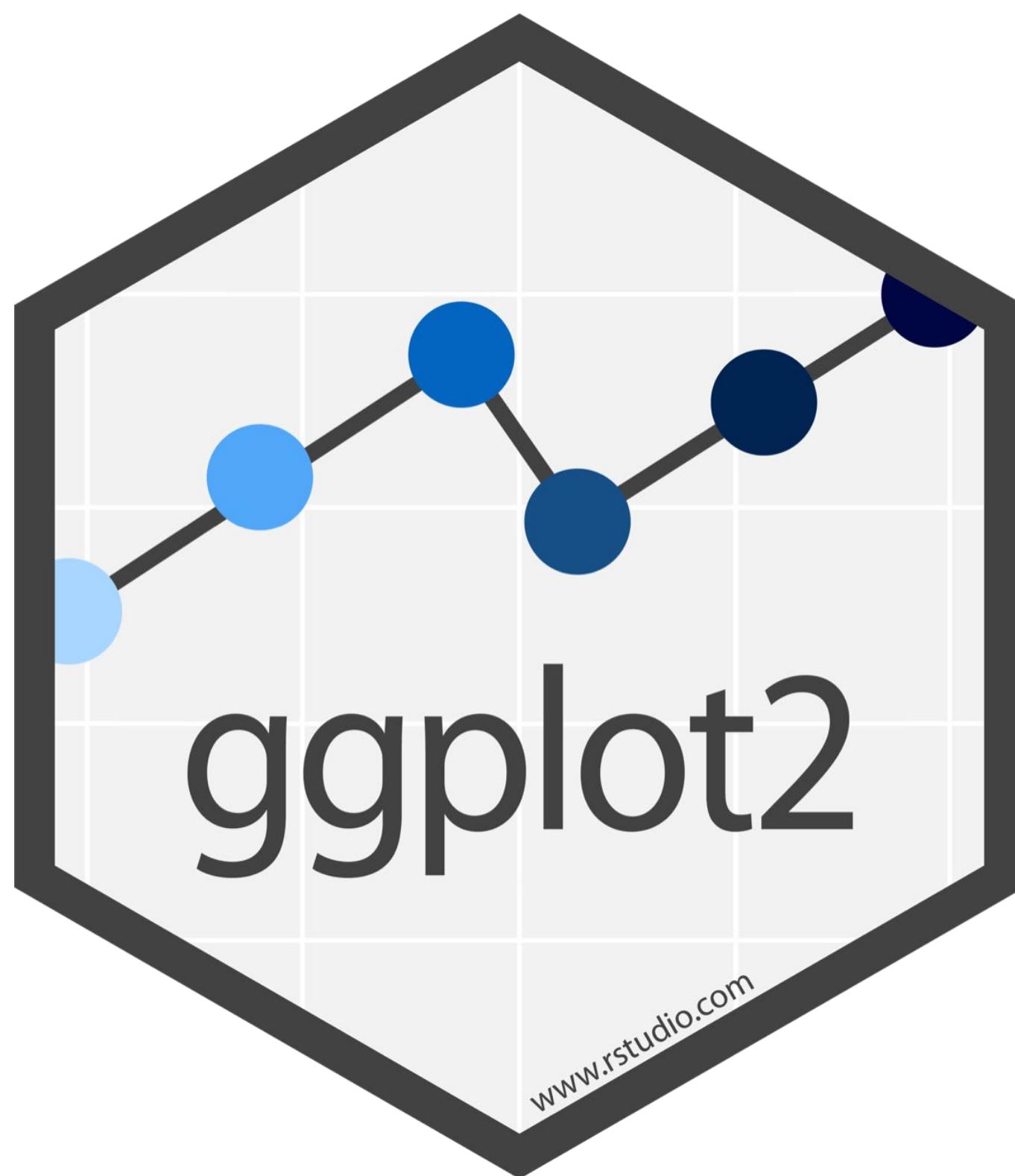


Visualize Data with

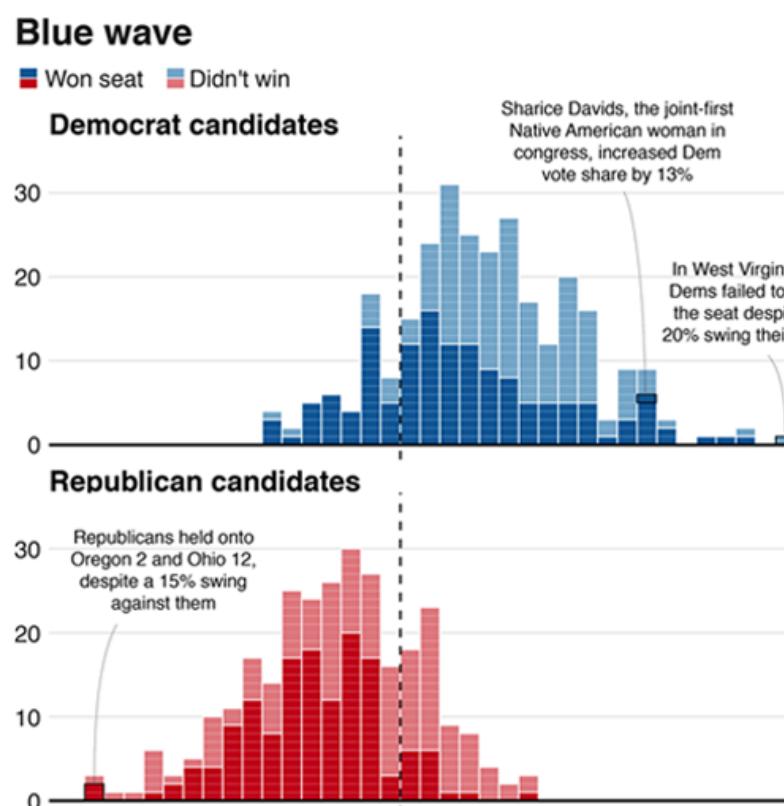


What can ggplot do?

