

Learning `ggplot2` in R

Abu Bakar Siddique, SLUBI

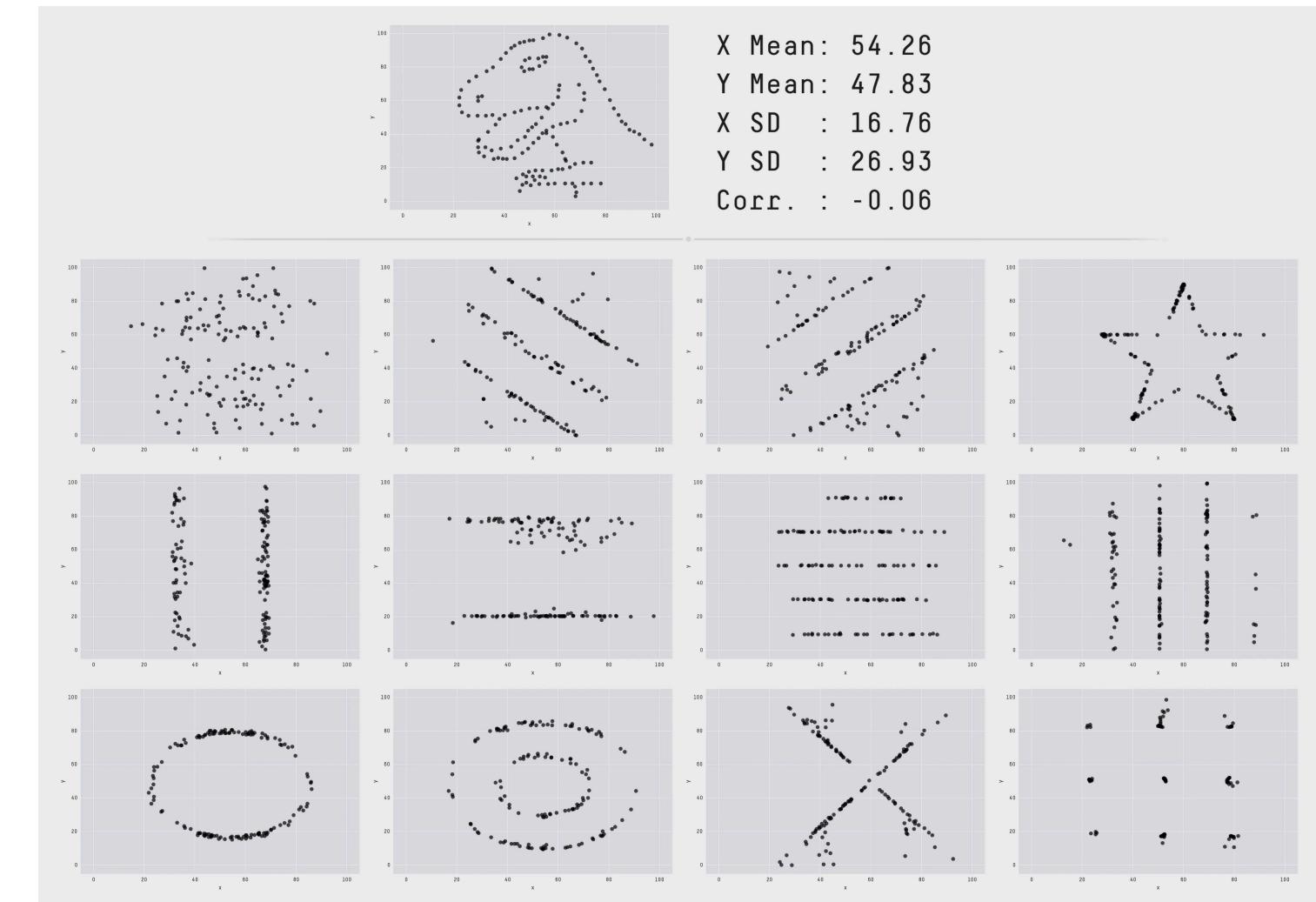
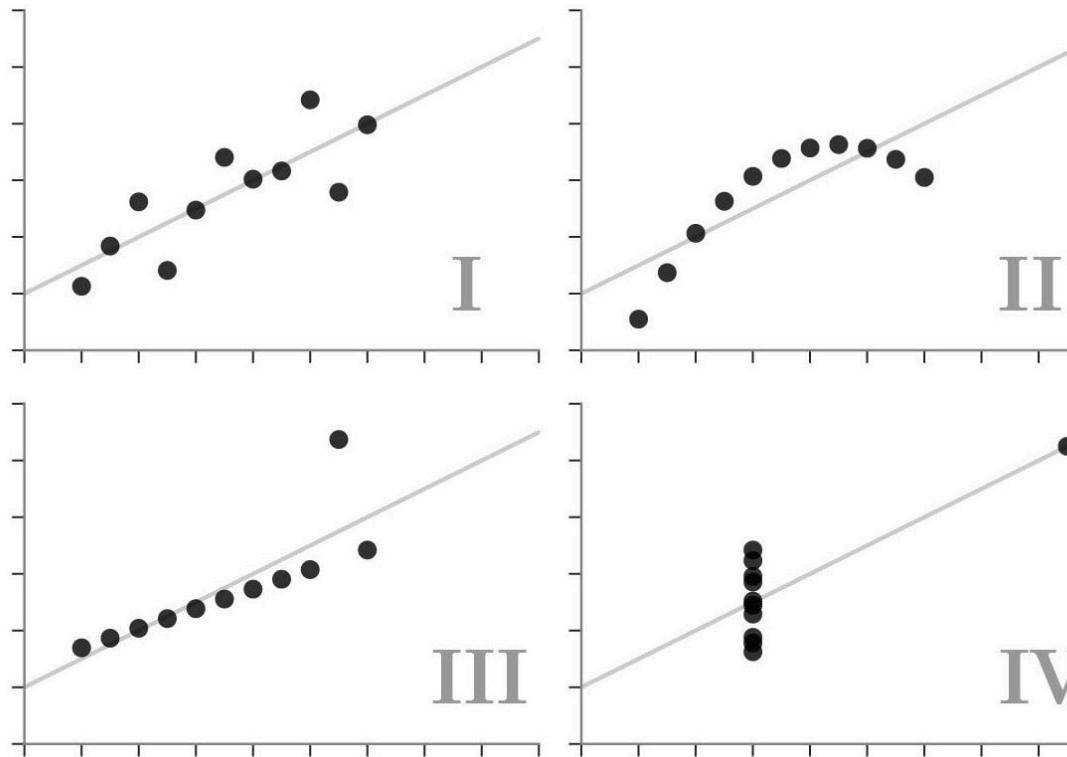
2025-04-07

Graphs

- Essential part of data analyses
- Data with same summary statistics can look very different when plotted out.

Anscombe's Quartet

Each dataset has the same summary statistics (mean, standard deviation, correlation), and the datasets are *clearly different*, and *visually distinct*.

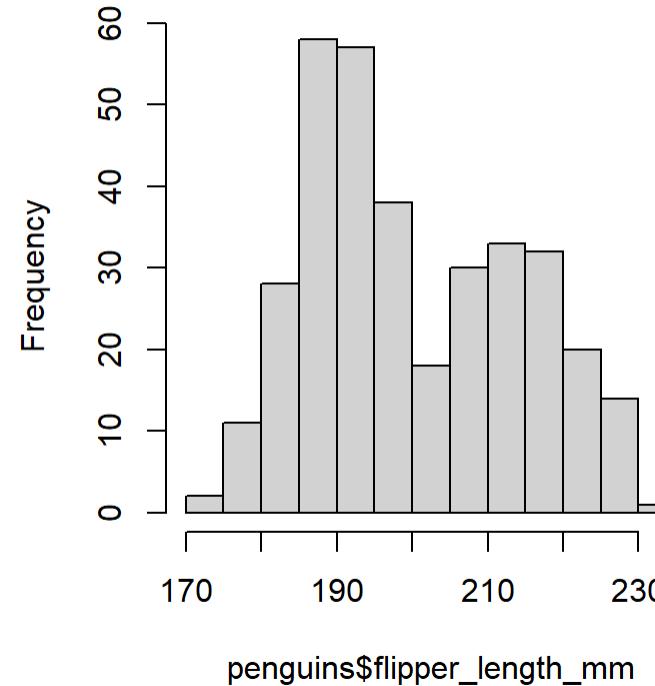


Anscombe's quartet, Datasaurus

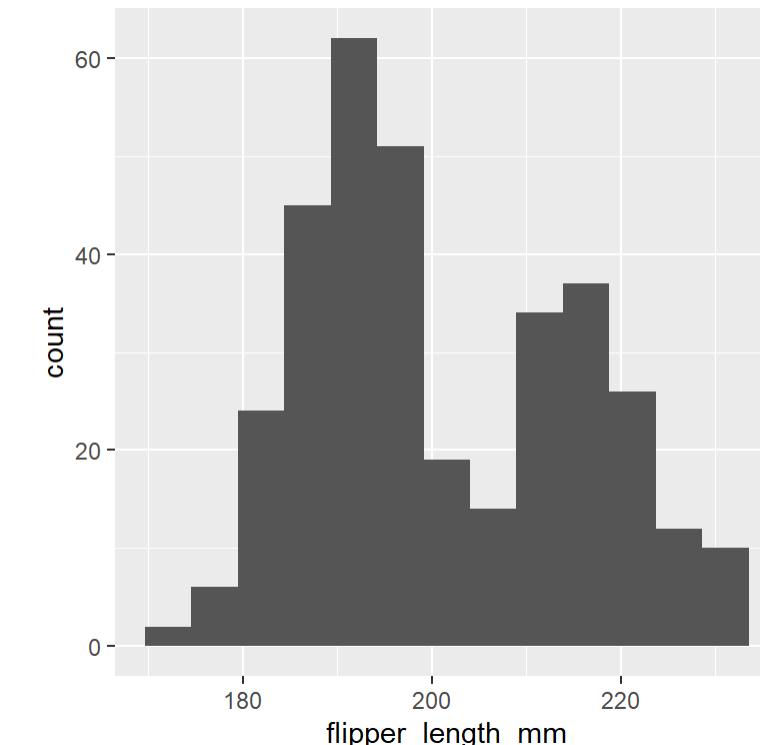
Base Graphics vs ggplot2

```
1 hist(penguins$flipper_length_mm)
```

Histogram of penguins\$flipper_length_mm



```
1 ggplot(penguins, aes(flipper_length_mm)) +  
2   geom_histogram(bins = 13)
```



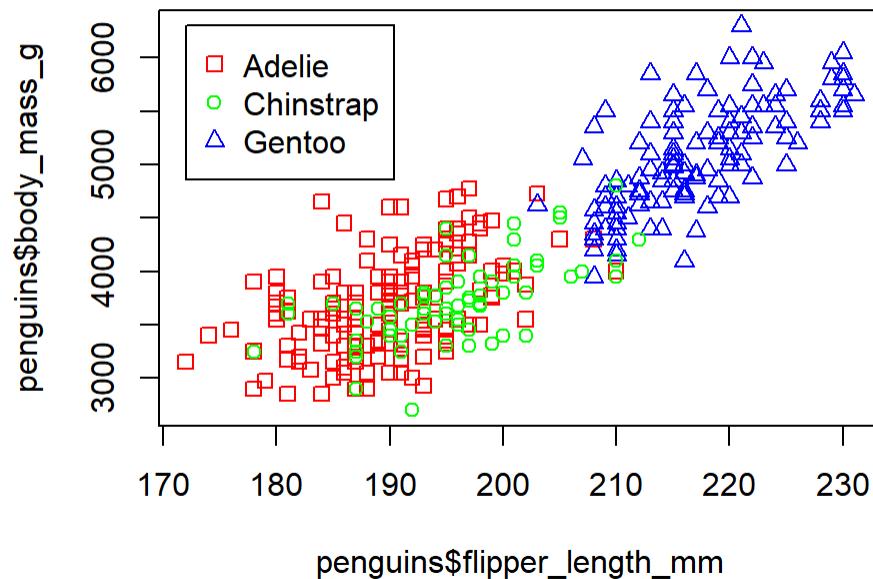
Base Graphics vs ggplot2

basic r plot

```

1 plot(penguins$flipper_length_mm, penguins$body_mass_g,
2       col=c("red", "green", "blue") [penguins$species],
3       pch=c(0,1,2) [penguins$species])
4 legend(x=172, y=6300,
5        legend=c("Adelie", "Chinstrap", "Gentoo"),
6        pch=c(0,1,2), col=c("red", "green", "blue"))

```

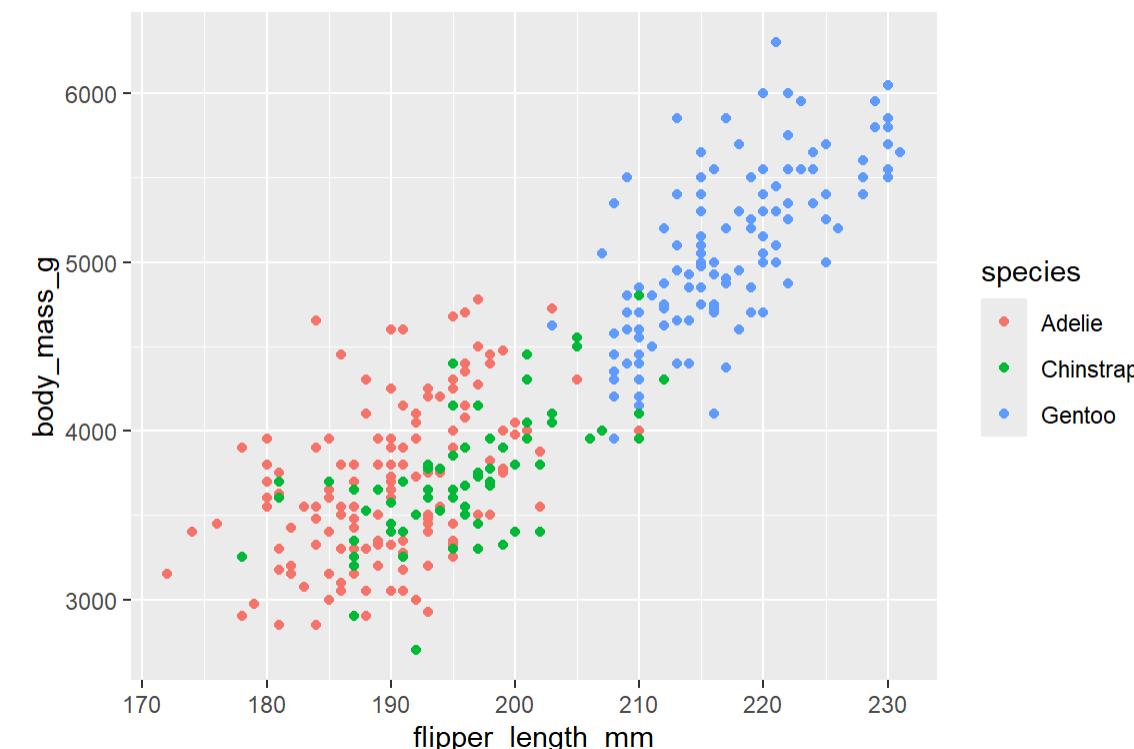


ggplot

```

1 ggplot(penguins, aes(flipper_length_mm, body_mass_g,
2   color=species)) +
3   geom_point()

```

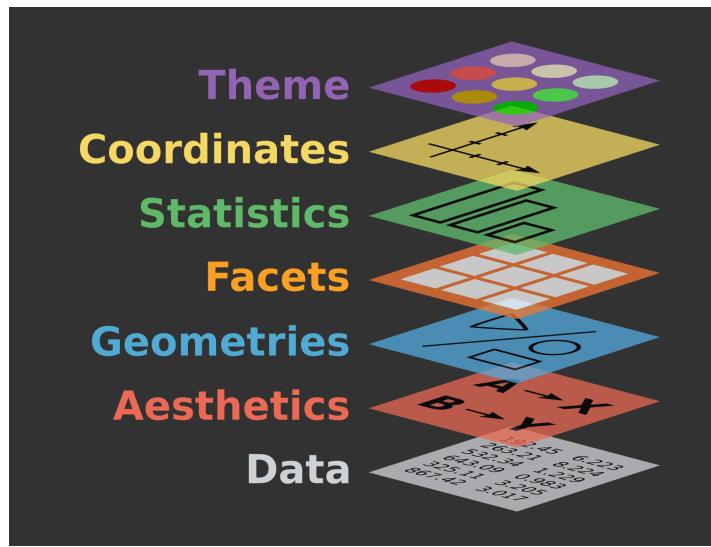


Why ggplot2?

- Consistent code
- Flexible (Add/remove components)
- Automatic legends, colors etc
- Save plot objects
- Themes for reusing styles
- Numerous add-ons/extensions
- Nearly complete structured graphing solution
- Adapted to other programming languages
 - **Gadfly** in Julia, **gramm** in MatLab, **GGPlot** in Perl, **Vega** in Javascript, **PlotNine**, **ggpy**, **lets-plot** in Python.

Grammar Of Graphics

Building A Graph: • Syntax



```
ggplot (data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
  stat = <STAT>, position = <POSITION>) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION> +
  <SCALE_FUNCTION> +
  <THEME_FUNCTION>
```

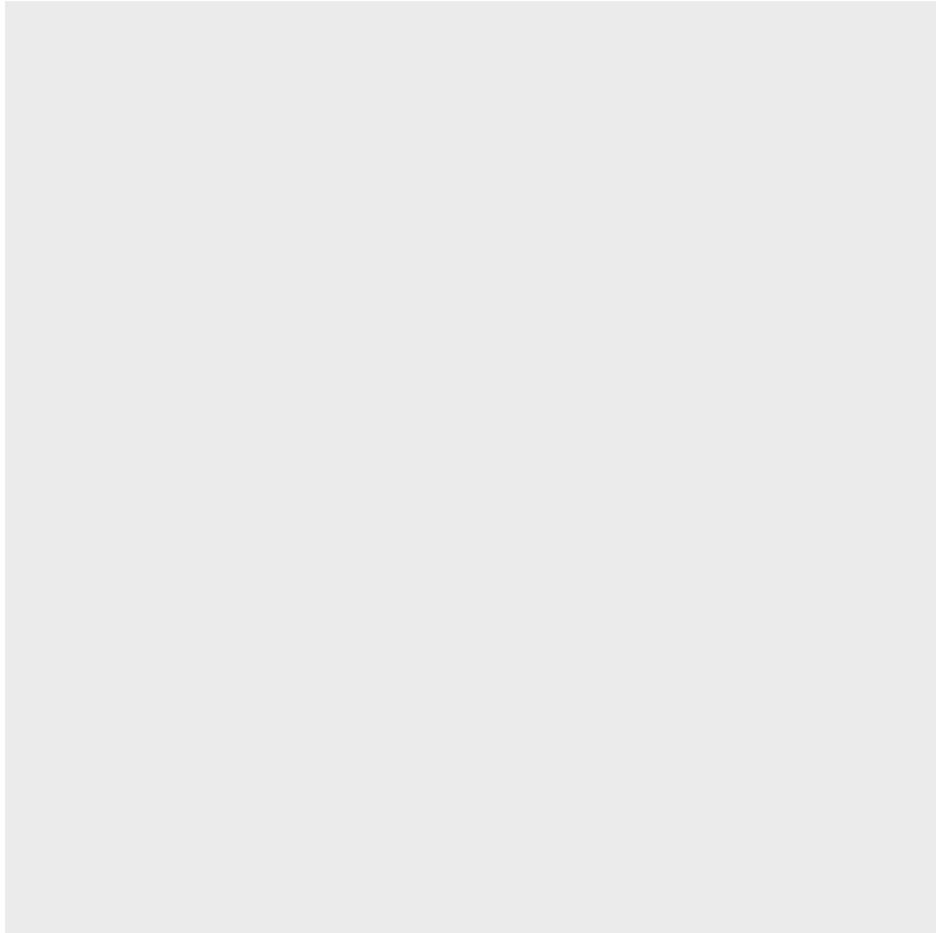
] required
]
Not required, sensible defaults supplied

Building A Graph

```
1 require(ggplot2)          # load ggplot2  
2 require(palmerpenguins) # load penguins data pack  
3  
4 data("penguins")        # load penguins data
```

