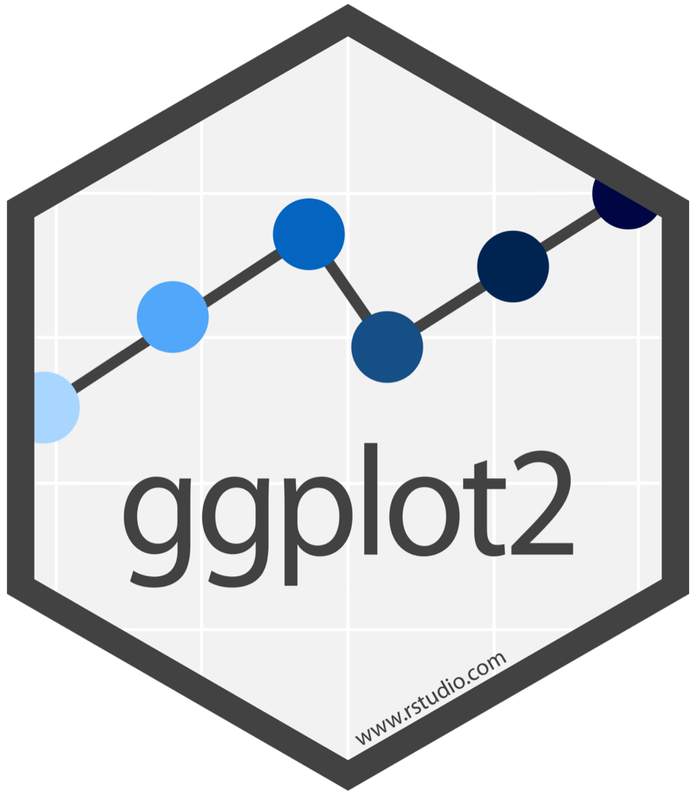
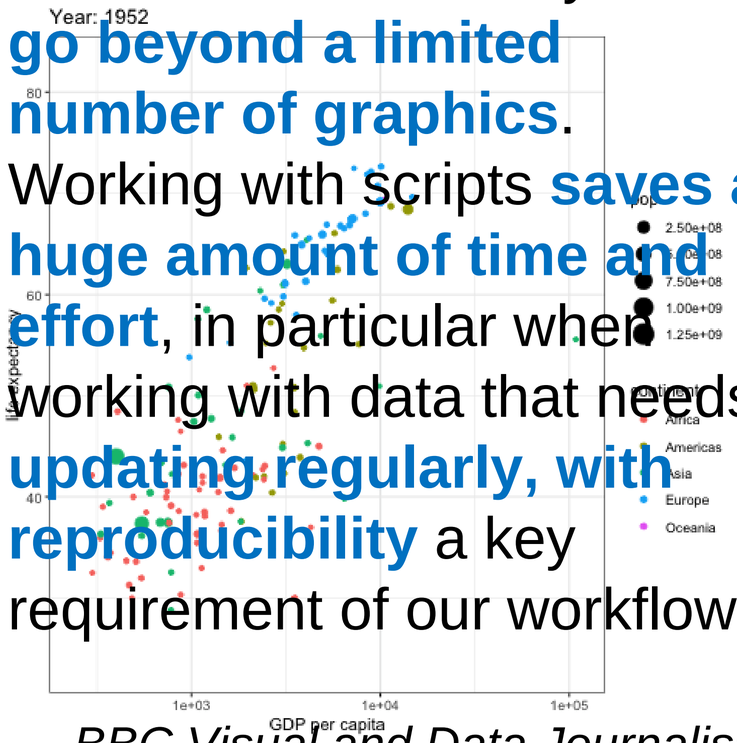
Visualize Data with



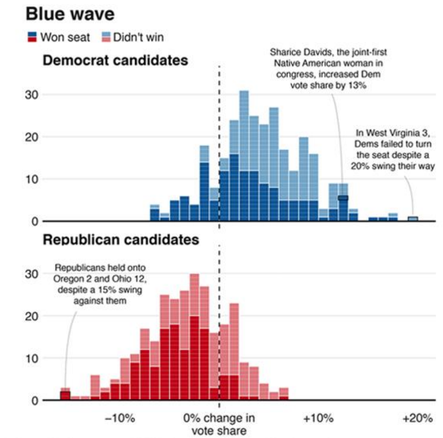
What can ggplot do?

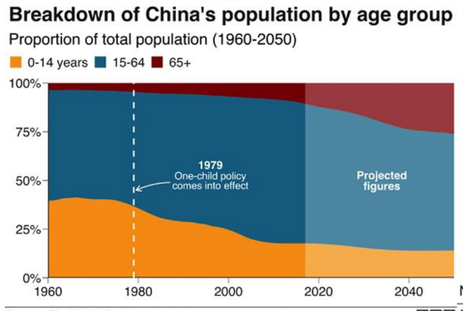
“ggplot2 gives you far more control and creativity than a chart tool and allows you to **go beyond a limited number of graphics**. Working with scripts **saves a huge amount of time and effort**, in particular when working with data that needs **updating regularly, with reproducibility** a key requirement of our workflow.”

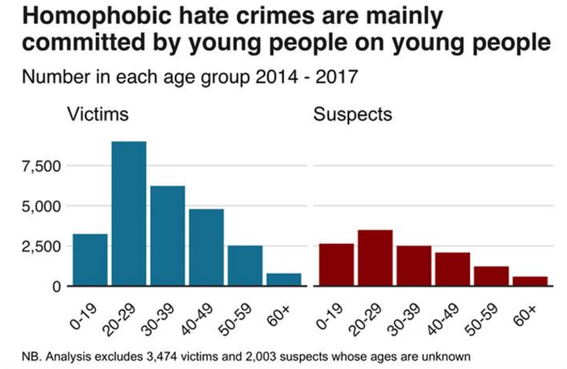


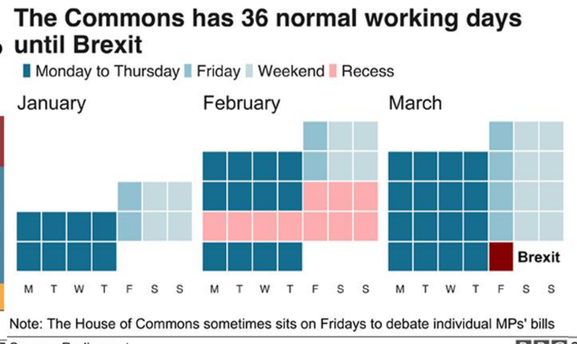
-BBC Visual and Data Journalism

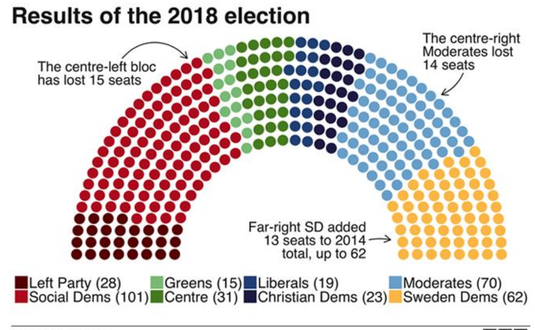
How the BBC Visual and Data Journalism team works with graphics in R

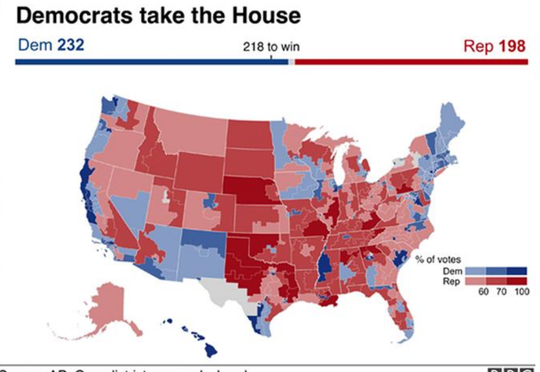












"The simple graph has brought more information to the data analyst’s mind than any other device. "

- John Tukey

genome

Basic data for all human chromosomes.

```
genome
```

	chromosome	length_mm	basepairs	variations	protein_codingg~	pseudo_genes	totallongnc_rna	totalsmallnc_rna	mi_rna	r_rna	sn_rna
	<chr>	<dbl>	<dbl>	<dbl>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
1	1	85	248956422	12151146	2058	1220	1200	496	134	66	221
2	2	83	242193529	12945965	1309	1023	1037	375	115	40	161
3	3	67	198295559	10638715	1078	763	711	298	99	29	138
4	4	65	190214555	10165685	752	727	657	228	92	24	120
5	5	62	181538259	9519995	876	721	844	235	83	25	106
6	6	58	170805979	9130476	1048	801	639	234	81	26	111
7	7	54	159345973	8613298	989	885	605	208	90	24	90
8	8	50	145138636	8221520	677	613	735	214	80	28	86
9	9	48	138394717	6590811	786	661	491	190	69	19	66
10	10	46	133797422	7223944	733	568	579	204	64	32	87

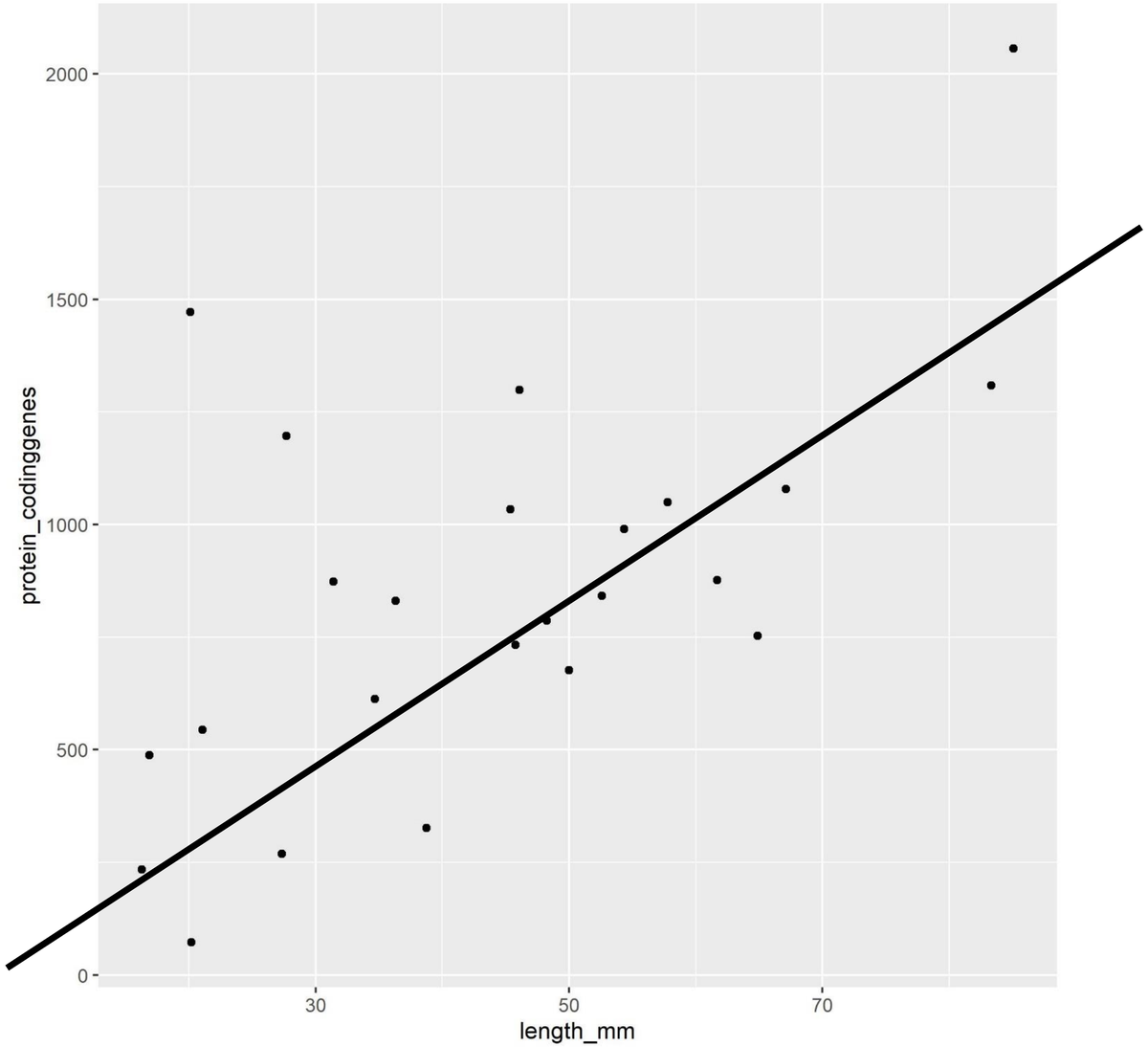
Question

What relationship do you expect to see between chromosome size (length_mm) and number of genes (protein_codinggenes)?

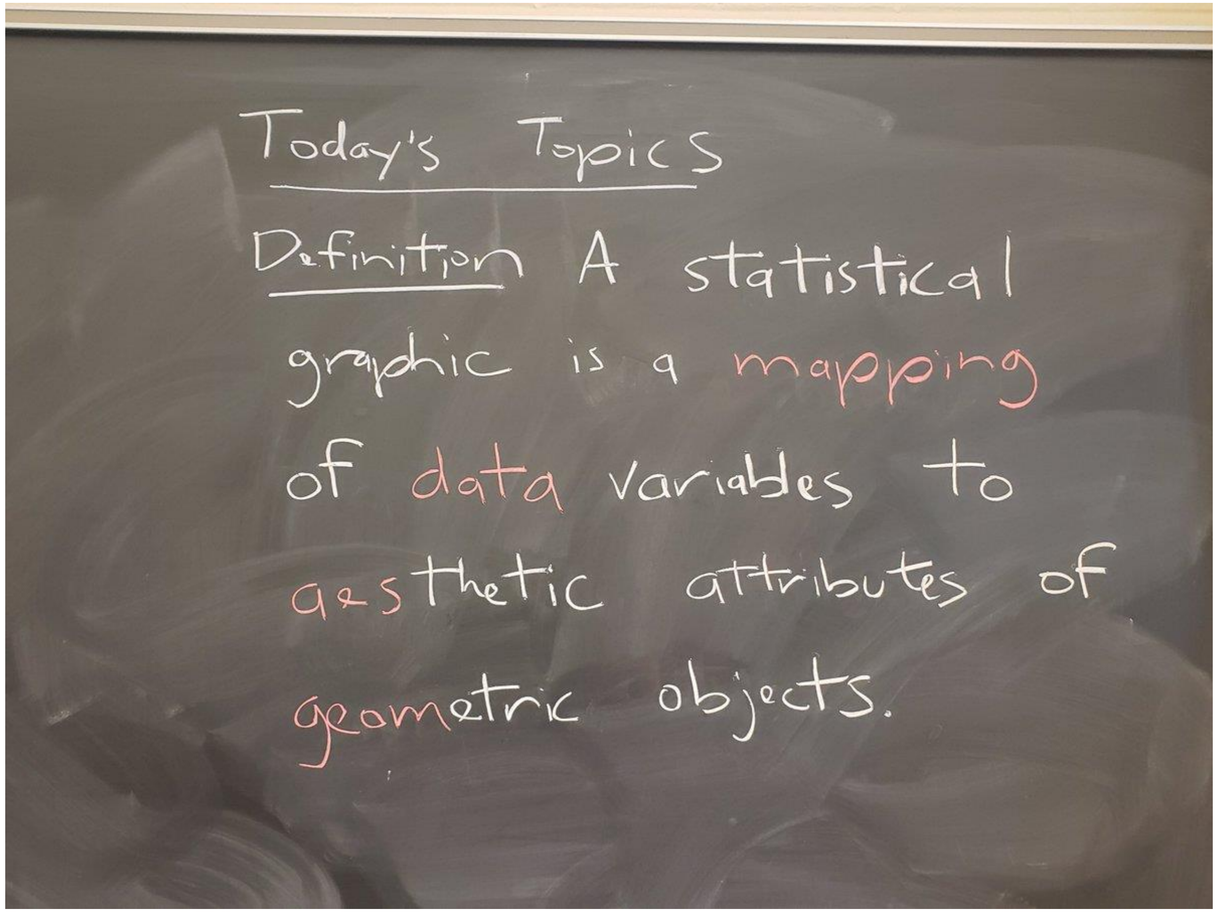
Exercise 1

Run this code in **02-Visualize-Exercises.Rmd** to make a graph. Pay strict attention to spelling, capitalization, and parentheses!

```
ggplot(data = genome) +
  geom_point(mapping=aes(x = length_mm, y = protein_codinggenes))
```




```
ggplot(data = genome) +
   geom_point(mapping=aes(x = length_mm, y = protein_codinggenes))
```