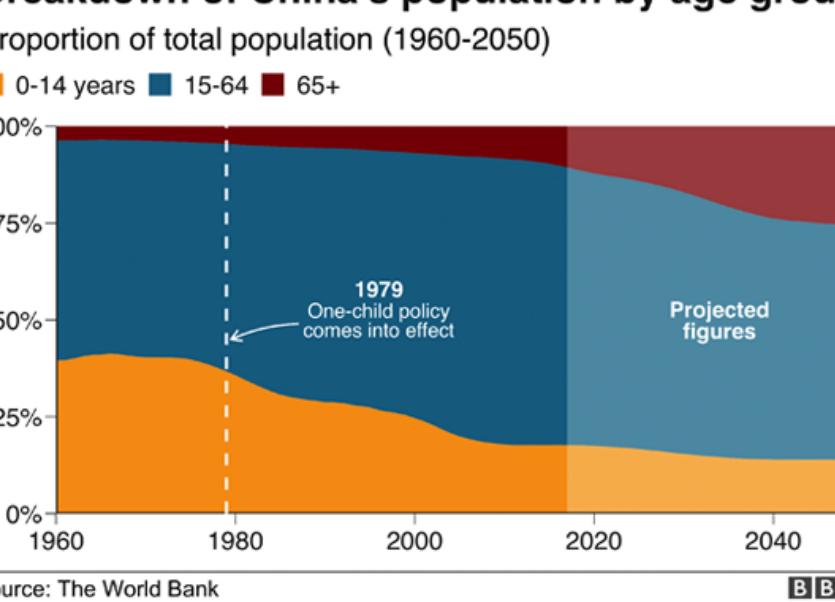
“ggplot2 gives you far more control and creativity than a chart tool and allows you to **go beyond a limited number of graphics**. Working with scripts **saves a huge amount of time and effort**, in particular when working with data that needs **updating regularly, with reproducibility** a key requirement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

-BBC Visual and Data Journalism

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



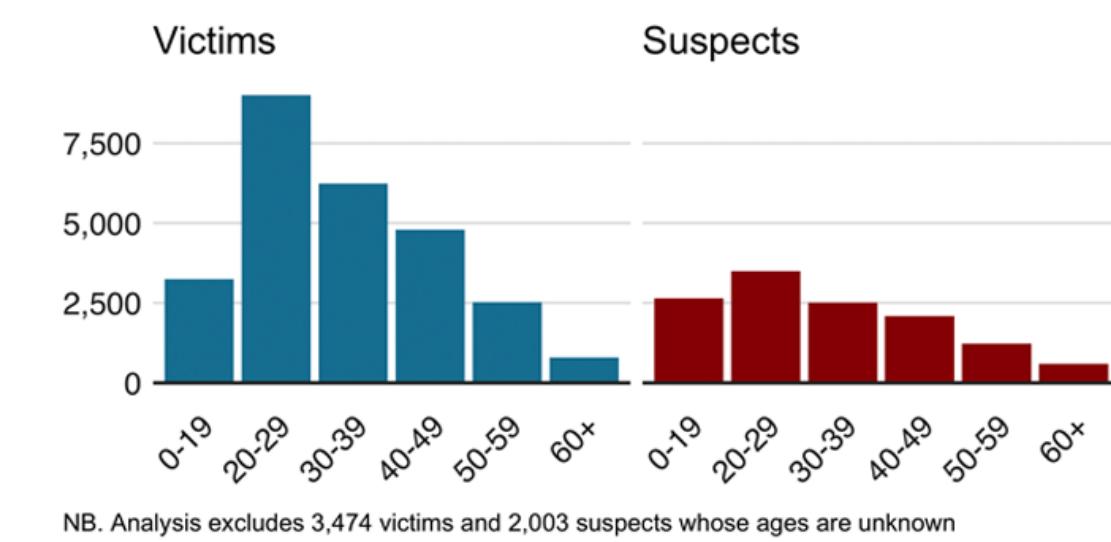
Breakdown of China's population by age group



Source: The World Bank

BBC Source: Parliament

Homophobic hate crimes are mainly committed by young people on young people
Number in each age group 2014 - 2017