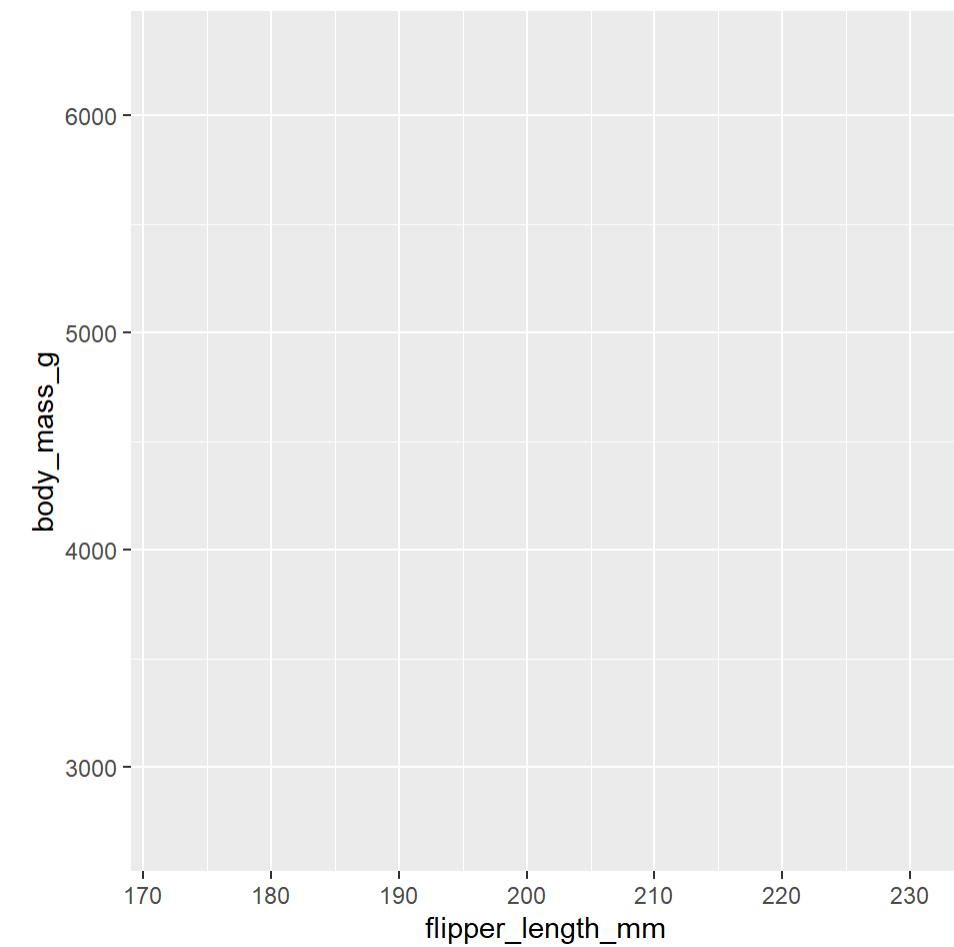
Building A Graph

```
1 ggplot(data = penguins)
```



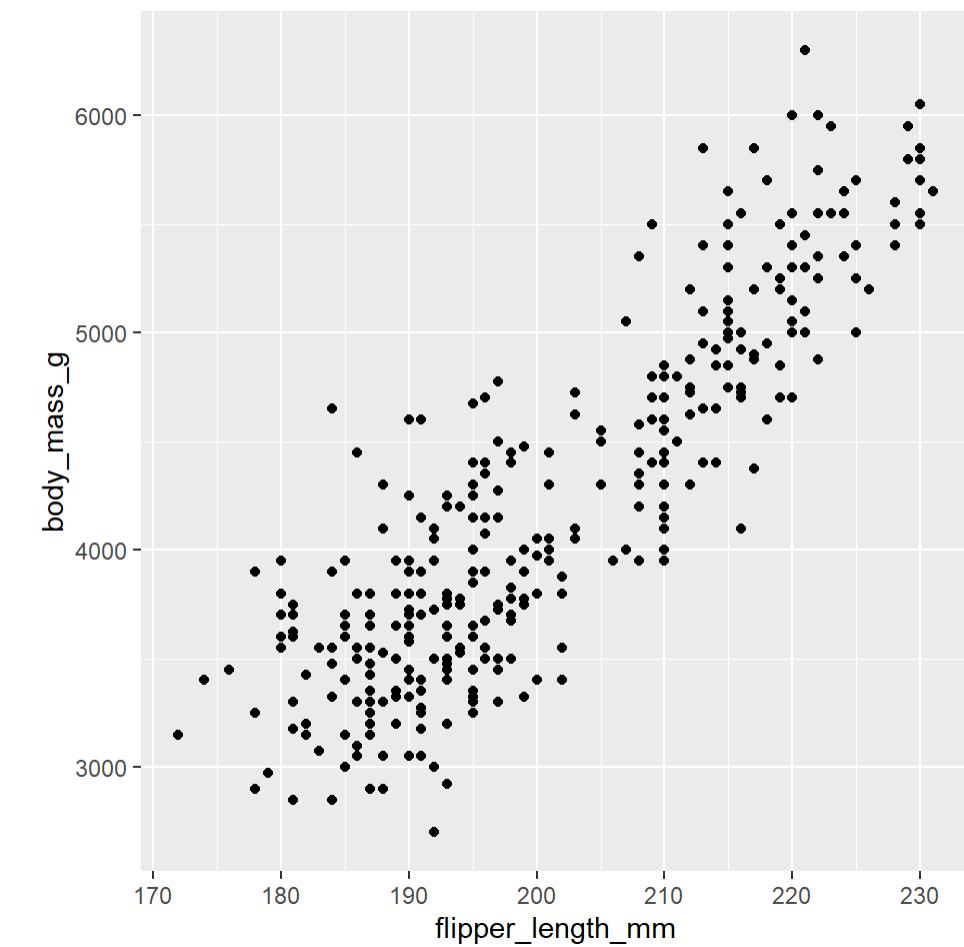
Building A Graph

```
1 ggplot(data = penguins,  
2 mapping = aes(x = flipper_length_mm,  
3                 y = body_mass_g))
```



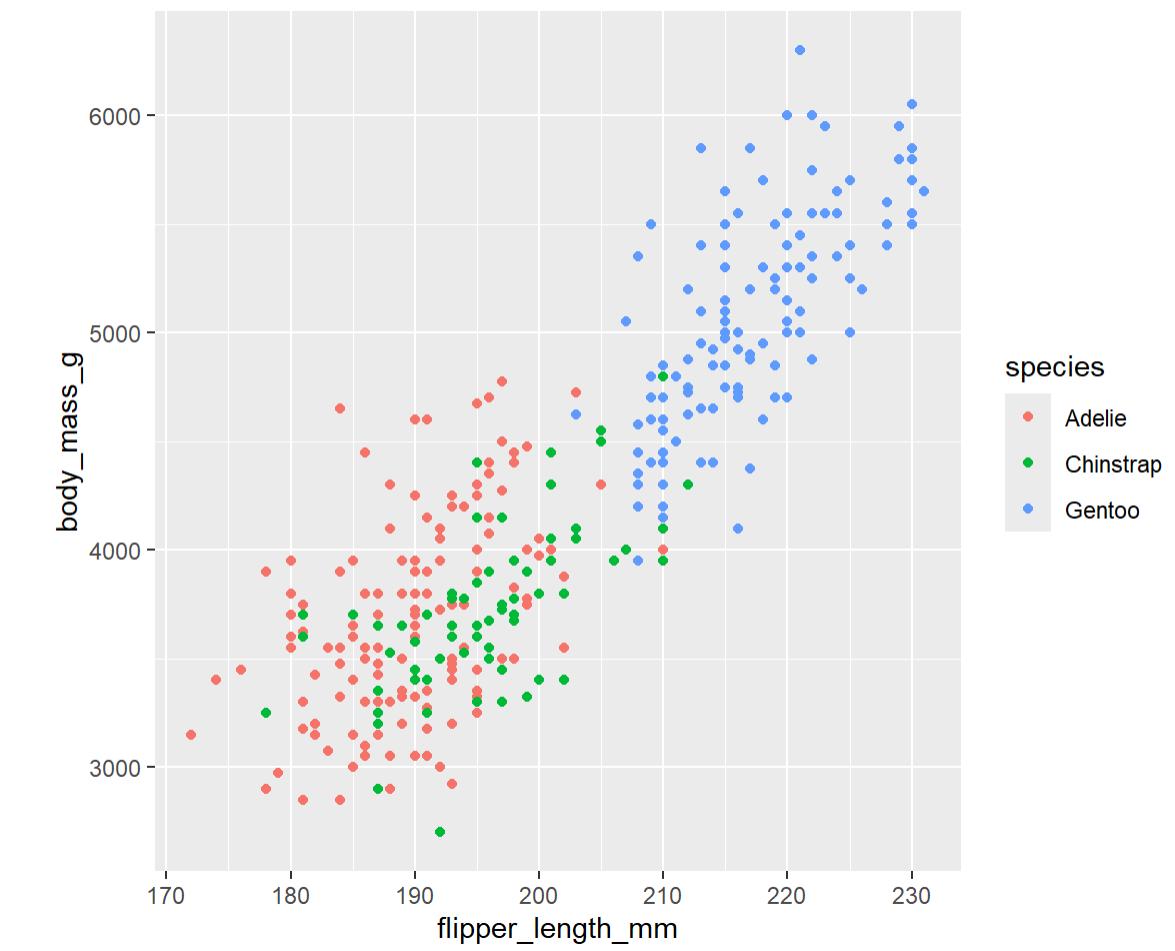
Building A Graph

```
1 ggplot(data = penguins,  
2 mapping = aes(x = flipper_length_mm,  
3                 y = body_mass_g)) +  
4 geom_point()
```



Building A Graph

```
1 ggplot(data = penguins,
2 mapping = aes(x = flipper_length_mm,
3                 y = body_mass_g,
4                 colour = species)) +
5 geom_point()
```



Building A Graph

```

1 ggplot(data = penguins,
2 mapping = aes(x = flipper_length_mm,
3                 y = body_mass_g,
4                 colour = species)) +
5 geom_point()

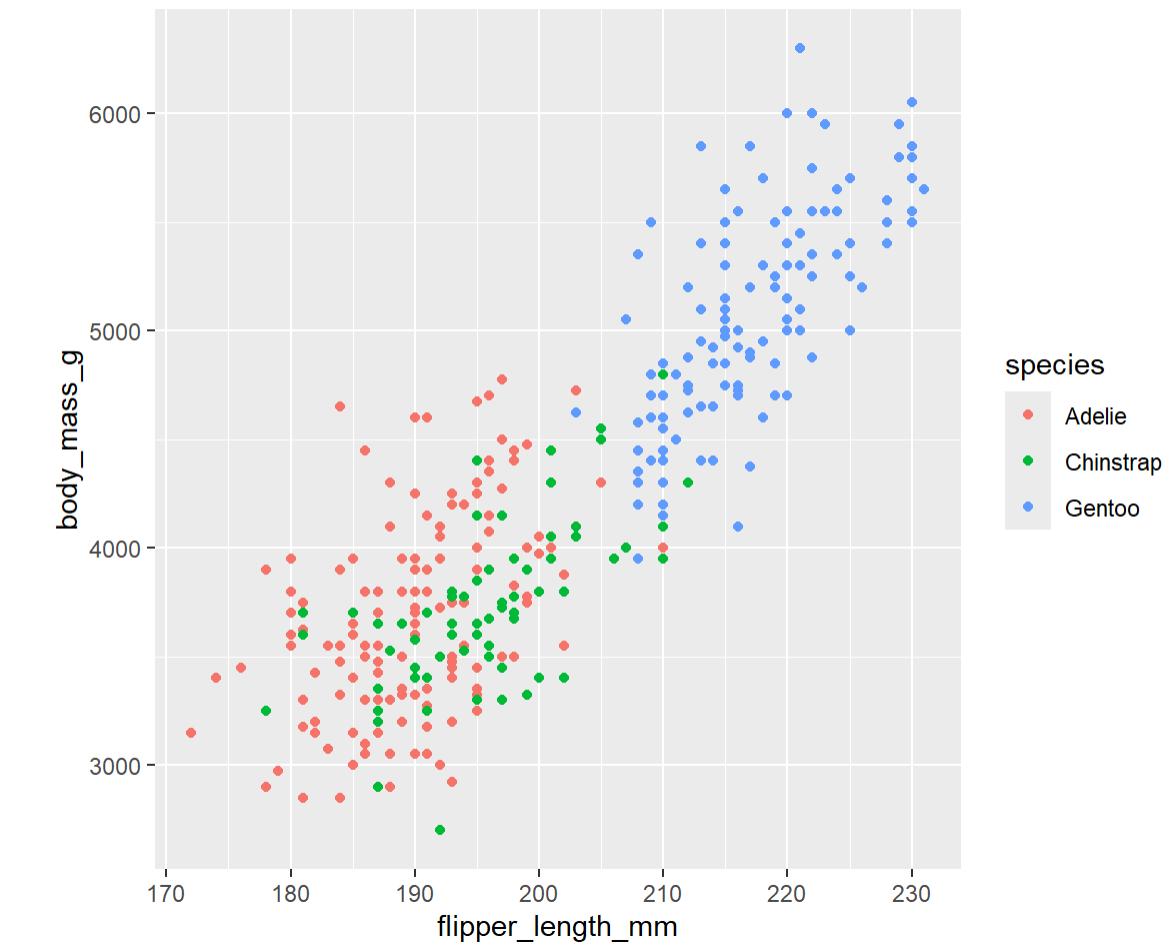
```

Or

```

1 ggplot(data = penguins) +
2 geom_point(mapping = aes(x = flipper_length_mm,
3                           y = body_mass_g,
4                           colour = species))

```



Building A Graph

```

1 ggplot(data = penguins,
2 mapping = aes(x = flipper_length_mm,
3                 y = body_mass_g,
4                 colour = species)) +
5 geom_point()

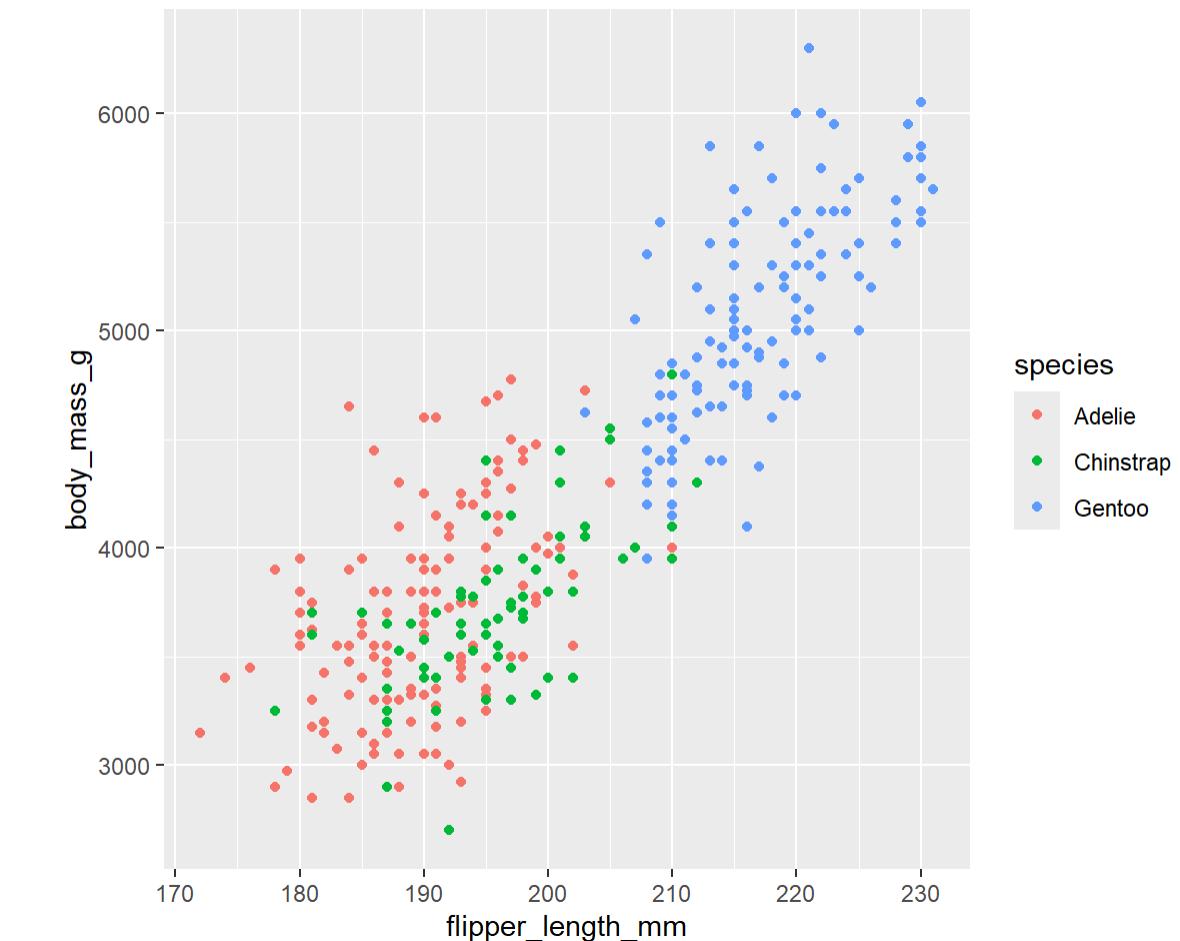
```

Or

```

1 ggplot(data = penguins) +
2 geom_point(mapping = aes(x = flipper_length_mm,
3                           y = body_mass_g,
4                           colour = species))

```



```

ggplot (data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
  stat =<STAT>, position =<POSITION>) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>

```

[required] [Not required, sensible defaults supplied]

Data • penguins

- Input data is always an R `data.frame` object

species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
Adelie	Torgersen	39.1	18.7	181	3750
Adelie	Torgersen	39.5	17.4	186	3800
Adelie	Torgersen	40.3	18.0	195	3250

```
1 str(penguins)
```

```
tibble [344 x 8] (S3: tbl_df/tbl/data.frame)
$ species      : Factor w/ 3 levels "Adelie", "Chinstrap", ...: 1 1 1 1 1 1 1 1 1 ...
$ island       : Factor w/ 3 levels "Biscoe", "Dream", ...: 3 3 3 3 3 3 3 3 3 ...
$ bill_length_mm: num [1:344] 39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ...
$ bill_depth_mm: num [1:344] 18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 20.2 ...
$ flipper_length_mm: int [1:344] 181 186 195 NA 193 190 181 195 193 190 ...
$ body_mass_g   : int [1:344] 3750 3800 3250 NA 3450 3650 3625 4675 3475 4250 ...
$ sex          : Factor w/ 2 levels "female", "male": 2 1 1 NA 1 2 1 2 NA NA ...
$ year         : int [1:344] 2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
```

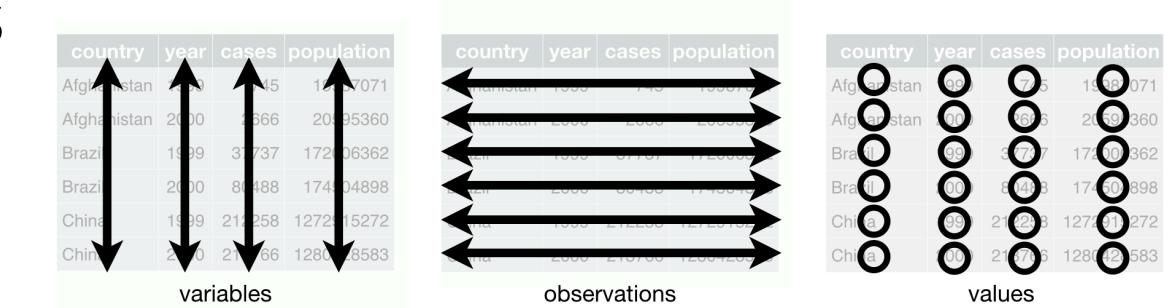
Data • diamonds

carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48

```
tibble [53,940 x 10] (S3: tbl_df/tbl/data.frame)
$ carat   : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
$ cut     : Ord.factor w/ 5 levels "Fair" < "Good" < ...: 5 4 2 4 2 3 3 3 1 3 ...
$ color   : Ord.factor w/ 7 levels "D" < "E" < "F" < "G" < ...: 2 2 2 6 7 7 6 5 2 5 ...
$ clarity: Ord.factor w/ 8 levels "I1" < "SI2" < "SI1" < ...: 2 3 5 4 2 6 7 3 4 5 ...
$ depth   : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
$ table   : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
$ price   : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
$ x       : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
$ y       : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
$ z       : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

Data • format

- Transforming data into ‘long’ or ‘wide’ formats



Wide

```
1 head(penguins, n=4)

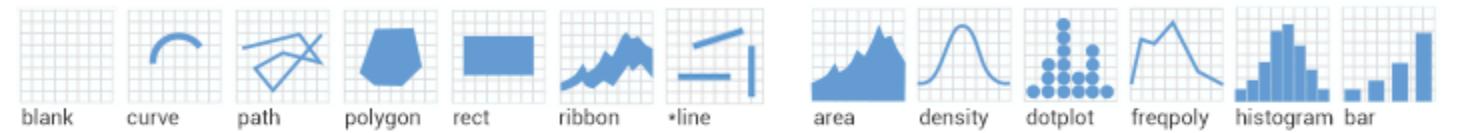
# A tibble: 4 × 8
  species   island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
  <fct>     <fct>        <dbl>          <dbl>            <int>        <int>
1 Adelie    Torgersen      39.1           18.7             181         3750
2 Adelie    Torgersen      39.5           17.4             186         3800
3 Adelie    Torgersen      40.3           18               195         3250
4 Adelie    Torgersen       NA             NA                NA         NA
# i 2 more variables: sex <fct>, year <int>
```

Long

	species	island	sex	year	variables	value
1	Adelie	Torgersen	male	2007	bill_length_mm	39.1
2	Adelie	Torgersen	male	2007	bill_depth_mm	18.7
3	Adelie	Torgersen	male	2007	flipper_length_mm	181.0
4	Adelie	Torgersen	male	2007	body_mass_g	3750.0

Geoms • types

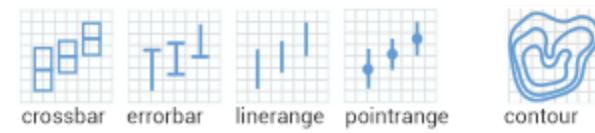
Basic



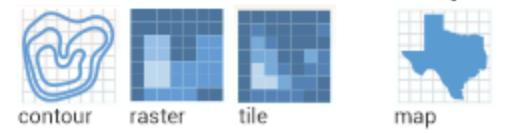
Two variables



Error



Three variables



Map



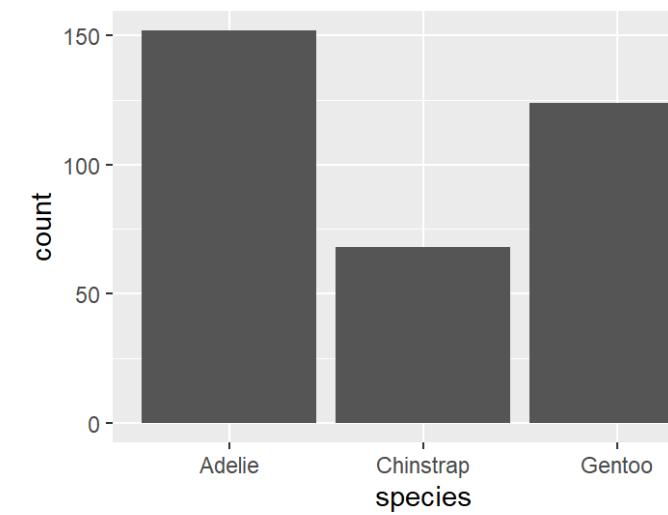
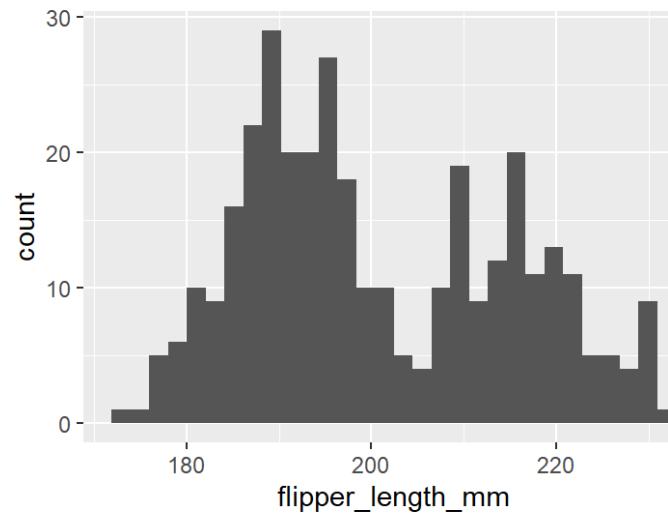
Stats

- Stats compute new variables from input data.
- Geoms have default stats.
- Plots can be built with stats.