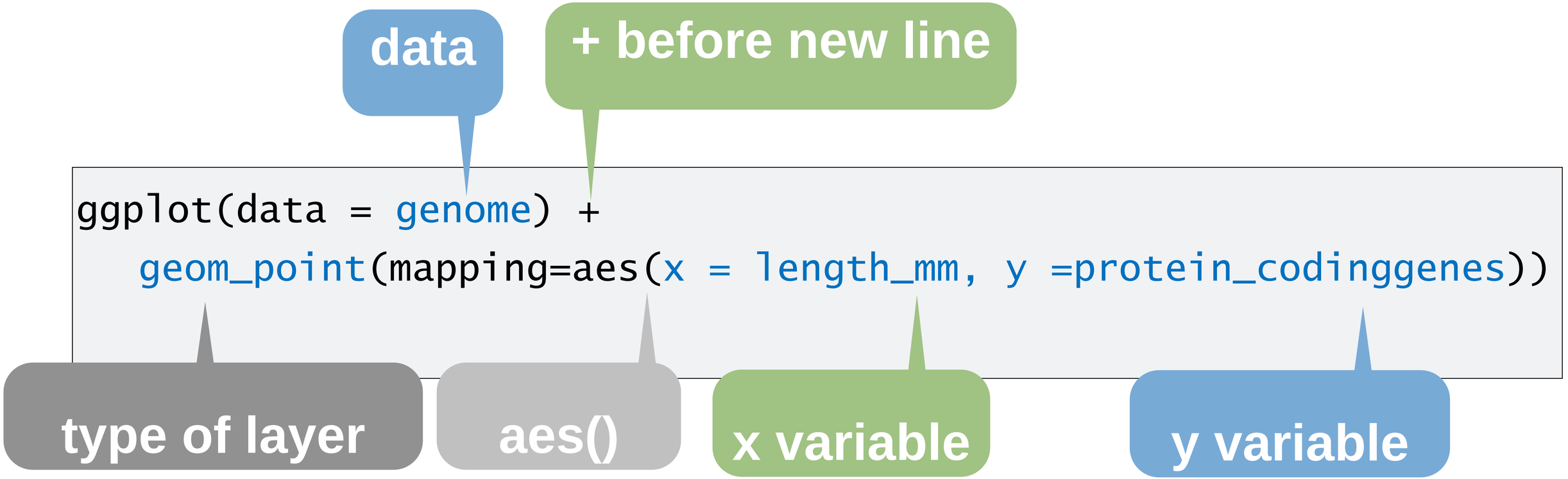

```
ggplot(data = genome) +
   geom_point(mapping=aes(x = length_mm, y = protein_codinggenes))
```

1. "Initialize" a plot with ggplot()
2. Add layers with geom_ functions

```
ggplot(data = genome) +
   geom_point(mapping=aes(x = length_mm, y =protein_codinggenes))
```

Pro tip: Always put the + at the end of a line, Never at the start

```
ggplot(data = genome) +
   geom_point(mapping=aes(x = length_mm, y =protein_codinggenes))
```


data
+ before new line
ggplot(data = genome) +
geom_point(mapping=aes(x = length_mm, y =protein_codinggenes))
type of layer
aes()
x variable
y variable

A template

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

```
  geom_point(mapping=aes(x = length_mm, y =protein_codinggenes))
```

A template

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

A template

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

A template

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

=

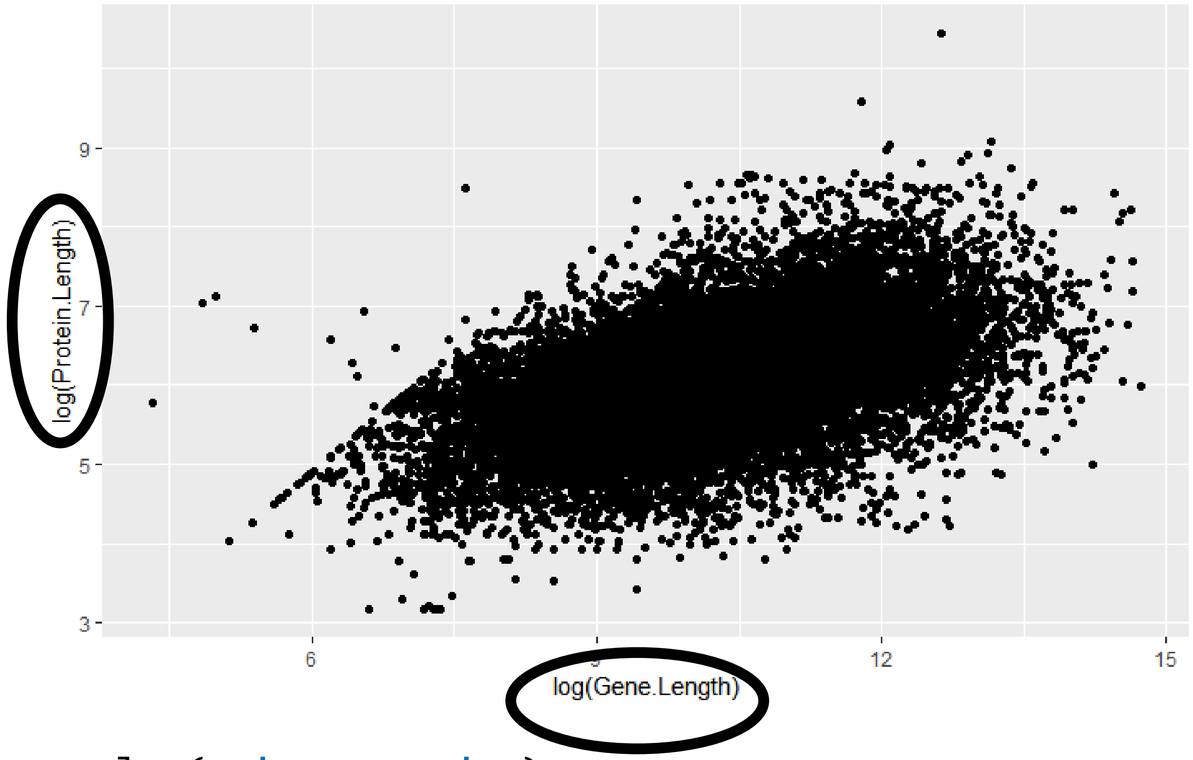
```
ggplot(<DATA>) +
  <GEOM_FUNCTION>(aes(<MAPPINGS>))
```

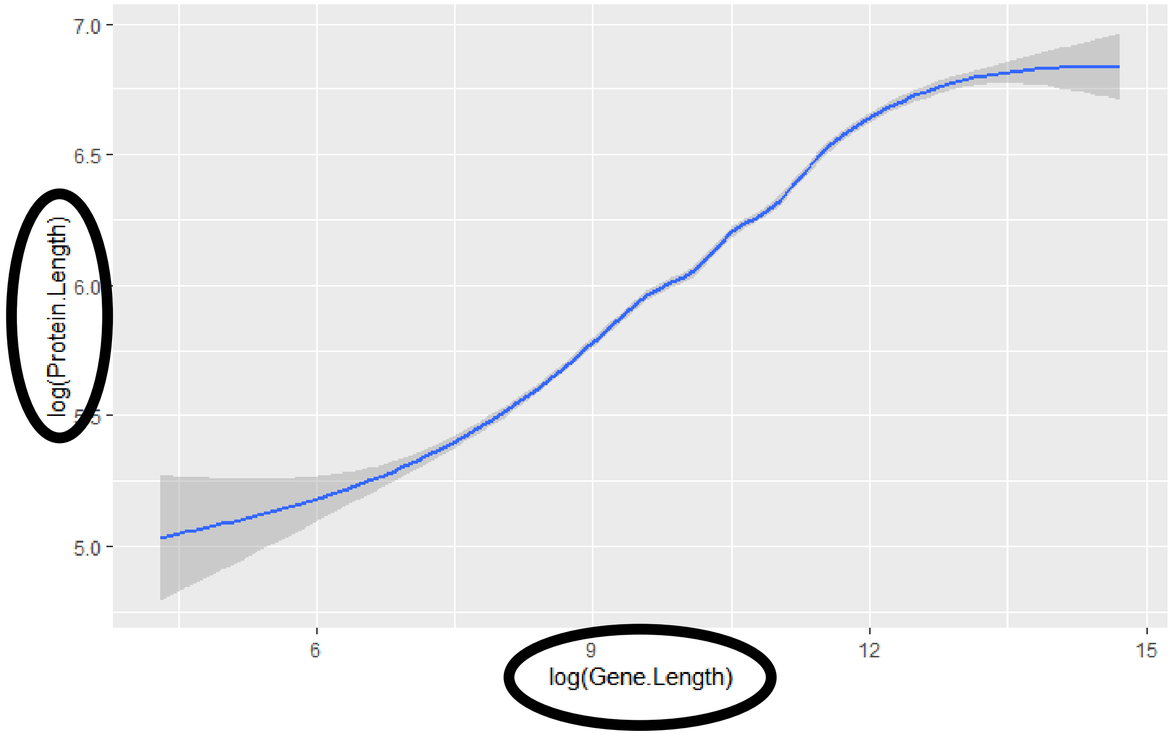
Mappings

Geoms

How are these plots similar?

Same: x var, y var, data





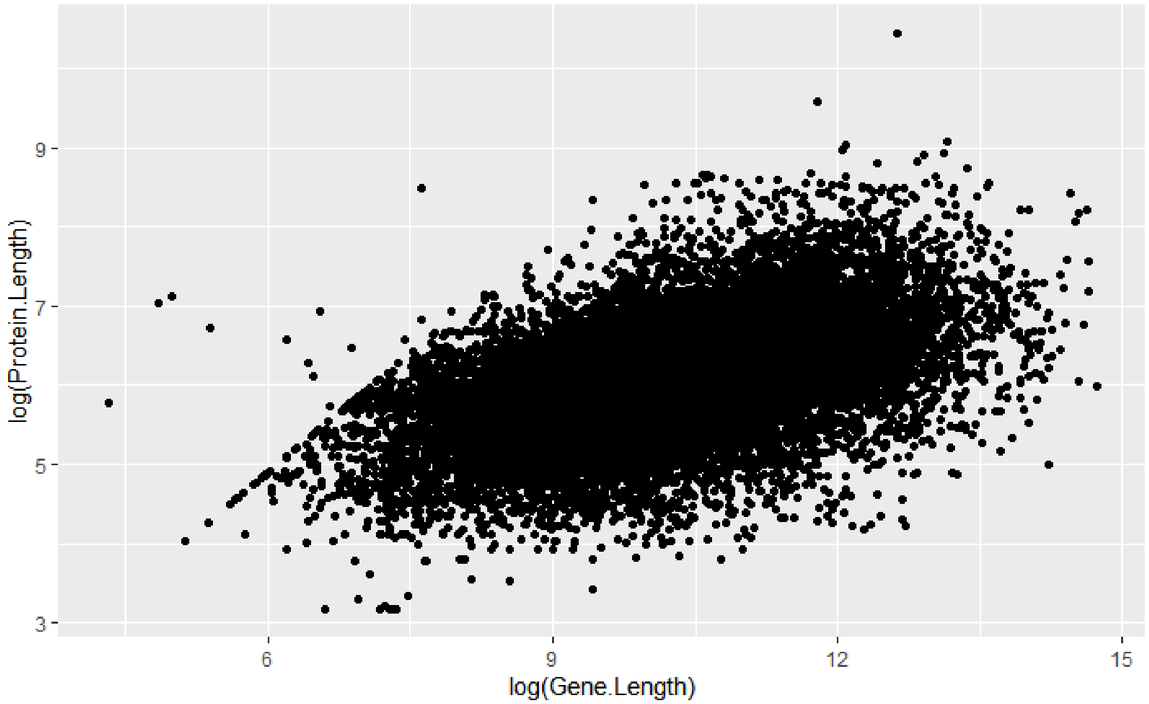

```
ggplot(uniprot_region)+
  geom_point(aes(x=log(Gene.Length),
                 y=log(Protein.Length)))
```

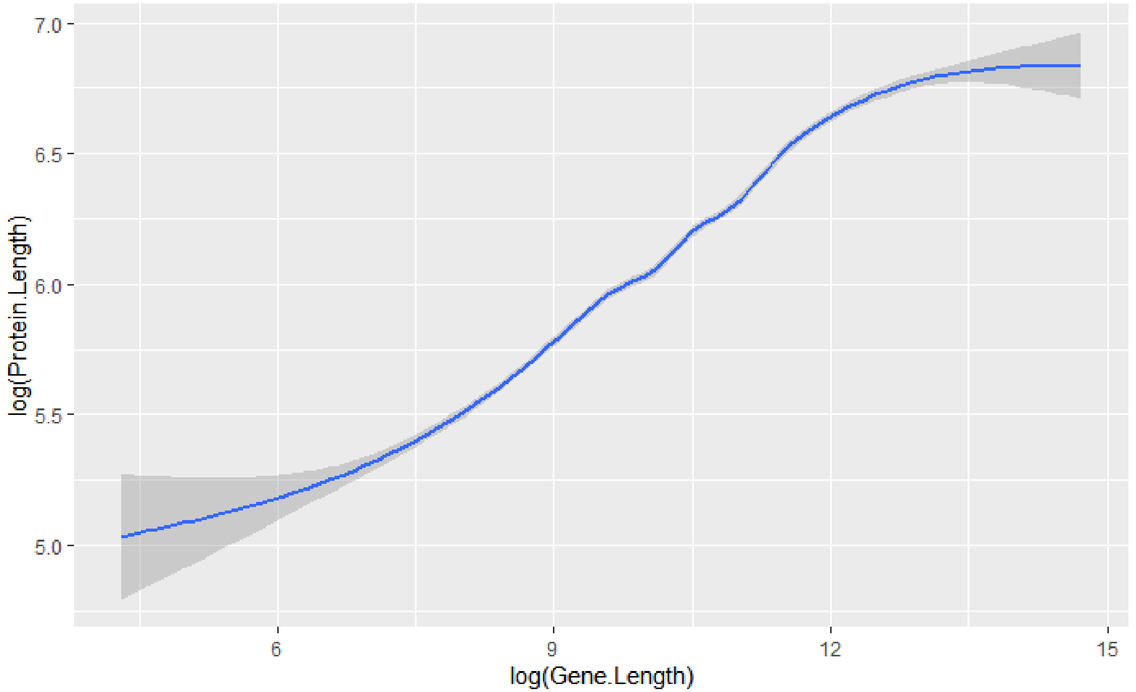
```
ggplot(uniprot_region)+
  geom_smooth(aes(x=log(Gene.Length),
                  y=log(Protein.Length)))
```

How are these plots different?