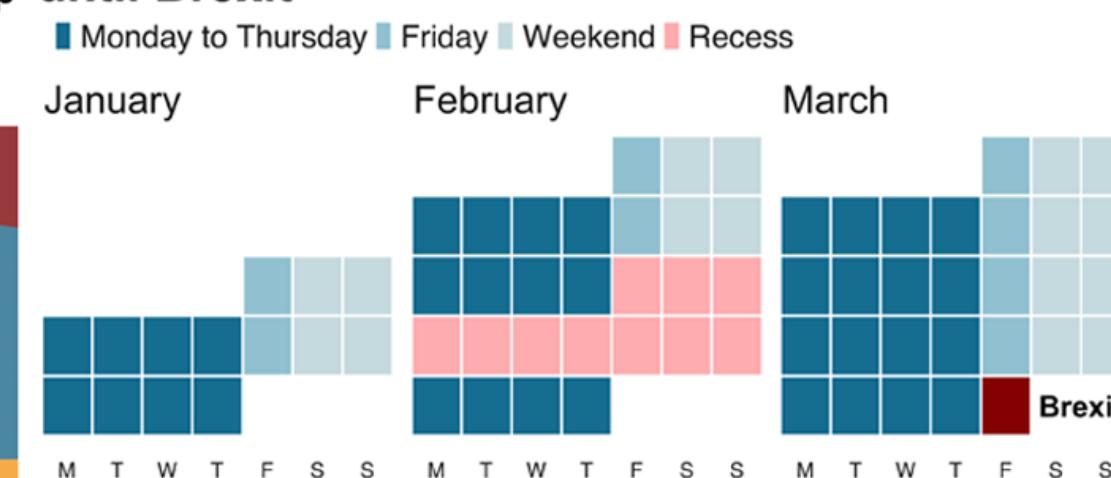


NB. Analysis excludes 3,474 victims and 2,003 suspects whose ages are unknown

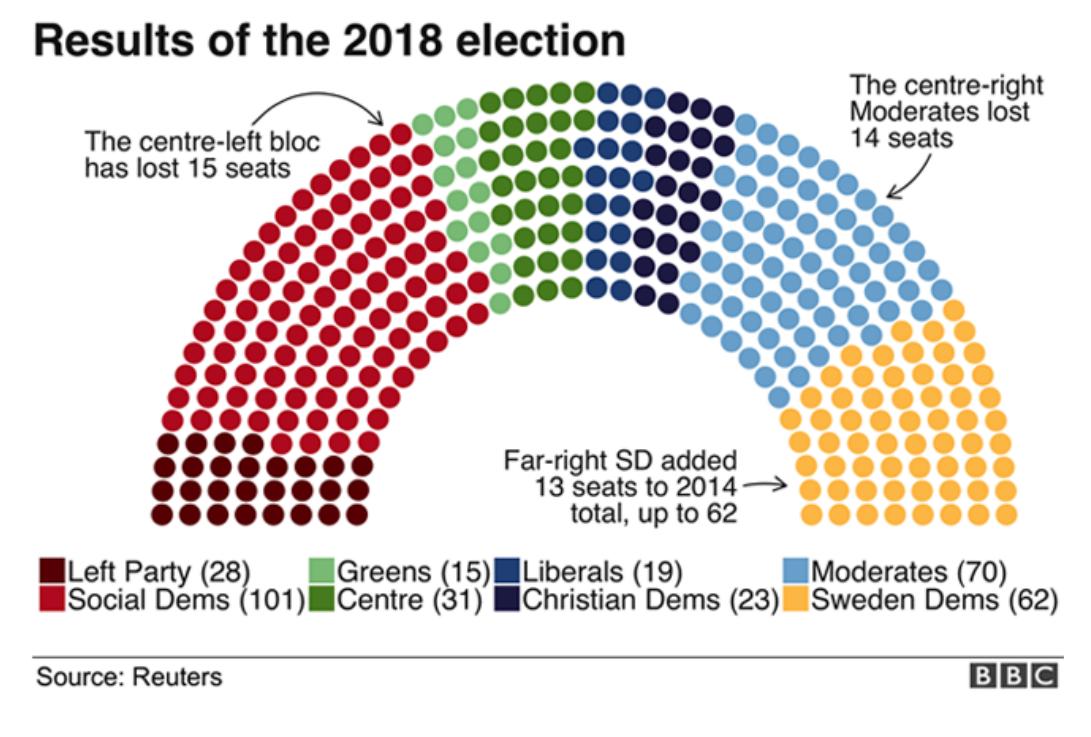
Source: BBC Freedom of Information requests to UK police forces

BBC

The Commons has 36 normal working days until Brexit

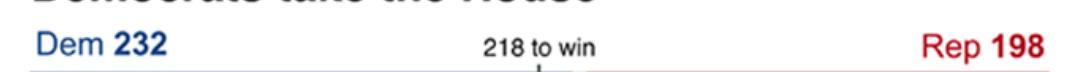


Note: The House of Commons sometimes sits on Fridays to debate individual MPs' bills