```

1 x <- ggplot(data = penguins) +
2   geom_bar(aes(x=flipper_length_mm), stat="bin")
3
4 y <- ggplot(data = penguins) +
5   geom_bar(aes(x=species), stat="count")
6
7 wrap_plots(x,y,nrow=1)

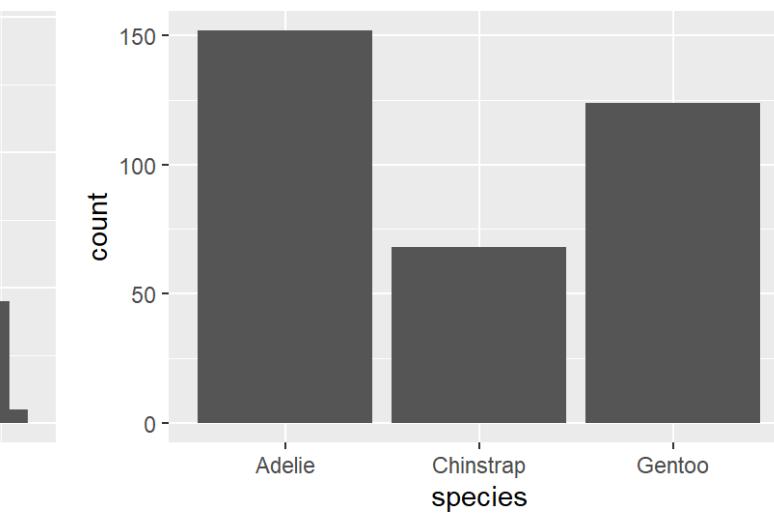
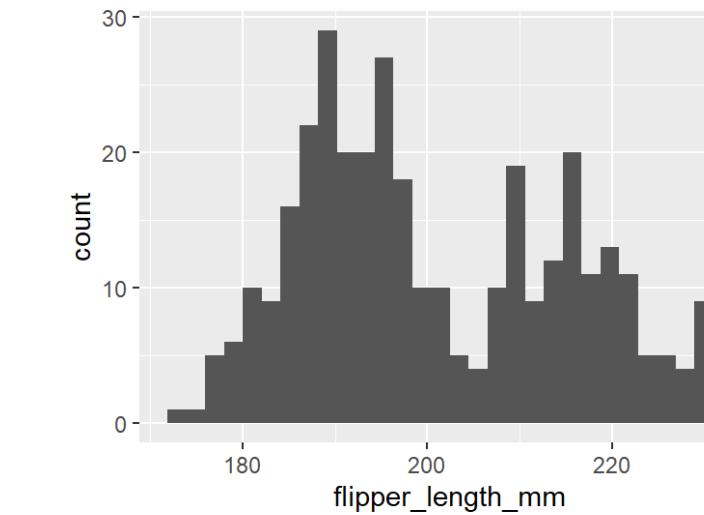
```



```

1 x <- ggplot(data = penguins) +
2   stat_bin(aes(x=flipper_length_mm), geom="bar")
3
4 y <- ggplot(data = penguins) +
5   stat_count(aes(x=species), geom="bar")
6
7 wrap_plots(x,y,nrow=1)

```



Stats

- Stats have default geoms.

plot	stat	geom
histogram	bin	bar
smooth	smooth	line
boxplot	boxplot	boxplot
density	density	line
freqpoly	freqpoly	line

Use `args(geom_bar)` to check arguments.

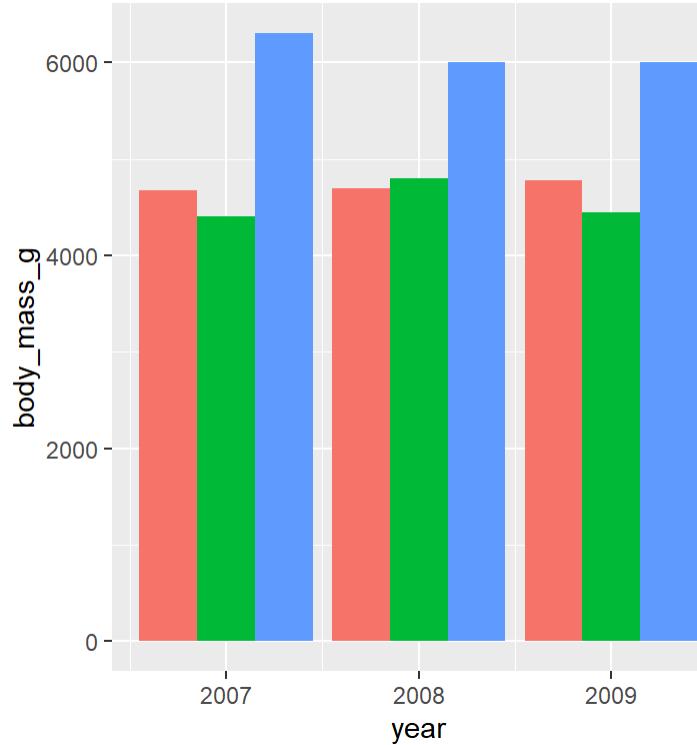
Position

```
1 p <- ggplot(penguins, aes(x=year, y=body_mass_g, fill=species))
```

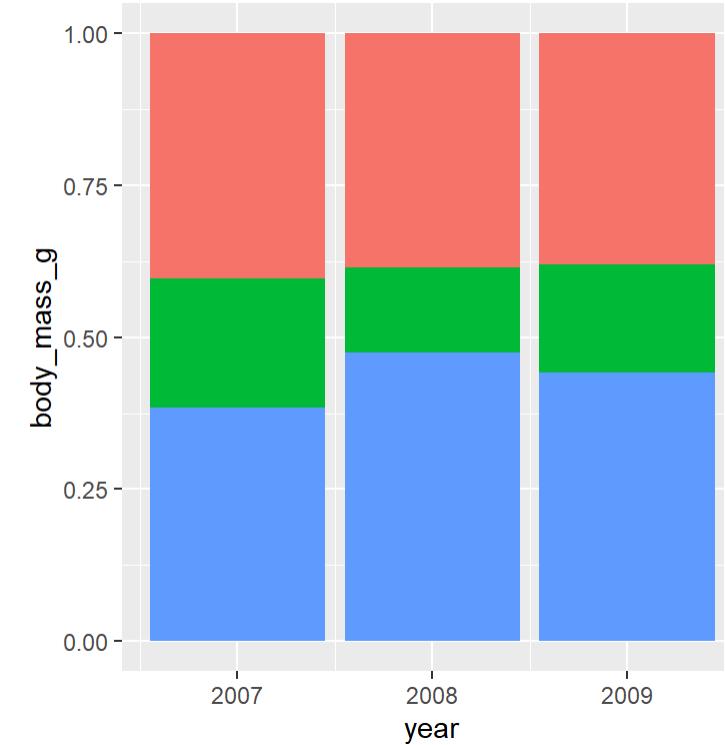
```
1 p + geom_bar(stat="identity",
2               position="stack")
```



```
1 p + geom_bar(stat="identity",
2               position="dodge")
```



```
1 p + geom_bar(stat="identity",
2               position="fill")
```



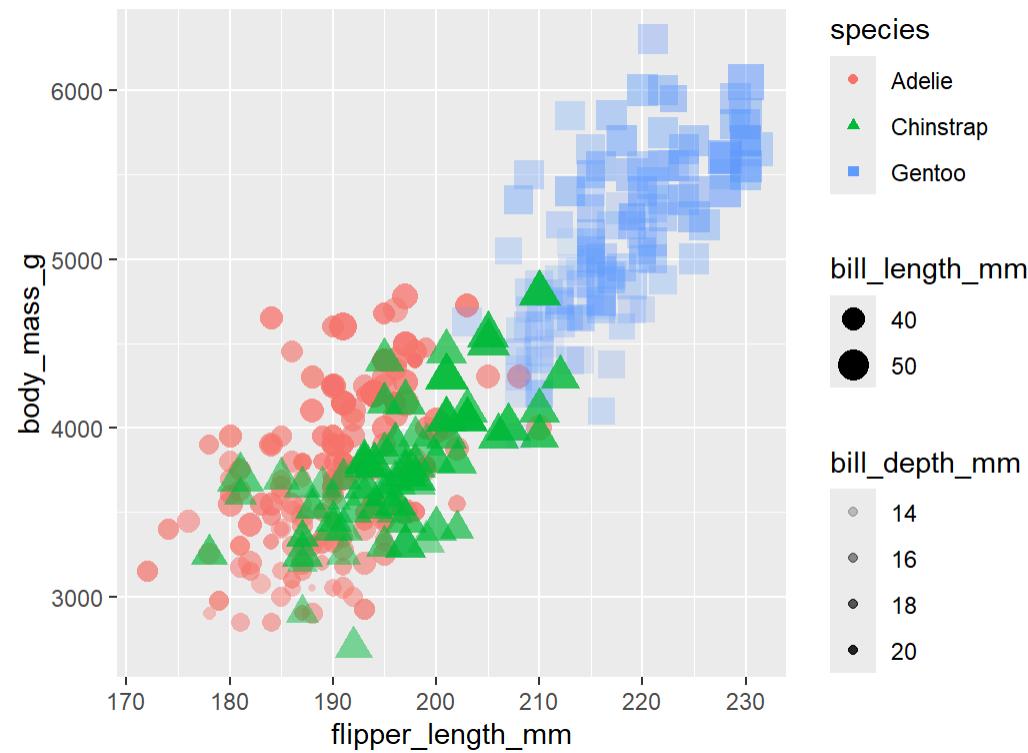
Aesthetics

- Aesthetic mapping

```

1 ggplot(data = penguins) +
2   geom_point(aes(x=flipper_length_mm,
3                   y=body_mass_g,
4                   size=bill_length_mm,
5                   alpha=bill_depth_mm,
6                   shape=species,
7                   color=species))

```

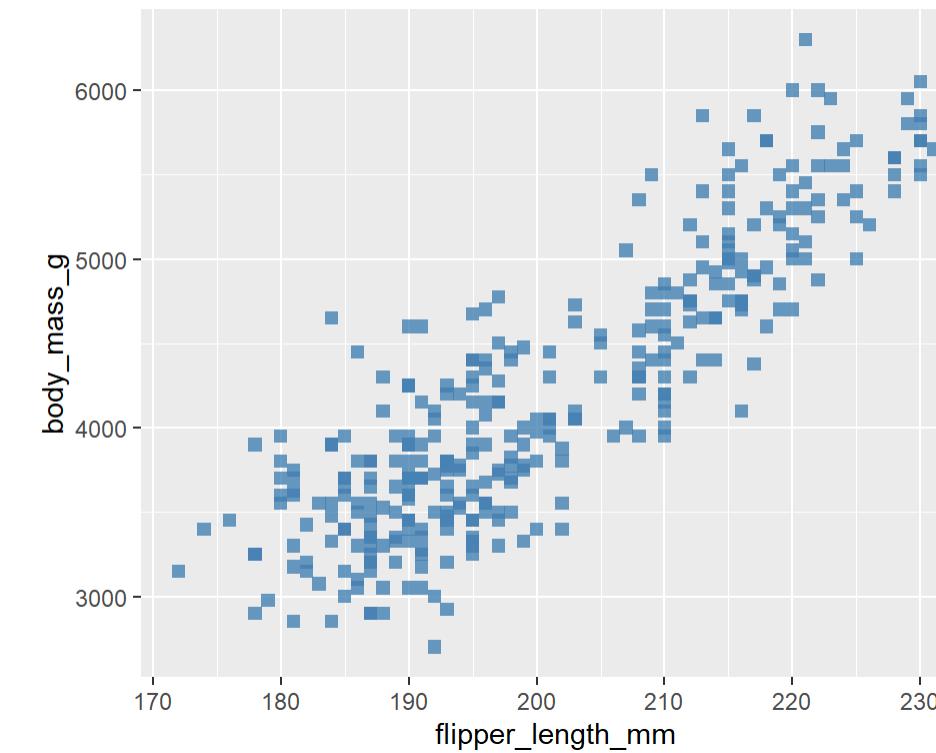


- Aesthetic parameter

```

1 ggplot(data = penguins) +
2   geom_point(aes(x=flipper_length_mm,
3                   y=body_mass_g),
4               size=2,
5               alpha=0.8,
6               shape=15,
7               color="steelblue")

```

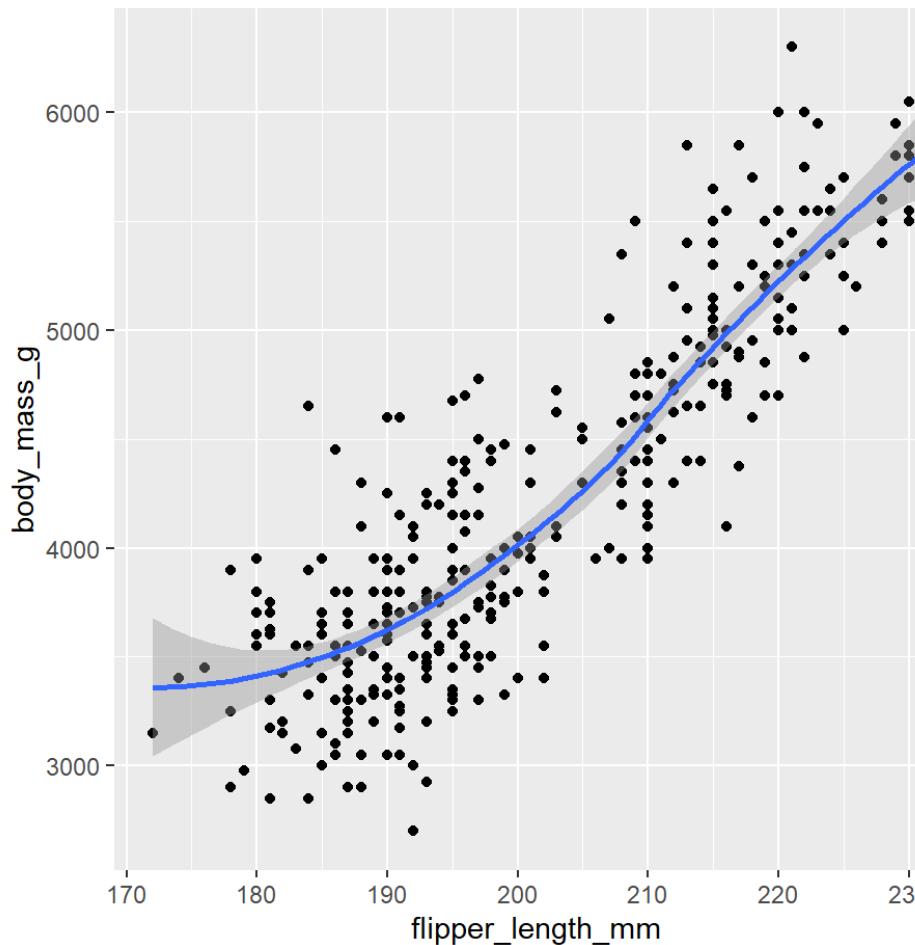
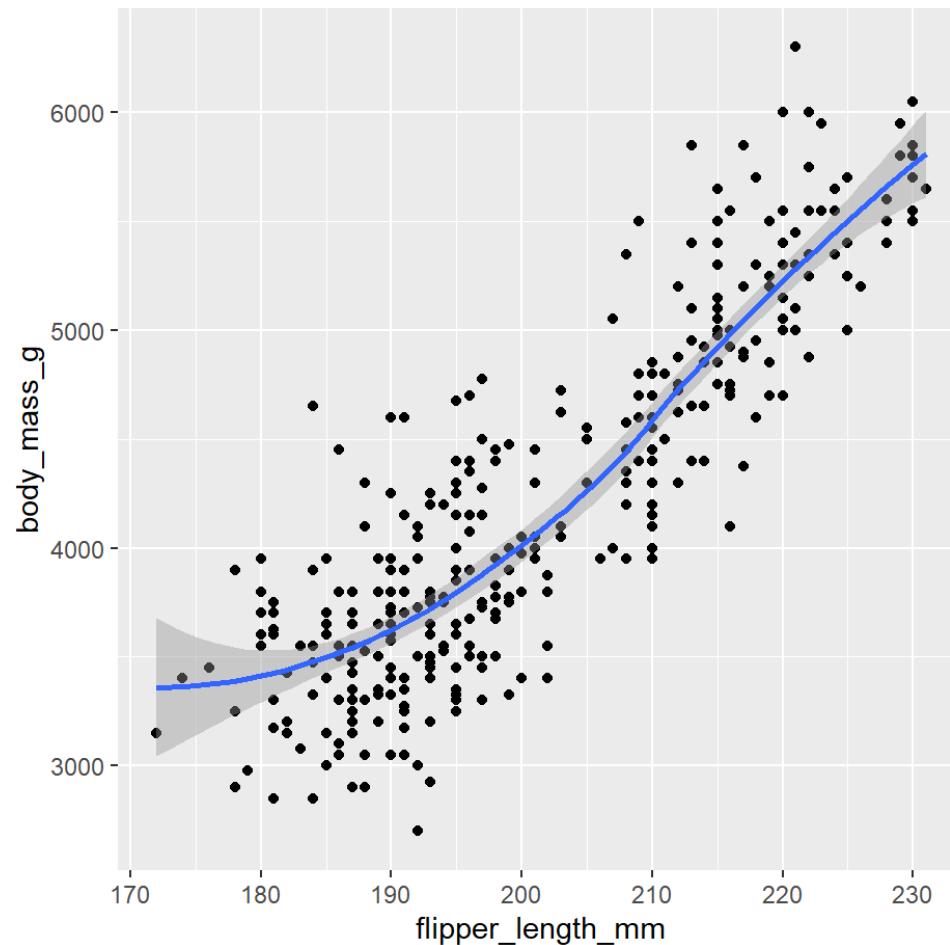


Aesthetics