Different: geometric object (geom),

e.g. the visual object used to represent the data






```
ggplot(uniprot_region)+
  geom_point(aes(x=log(Gene.Length),
                 y=log(Protein.Length)))
```

```
ggplot(uniprot_region)+
  geom_smooth(aes(x=log(Gene.Length),
                  y=log(Protein.Length)))
```

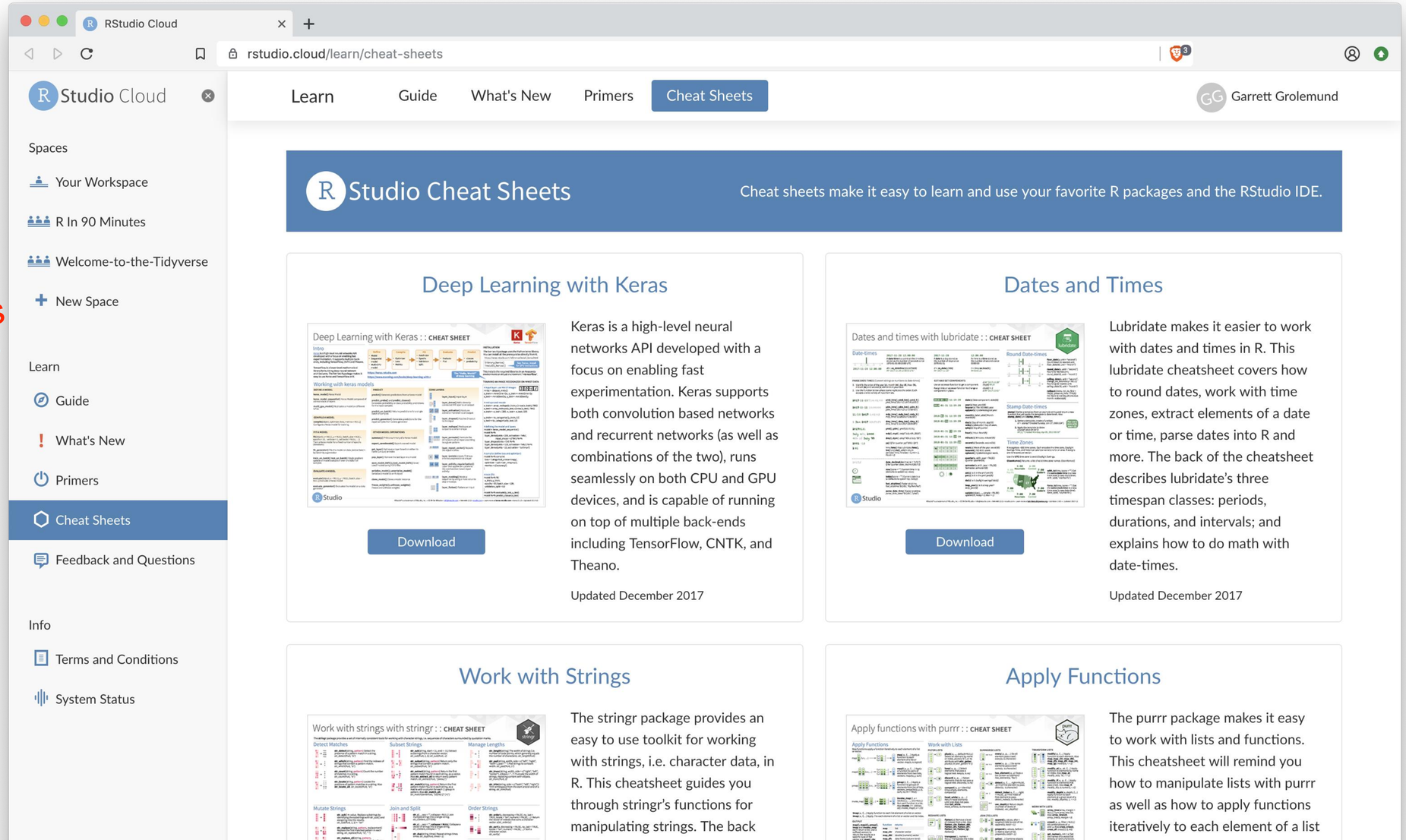
geoms

```
ggplot(<DATA>) +
  <GEOM_FUNCTION>(aes(<MAPPINGS>))
```

How do I know which geom to use?

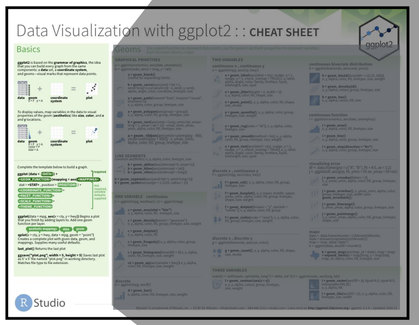
rstudio.cloud/learn/cheat-sheets

Click CheAT Sheets in the sidebar



geom_ functions

Each requires a mapping argument.



Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

ggplot2

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1,curvature=z)**)** - 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(group = group)**)**
x, y, alpha, color, fill, group, 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

continuous x , continuous y

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

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

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

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, check_overlap = TRUE**)**, x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

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

discrete x , discrete y

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

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

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_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, 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))

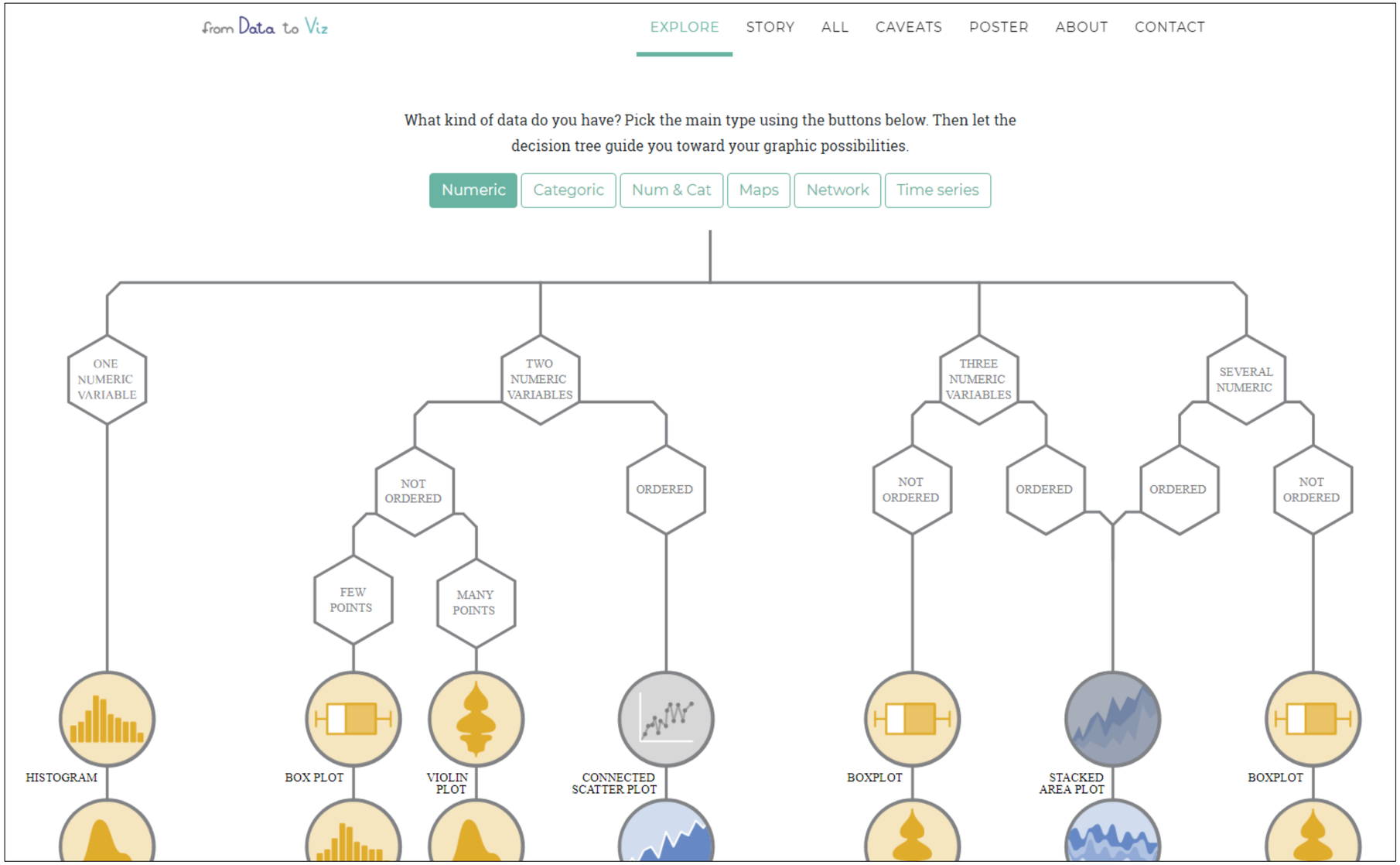
l + geom_contour(aes(z = z))
x, y, z, alpha, colour, group, linetype, size, weight

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

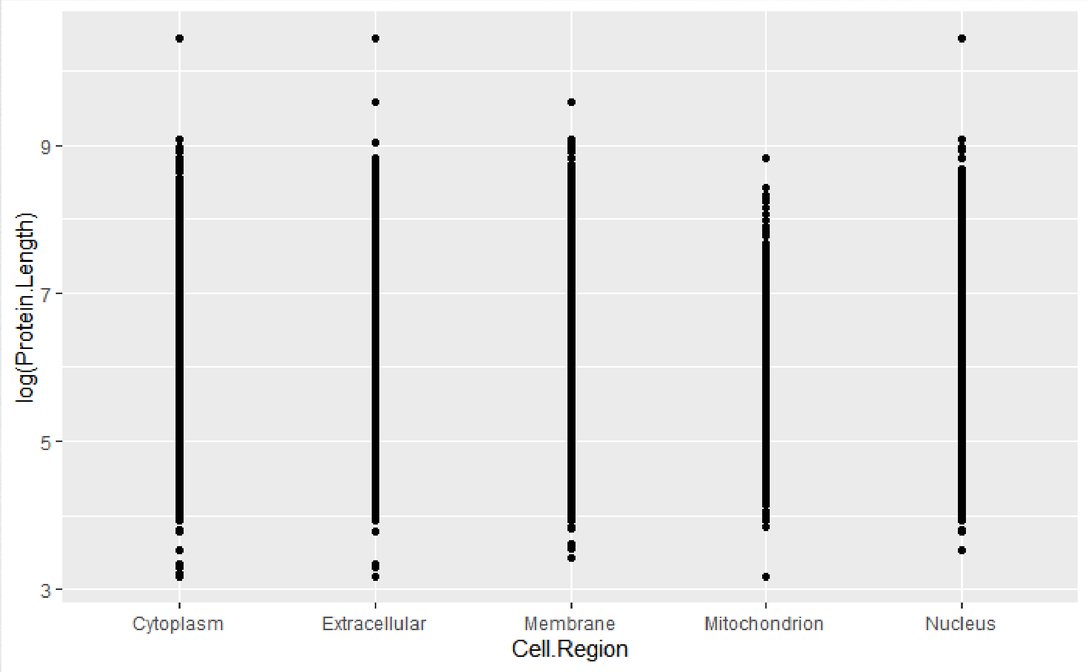
ggplot2

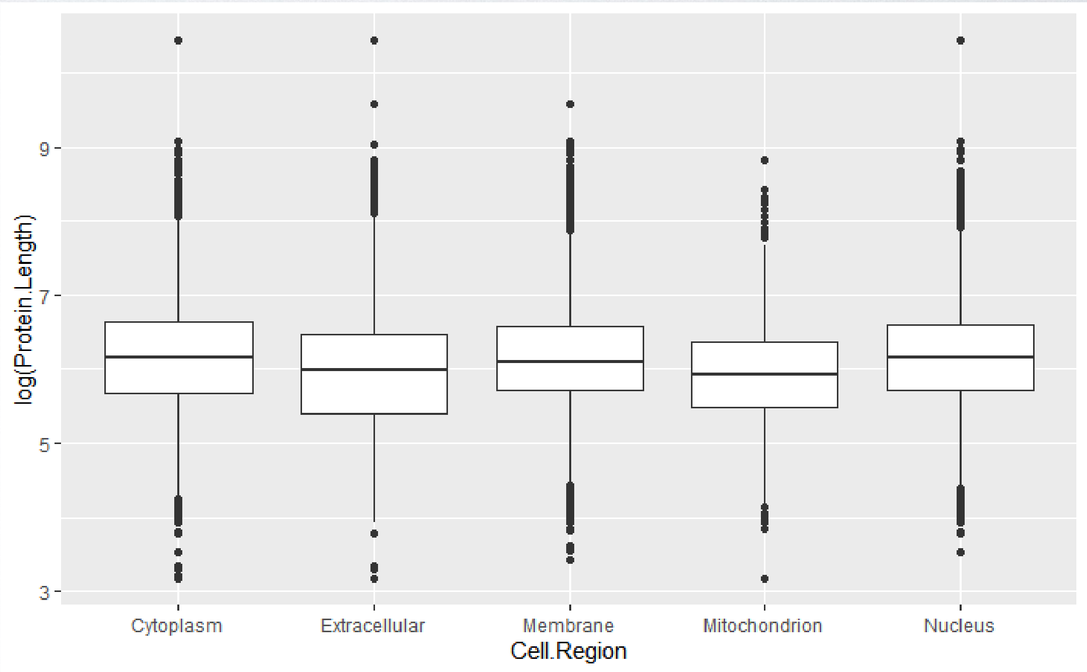
https://www.data-to-viz.com



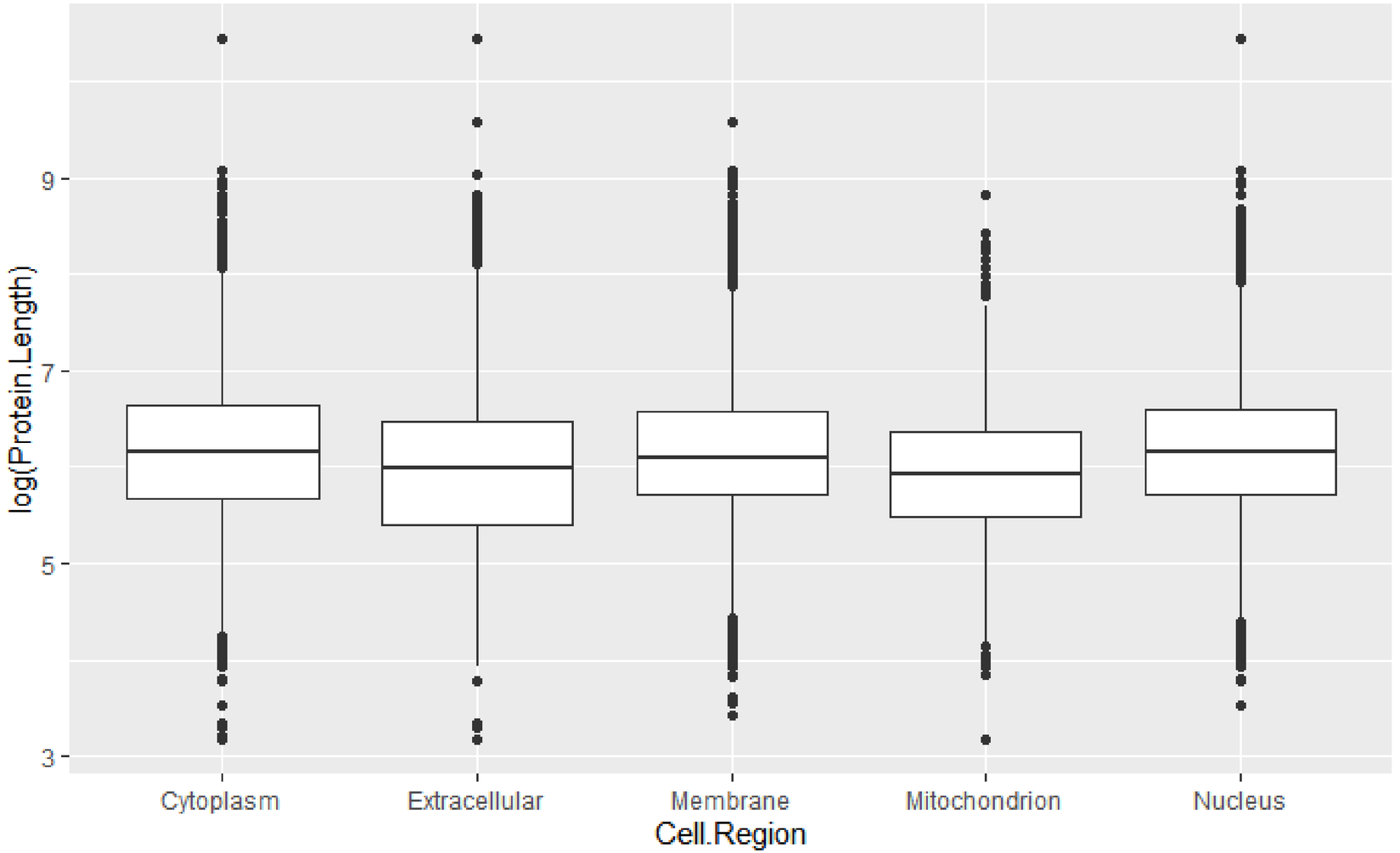
Exercise 2

Decide how to replace this scatterplot with one that draws boxplots. Use the cheatsheet. Try your best guess.






```
ggplot(uniprot_region) +
geom_point(aes(x=Cell.Region,y=log(Protein.Length)))
```




```
ggplot(uniprot_region) +
geom_boxplot(aes(x=Cell.Region,y=log(Protein.Length)))
```