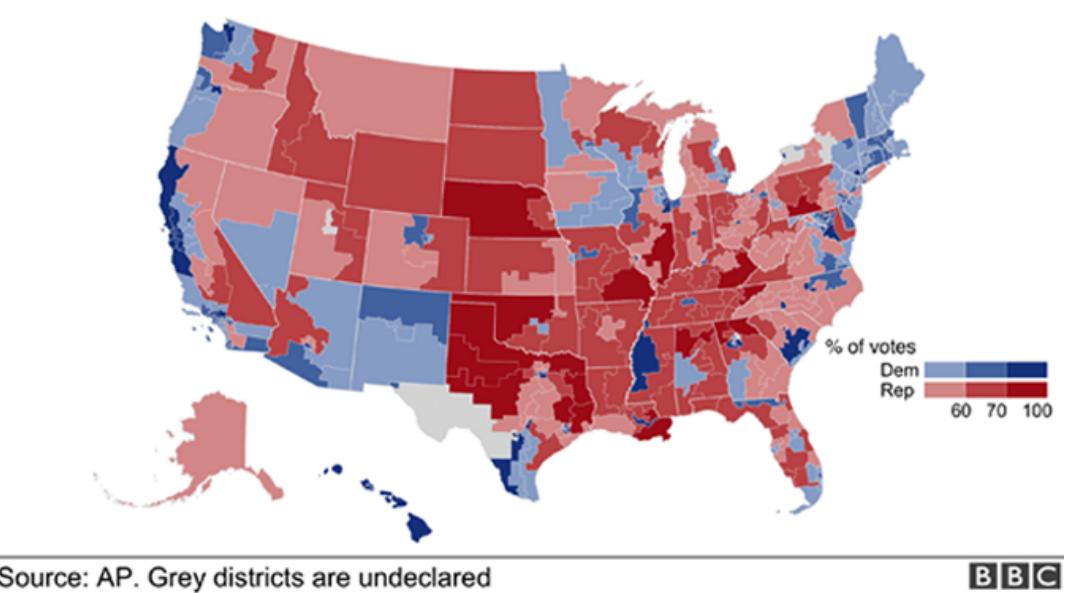


Source: Reuters

Democrats take the House



BBC



BBC Source: AP. Grey districts are undeclared

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

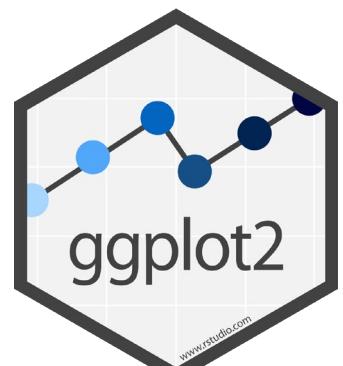
- John Tukey

chromosome

Basic data for all human chromosomes.

chromosome

chromosome	length_mm	basepairs	variations	protein_codinggg~	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



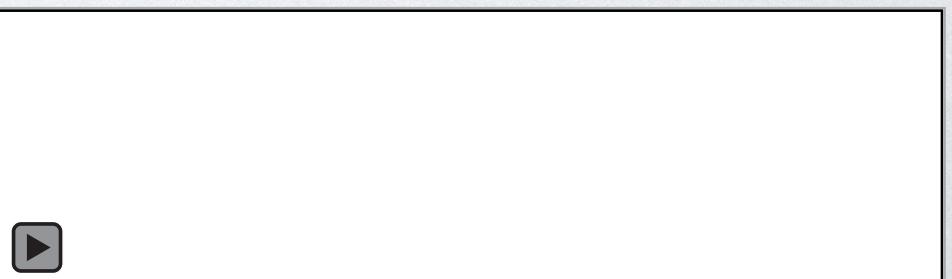
Question

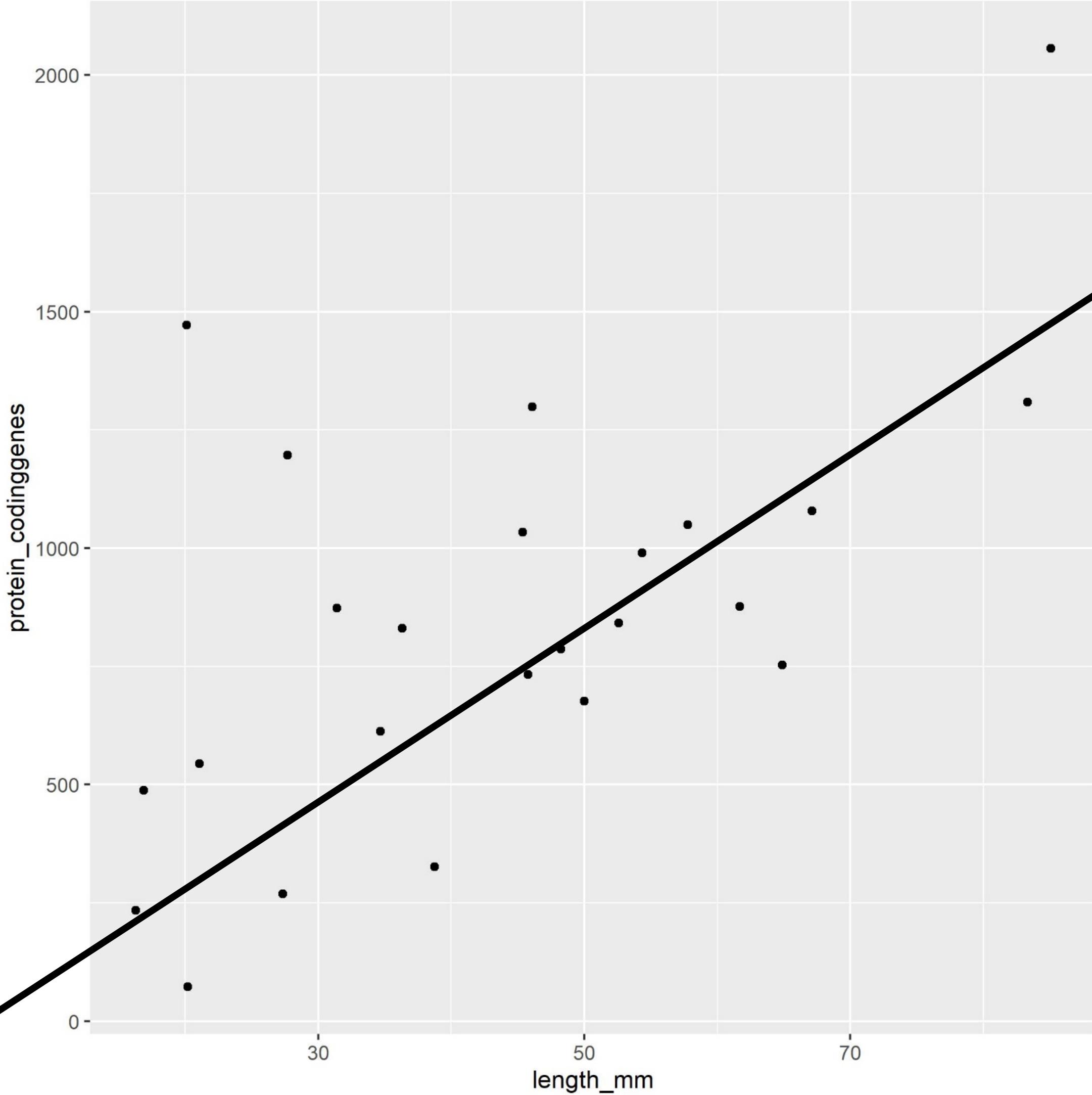
What relationship do you expect to see between chromosome size (`length_mm`) and number of genes (`protein_codinggenes`)?

