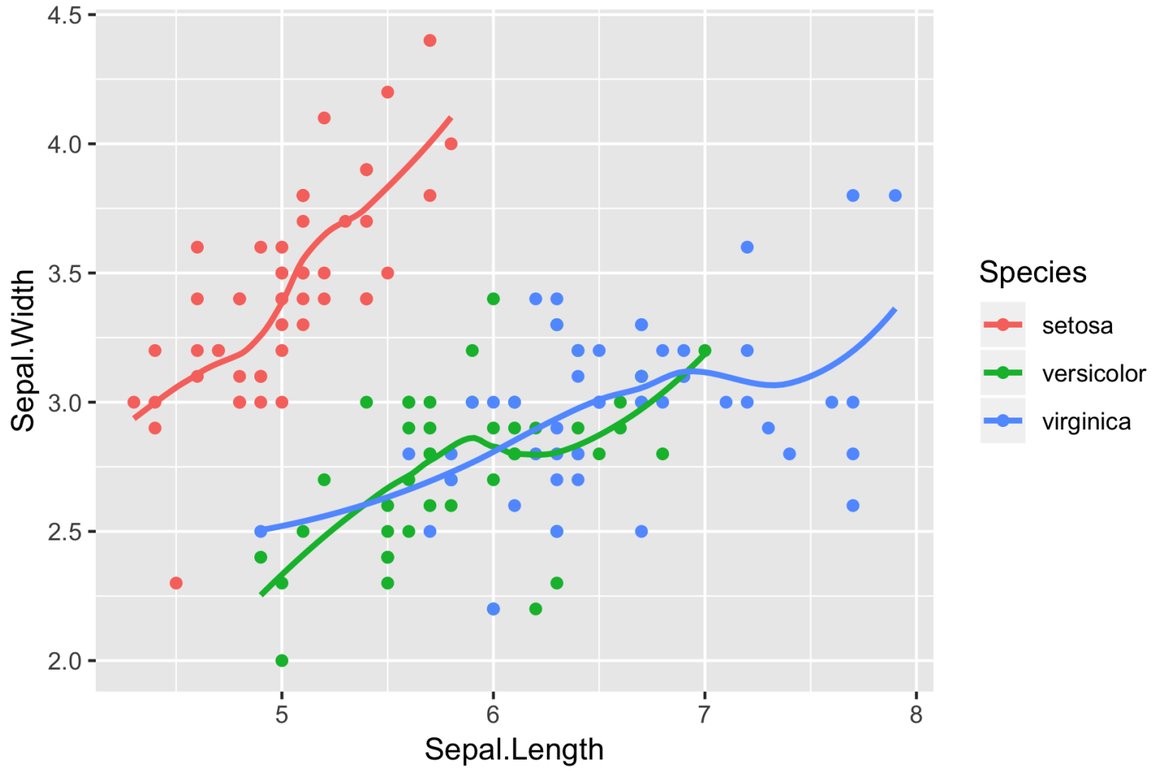
```
geom_smooth() using method = 'loess' and
formula 'y ~ x'
```



stat_smooth(se = FALSE)

```
ggplot(iris, aes(x = Sepal.Length,
                 y = Sepal.Width,
                 color = Species)) +
  geom_point() +
  geom_smooth(se = FALSE)
```

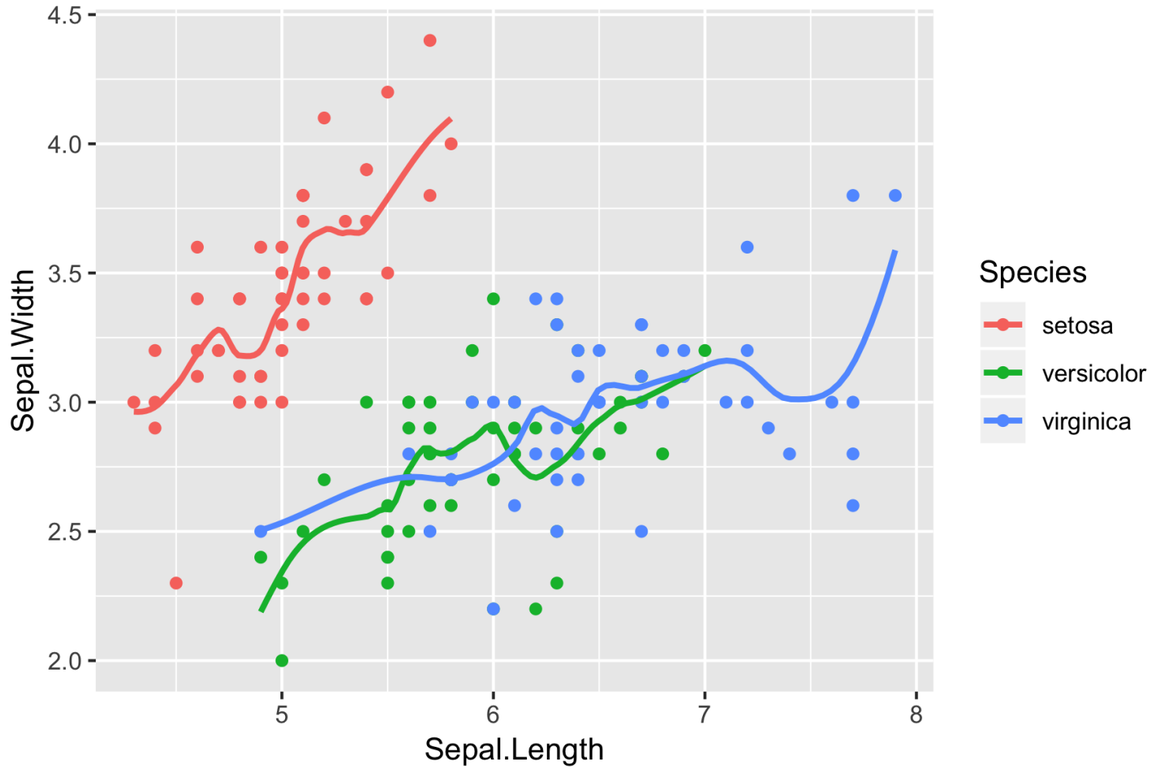
```
geom_smooth() using method = 'loess' and
formula 'y ~ x'
```



geom_smooth(span = 0.4)

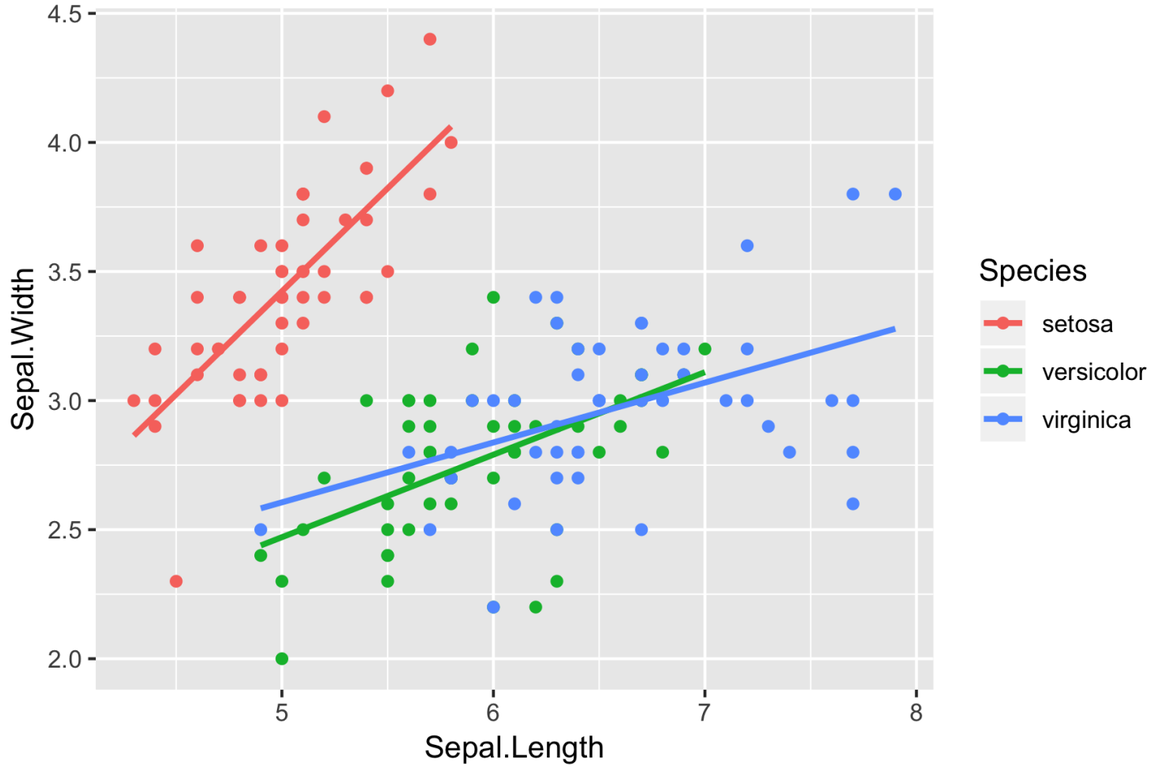
```
ggplot(iris, aes(x = Sepal.Length,
                 y = Sepal.Width,
                 color = Species)) +
  geom_point() +
  geom_smooth(se = FALSE, span = 0.4)
```

```
geom_smooth() using method = 'loess' and
formula 'y ~ x'
```