"Help" pages

To open the documentation for a function, type

```
?geom_histogram
```

? — **function name** (no parentheses)

geom_freqpoly {ggplot2} R Documentation

Histograms and frequency polygons

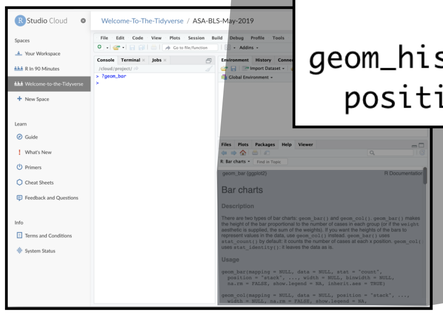
Description

Visualise the distribution of a single continuous variable by dividing the x axis into bins and counting the number of observations in each bin. Histograms (`geom_histogram()`) display the counts with bars; frequency polygons (`geom_freqpoly()`) display the counts with lines. Frequency polygons are more suitable when you want to compare the distribution across the levels of a categorical variable.

Usage

```
geom_freqpoly(mapping = NULL, data = NULL, stat = "bin",
  position = "identity", ..., na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE)

geom_histogram(mapping = NULL, data = NULL, stat = "bin",
  position = "stack", ..., binwidth = NULL, bins = NULL,
```



ggplot2.tidyverse.org

Reference

Plot basics

All ggplot2 plots begin with a call to `ggplot()`, supplying default data and aesthethic mappings, specified by `aes()`. You then add layers, scales, coords and facets with `+`. To save a plot to disk, use `ggsave()`.

`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

Layer: geoms

A layer combines data, aesthetic mapping, a geom (geometric object), a stat (statistical transformation), and a position adjustment. Typically, you will create layers using a `geom_` function, overriding the default position and stat if needed.

`geom_abline()` `geom_hline()` `geom_vline()`	Reference lines: horizontal, vertical, and diagonal
`geom_bar()` `geom_col()`	Bar charts

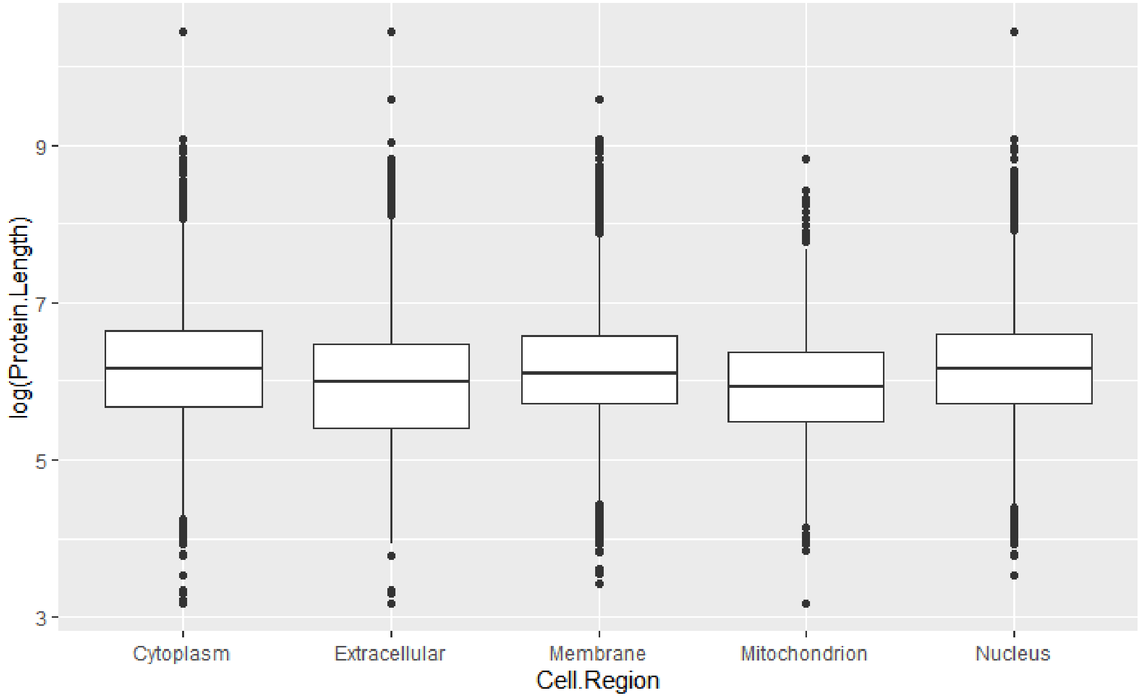
Contents

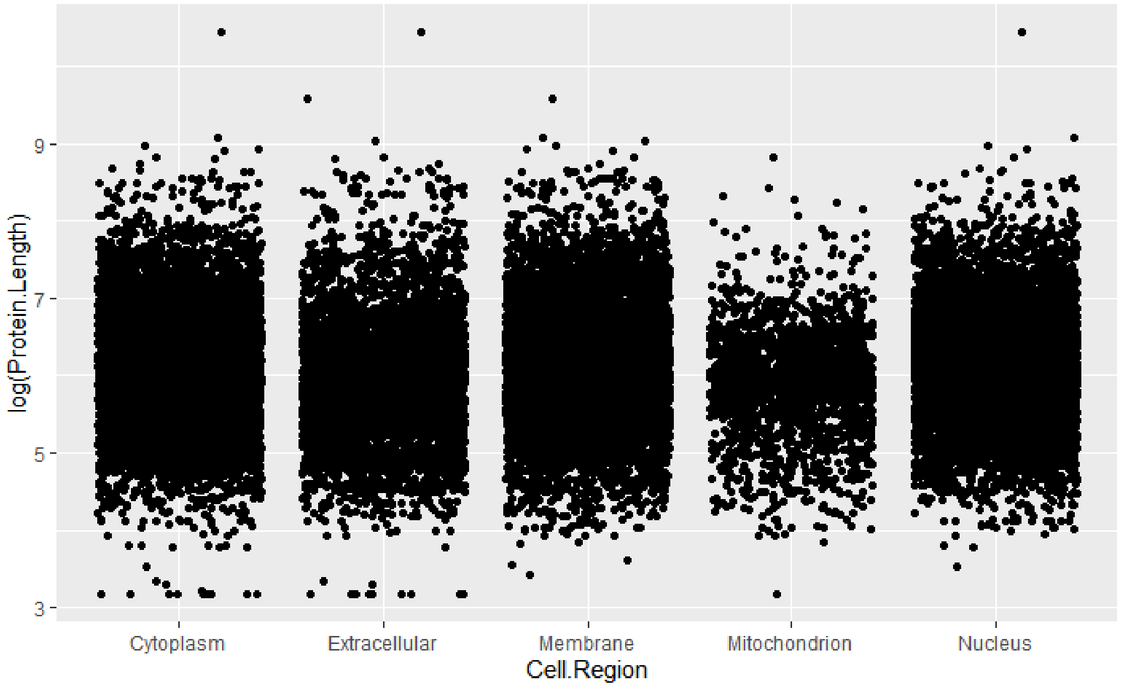
Question?

What will this code do?

```
ggplot(uniprot_region)+
  geom_jitter(aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_boxplot(aes(x=Cell.Region, y=log(Protein.Length)))
```

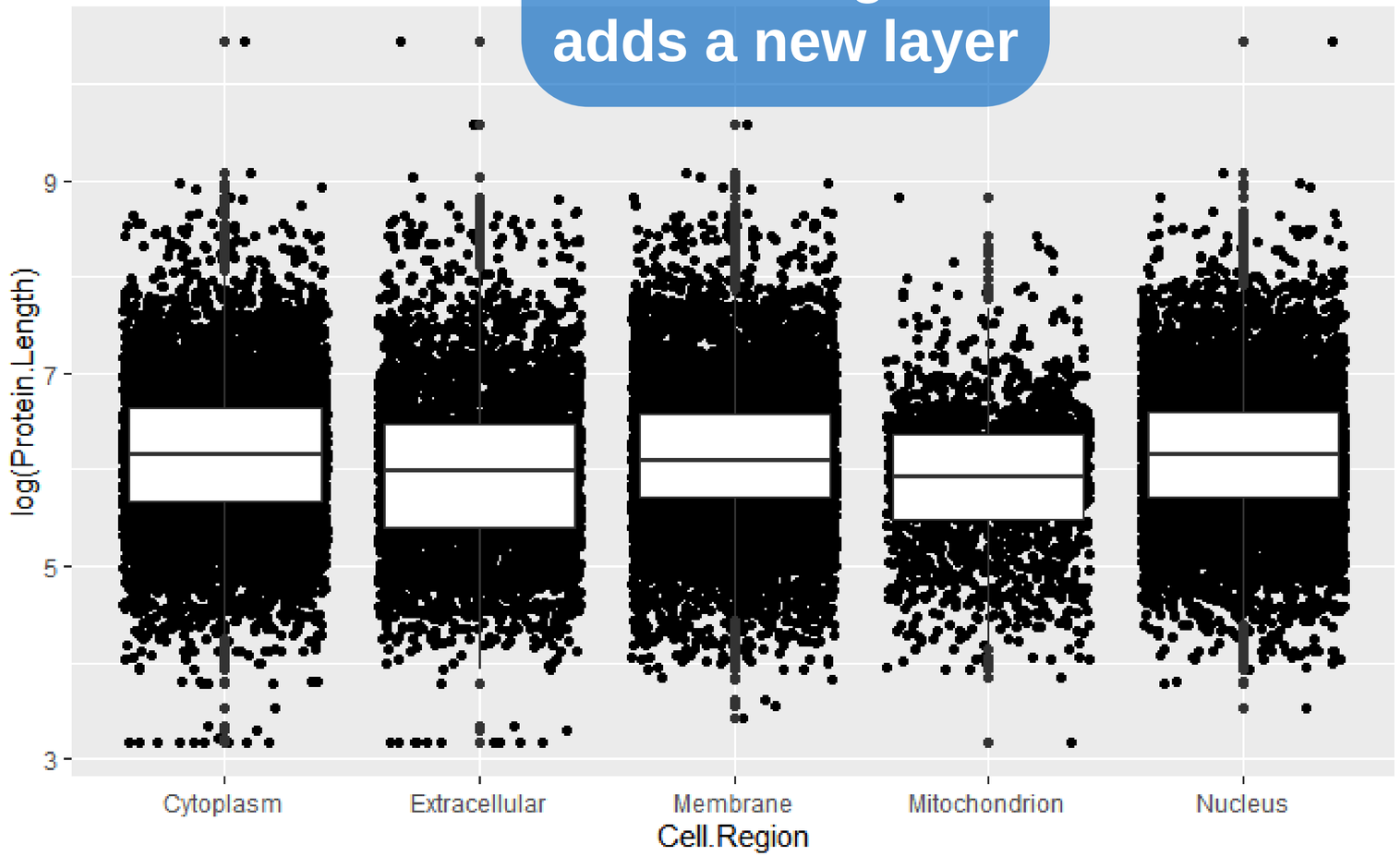
Each new geom adds a new layer






```
ggplot(uniprot_region)+
  geom_jitter(aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_boxplot(aes(x=Cell.Region, y=log(Protein.Length)))
```

Each new geom adds a new layer



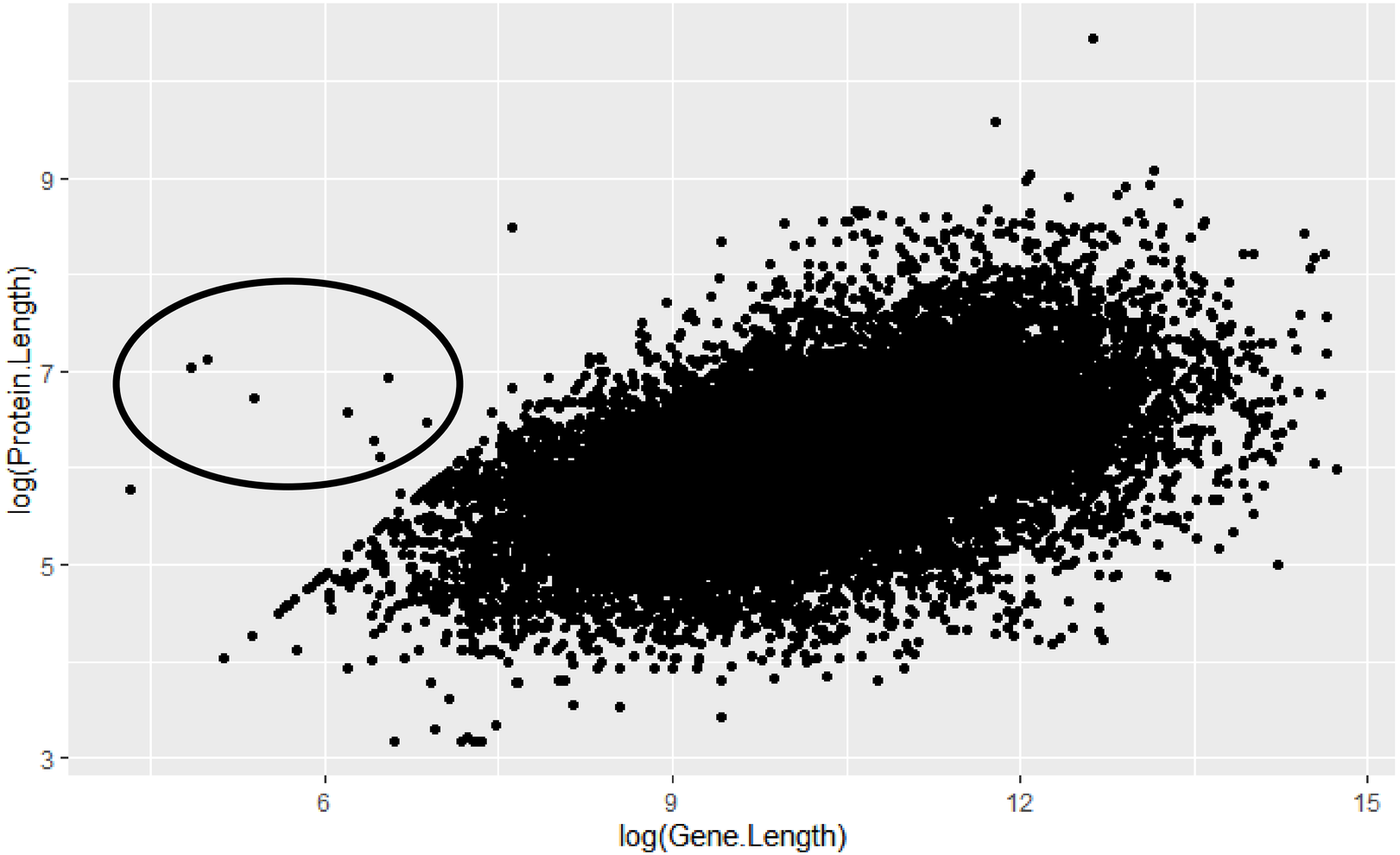

```
ggplot(uniprot_region)+
  geom_jitter(aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_boxplot(aes(x=Cell.Region, y=log(Protein.Length)))
```

Aesthetics

"The greatest value of a picture is when it forces us to notice what we never expected to see."