Exercise 1

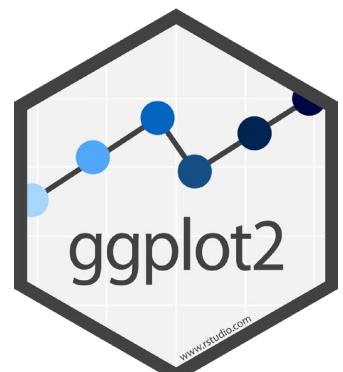
Run this code in **Day-3---ggplot2-lecture.Rmd** to make a graph. Pay strict attention to spelling, capitalization, and parentheses!

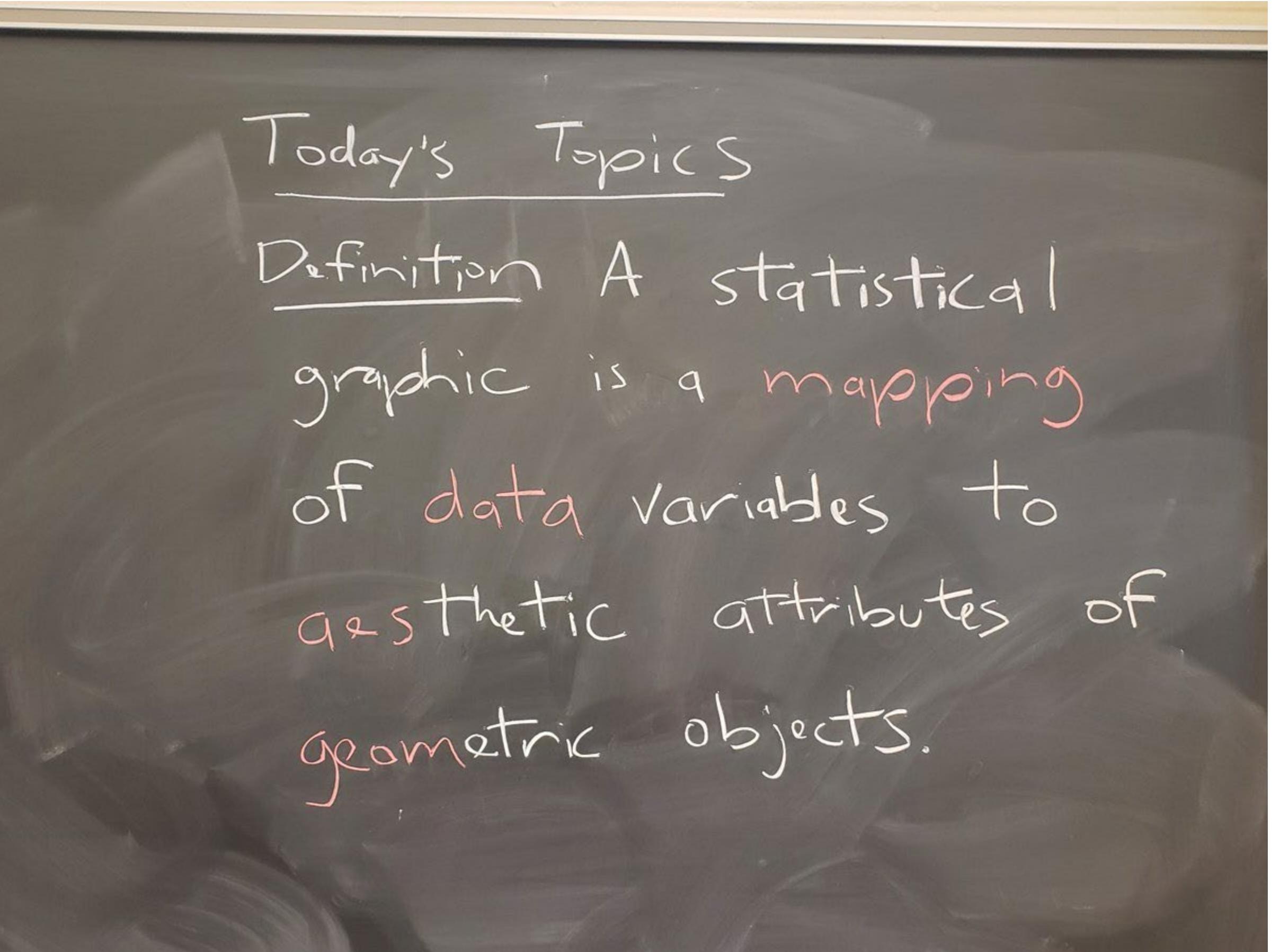
```
ggplot(data = chromosome) +  
  geom_point(mapping=aes(x = length_mm, y = protein_codinggenes))
```