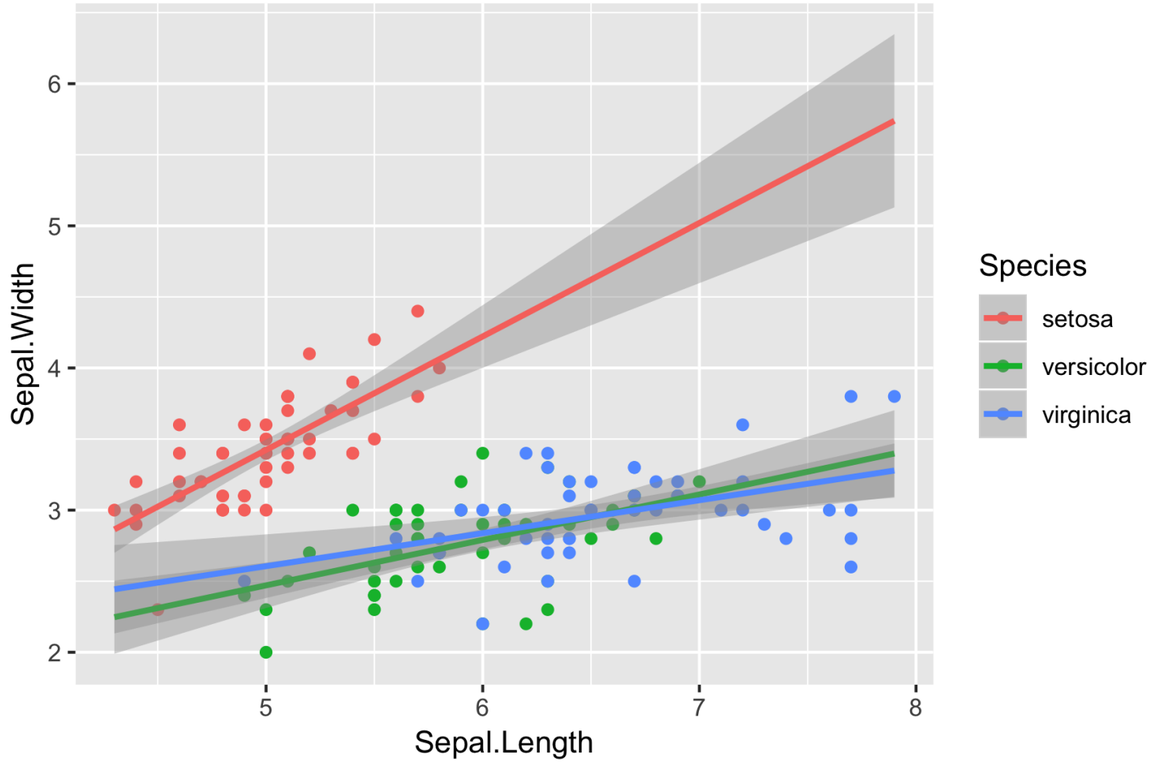
geom_smooth(method = "lm")

```
ggplot(iris, aes(x = Sepal.Length,
                 y = Sepal.Width,
                 color = Species)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```



geom_smooth(fullrange = TRUE)