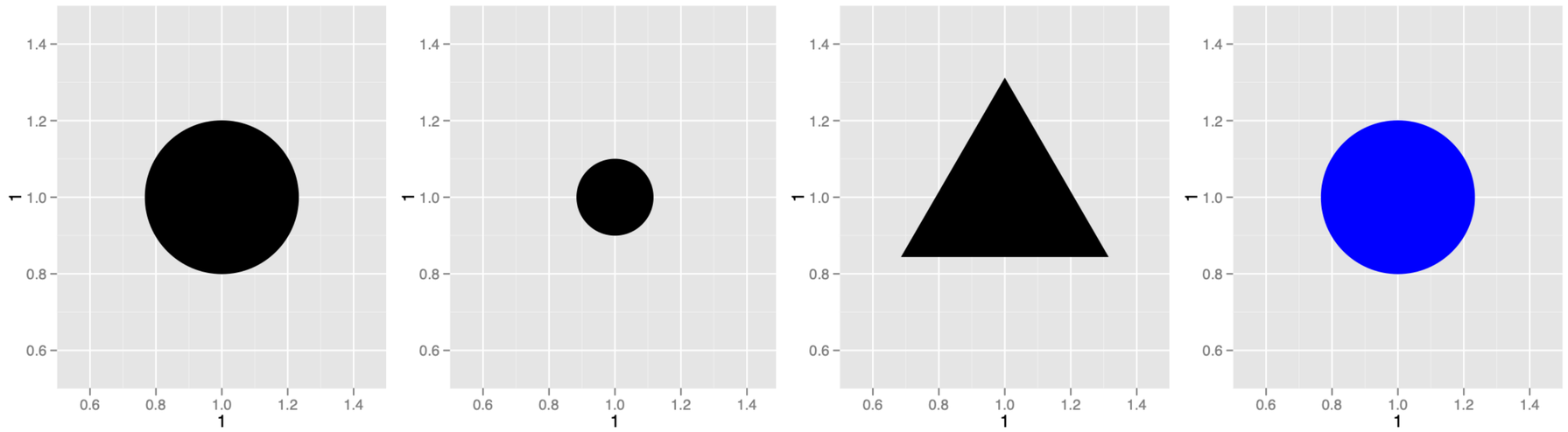
- John Tukey

What are these smaller genes that make large proteins?

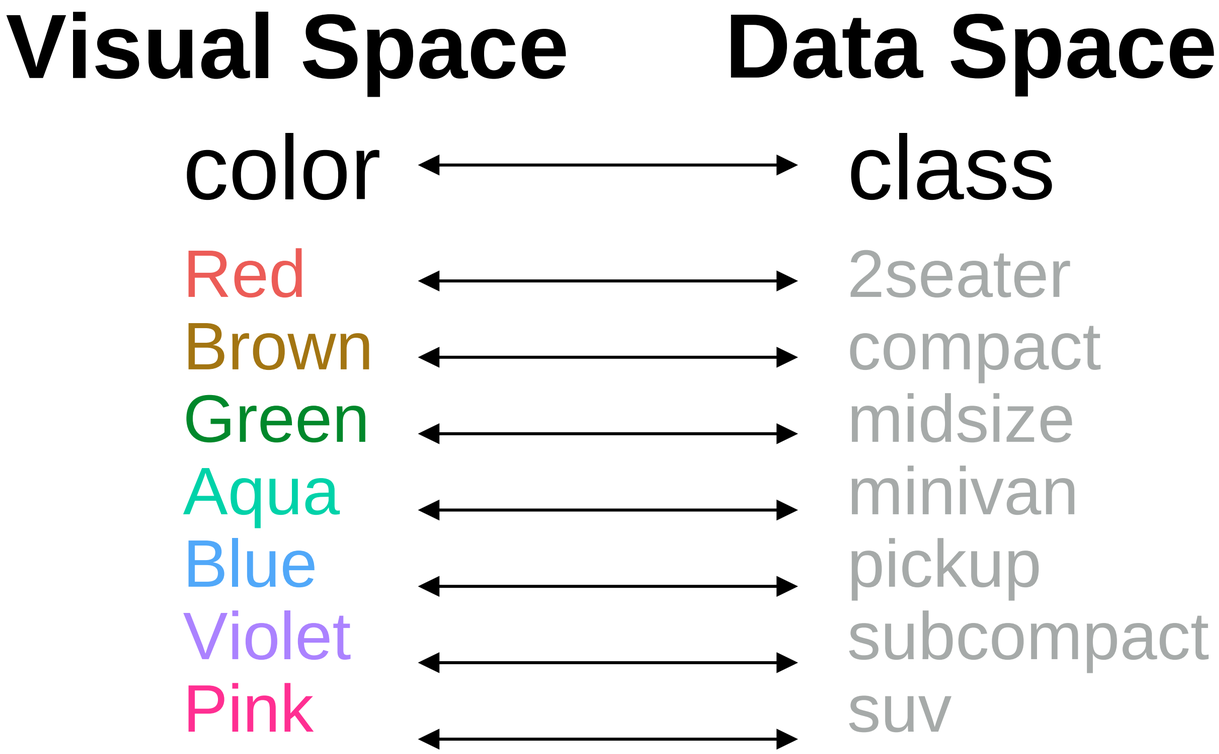



```
ggplot(uniprot_region) +
  geom_point(aes(log(Gene.Length),log(Protein.Length)))
```

Aesthetics- Shape/Size



Aesthetics- Color



Aesthetics

aesthetic property

Variable to map it to

```
ggplot(uniprot_region) +
  geom_point(aes(log(Gene.Length),log(Protein.Length), color= Cell.Region))
ggplot(uniprot_region) +
  geom_point(aes(log(Gene.Length),log(Protein.Length), size = Cell.Region))

ggplot(uniprot_region) +
  geom_point(aes(log(Gene.Length),log(Protein.Length), shape = Cell.Region))

ggplot(uniprot_region) +
  geom_point(aes(log(Gene.Length),log(Protein.Length), alpha = Cell.Region))

ggplot(uniprot_region) +
  geom_point(aes(log(Gene.Length),log(Protein.Length), fill = Cell.Region))
```

NOTE: Color vs Fill- depends on shape!

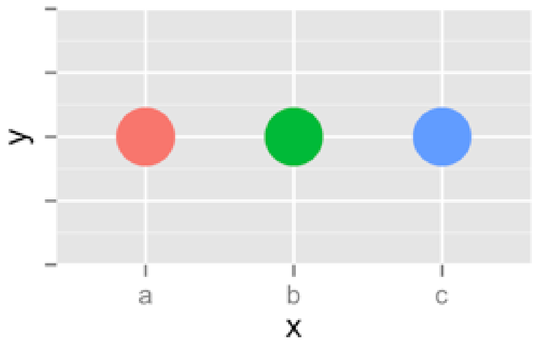
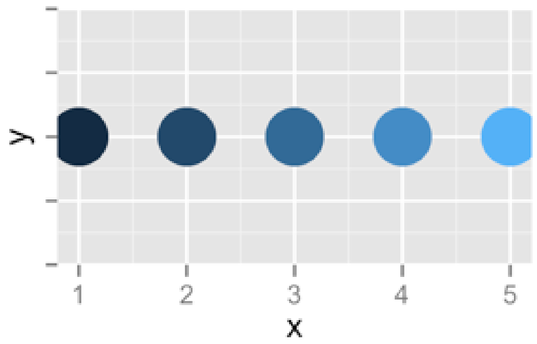
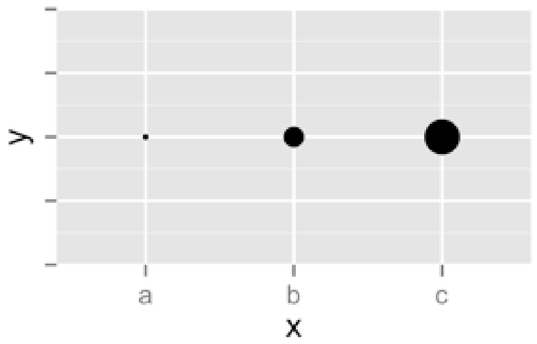
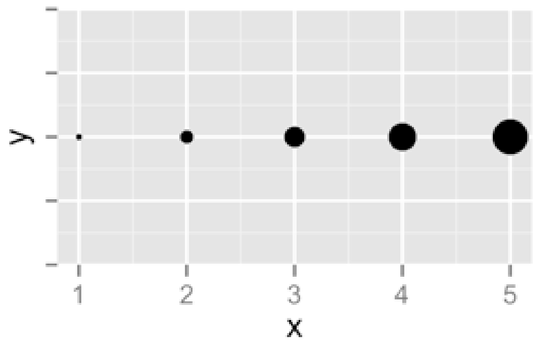
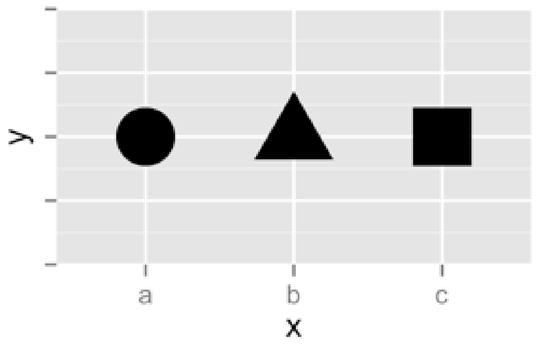
color= outline color, **fill =** fill color

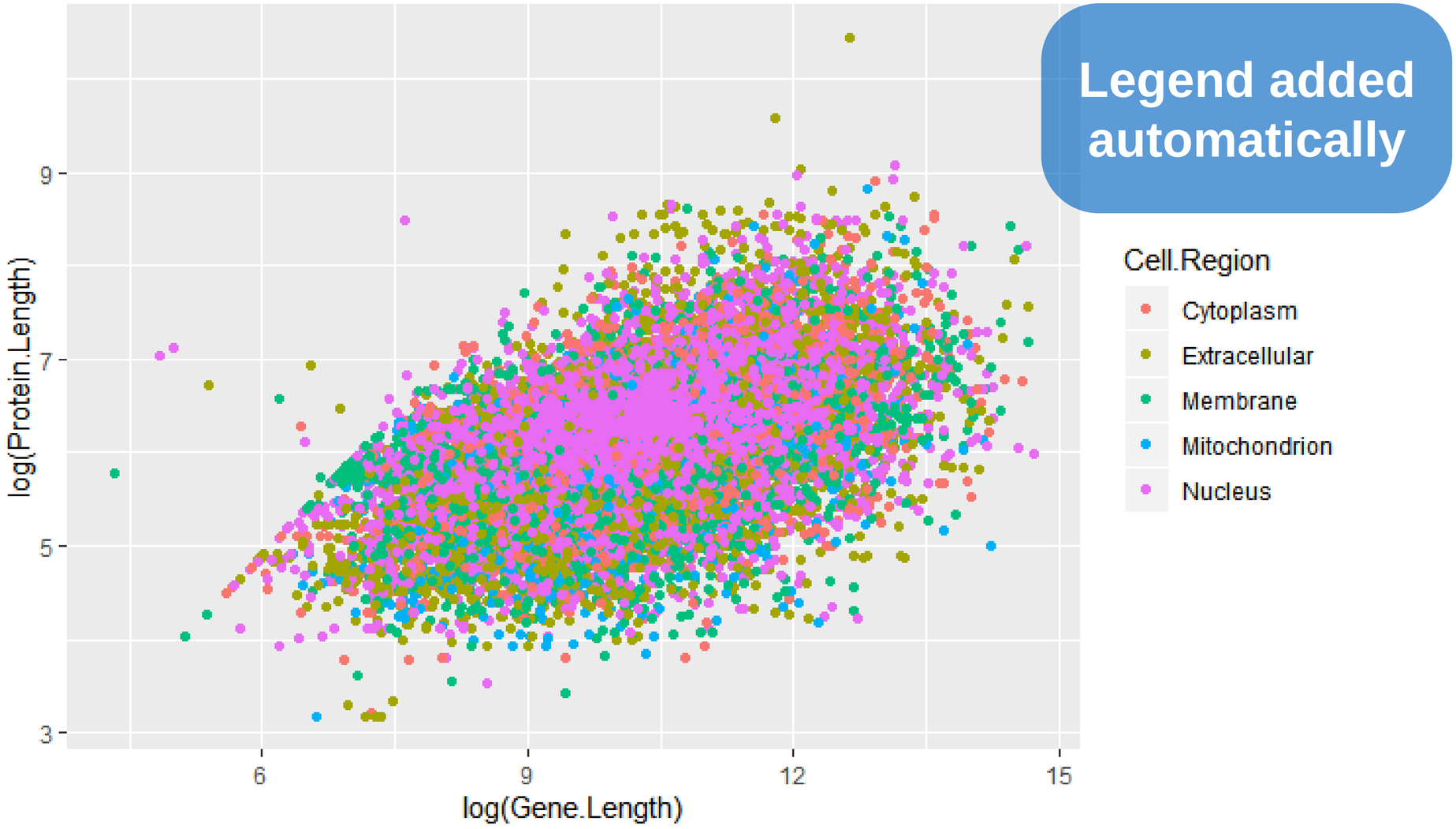
Exercise 3

In the next chunk, add color, size, alpha, and shape aesthetics to your graph. Experiment.

Do different things happen when you map aesthetics to discrete and continuous variables?

What happens when you use more than one aesthetic?