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

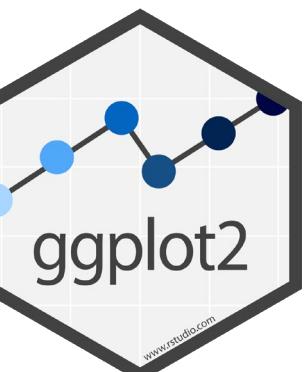




```
ggplot(data = chromosome) +  
  geom_point(mapping=aes(x = length_mm, y = protein_codinggenes))
```

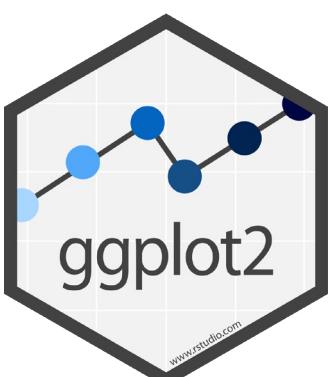
1. "Initialize" a plot with `ggplot()`
2. Add layers with `geom_` functions

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



data

+ before new line

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

type of layer

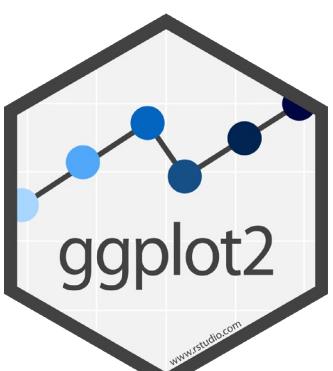
aes()