```
ggplot(iris, aes(x = Sepal.Length,
                 y = Sepal.Width,
                 color = Species)) +
  geom_point() +
  geom_smooth(method = "lm",
              fullrange = TRUE)
```



The geom_/stat_ connection

stat_	geom_
`stat_bin()`	`geom_histogram()`, `geom_freqpoly()`
`stat_count()`	`geom_bar()`
`stat_smooth()`	`geom_smooth()`

Other stat_ functions

stat_	geom_
`stat_boxplot()`	`geom_boxplot()`

Other stat_ functions

stat_	geom_
`stat_boxplot()`	`geom_boxplot()`
`stat_bindot()`	`geom_dotplot()`
`stat_bin2d()`	`geom_bin2d()`
`stat_binhex()`	`geom_hex()`

Other stat_ functions

stat_	geom_
`stat_boxplot()`	`geom_boxplot()`
`stat_bindot()`	`geom_dotplot()`
`stat_bin2d()`	`geom_bin2d()`
`stat_binhex()`	`geom_hex()`
`stat_contour()`	`geom_contour()`
`stat_quantile()`	`geom_quantile()`
`stat_sum()`	`geom_count()`

Let's practice!

INTERMEDIATE DATA VISUALIZATION WITH GGPLOT2

Stats: sum and quantile

INTERMEDIATE DATA VISUALIZATION WITH GGPLOT2

Rick Scavetta
Founder, Scavetta Academy

Recall from course 1

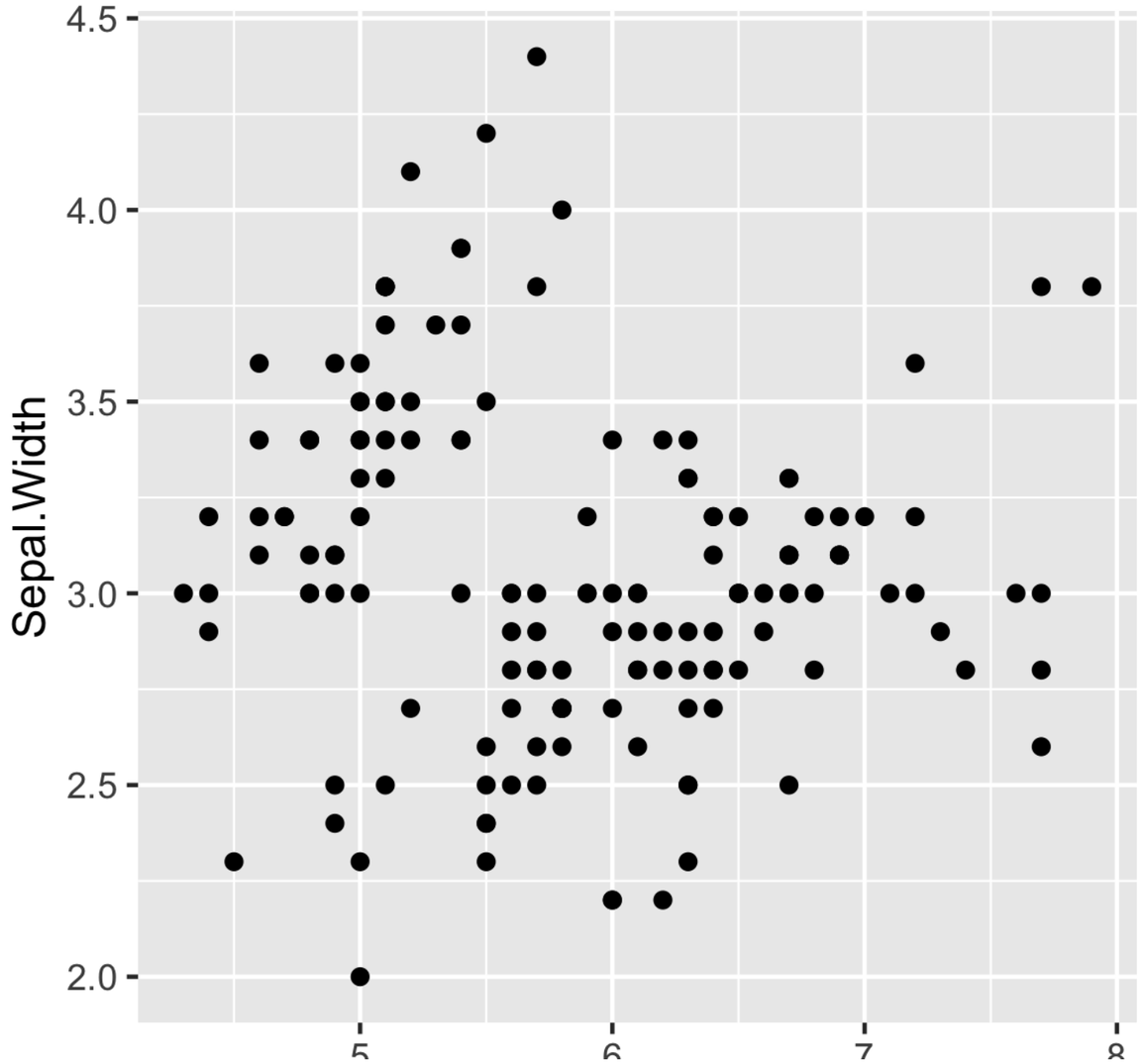
	Cause of Over-plotting	Solutions
1.	Large datasets	Alpha-blending, hollow circles, point size
2.	Aligned values on a single axis	As above, plus change position
3.	Low-precision data	Position: jitter
4.	Integer data	Position: jitter

Plot counts to overcome over-plotting

	Cause of Over-plotting	Solutions	Here...
1.	Large datasets	Alpha-blending, hollow circles, point size	
2.	Aligned values on a single axis	As above, plus change position	
3.	Low-precision data	Position: jitter	`geom_count()`
4.	Integer data	Position: jitter	`geom_count()`

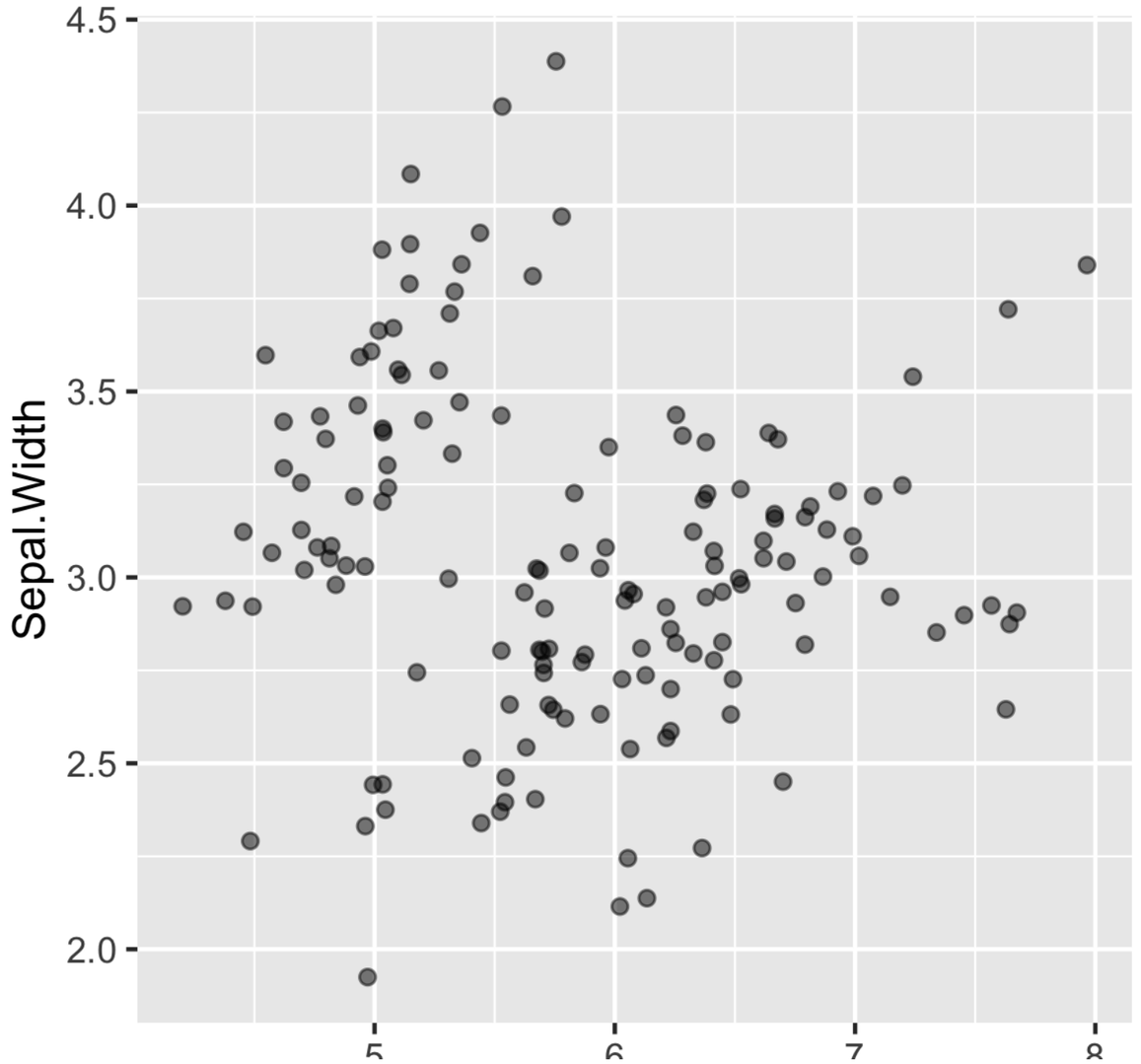
Low precision (& integer) data

```
p <- ggplot(iris, aes(Sepal.Length,
                      Sepal.Width))

p + geom_point()
```



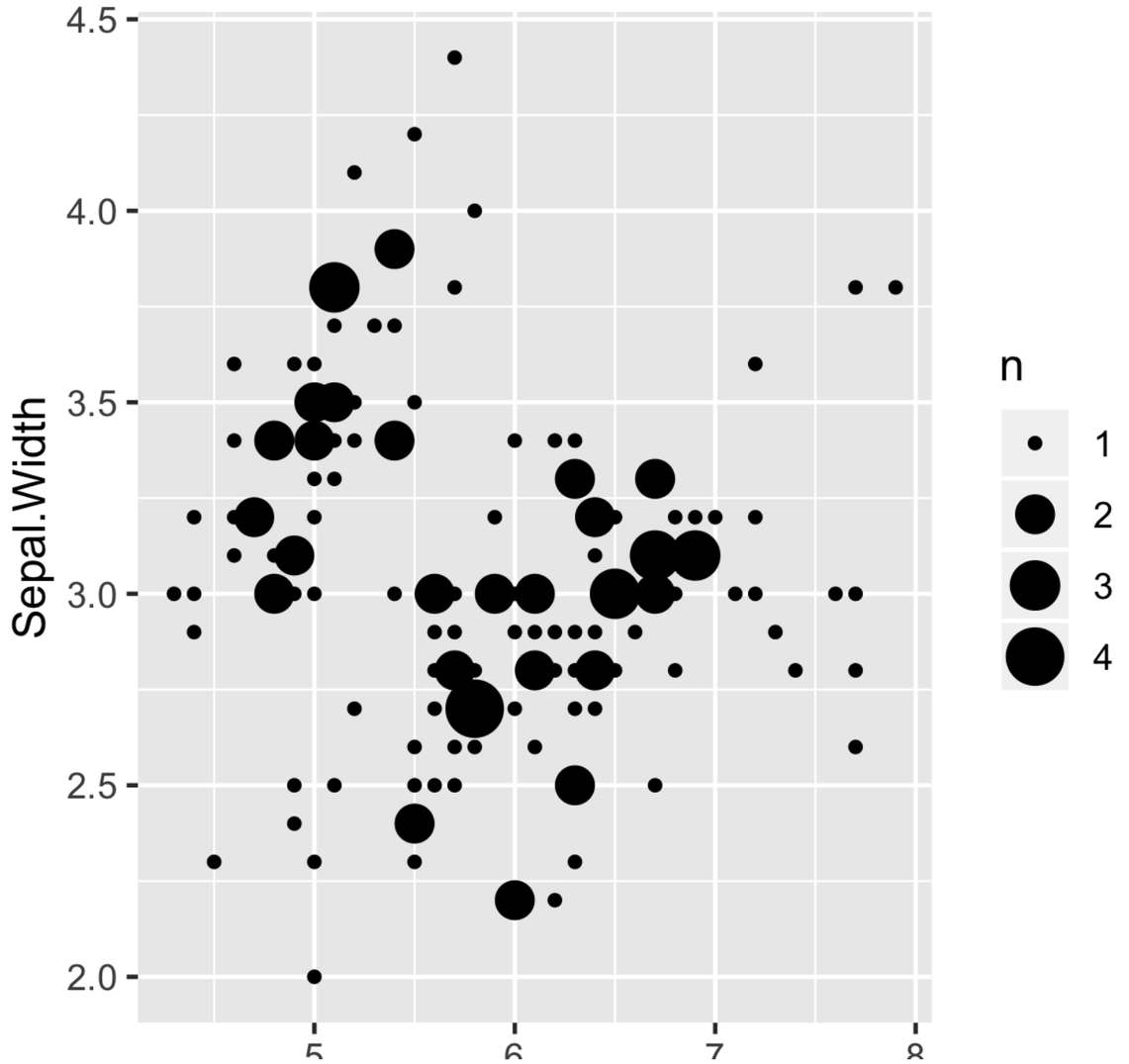
Jittering may give a wrong impressions