	Discrete	Continuous
Color		
Size		
Shape		



Legend added automatically

```
ggplot(uniprot_region) +
  geom_point(aes(log(Gene.Length),log(Protein.Length), color= Cell.Region))
```

Exercise 4

What happens when you change the alpha aes?

```
ggplot(uniprot_region) +
  geom_point(aes(log(Gene.Length),log(Protein.Length), fill= Cell.Region,
alpha=0.5))
```

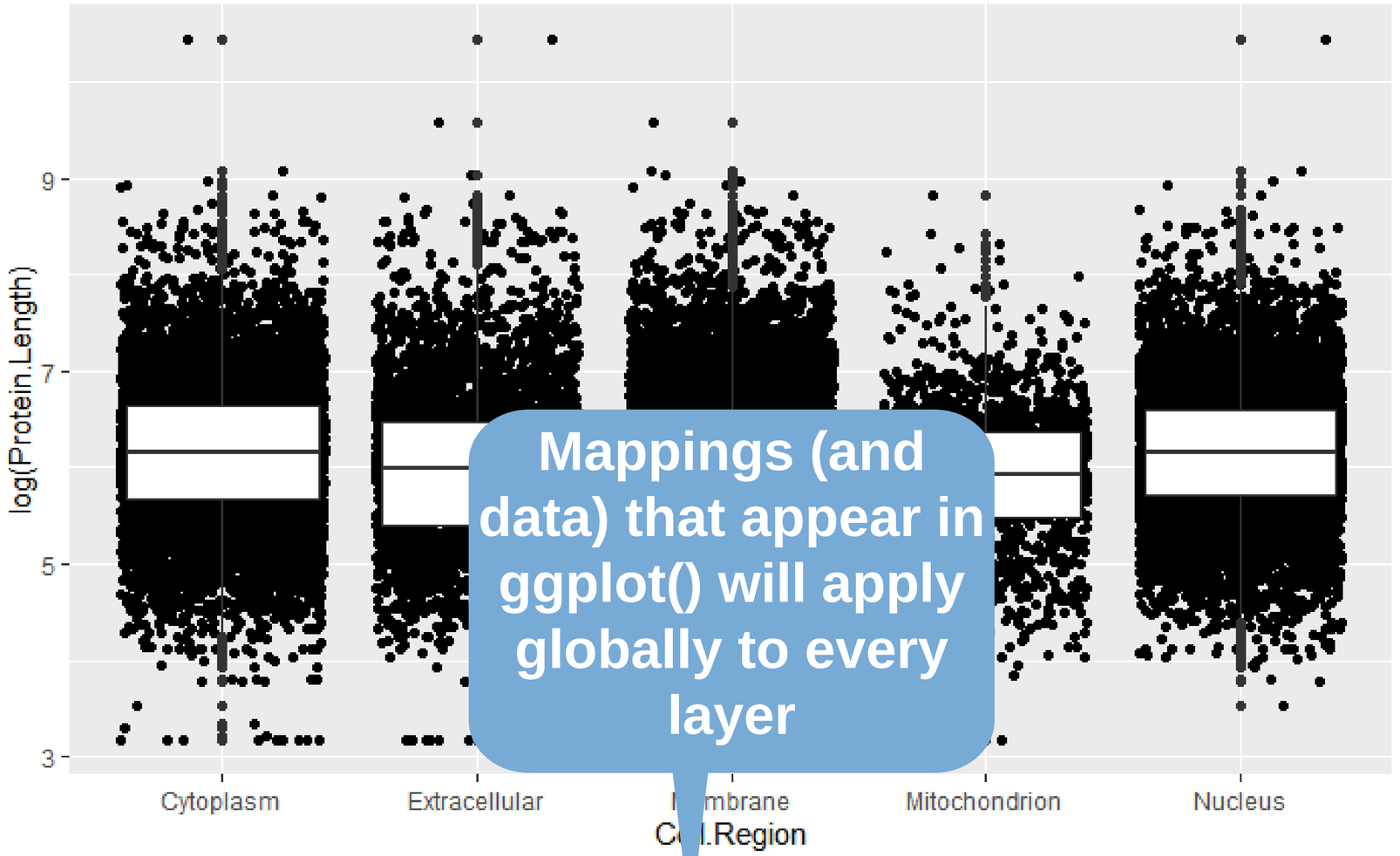
Changes transparency of points


```
ggplot(uniprot_region) +
  geom_point(aes(log(Gene.Length),log(Protein.Length), color= Cell.Region,
alpha=0.5))
```

Error!

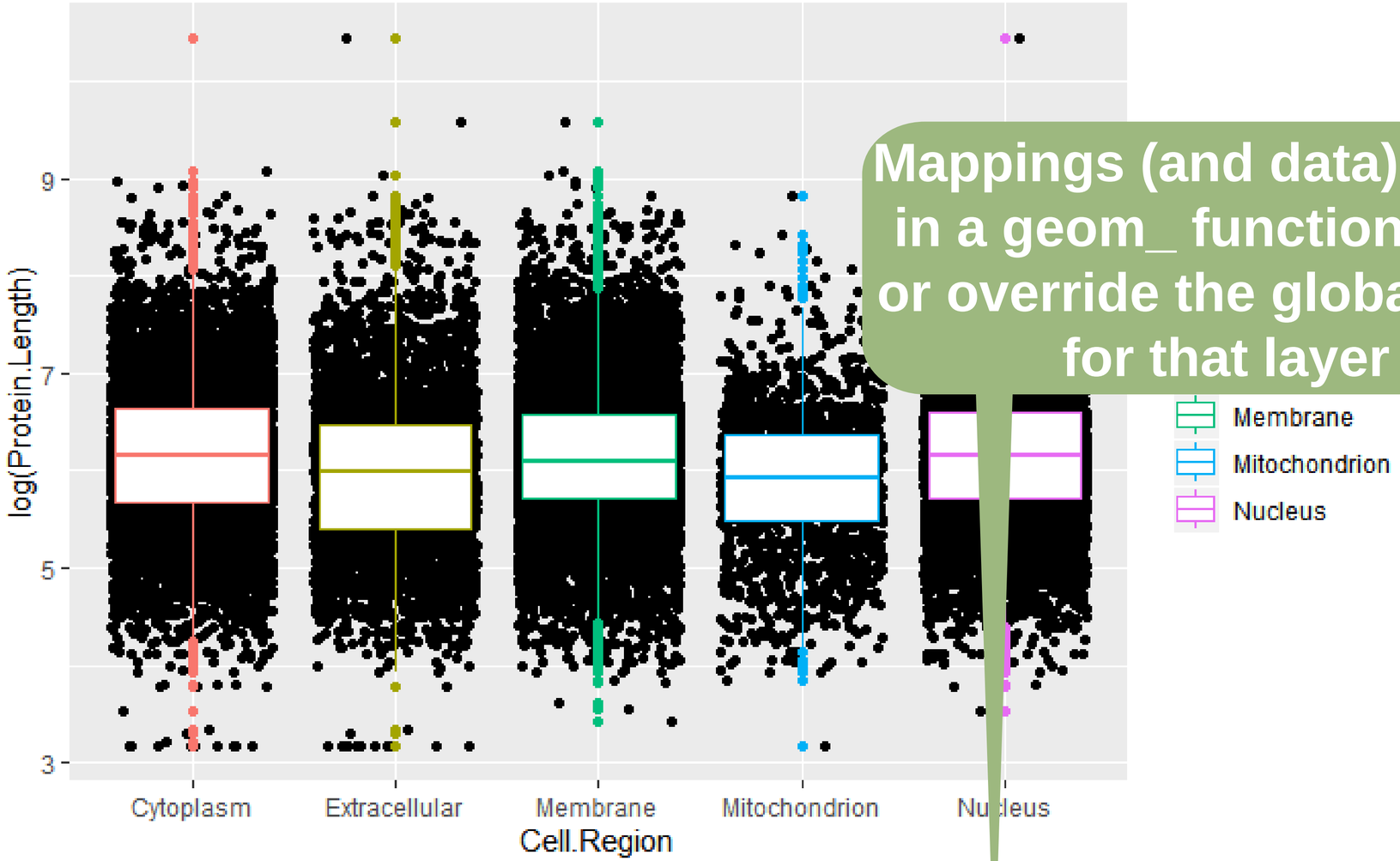
```
ggplot(uniprot_region) +
  geom_point(aes(log(Gene.Length),log(Protein.Length)), fill= Cell.Region)
```

global vs. local



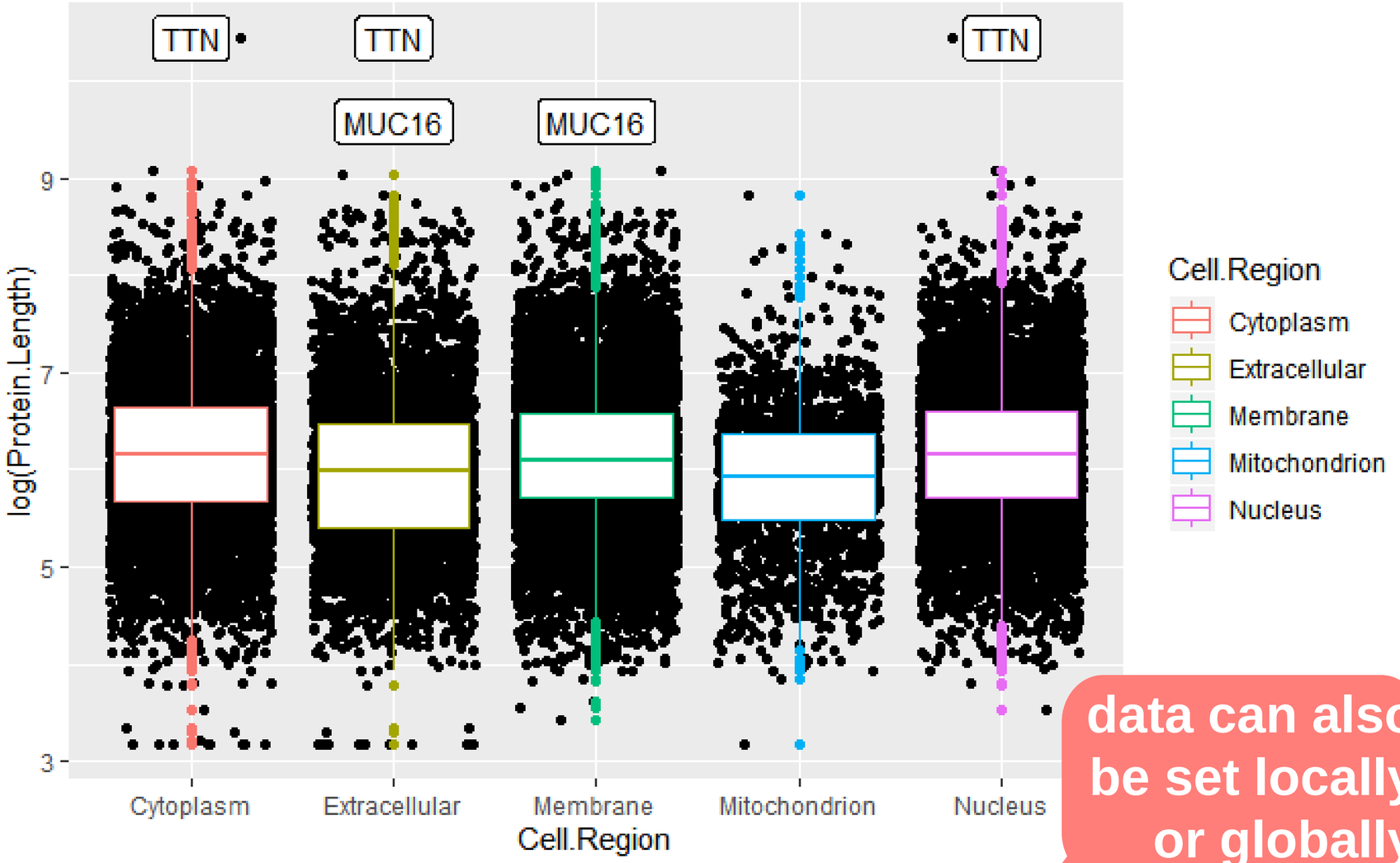
Mappings (and data) that appear in ggplot() will apply globally to every layer

```
ggplot(uniprot_region, aes(x=Cell.Region,
y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot()
```



Mappings (and data) that appear in a geom_ function will add to or override the global mappings for that layer only

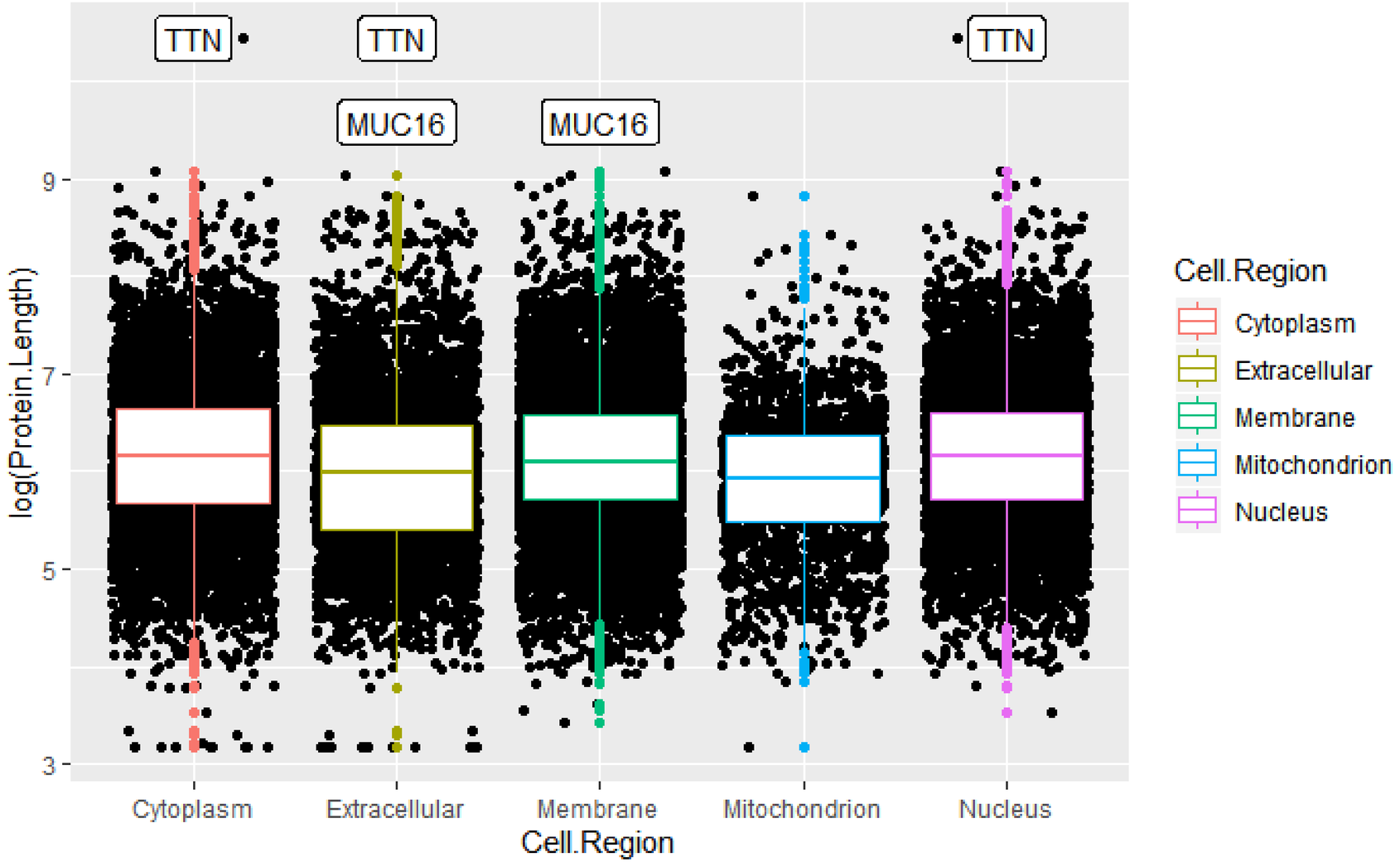
```
ggplot(uniprot_region, aes(x=Cell.Region,
y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=Cell.Region))
```