x variable

y variable

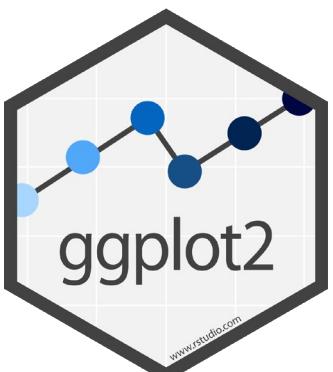
A template

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))  
ggplot(data = chromosome) +  
  geom_point(mapping=aes(x = length_mm, y =protein_codinggenes))
```



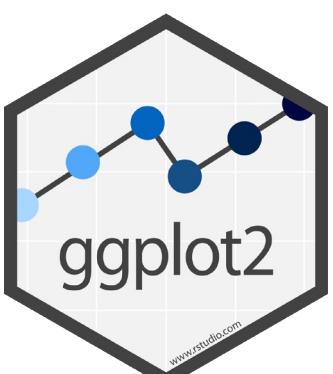
A template

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



A template

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

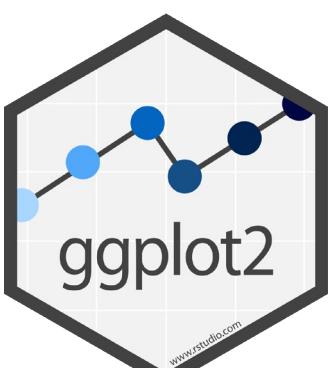


A template

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