```

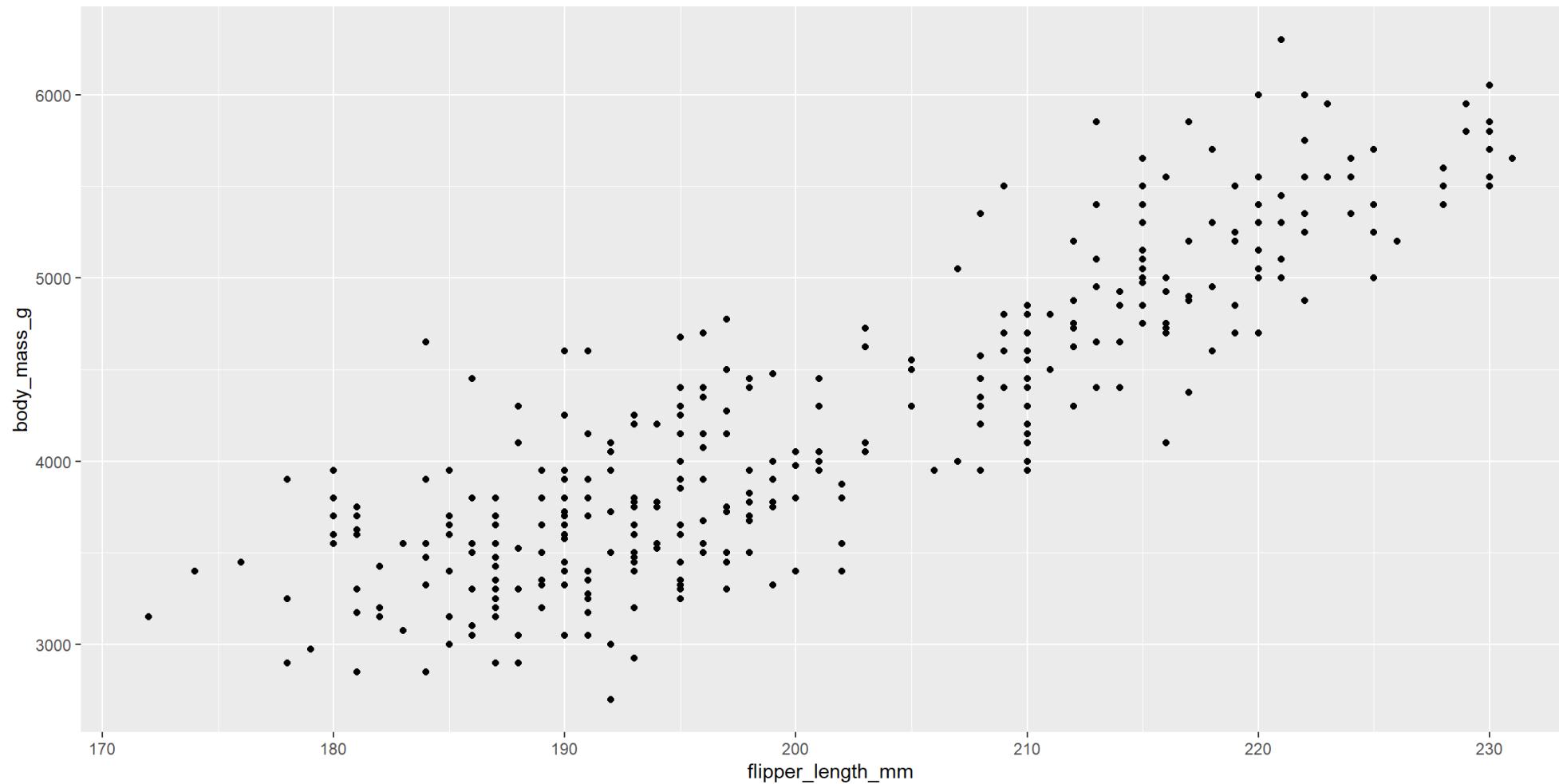
1 x1 <- ggplot(penguins) +
2   geom_point(aes(x=flipper_length_mm, y=body_mass_g)) +
3   stat_smooth(aes(x=flipper_length_mm, y=body_mass_g))
4
5 x2 <- ggplot(penguins, aes(x=flipper_length_mm, y=body_mass_g)) +
6   geom_point() +
7   geom_smooth()
8
9 wrap_plots(x1,x2,nrow=1,ncol=2)

```



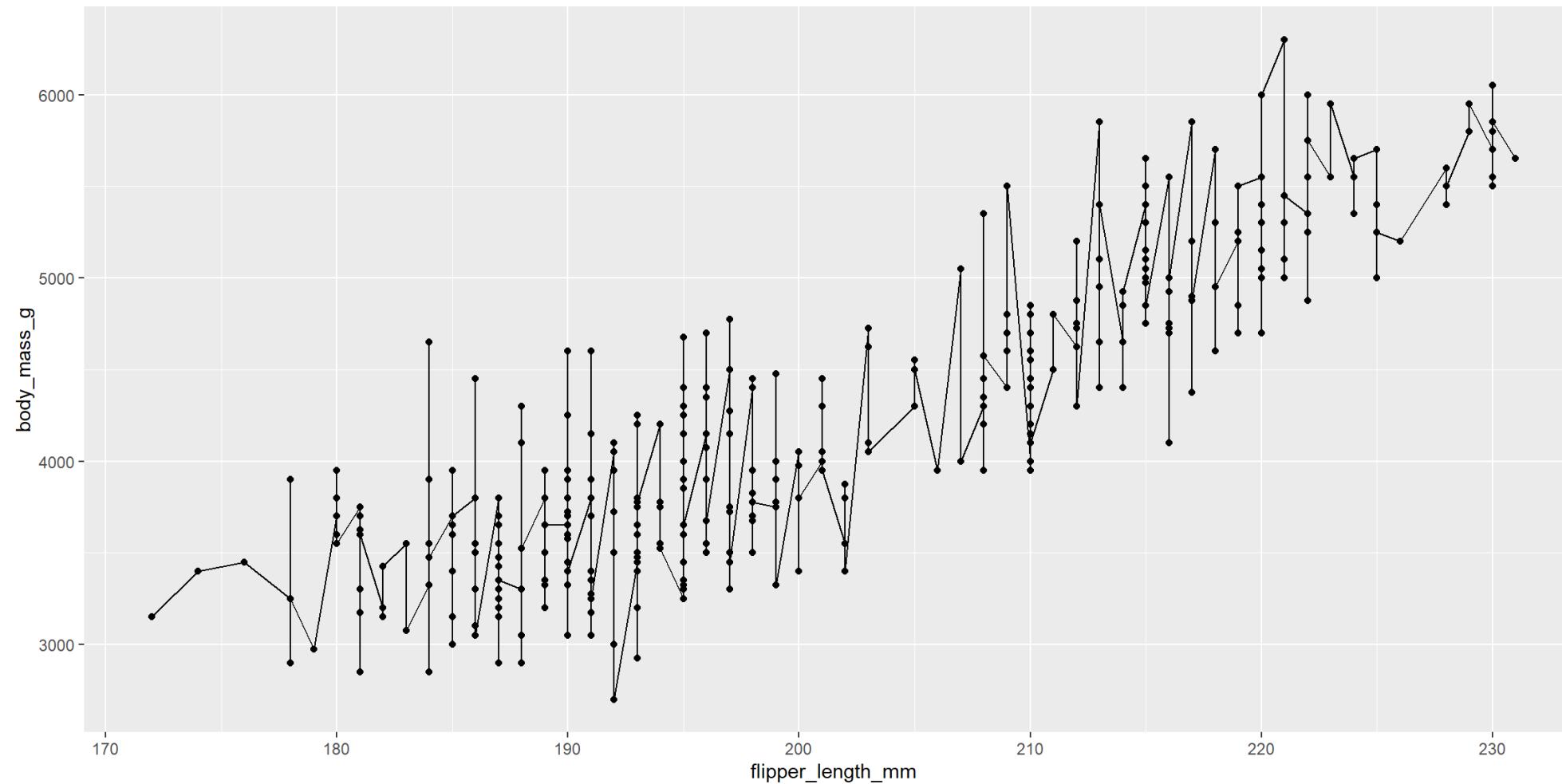
Multiple Geoms

```
1 p <- ggplot(penguins,aes(x=flipper_length_mm,y=body_mass_g)) +  
2   geom_point()  
3  
4  
5  
6  
7  
8  
9 p
```



Multiple Geoms

```
1 p <- ggplot(penguins,aes(x=flipper_length_mm,y=body_mass_g)) +  
2   geom_point() +  
3   geom_line()  
4  
5  
6  
7  
8  
9 p
```

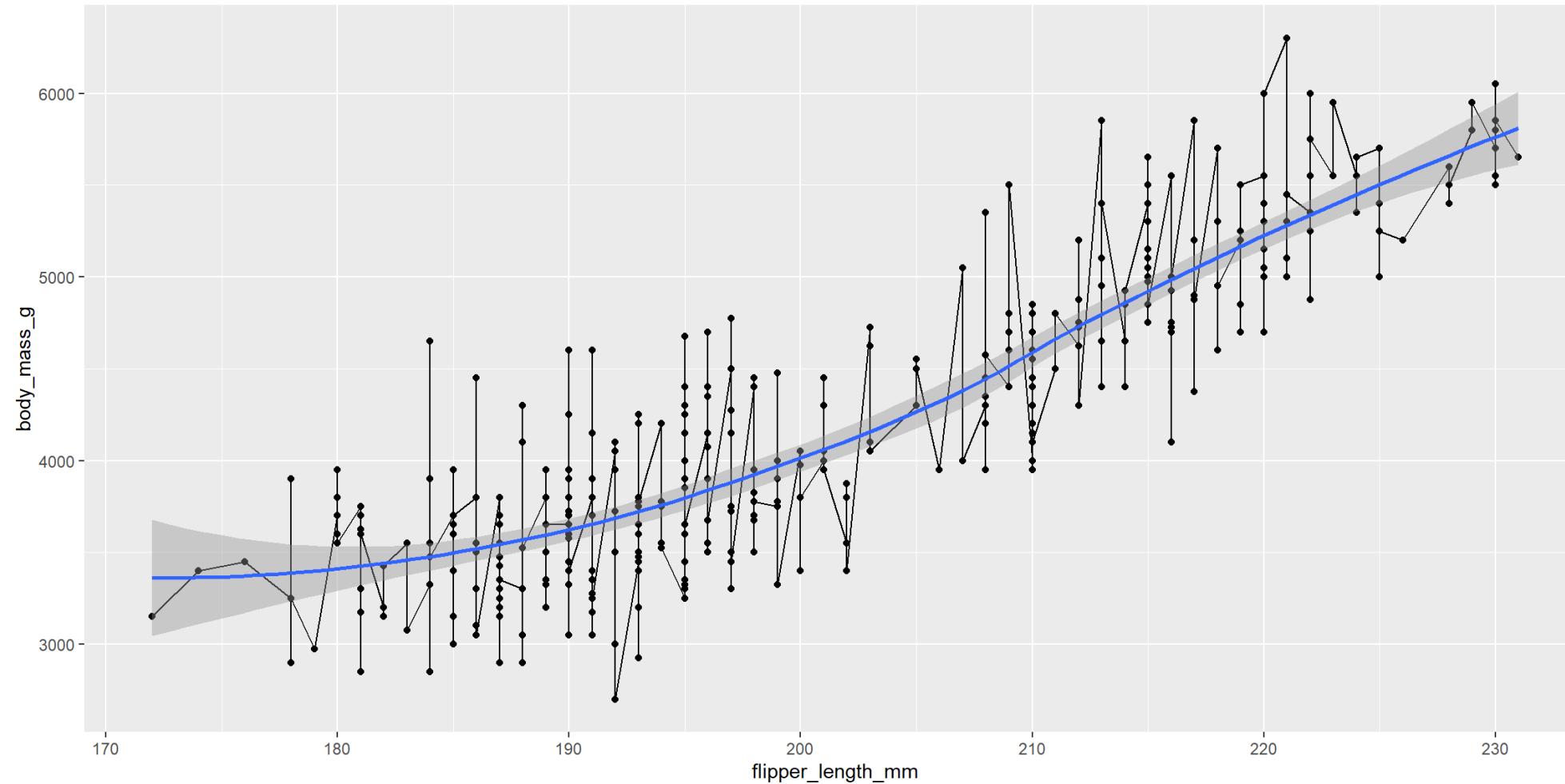


Multiple Geoms

```

1 p <- ggplot(penguins,aes(x=flipper_length_mm,y=body_mass_g))+
2   geom_point()+
3   geom_line()+
4   geom_smooth()
5
6
7
8
9 p

```

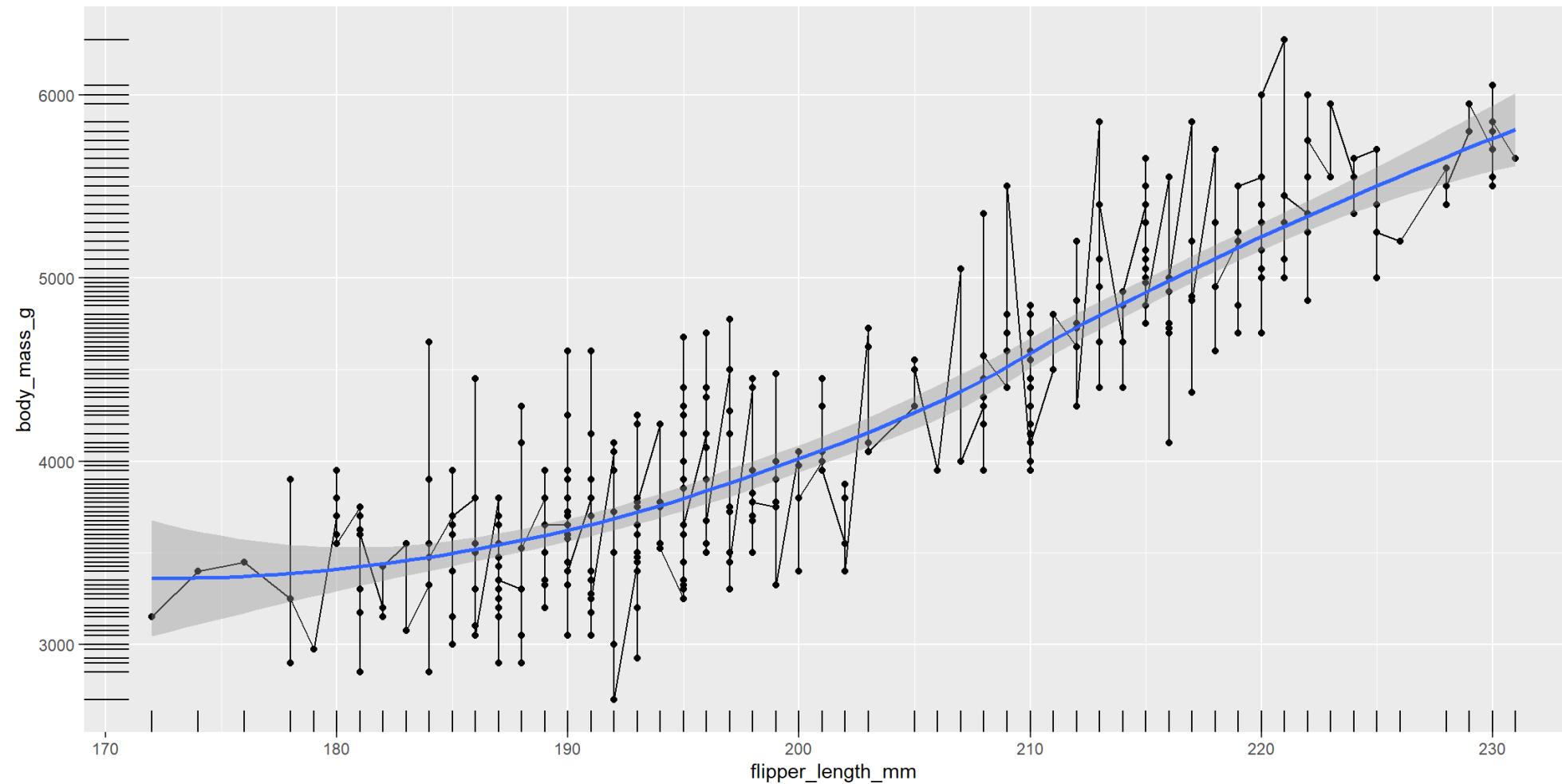


Multiple Geoms

```

1 p <- ggplot(penguins,aes(x=flipper_length_mm,y=body_mass_g)) +
2   geom_point() +
3   geom_line() +
4   geom_smooth() +
5   geom_rug()
6
7
8
9 p

```

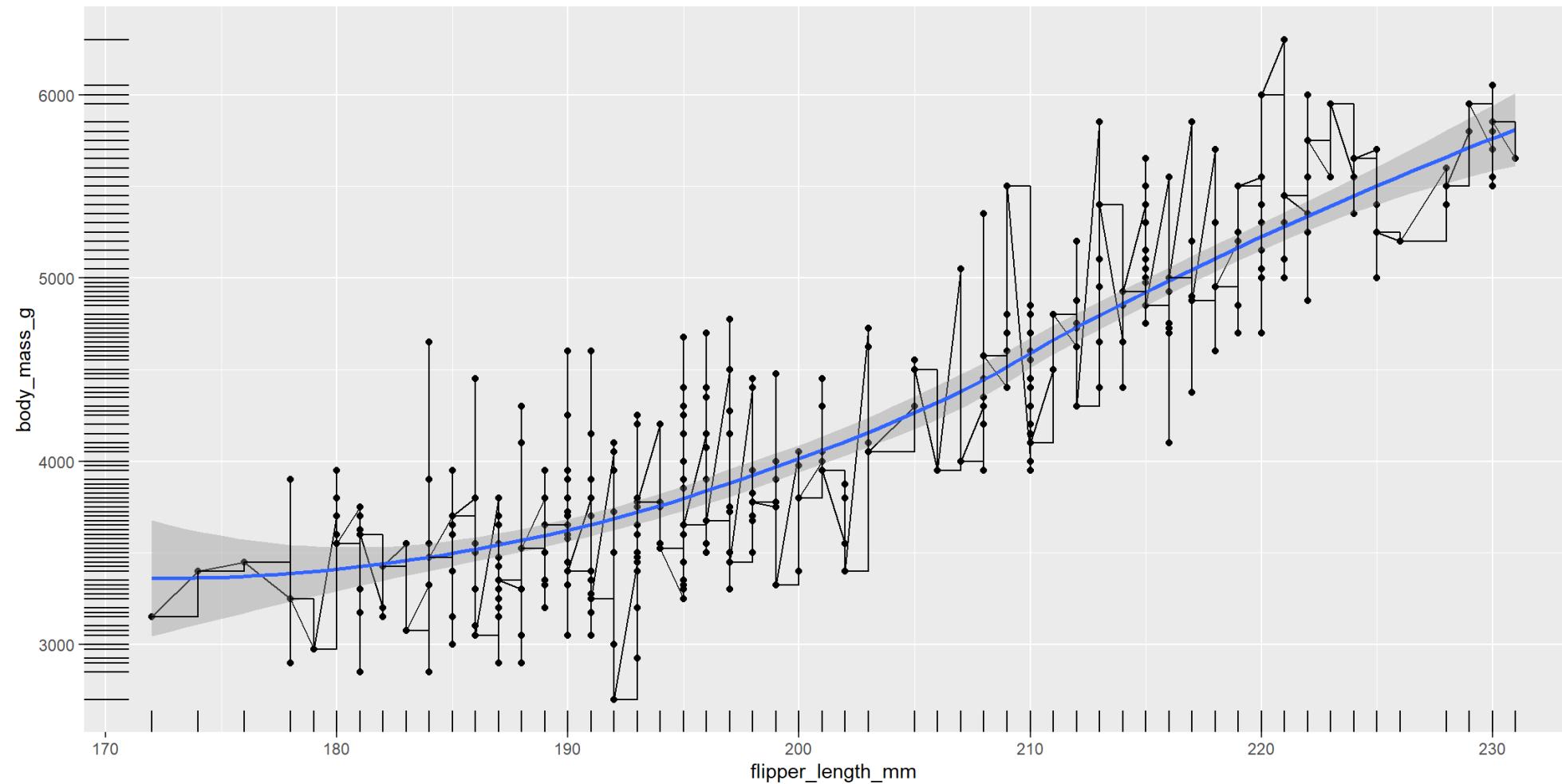


Multiple Geoms

```

1 p <- ggplot(penguins,aes(x=flipper_length_mm,y=body_mass_g))+
2   geom_point()+
3   geom_line()+
4   geom_smooth()+
5   geom_rug()+
6   geom_step()
7
8
9 p

```

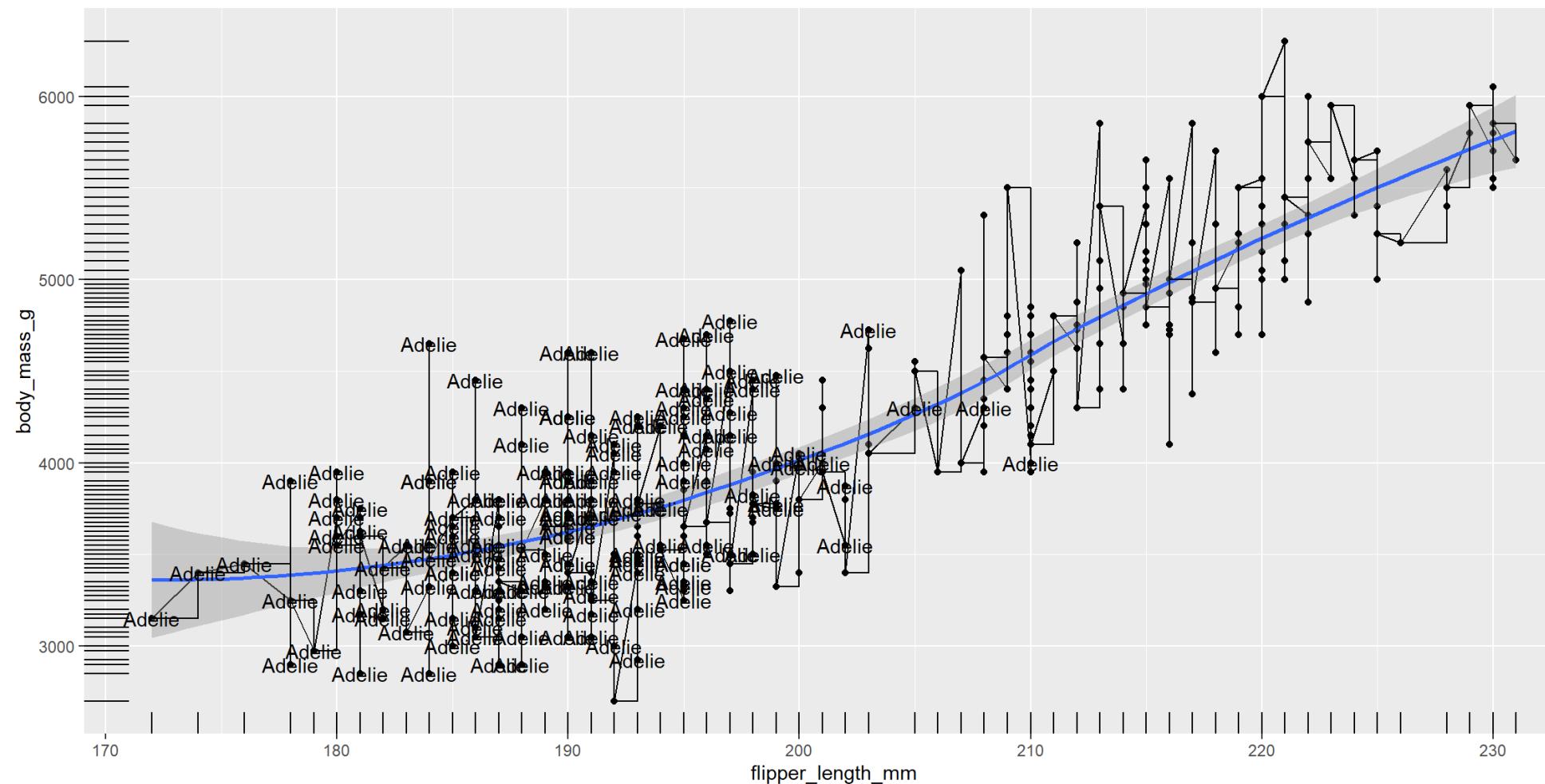


Multiple Geoms