```
p + geom_jitter(alpha = 0.5,
                width = 0.1,
                height = 0.1)
```



geom_count()

```
p +
  geom_count()
```

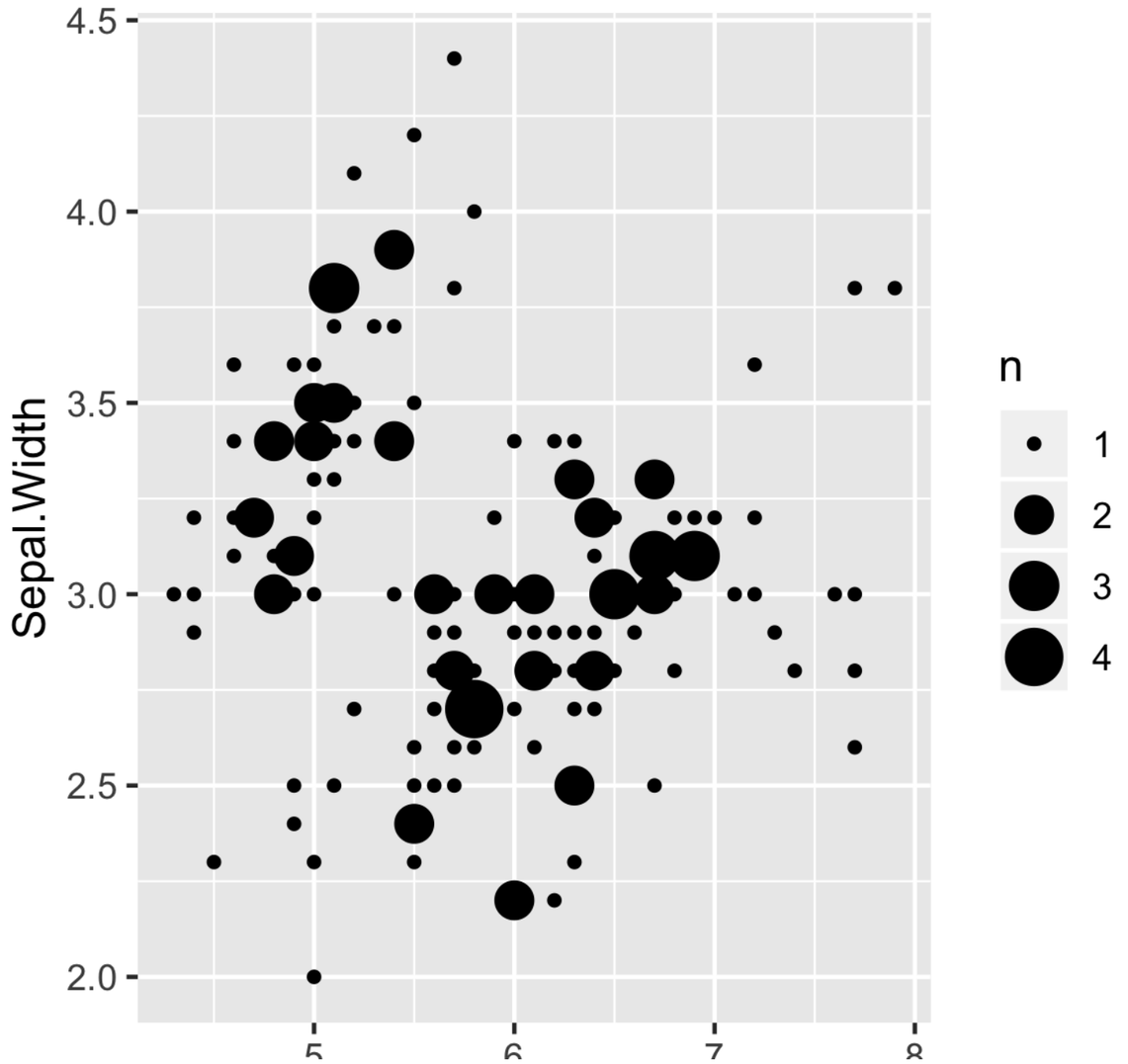


The geom/stat connection

geom_	stat_
`geom_count()`	`stat_sum()`

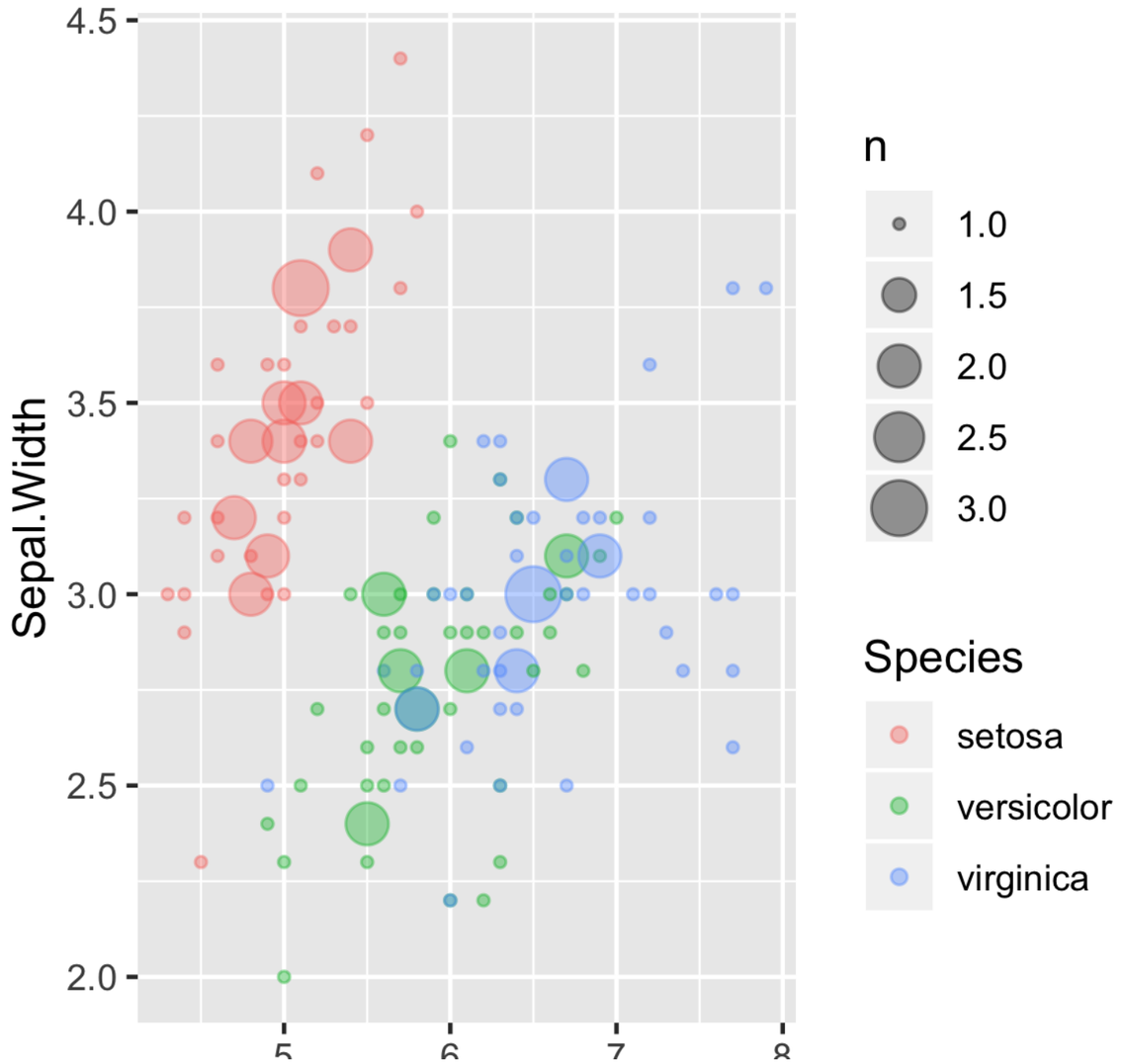
stat_sum()

```
p +
  stat_sum()
```



Over-plotting can still be a problem!