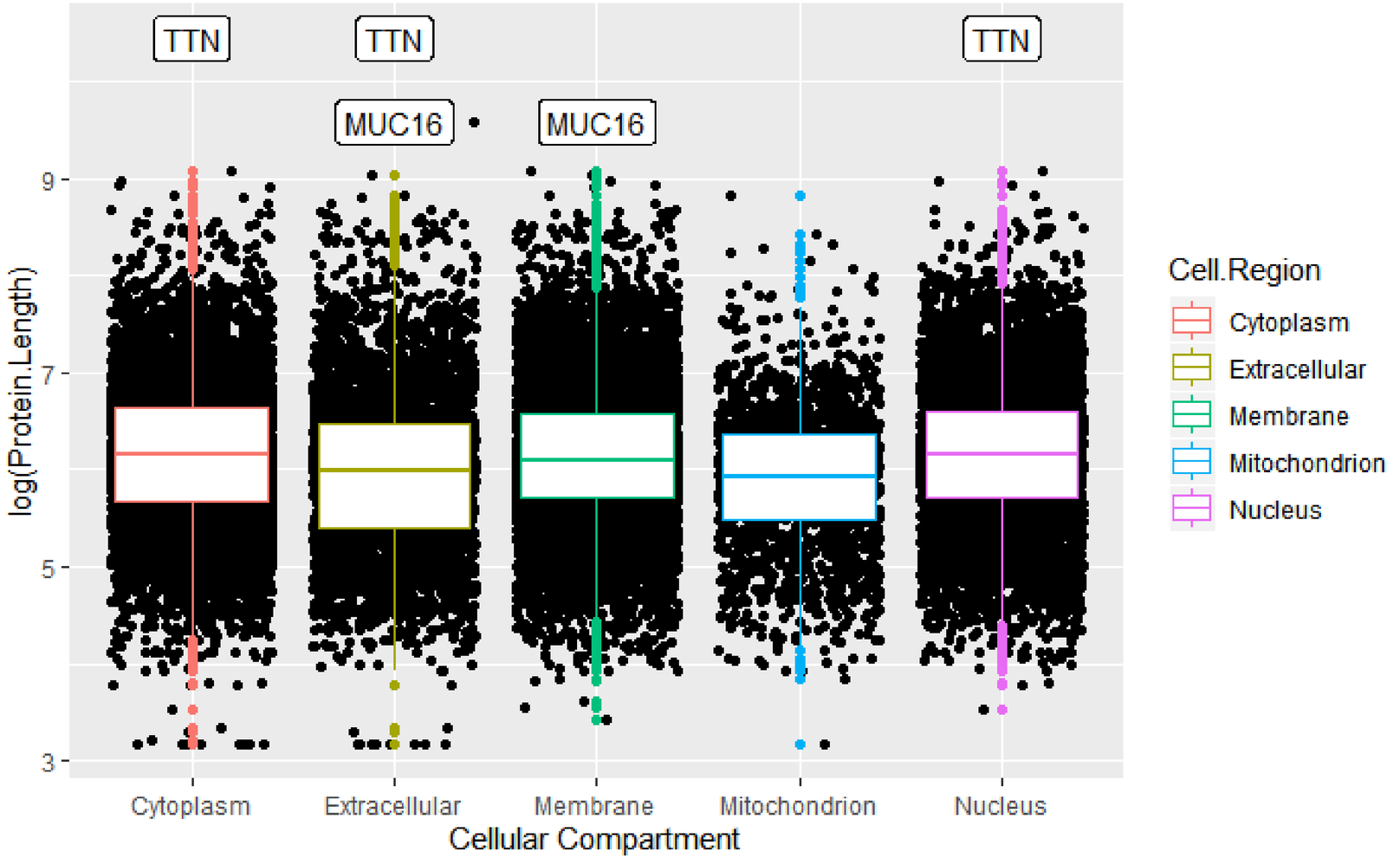
data can also be set locally or globally

```
ggplot(uniprot_region, aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=Cell.Region))+
  geom_label(data=large_prot, aes(label= Gene.Name))
```

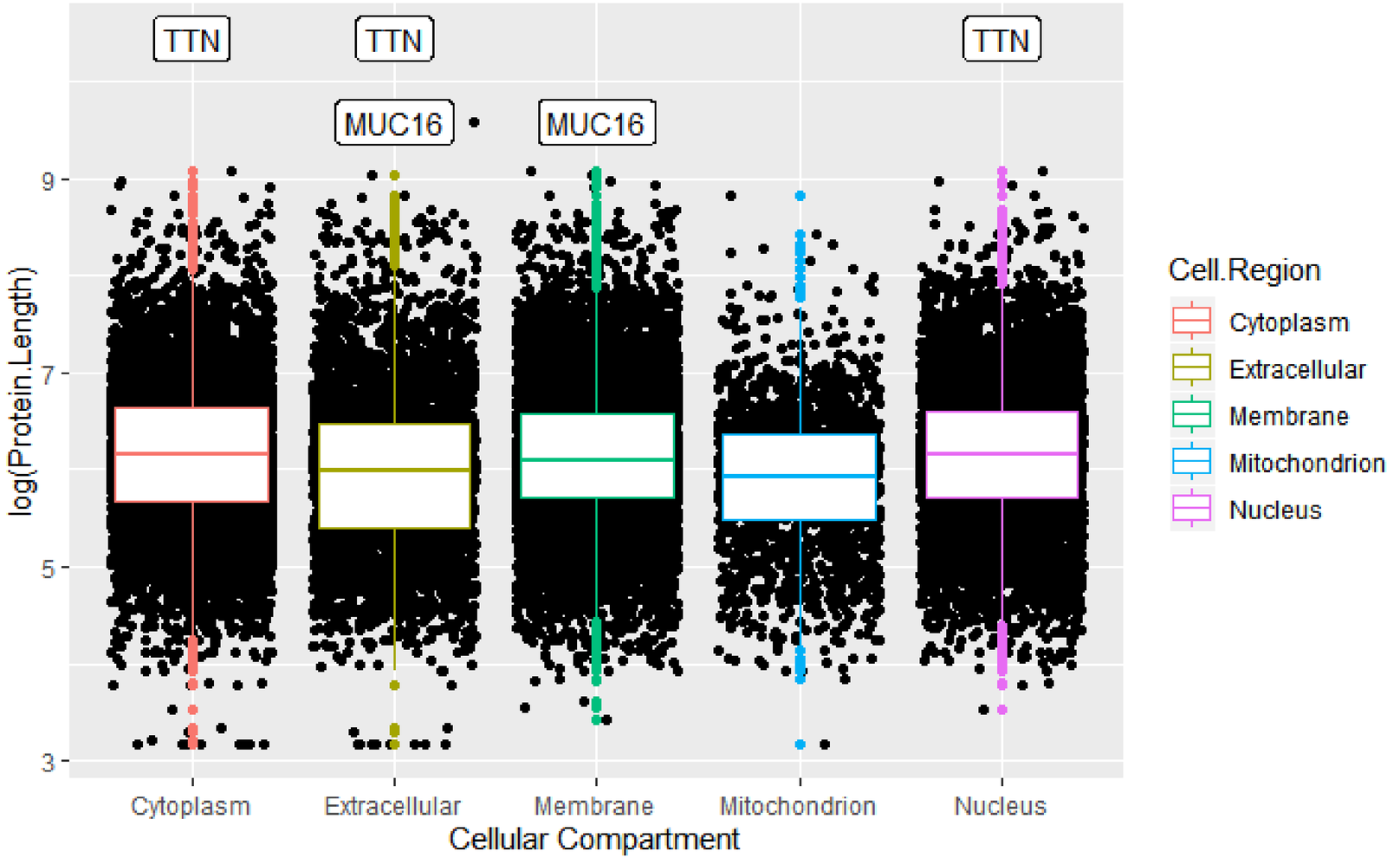
Labels and Legends



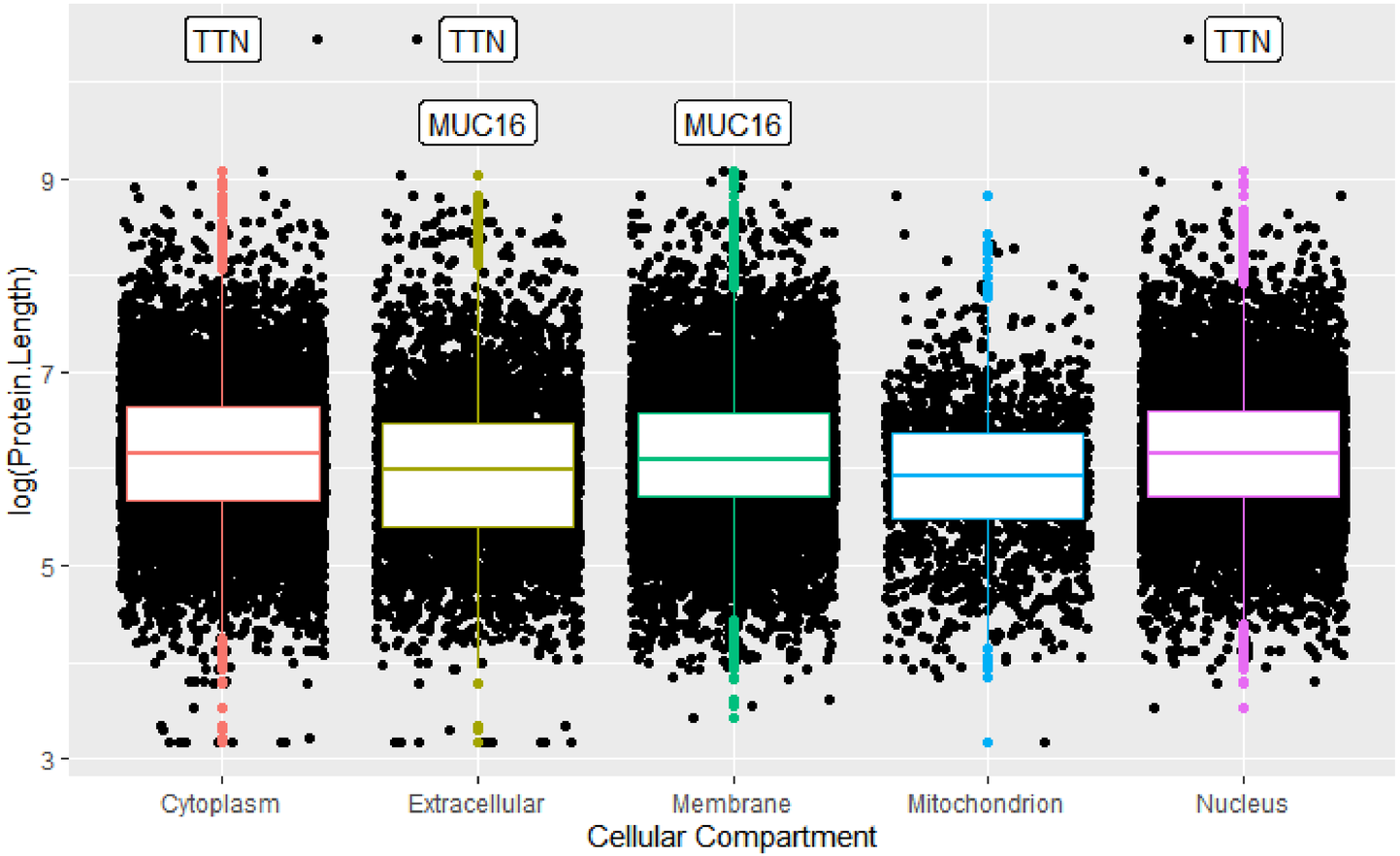

```
ggplot(uniprot_region, aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=Cell.Region))+
  geom_label(data=large_prot, aes(label= Gene.Name))
```




```
ggplot(uniprot_region, aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=Cell.Region))+
  geom_label(data=large_prot, aes(label= Gene.Name))+
  labs(x="Cellular Compartment")
```




```
ggplot(uniprot_region, aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=Cell.Region))+
  geom_label(data=large_prot, aes(label= Gene.Name))+
  labs(x="Cellular Compartment")
```

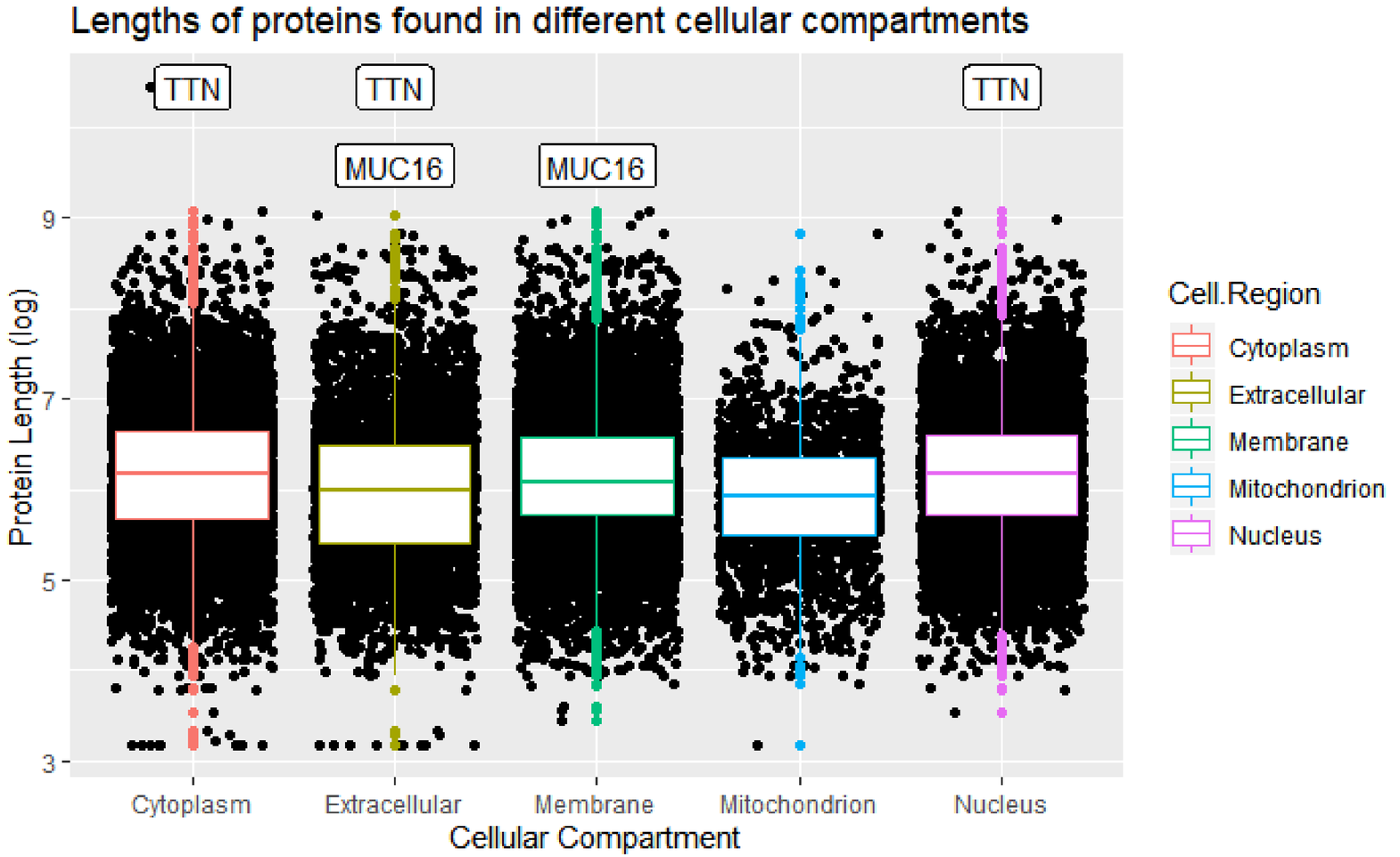



```
ggplot(uniprot_region, aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=Cell.Region))+
   geom_label(data=large_prot, aes(label= Gene.Name))+
   labs(x="Cellular Compartment")+
   guides(fill="none")
```