```

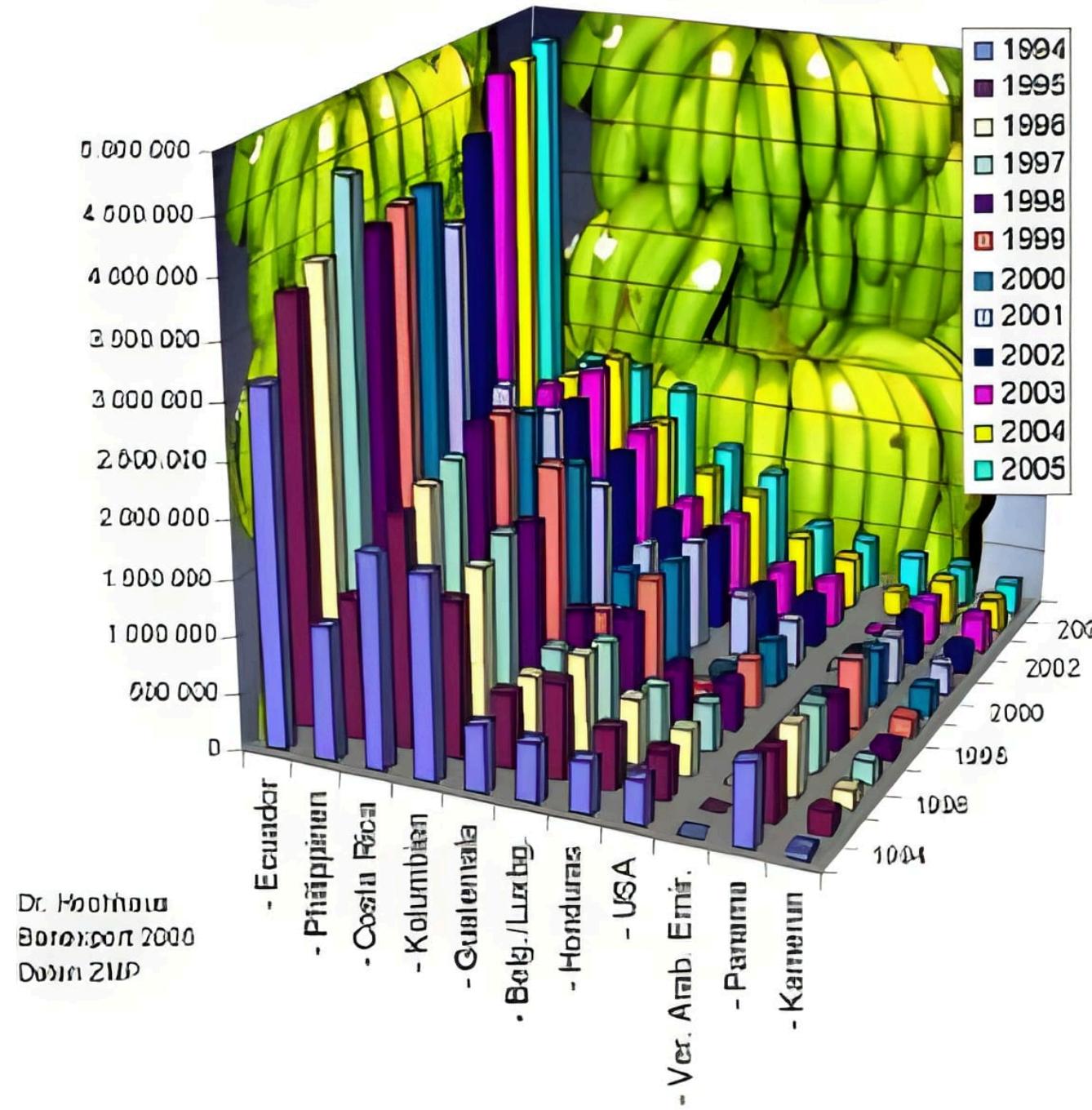
1 p <- ggplot(penguins,aes(x=flipper_length_mm,y=body_mass_g)) +
2   geom_point() +
3   geom_line() +
4   geom_smooth() +
5   geom_rug() +
6   geom_step() +
7   geom_text(data=subset(penguins,penguins$species=="Adelie"),
8             aes(label=species))
9 p

```

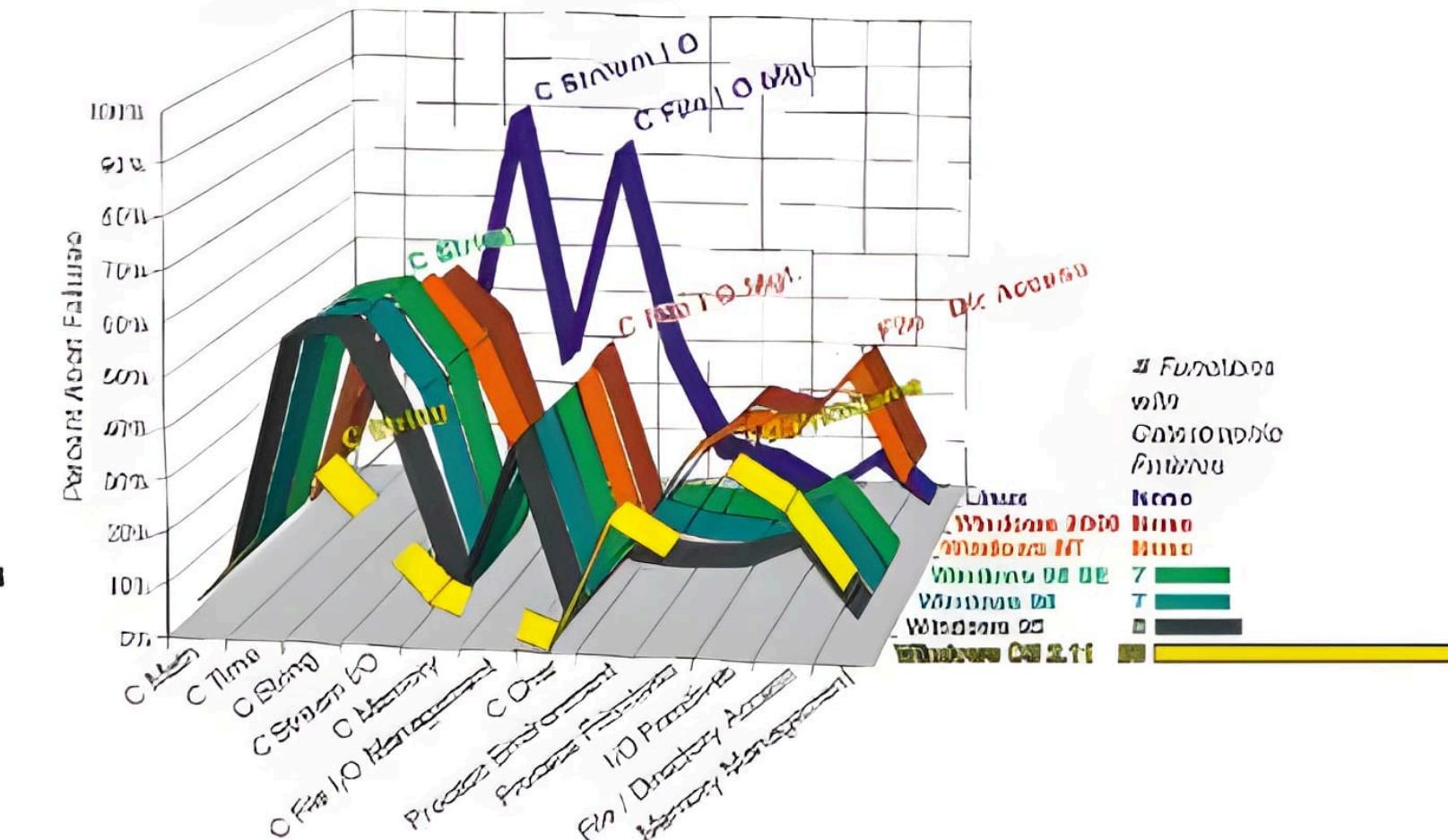


Just because you can doesn't mean you should!

Export von Bananen in Tonnen von 1994-2000



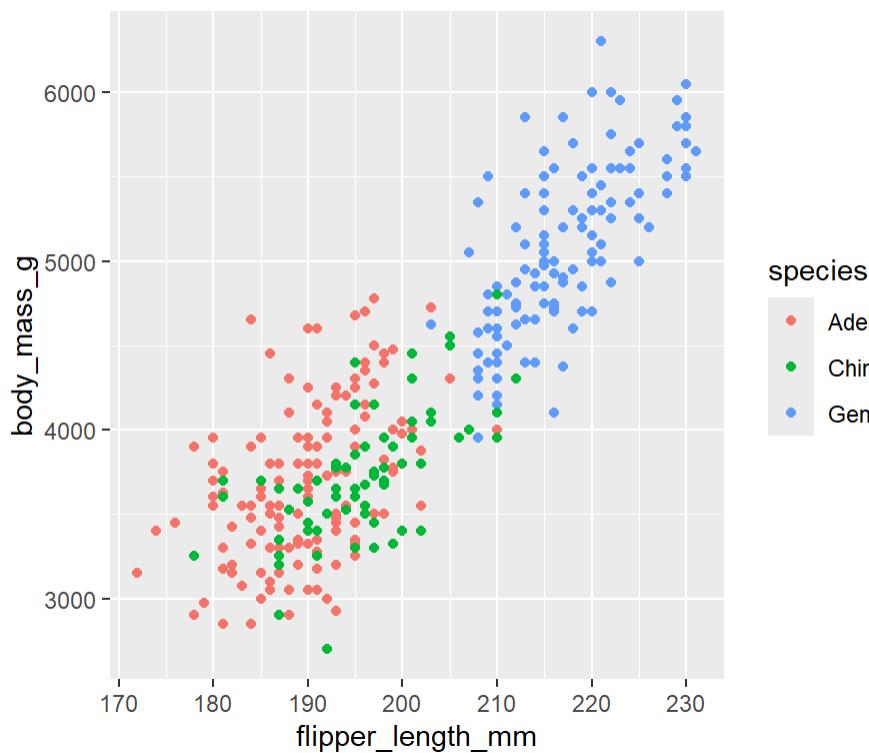
Percsent Failures by Functional Group



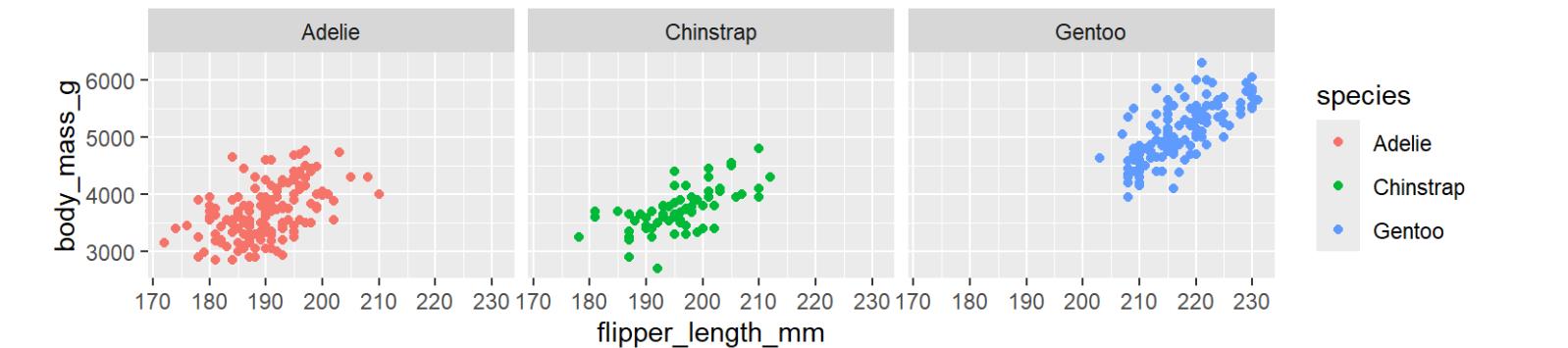
Facets • facet_wrap

- Split to subplots based on variable(s),
- Faceting in one dimension

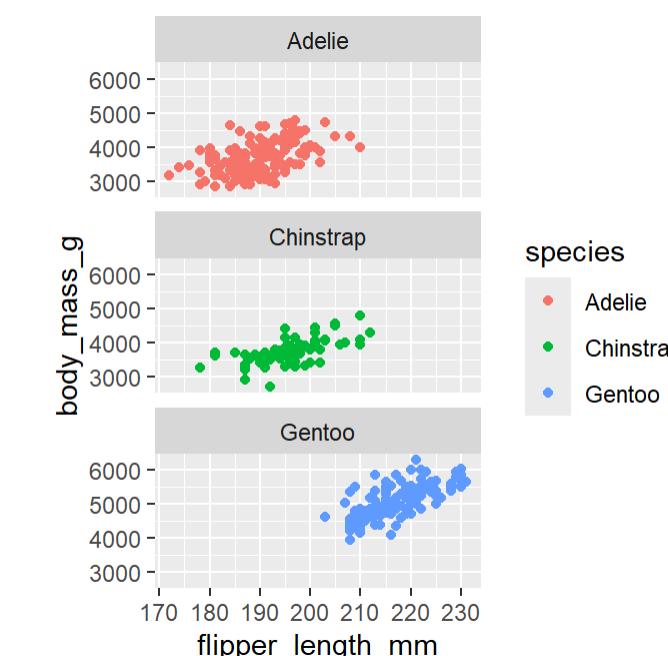
```
1 p <- ggplot(penguins) +
  geom_point(aes(x=flipper_length_mm,
                  y=body_mass_g,
                  color=species))
```



```
1 p + facet_wrap(~species)
```



```
1 p + facet_wrap(~species, nrow=3)
```



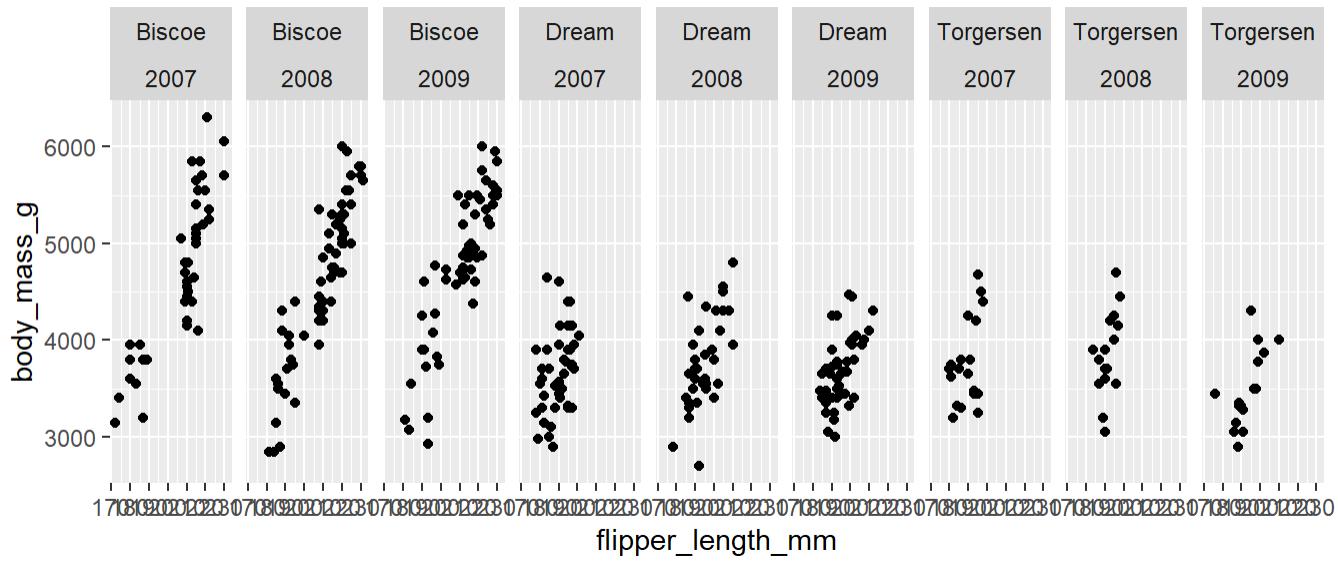
Facets • facet_grid

- Faceting in two dimensions

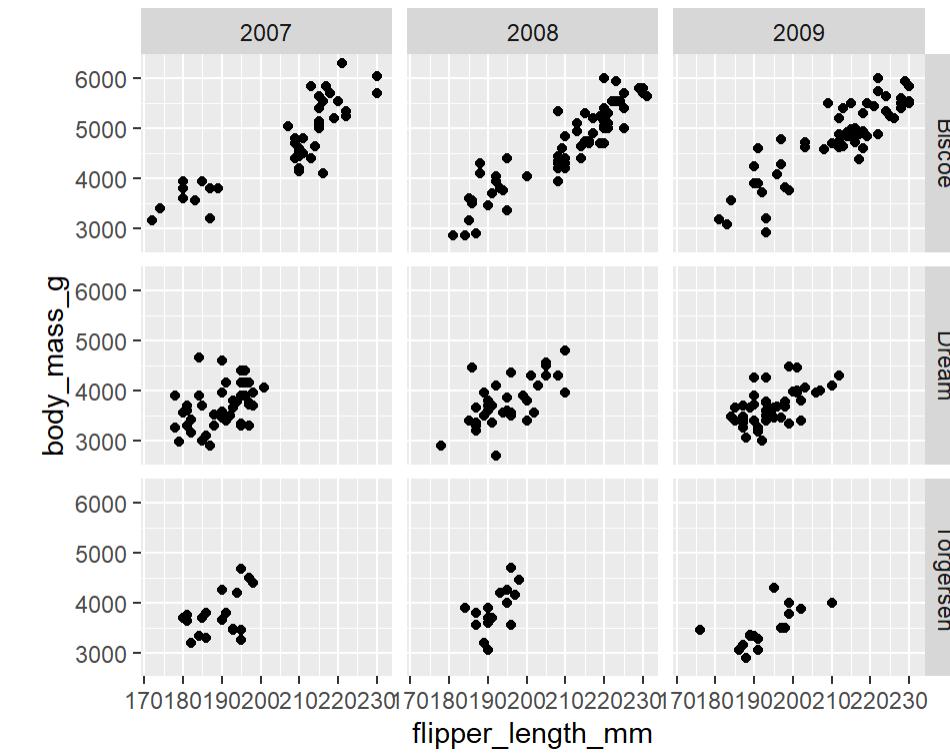
```

1 p <- ggplot(data = penguins, aes(x=flipper_length_mm,
2                                     y=body_mass_g))+
3   geom_point()
4 p + facet_grid(~island+year)