```
ggplot(iris, aes(Sepal.Length,
                 Sepal.Width,
                 color = Species)) +
  geom_count(alpha = 0.4)
```

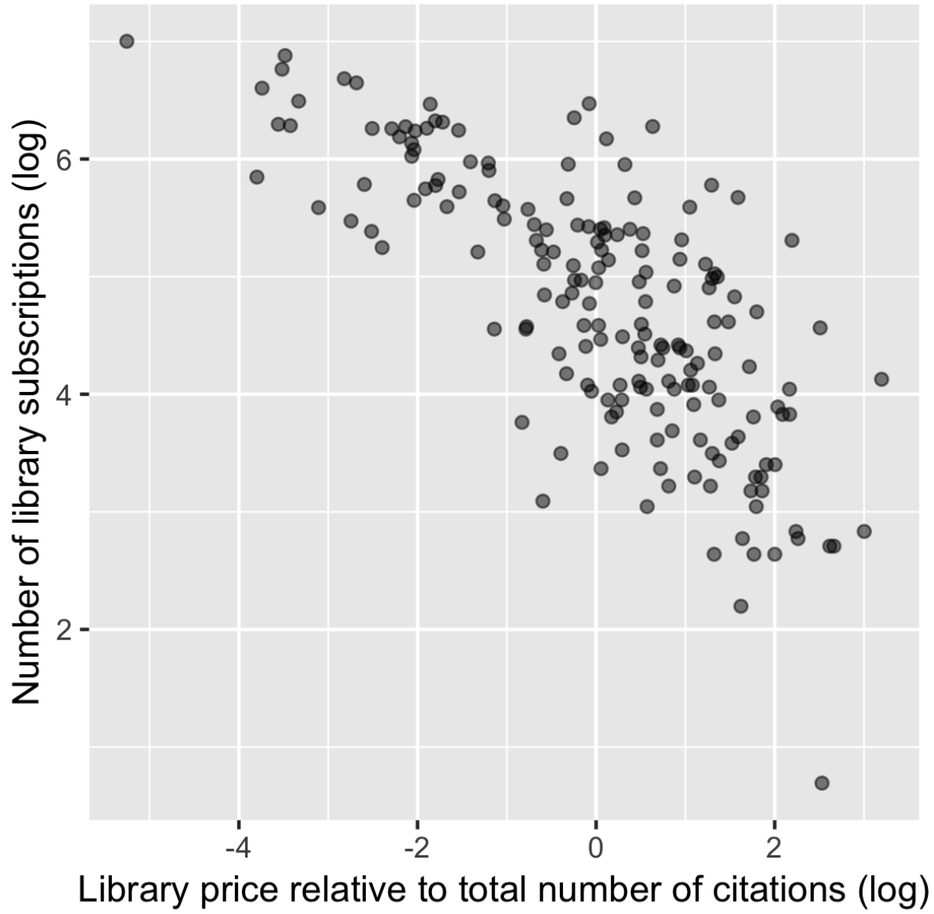


geom_quantile()

```
ggplot(iris, aes(Sepal.Length,
                 Sepal.Width,
                 color = Species)) +
  geom_count(alpha = 0.4)
```

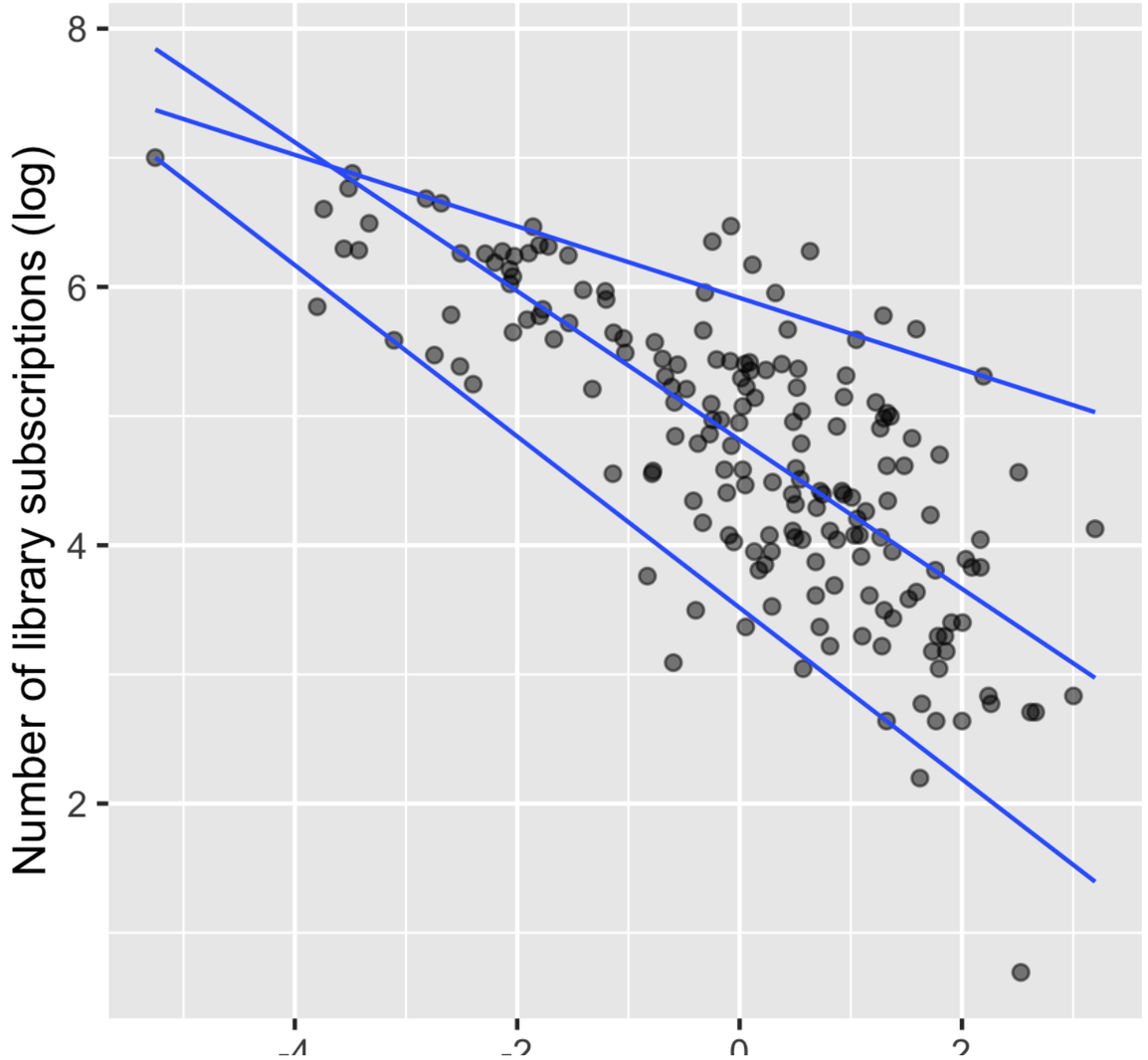
Dealing with heteroscedasticity