Exercise 5

Change the x and y axis labels and add a title

```
ggplot(uniprot_region, aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=Cell.Region))+
  geom_label(data=large_prot, aes(label= Gene.Name))
```

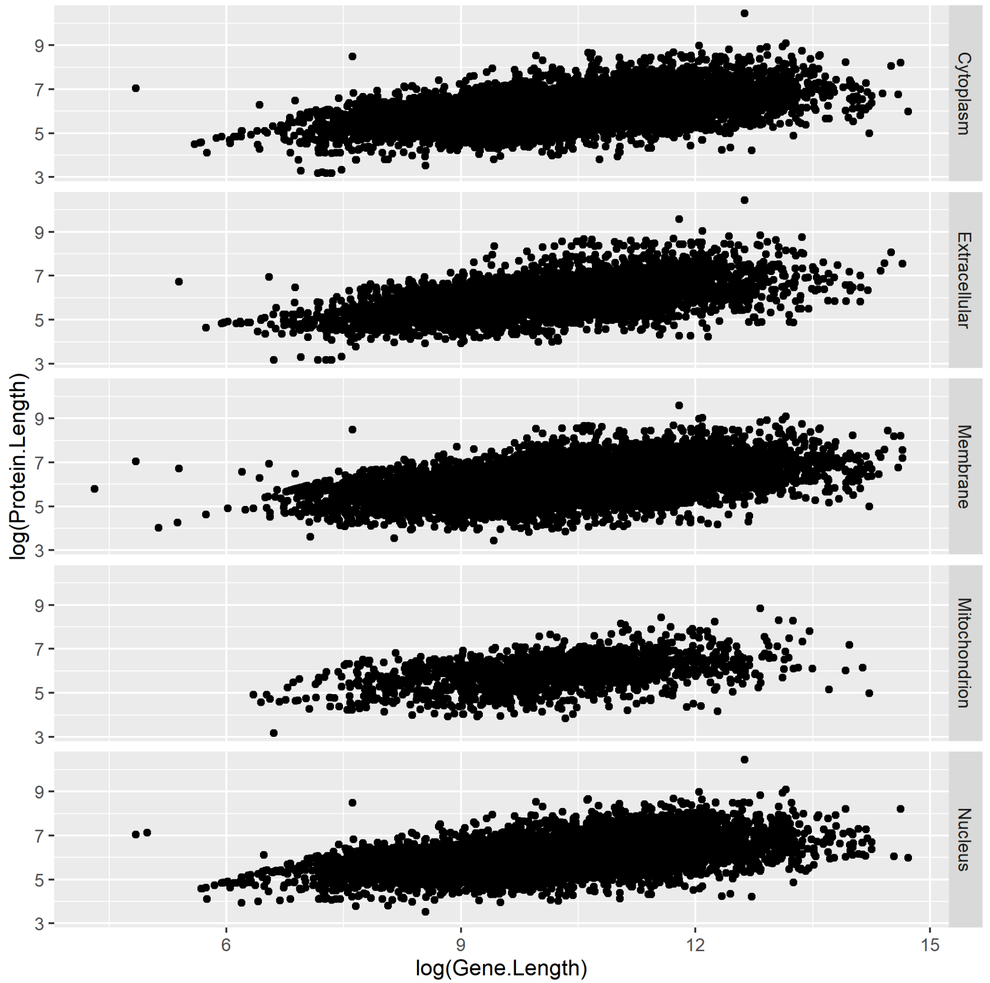



```
ggplot(uniprot_region, aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=Cell.Region))+
  geom_label(data=large_prot, aes(label= Gene.Name))+
  labs(x="Cellular Compartment", y="Protein Length (log)", title= "Lengths of proteins found in different cellular compartments")
```

Facets

Facets

Subplots that display subsets of the data.



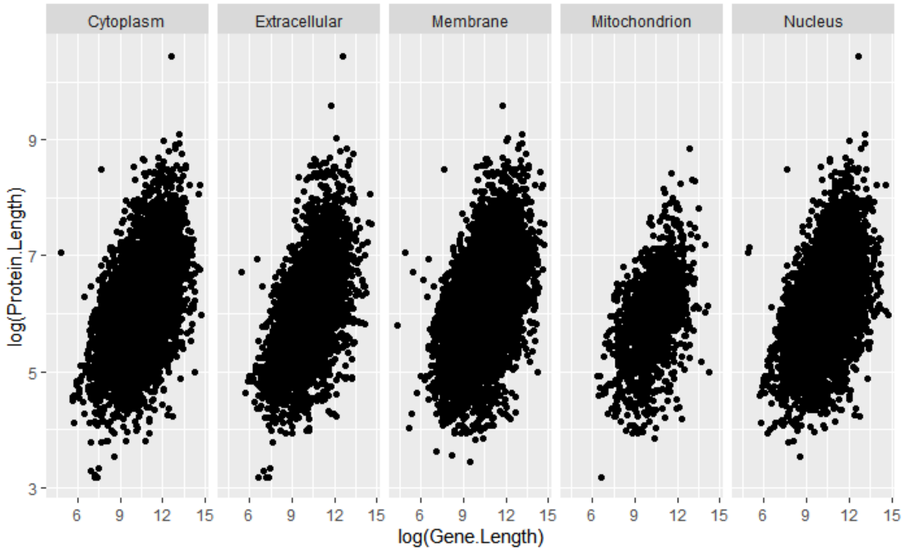

```
ggplot(uniprot_region)+
  geom_point(aes(x=log(Gene.Length), y=log(Protein.Length)))+
  facet_grid(rows = vars(Cell.Region))
```

Exercise 6

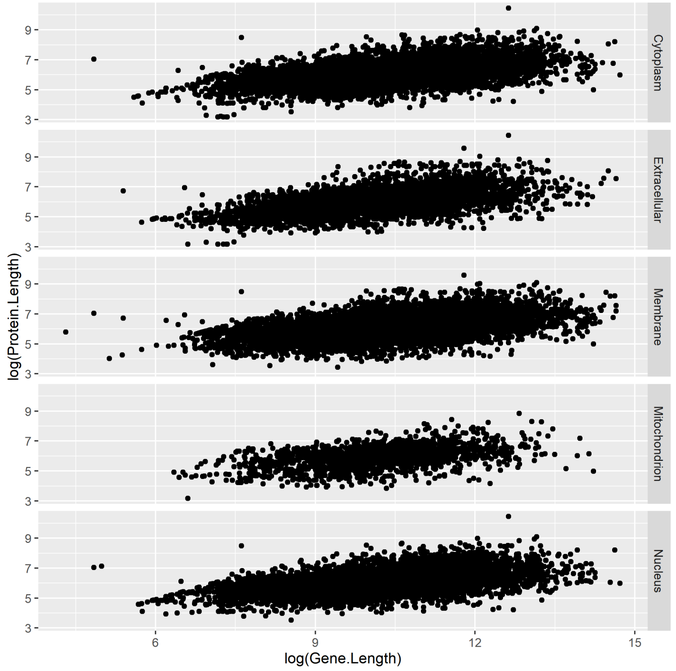
What do **facet_grid** and **facet_wrap** do?

```
q <- ggplot(uniprot_region) + geom_point(aes(x =
Gene.Length, y = Protein.Length))
q + facet_grid(. ~ Cell.Region)
q + facet_grid(Cell.Region ~ .)
q + facet_wrap(~ Cell.Region)
```

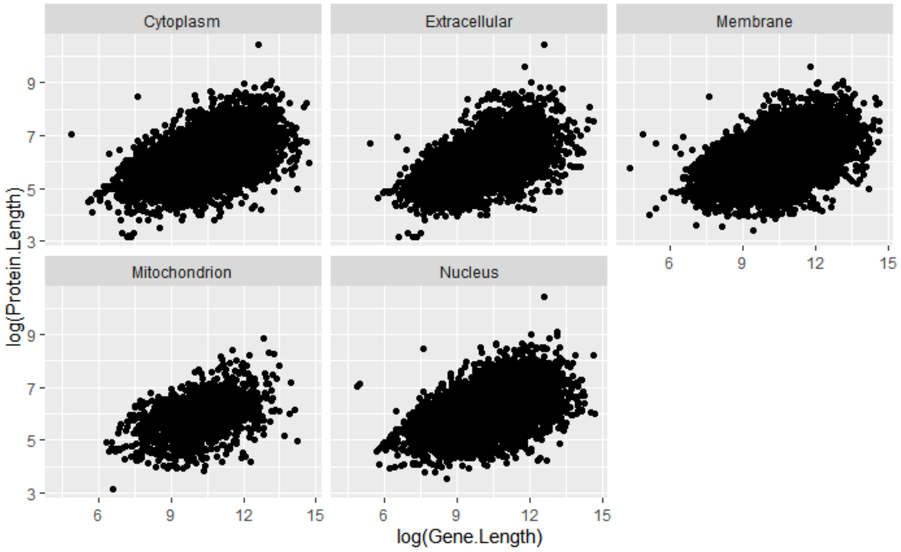
Facets




```
q + facet_grid(. ~ Cell.Region)
```




```
q + facet_grid(Cell.Region ~ .)
```



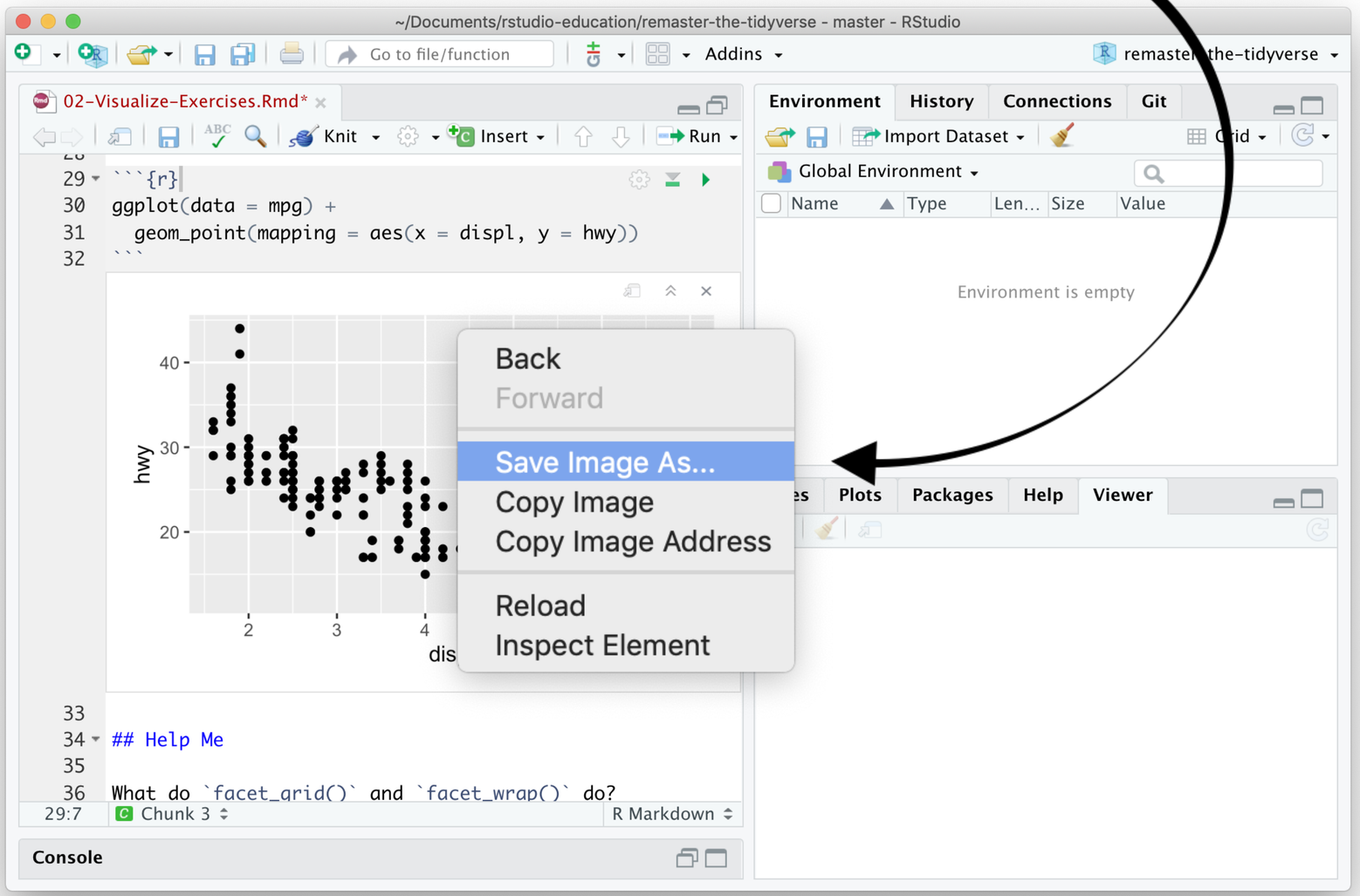

```
q + facet_wrap(~ Cell.Region)
```

facet_grid() - 2D grid, rows ~ cols, . for no split
facet_wrap() - 1D ribbon wrapped into 2D

Saving graphs

GUI method

Right click on the plot



Code method

ggsave() saves the last plot.
Uses size on screen:

```
ggsave("my-plot.pdf")
ggsave("my-plot.png")
```

Specify size in inches

```
ggsave("my-plot.pdf", width = 6, height
```

Q: But where will it save it?

A: Alongside your .Rmd

Themes

A ggplot2 template

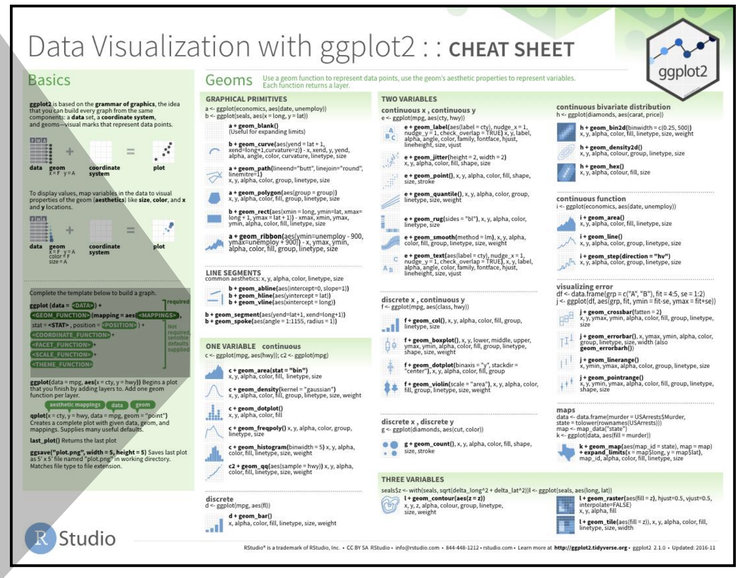
Make any plot by filling in the parameters of this template

Complete the template below to build a graph.

```
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



Your Turn

Open the **Day-3---ggplot2-hw.Rmd**