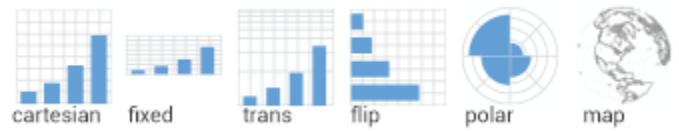
```



```
1 p + facet_grid(island~year)
```



Coordinate Systems



- `coord_cartesian(xlim=c(2,8))` for zooming in
- `coord_map` for controlling limits on maps
- `coord_polar` for polar coordinates

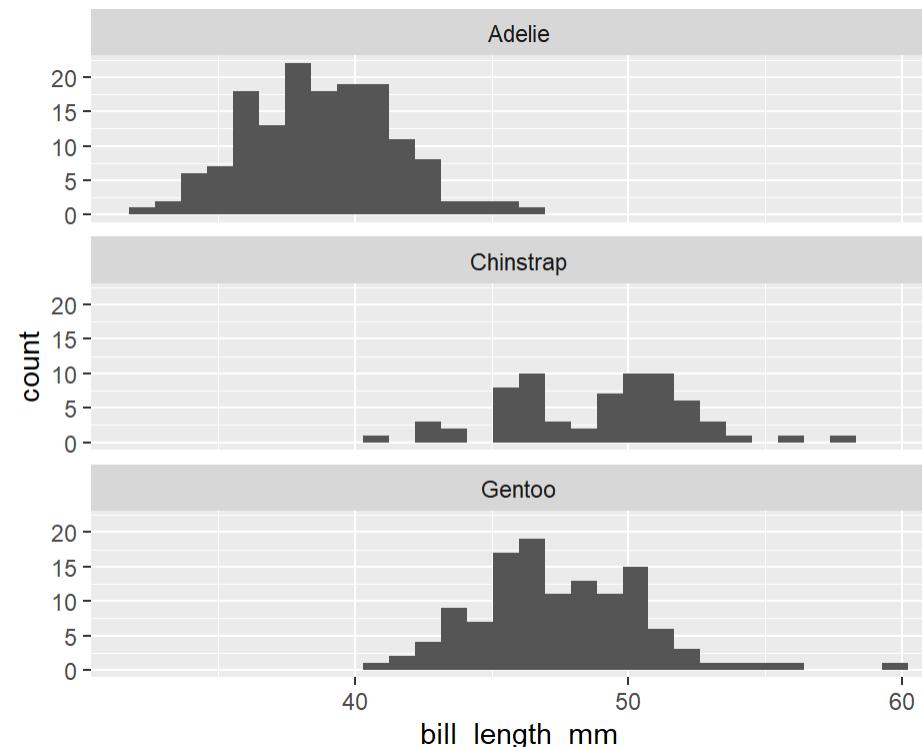
Theming

- Modify non-data plot elements/appearance
- Axis labels, panel colors, legend appearance etc
- Save a particular appearance for reuse
- `?theme`

```

1 ggplot(penguins, aes(x=bill_length_mm)) +
2   geom_histogram() +
3   facet_wrap(~species, ncol = 1) +
4   theme_grey()

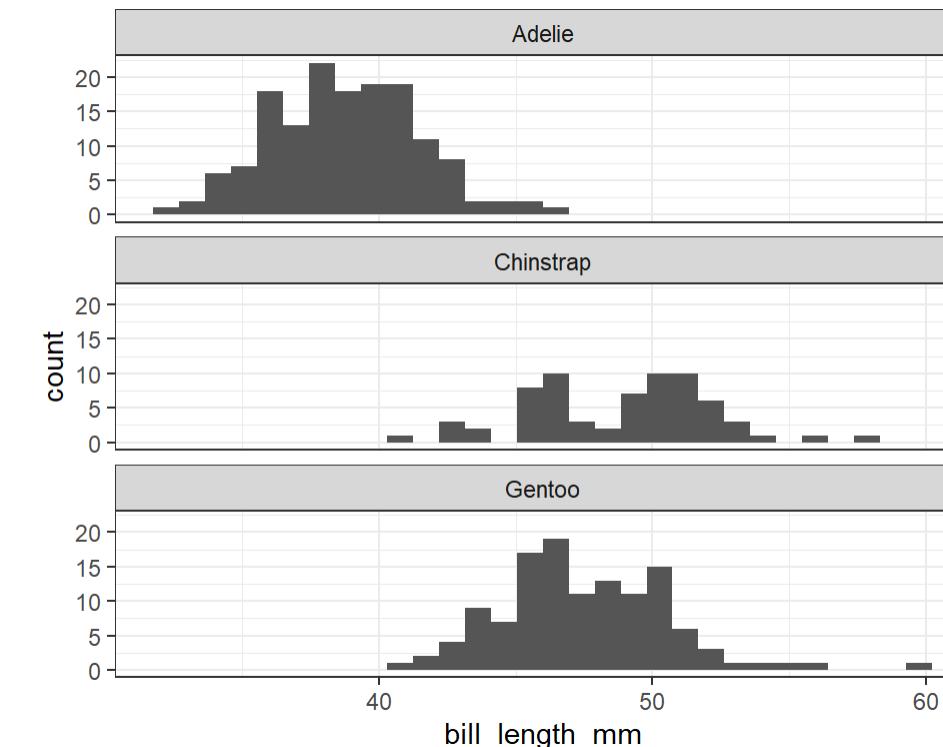
```



```

1 ggplot(penguins, aes(x=bill_length_mm)) +
2   geom_histogram() +
3   facet_wrap(~species, ncol = 1) +
4   theme_bw()

```

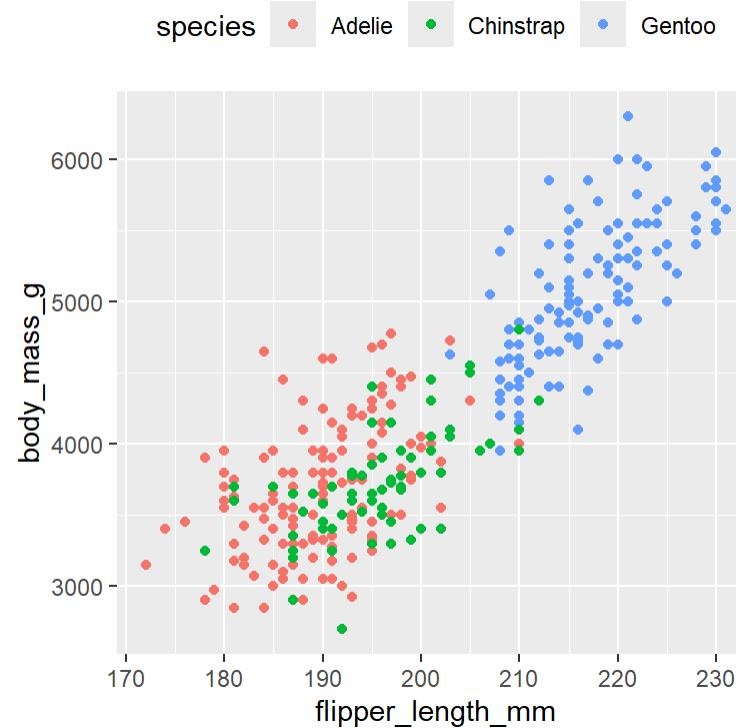


Theme • Legend

```
1 p <- ggplot(penguins) +
  geom_point(aes(x=flipper_length_mm,
                  y=body_mass_g,
                  color=species))
```

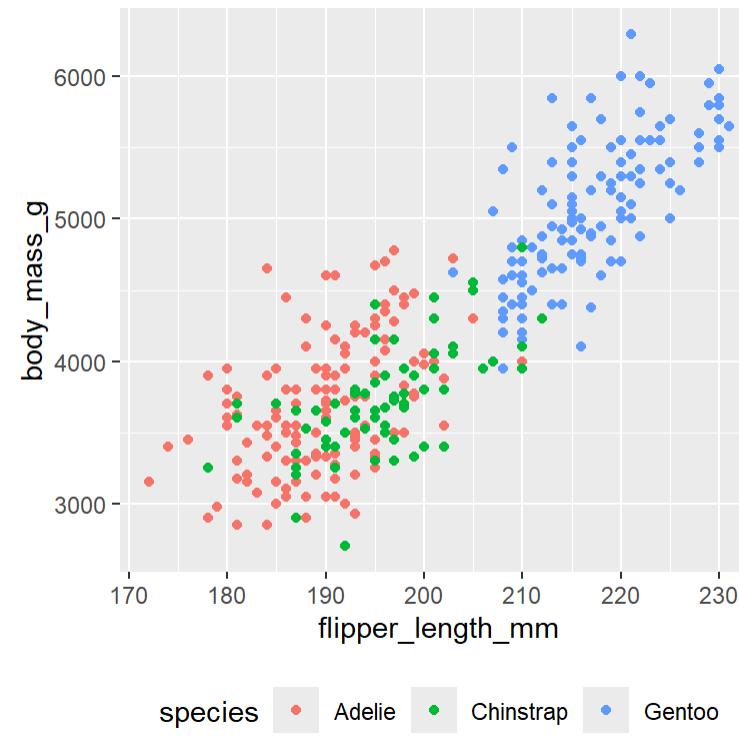
at top

```
1 p + theme(legend.position="top")
```



at bottom

```
1 p + theme(legend.position="bottom")
```

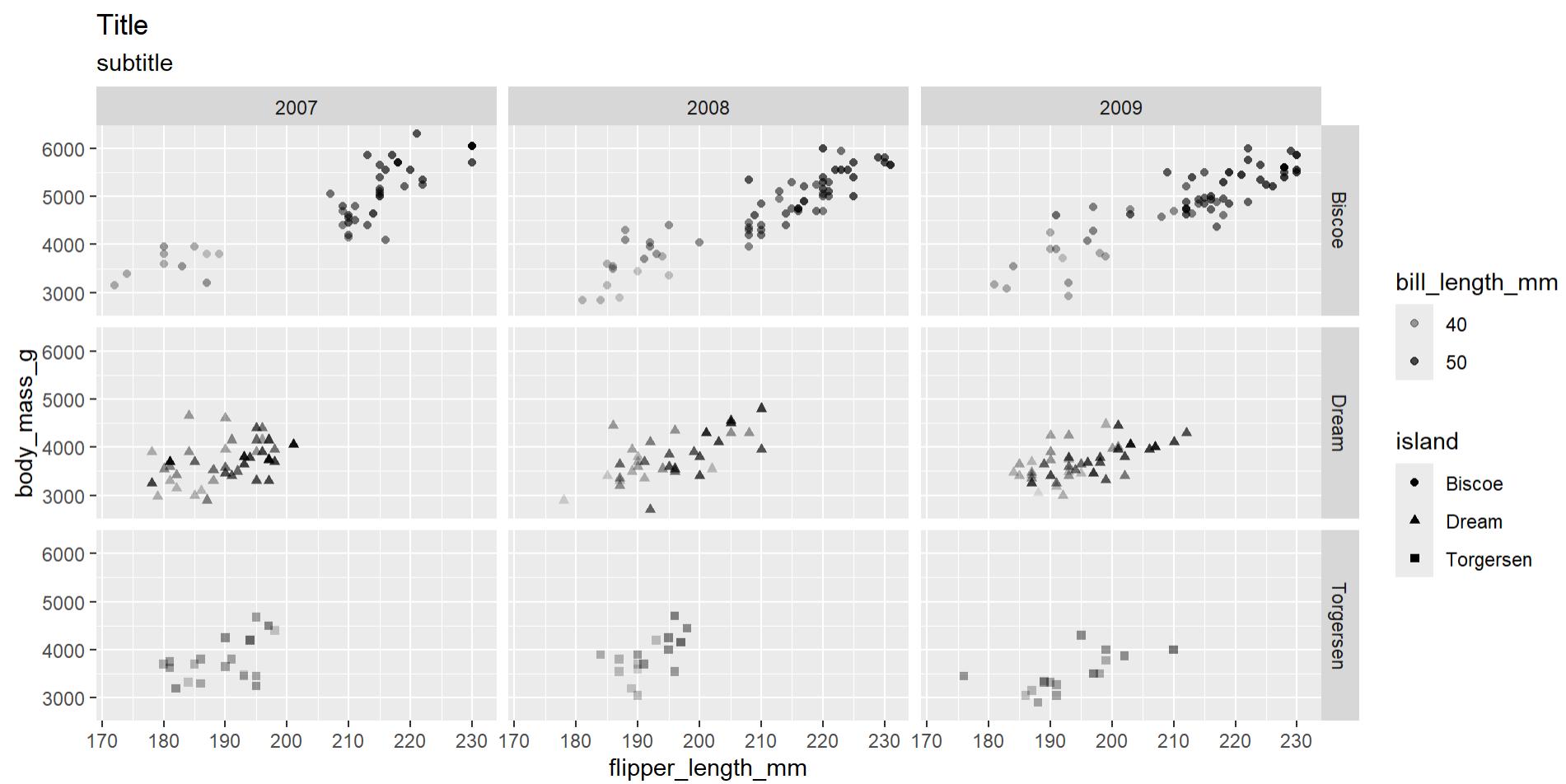


Theme • Text

```

1 p <- ggplot(penguins,
2   aes(x = flipper_length_mm,
3     y = body_mass_g,
4     alpha = bill_length_mm,
5     shape = island)) +
6   geom_point() +
7   facet_grid(island~year) +
8   labs(title="Title",
9     subtitle="subtitle")
10 p

```

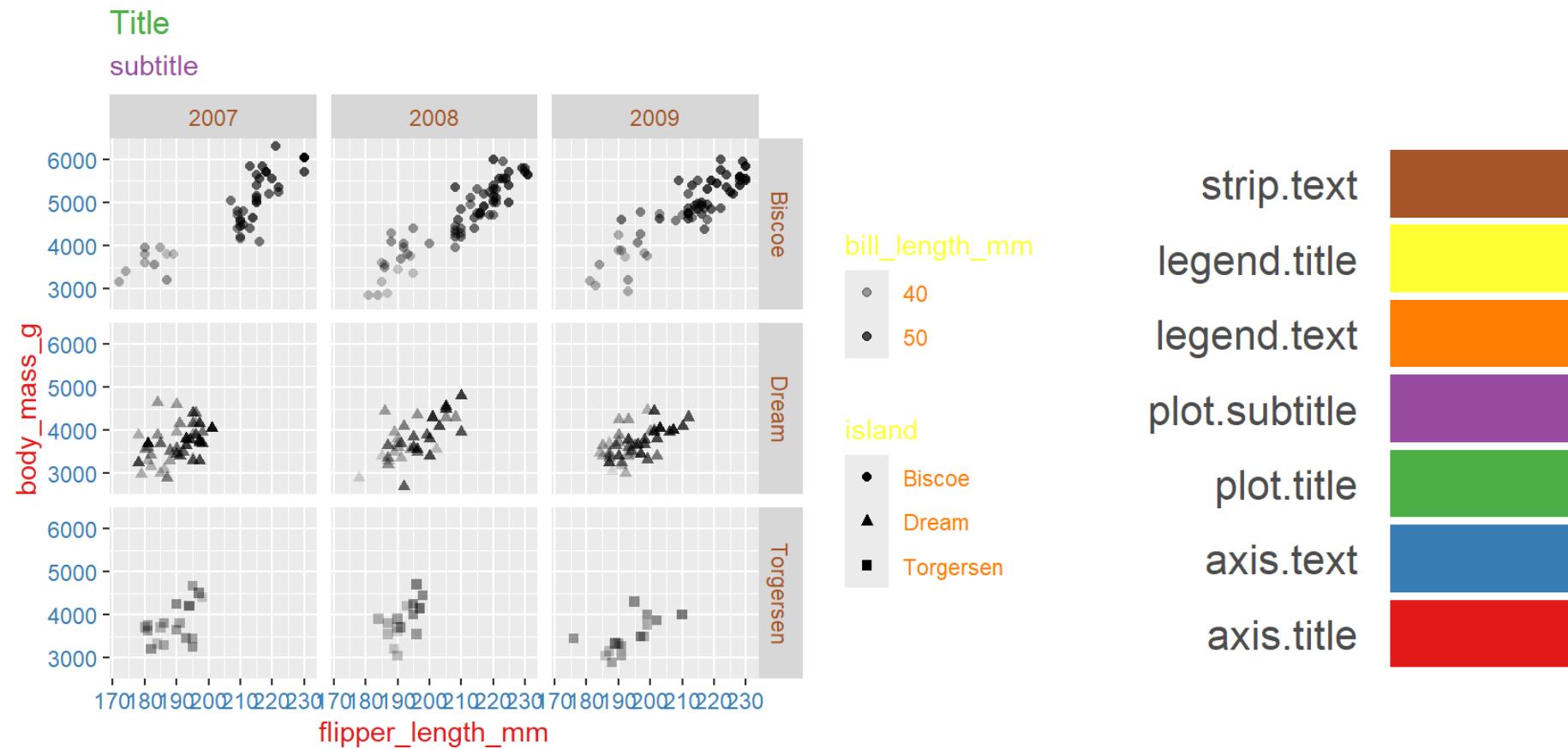


Theme • Text

```

1 p <- p + theme(
2   axis.title=element_text(color="#e41a1c"),
3   axis.text=element_text(color="#377eb8"),
4   plot.title=element_text(color="#4daf4a"),
5   plot.subtitle=element_text(color="#984ea3"),
6   legend.text=element_text(color="#ff7f00"),
7   legend.title=element_text(color="#ffff33"),
8   strip.text=element_text(color="#a65628")
9 )

```

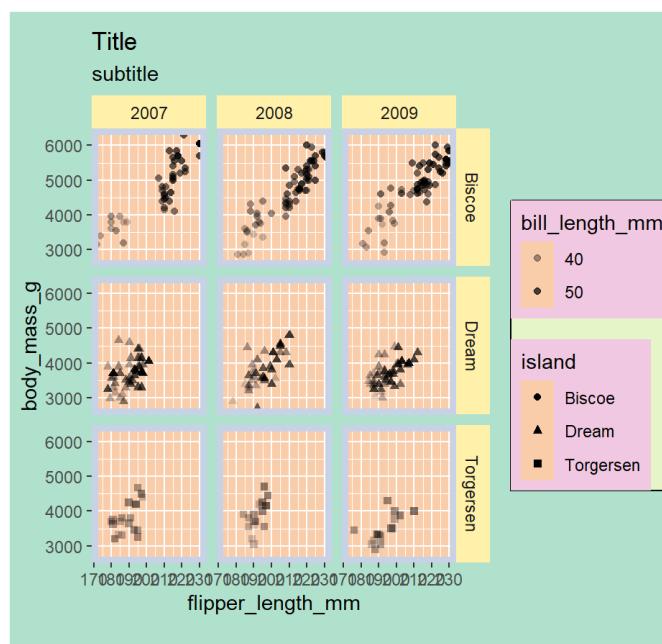


Theme • Rect

List of 5

```
$ fill      : NULL
$ colour    : NULL
$ linewidth : NULL
$ linetype   : NULL
$ inherit.blank: logi FALSE
- attr(*, "class")= chr [1:2] "element_rect" "element"
```

```
1 p <- p + theme(
2   plot.background=element_rect(fill="#b3e2cd"),
3   panel.background=element_rect(fill="#fdcdac"),
4   panel.border=element_rect(fill=NA,color="#cbd5e8",size=3),
5   legend.background=element_rect(fill="#f4cae4"),
6   legend.box.background=element_rect(fill="#e6f5c9"),
7   strip.background=element_rect(fill="#fff2ae")
8 )
```



strip.background	
legend.box.background	
legend.background	
panel.border	
panel.background	
plot.background	

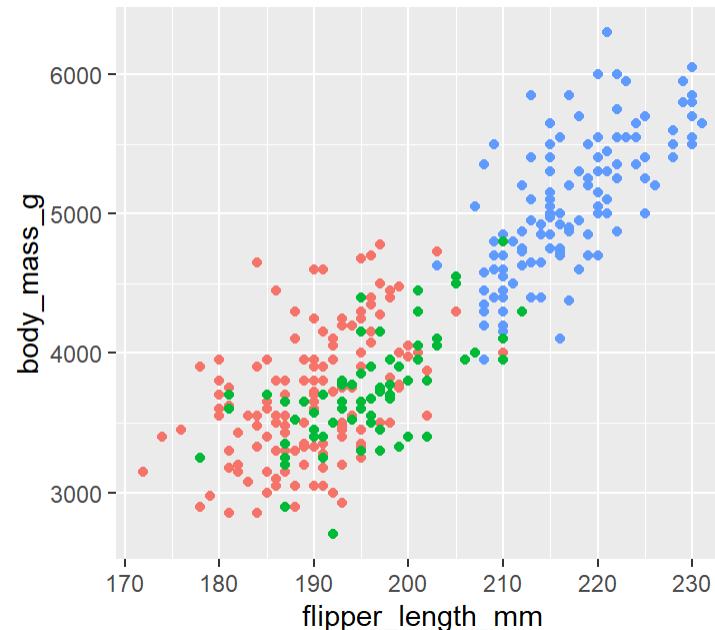
Theme • Reuse