```
library(AER)
data(Journals)

p <- ggplot(Journals,
            aes(log(price/citations),
                log(subs))) +
  geom_point(alpha = 0.5) +
  labs(...)

p
```



Using geom_quantiles

```
p +
  geom_quantile(quantiles =
                  c(0.05, 0.50, 0.95))
```



The geom/stat connection

geom_	stat_
`geom_count()`	`stat_sum()`
`geom_quantile()`	`stat_quantile()`

Ready for exercises!

INTERMEDIATE DATA VISUALIZATION WITH GGPLOT2

Stats outside geoms

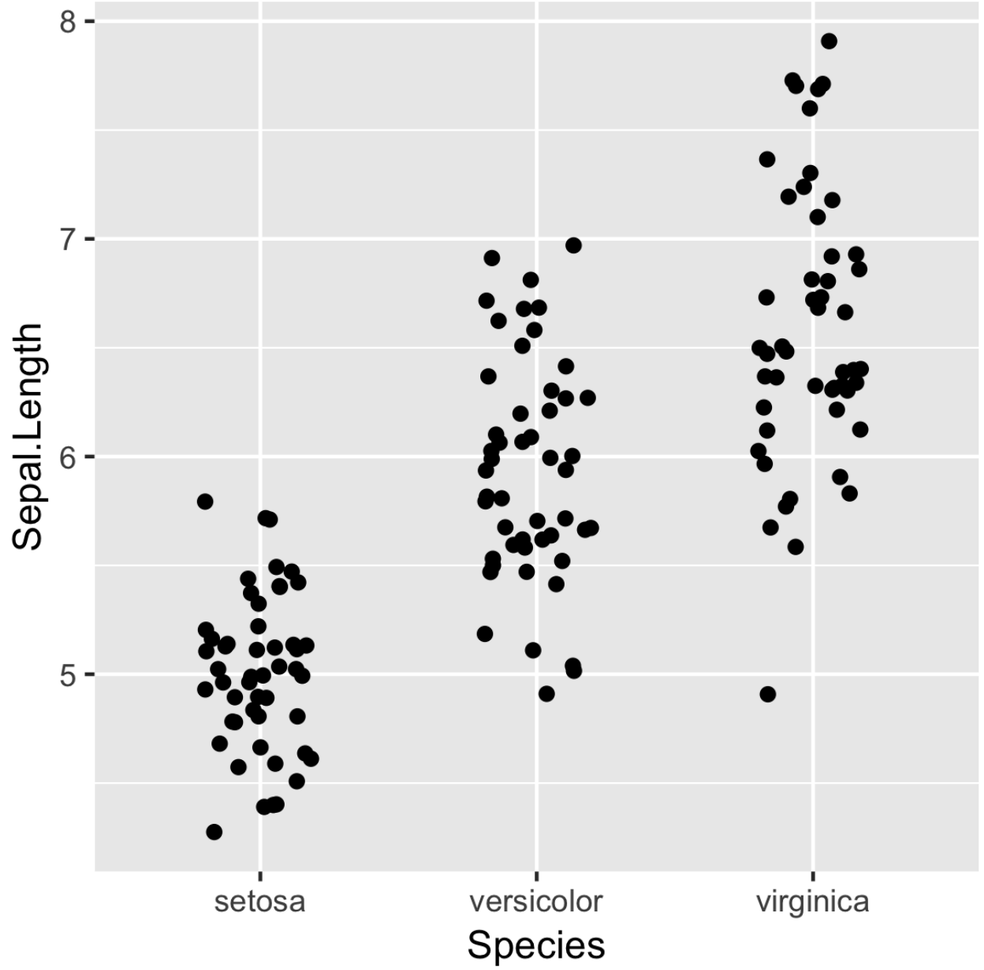
INTERMEDIATE DATA VISUALIZATION WITH GGPLOT2

Rick Scavetta
Founder, Scavetta Academy

datacamp

Basic plot

```
ggplot(iris, aes(x = Species,
                 y = Sepal.Length)) +
  geom_jitter(width = 0.2)
```



Calculating statistics

```
set.seed(123)
xx <- rnorm(100)
mean(xx)
```

```
[1] 0.09040591
```

```
mean(xx) + (sd(xx) * c(-1, 1))
```

```
[1] -0.822410  1.003222
```

Calculating statistics

```
set.seed(123)
xx <- rnorm(100)

# Hmisc
library(Hmisc)
smean.sdl(xx, mult = 1)
```

```
      Mean       Lower       Upper
 0.09040591 -0.82240997  1.00322179
```

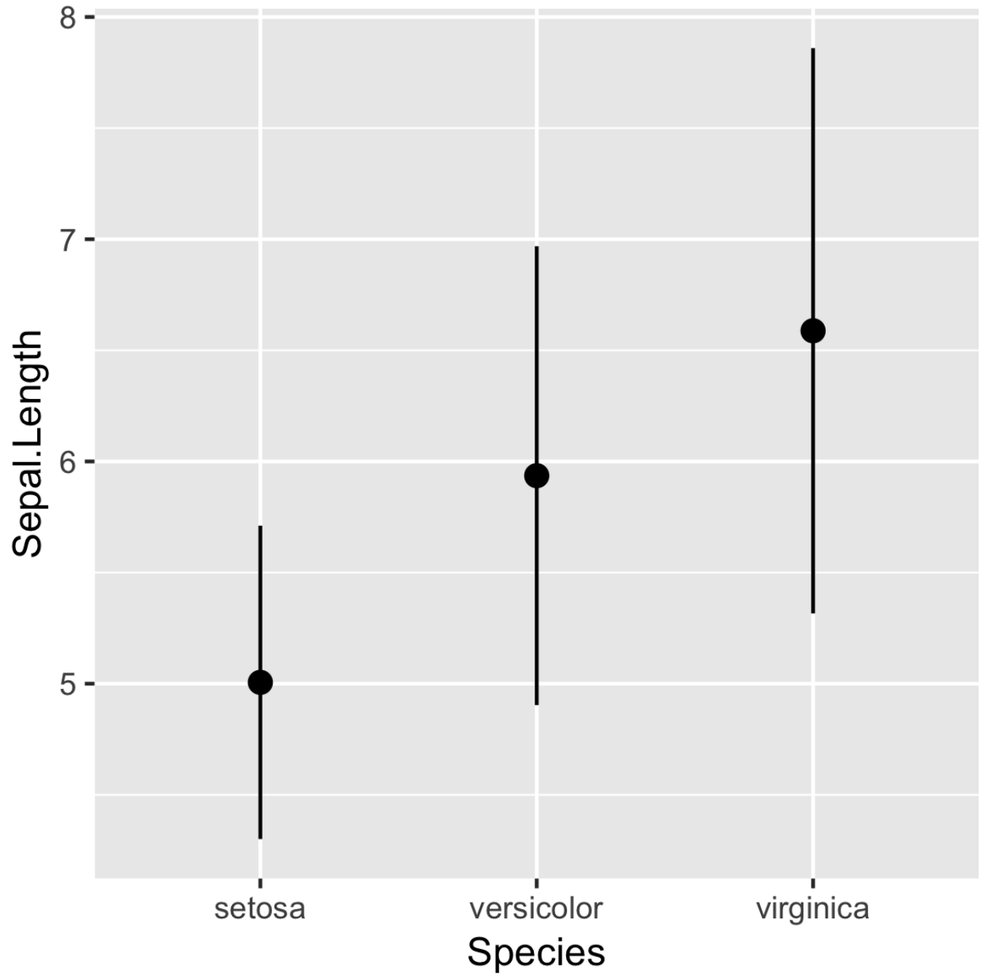
```
# ggplot2
mean_sdl(xx, mult = 1)
```