```

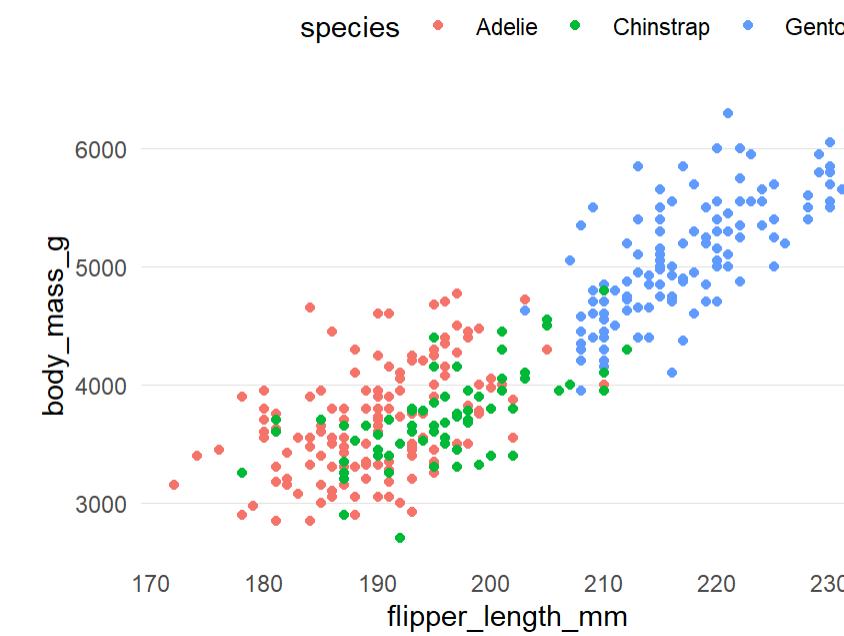
1 newtheme <- theme_bw() + theme(
2   axis.ticks=element_blank(), panel.background=element_rect(fill="white"),
3   panel.grid.minor=element_blank(), panel.grid.major.x=element_blank(),
4   panel.grid.major.y=element_line(size=0.3,color="grey90"), panel.border=element_blank(),
5   legend.position="top", legend.justification="right"
6 )

```

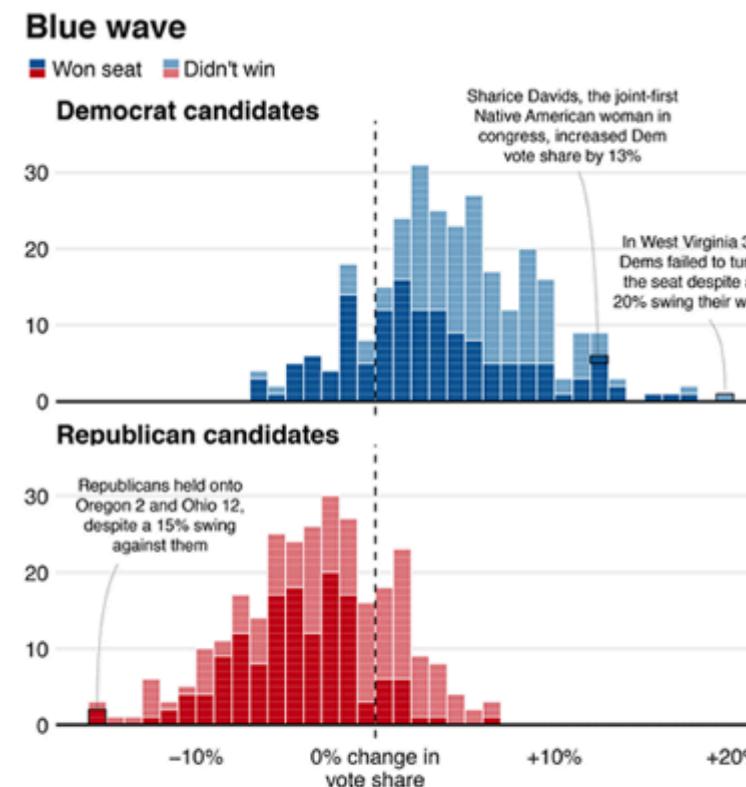
1 p



1 p + newtheme

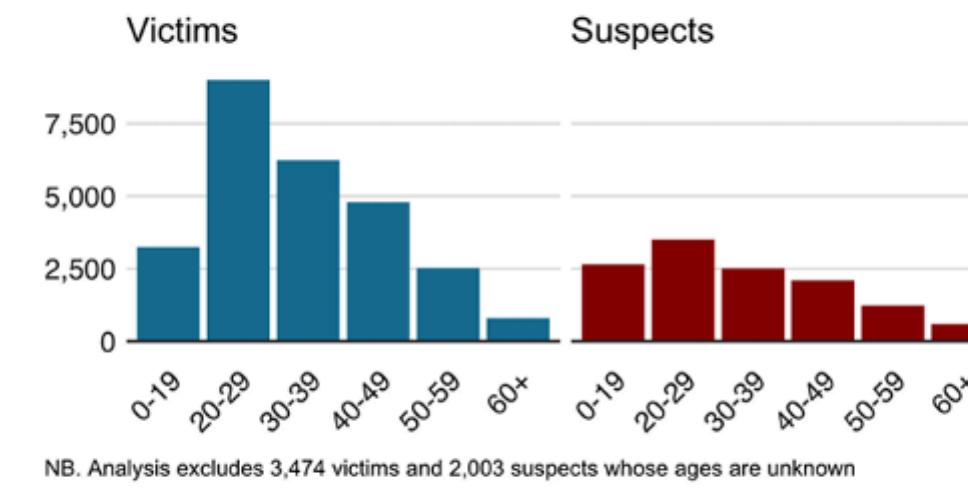


Professional themes

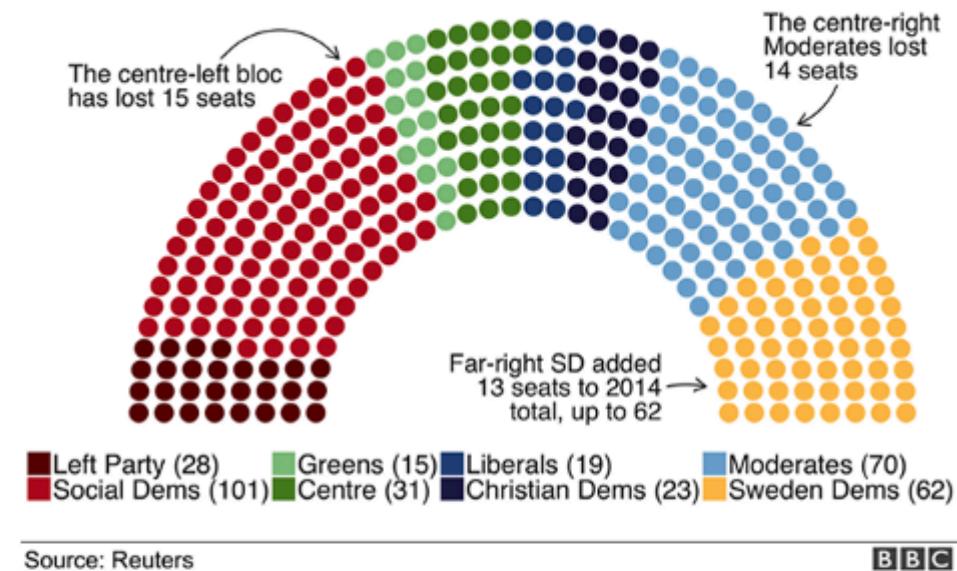


Homophobic hate crimes are mainly committed by young people on young people

Number in each age group 2014 - 2017



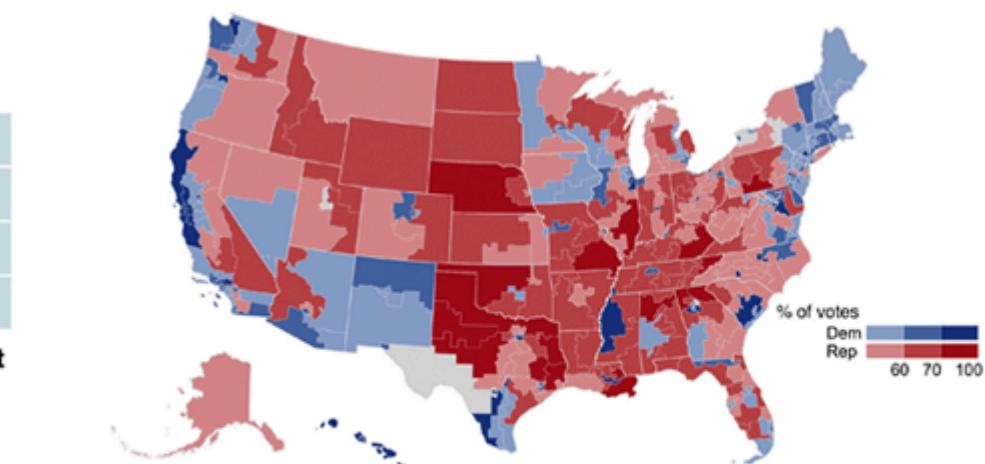
Results of the 2018 election



BBC

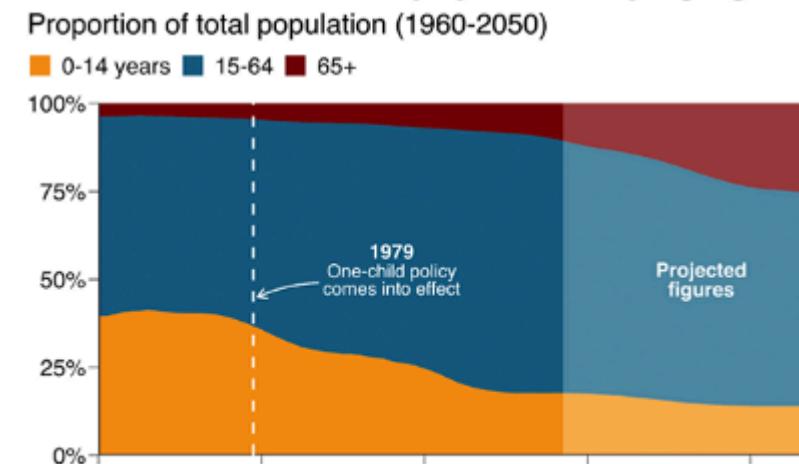
Democrats take the House

Dem 232 Rep 198

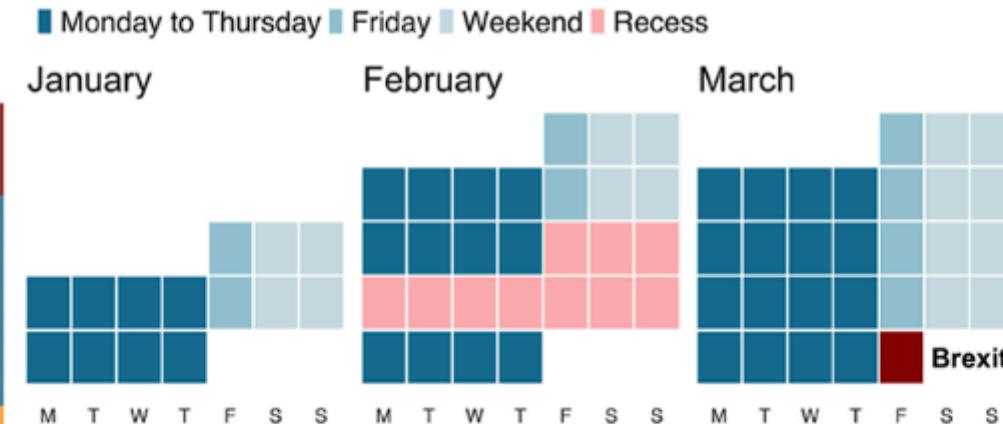


BBC

Breakdown of China's population by age group



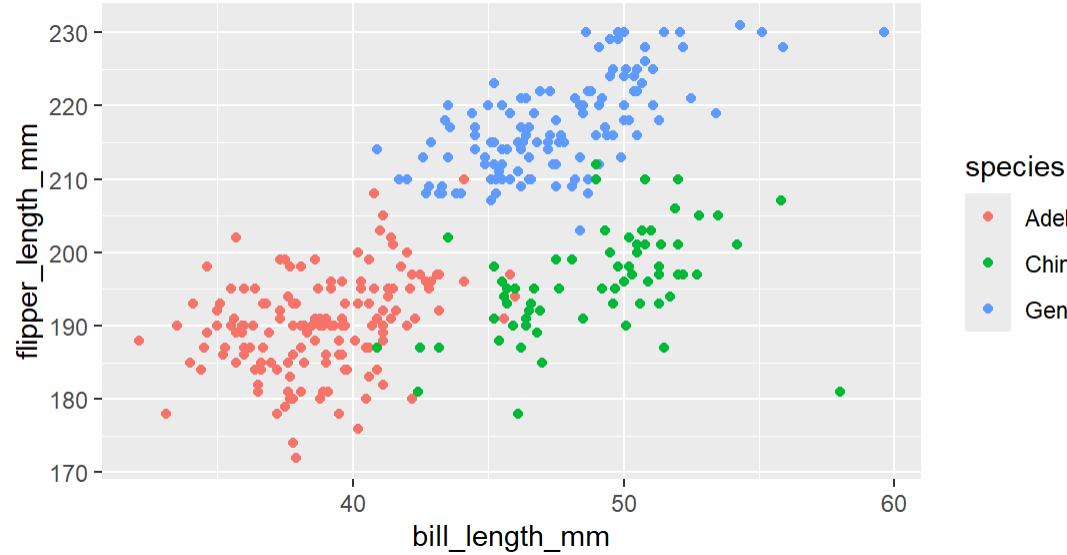
The Commons has 36 normal working days until Brexit



How BBC works with R graphics

Saving plots

```
1 p <- ggplot(penguins,aes(x=bill_length_mm,y=flipper_length_mm,color=species)) +
2   geom_point()
3 p
```



- **ggplot2** package offers a convenient function

```
1 ggsave("plot.png",p,height=5,width=7,units="cm",dpi=200)
2 # Note that default units in png is pixels while in ggsave it's inches
```

- **ggplot2** plots can be saved just like base plots

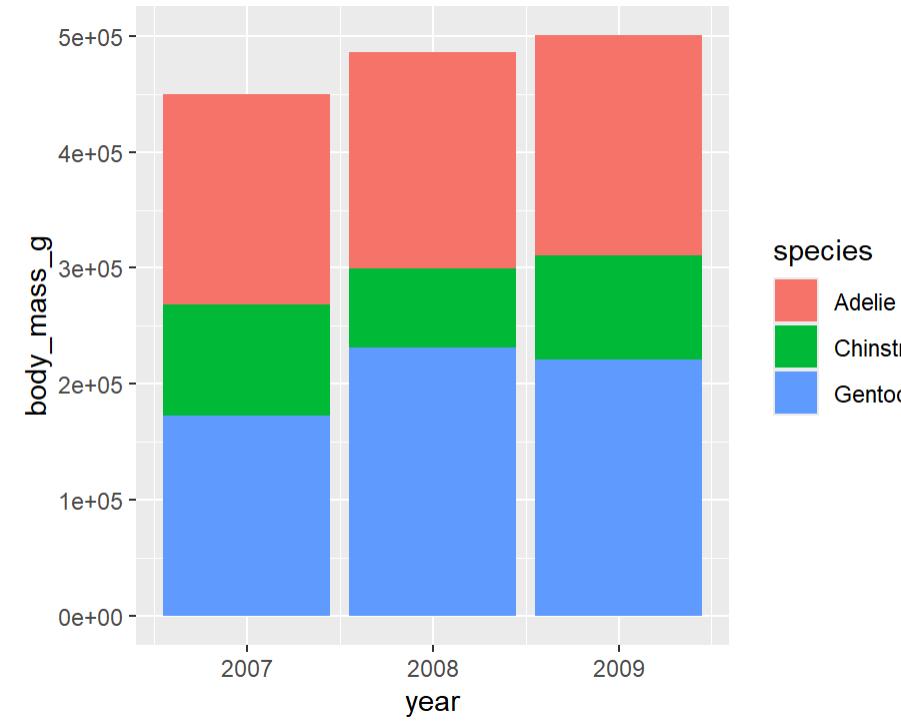
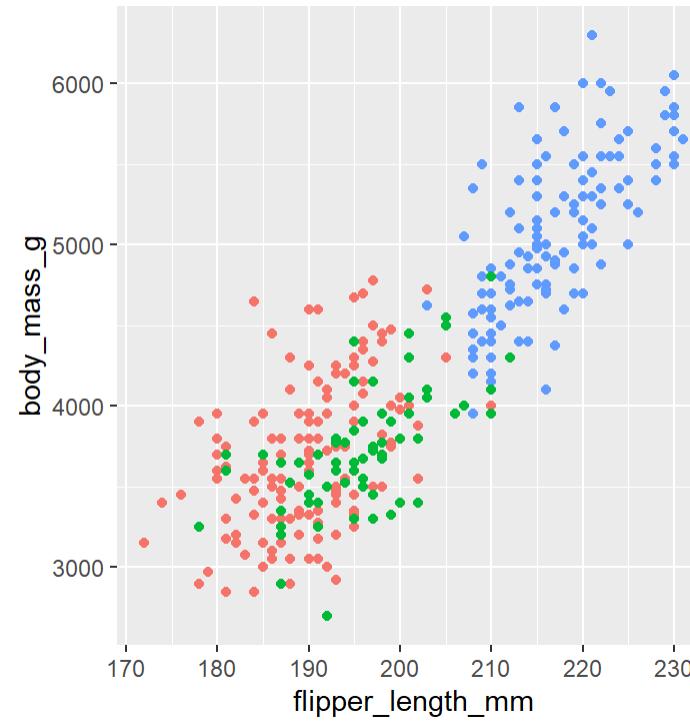
```
1 png("plot.png",height=5,width=7,units="cm",res=200)
2 print(p)
3 dev.off()
```

Combining Plots

```

1 p <- ggplot(penguins, aes(x=flipper_length_mm, y=body_mass_g, color=species)) + geom_point()
2 q <- ggplot(penguins, aes(x=year, y=body_mass_g, fill=species)) + geom_bar(stat="identity")
1 patchwork::wrap_plots(p,q)

```



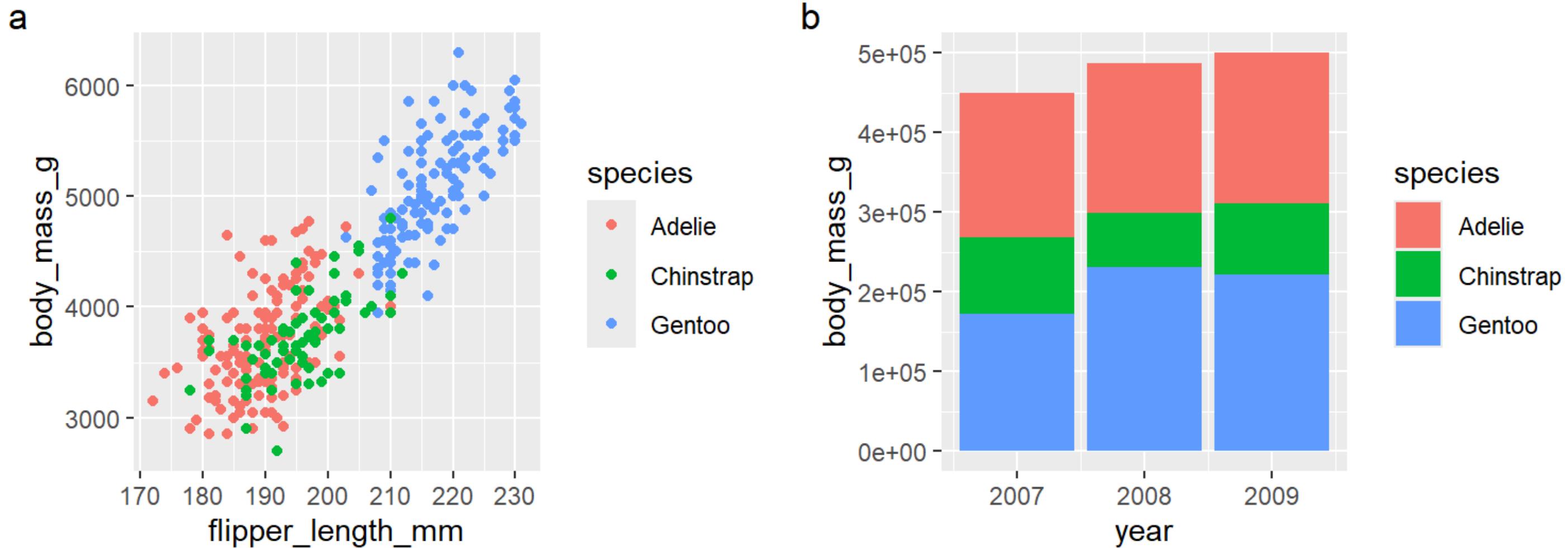
Combining Plots

```

1 p <- ggplot(penguins, aes(x=flipper_length_mm, y=body_mass_g, color=species)) + geom_point()
2 q <- ggplot(penguins, aes(x=year, y=body_mass_g, fill=species)) + geom_bar(stat="identity")