```
          y     ymin     ymax
1 0.09040591 -0.82241 1.003222
```

stat_summary()

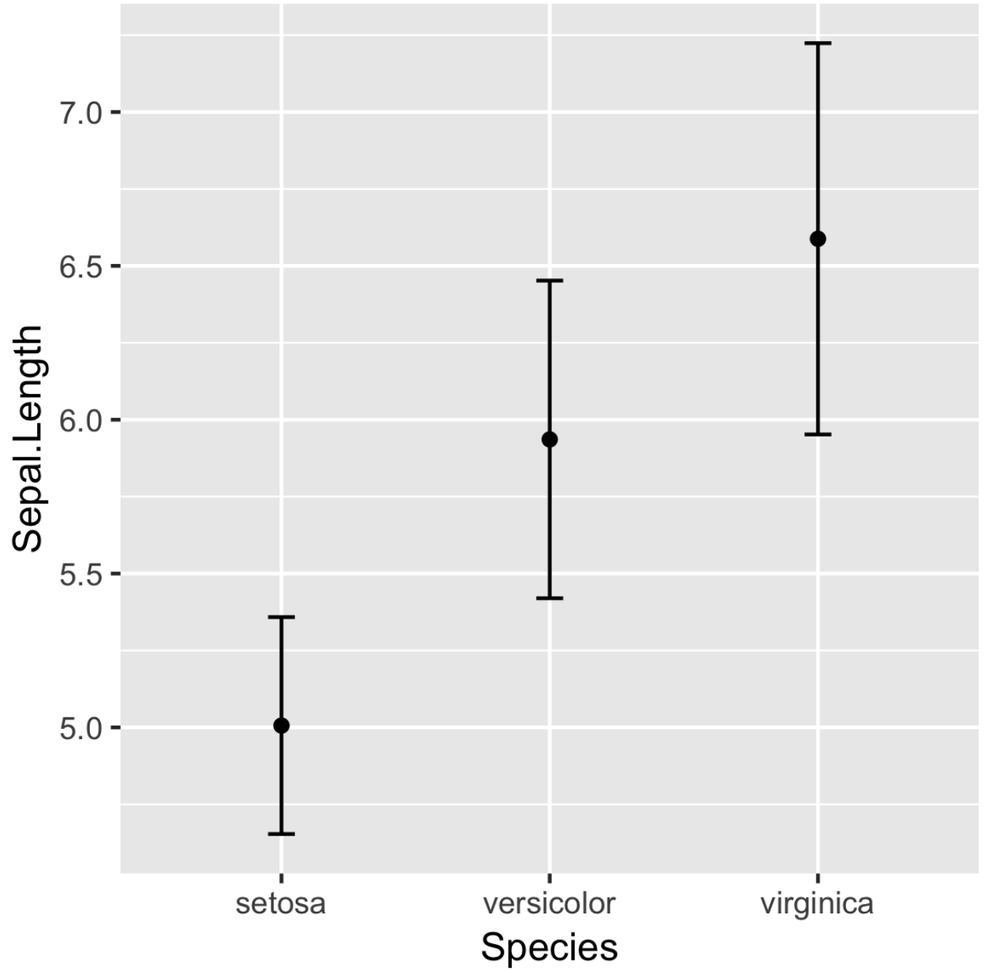
```
ggplot(iris, aes(x = Species,
                 y = Sepal.Length)) +
 stat_summary(fun.data = mean_sdl,
              fun.args = list(mult = 1))
```

- Uses `geom_pointrange()` by default

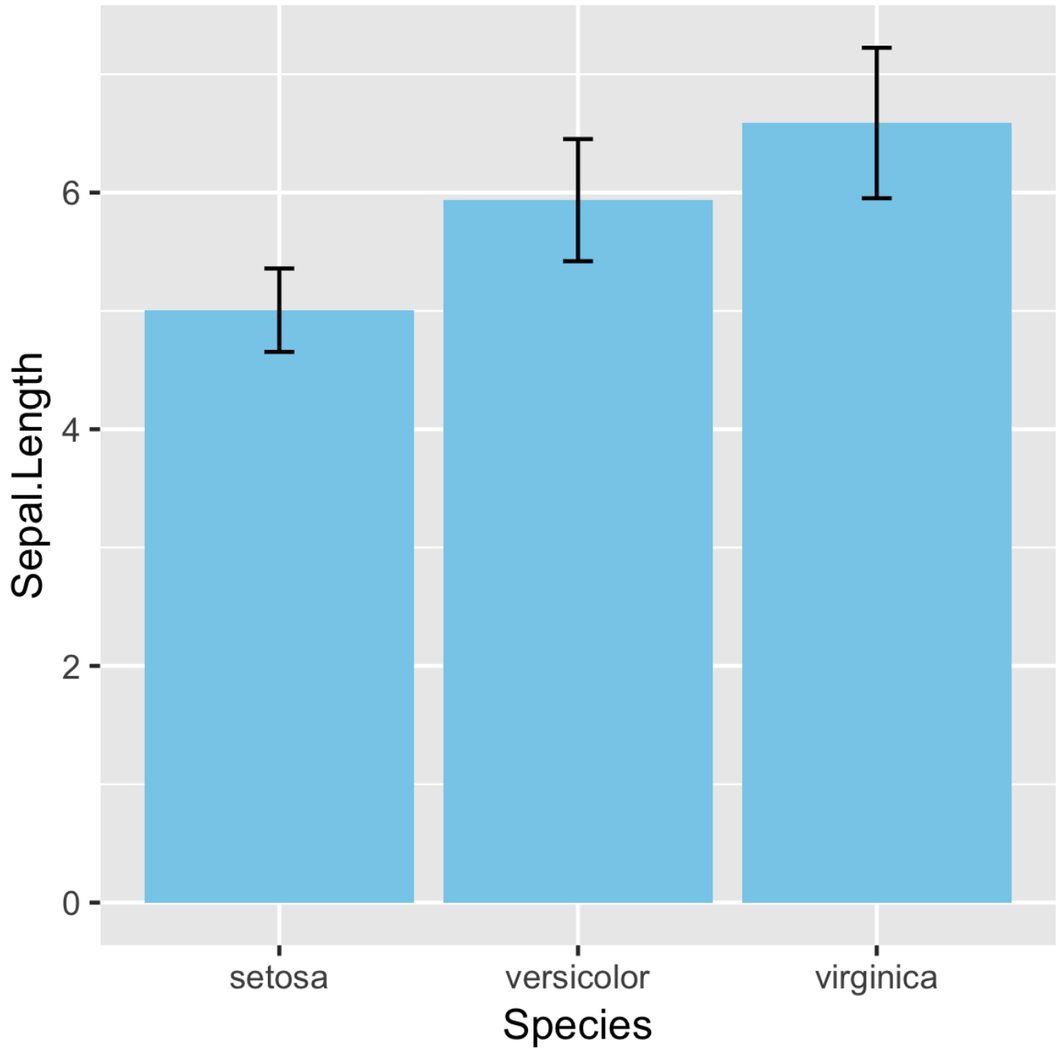


stat_summary()

```
ggplot(iris, aes(x = Species,
                 y = Sepal.Length)) +
  stat_summary(fun = mean,
               geom = "point") +
  stat_summary(fun.data = mean_sdl,
               fun.args = list(mult = 1),
               geom = "errorbar",
               width = 0.1)
```



Not recommended!



95% confidence interval

```
ERR <- qt(0.975, length(xx) - 1) * (sd(xx) / sqrt(length(xx)))
mean(xx)
```

```
0.09040591
```

```
mean(xx) + (ERR * c(-1, 1)) # 95% CI
```

```
-0.09071657  0.27152838
```

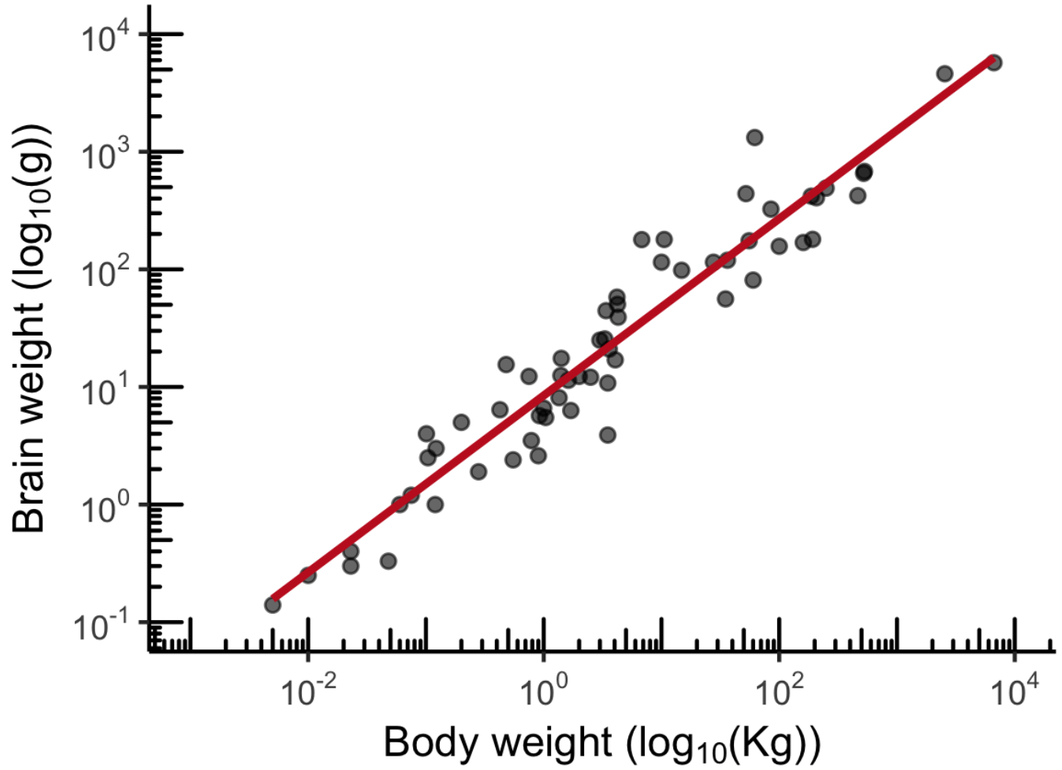
```
mean_cl_normal(xx)
```