1 patchwork::wrap_plots(p,q) +
2 plot_annotation(tag_levels = 'a')

```



[patchwork documentation.](#)

Extensions

- **patchwork**: Combining plots
- **ggrepel**: Text labels including overlap control
- **ggforce**: Circles, splines, hulls, voronoi etc
- **ggpmisc**: Miscellaneous features
- **ggthemes**: Set of extra themes
- **ggthemr**: More themes
- **ggsci**: Color palettes for scales
- **ggmap**: Dedicated to mapping
- **ggraph**: Network graphs
- **ggiraph**: Converting ggplot2 to interactive graphics

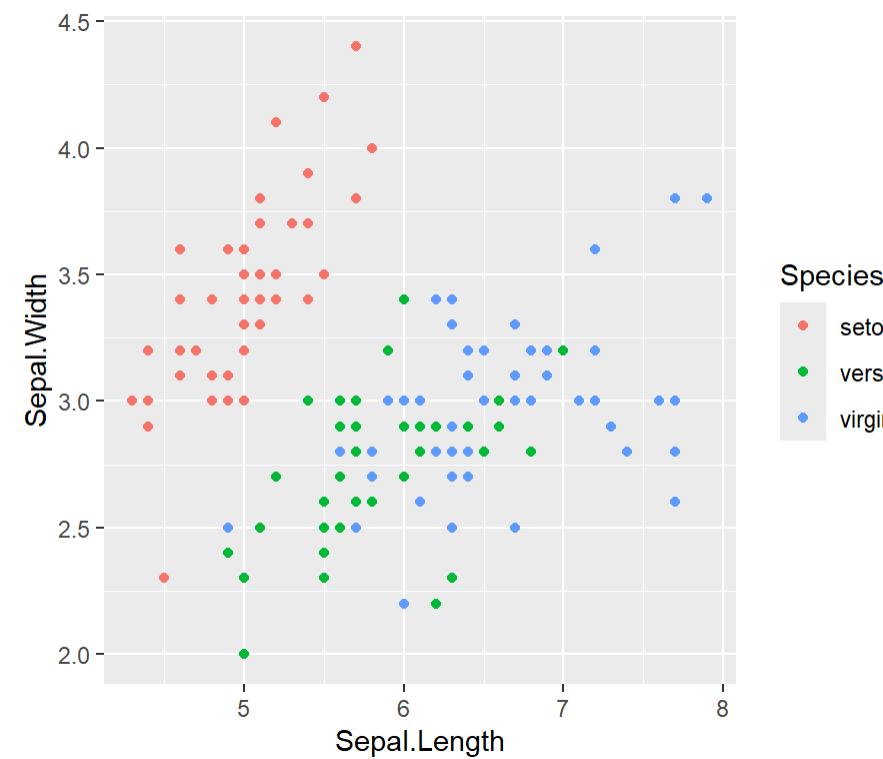
A collection of ggplot extension packages: <https://exts.ggplot2.tidyverse.org/>.

Curated list of ggplot2 links: <https://github.com/erikgahner/awesome-ggplot2>.

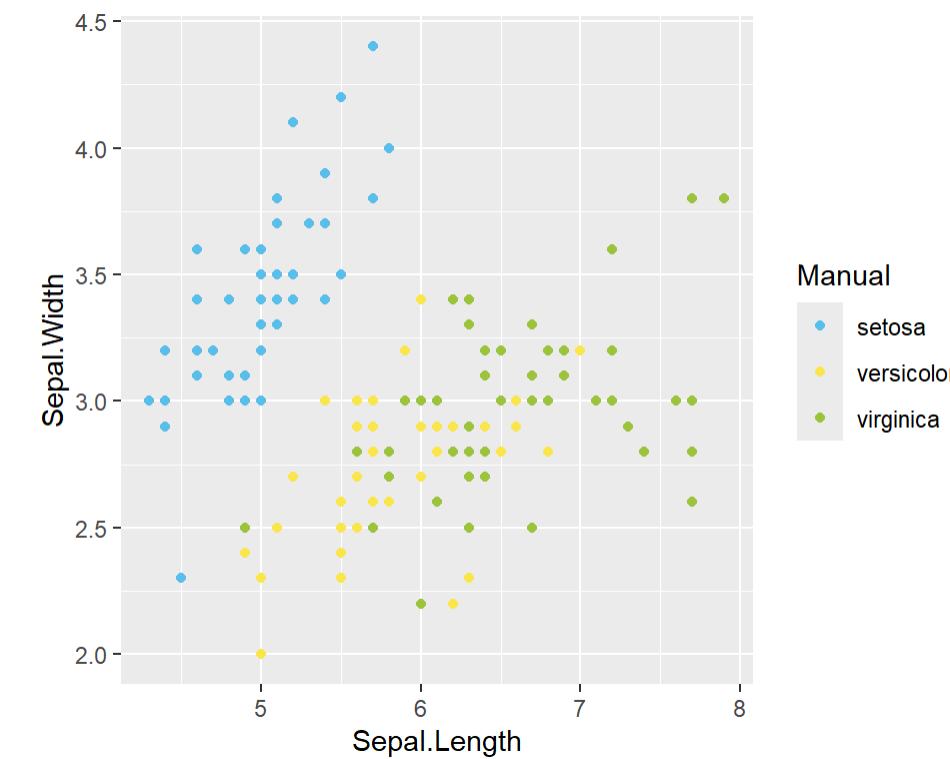
Scales • Discrete Colors

- scales: position, color, fill, size, shape, alpha, linetype
- syntax: `scale_<aesthetic>_<type>`

```
1 p <- ggplot(iris)+geom_point(aes(x=Sepal.Length,
2                               y=Sepal.Width,color=Species))
3 p
```



```
1 p + scale_color_manual(
2   name="Manual",
3   values=c ("#5BC0EB","#FDE74C","#9BC53D"))
```



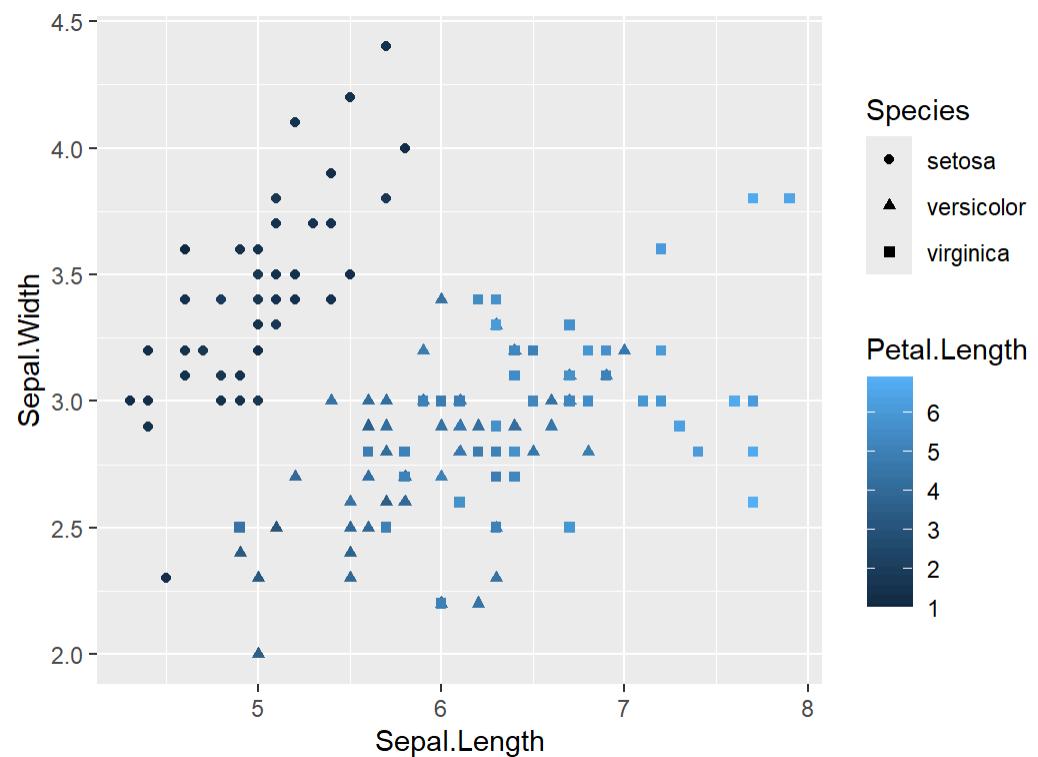
Scales • Continuous Colors

- In RStudio, type `scale_`, then press TAB

```

1 p <- ggplot(iris) +
2   geom_point(aes(x=Sepal.Length,
3                 y=Sepal.Width,
4                 shape=Species, color=Petal.Length))
5 p

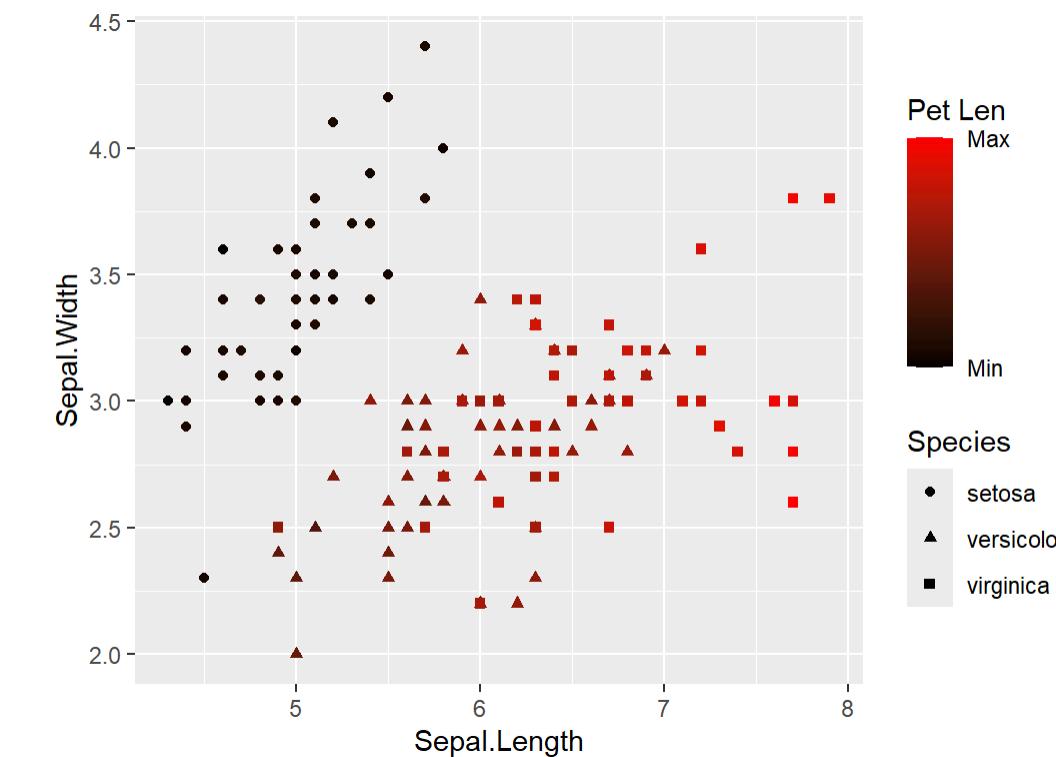
```



```

1 p +
2   scale_color_gradient(name="Pet Len",
3   breaks=range(iris$Petal.Length),
4   labels=c("Min","Max"),
5   low="black", high="red")

```

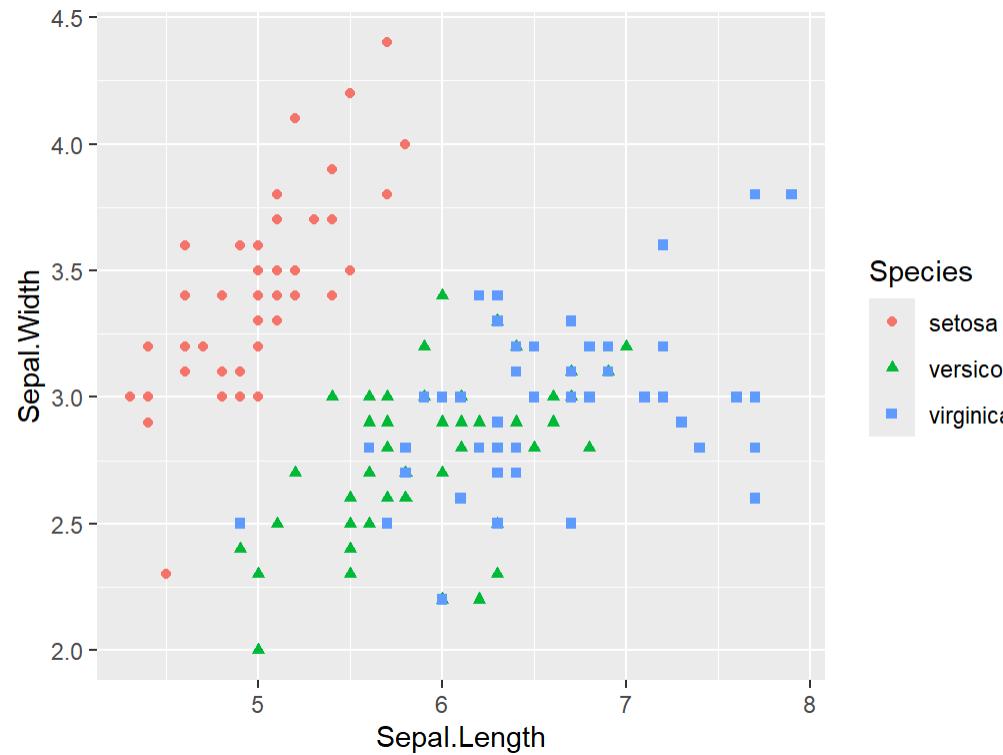


Scales • Shape

```

1 p <- ggplot(iris) +
2   geom_point(aes(x=Sepal.Length,
3                   y=Sepal.Width,
4                   shape=Species,color=Species))
5 p

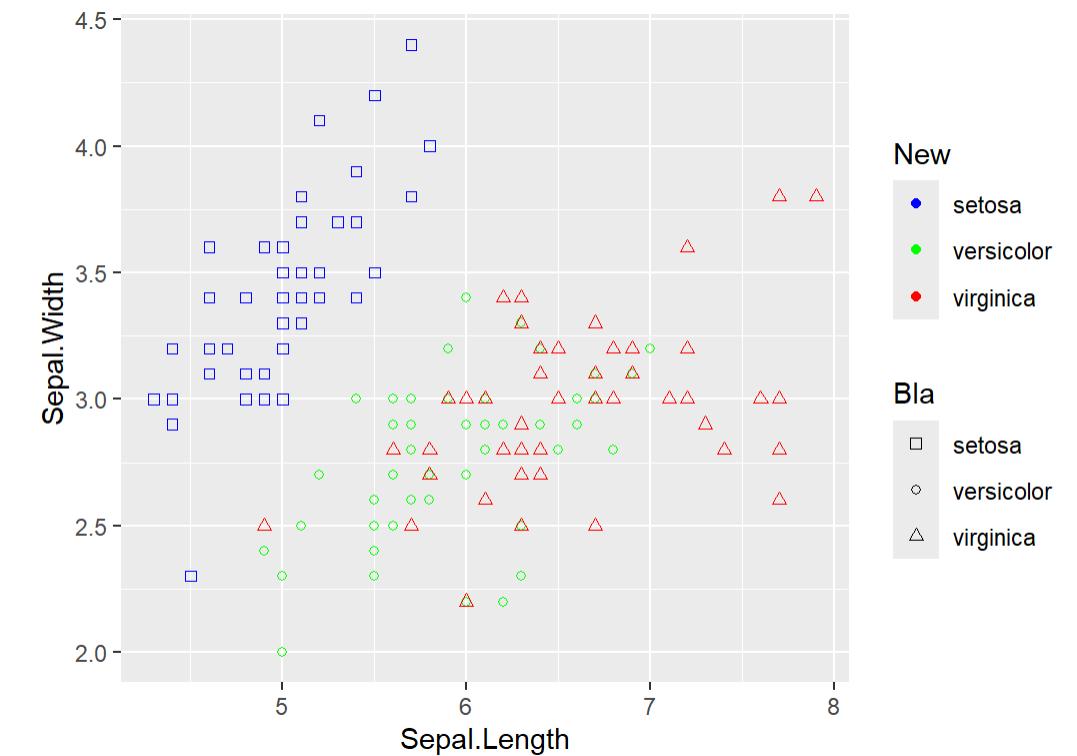
```



```

1 p +
2 scale_color_manual(name="New",
3                     values=c("blue","green","red")) +
4 scale_shape_manual(name="Bla",values=c(0,1,2))

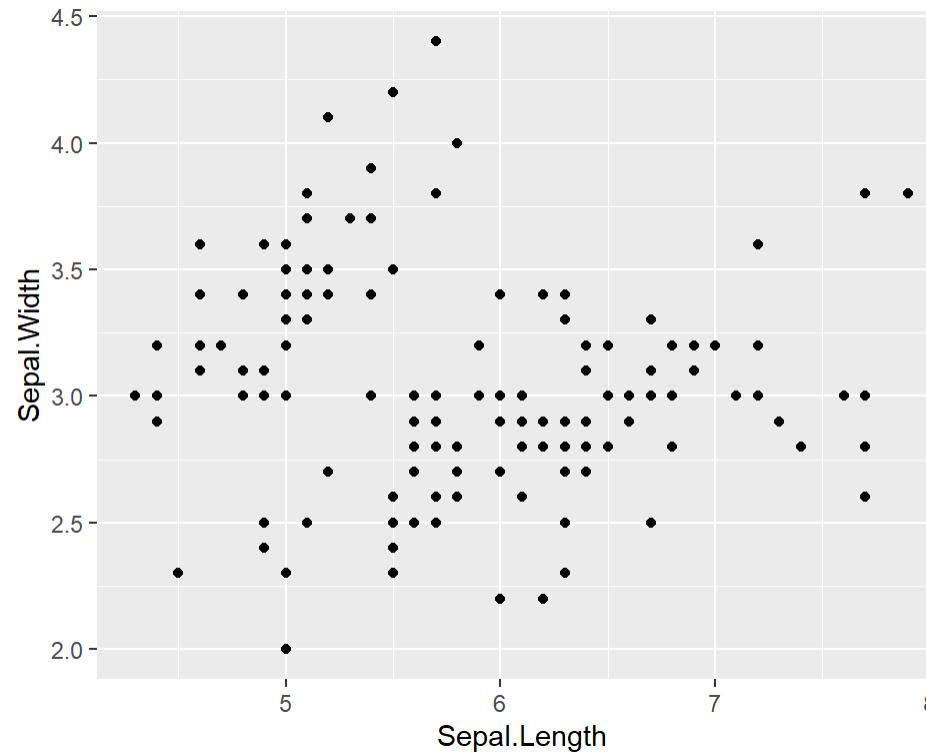
```



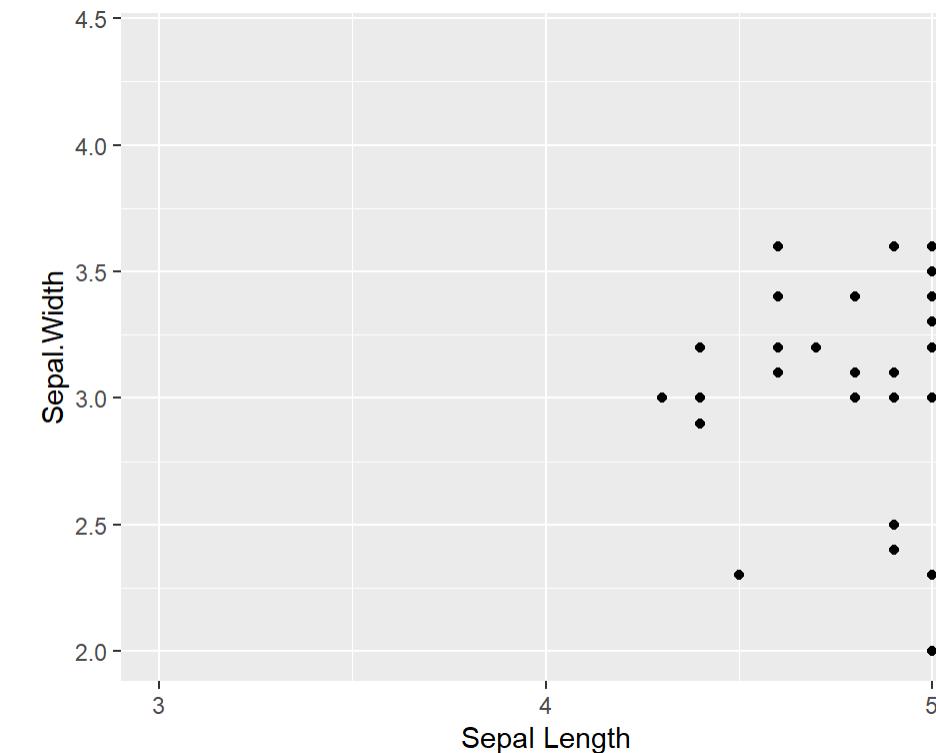
Scales • Axes

- scales: x, y
- syntax: `scale_<axis>_<type>`
- arguments: name, limits, breaks, labels

```
1 p <- ggplot(iris)+geom_point(  
2   aes(x=Sepal.Length,y=Sepal.Width))  
3 p
```

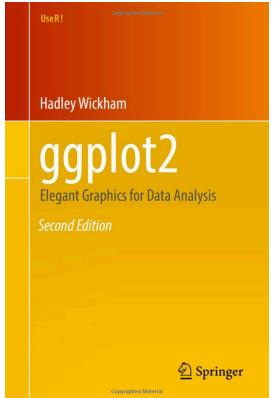


```
1 p + scale_x_continuous(name="Sepal Length",  
2                           breaks=seq(1,8),limits=c(3,5))
```

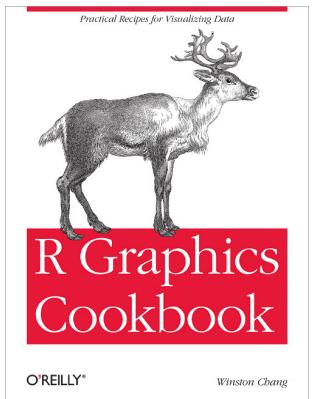


Help

- **ggplot2 book**



- **The R cookbook**



- **ggplot2 official reference**
- **RStudio cheatsheet**
- **r-statistics ggplot2 cheatsheet**
- **StackOverflow**
- **Blogs, R-Bloggers, Cedric Scherer etc.**

Thank you! Questions?



Acknowledgements:

- SLUBI • 3Bs • Slides adapted from RaukR • GPL-3 License