```
           y        ymin      ymax
0.09040591 -0.09071657 0.2715284
```

Other stat_ functions

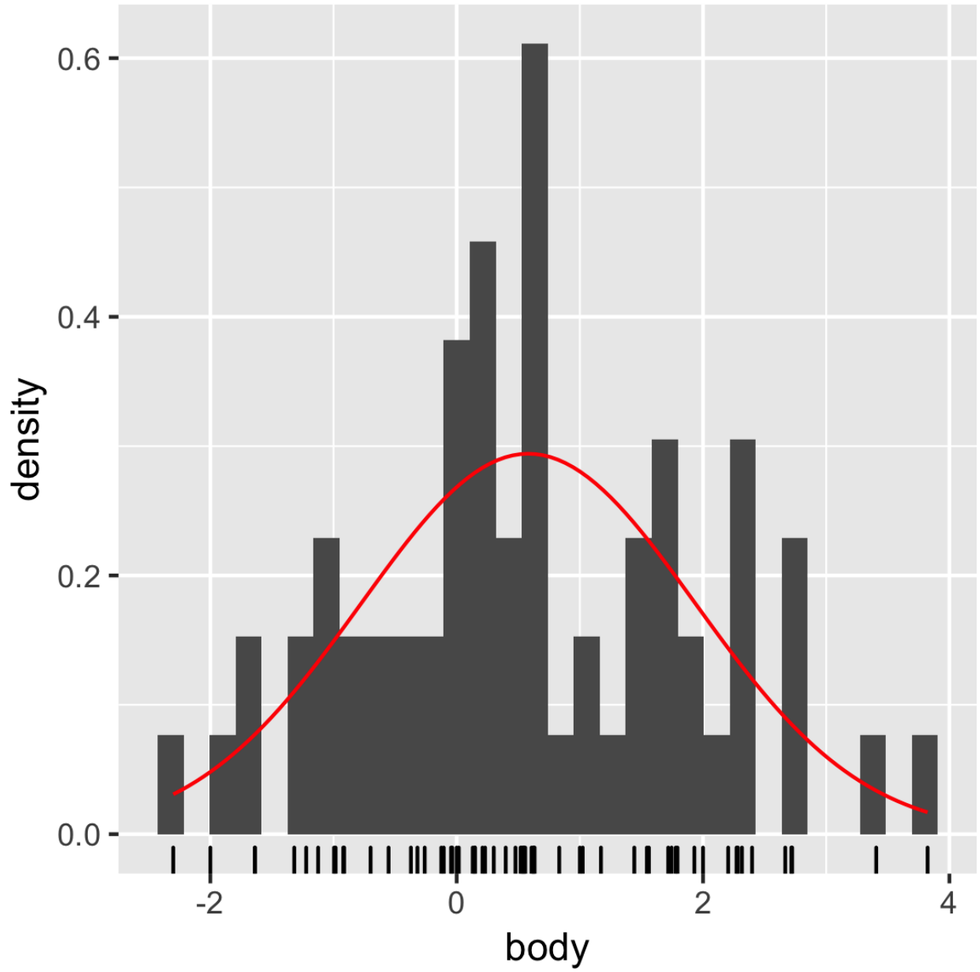
stat_	Description
`stat_summary()`	summarize y values at distinct x values.
`stat_function()`	compute y values from a function of x values.
`stat_qq()`	perform calculations for a quantile-quantile plot.

MASS::mammals



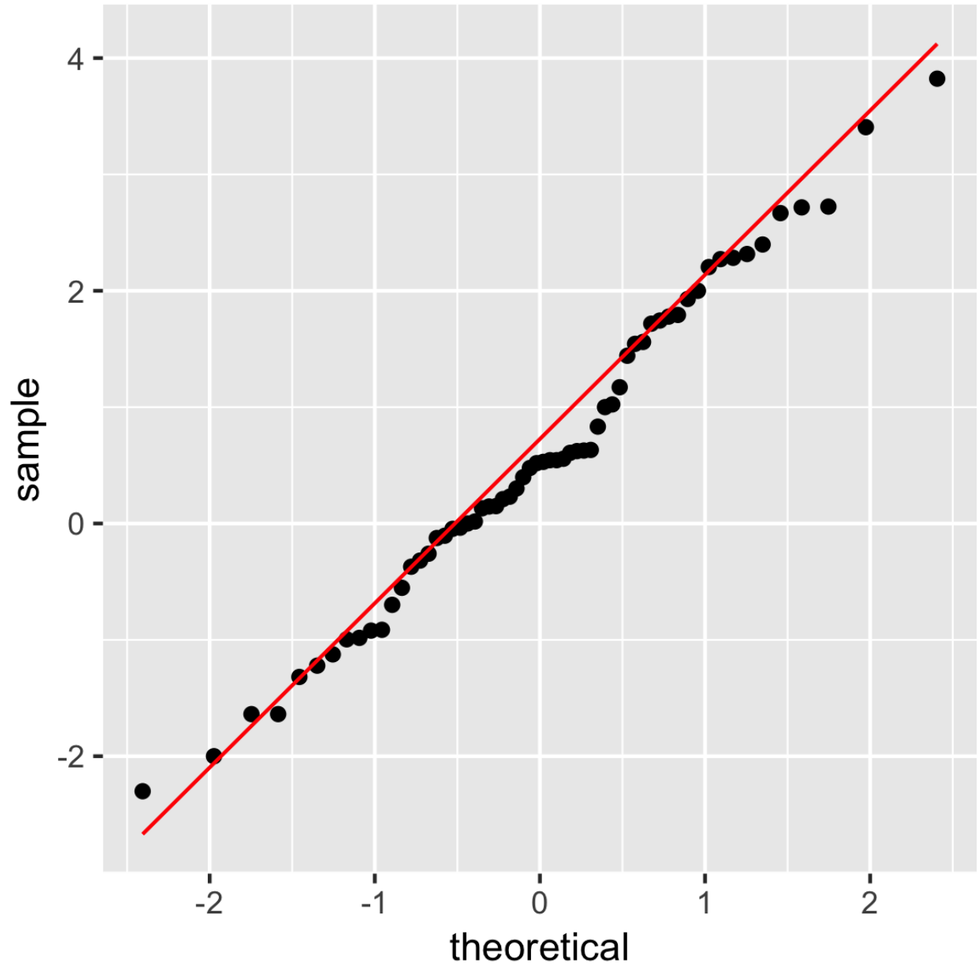
Normal distribution

```
mam.new <- data.frame(body = log10(mammals$body))

ggplot(mam.new, aes(x = body)) +
  geom_histogram(aes( y = ..density..)) +
  geom_rug() +
  stat_function(fun = dnorm, color = "red",
                args = list(mean = mean(mam.new$body),
                            sd = sd(mam.new$body)))
```



QQ plot

```
ggplot(mam.new, aes(sample = body)) +
  stat_qq() +
  geom_qq_line(col = "red")
```



Your turn!

INTERMEDIATE DATA VISUALIZATION WITH GGPLOT2