==

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

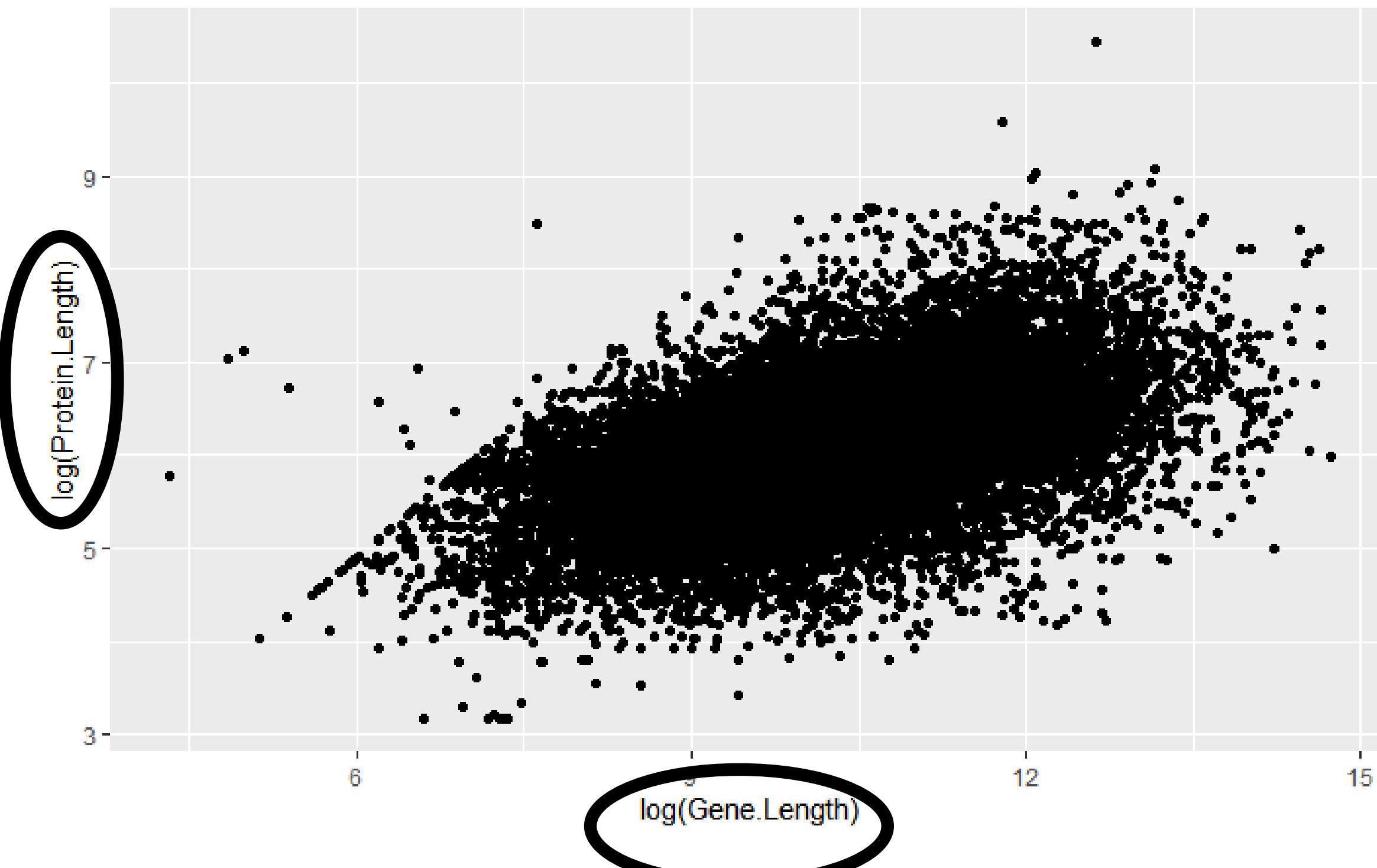


Mappings

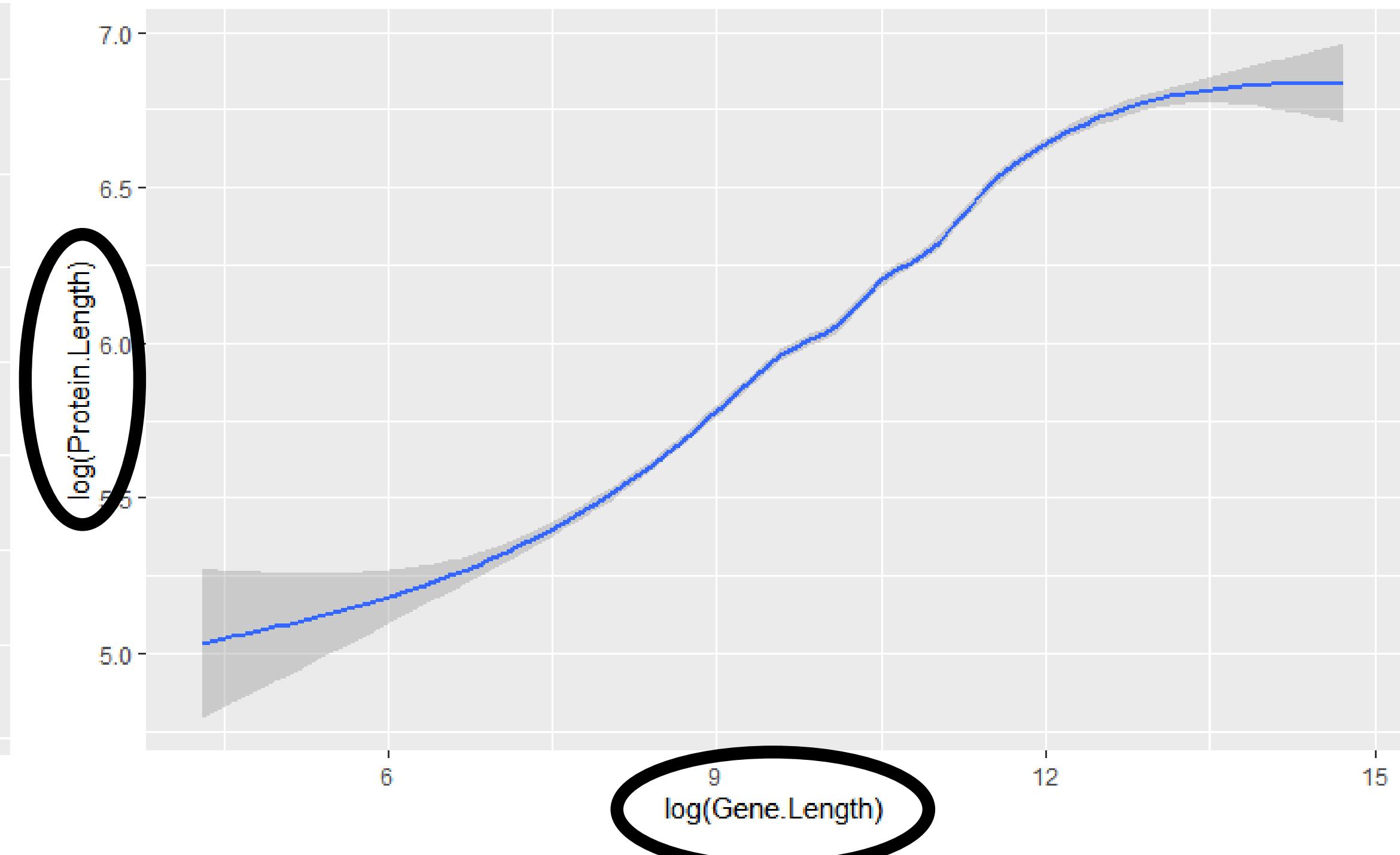
Geoms

How are these plots similar?

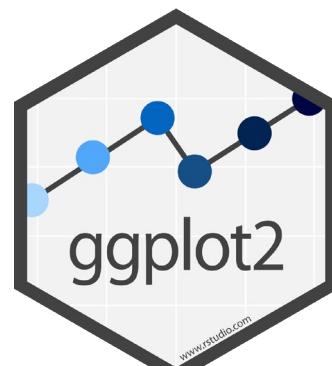
Same: x var, y var, data



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

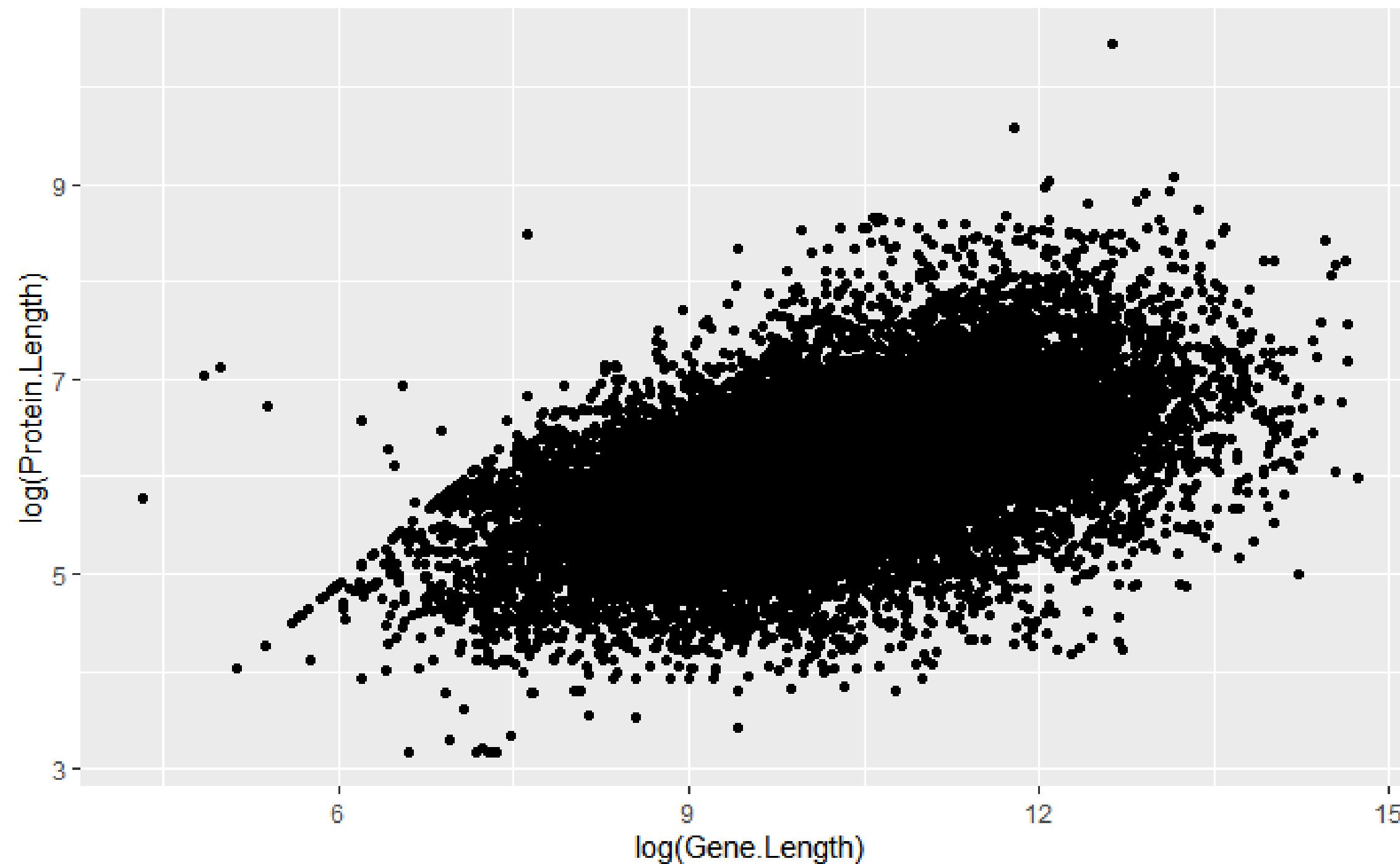


```
ggplot(uniprot_region)+  
  geom_smooth(aes(x=log(Gene.Length),  
                  y=log(Protein.Length)))
```

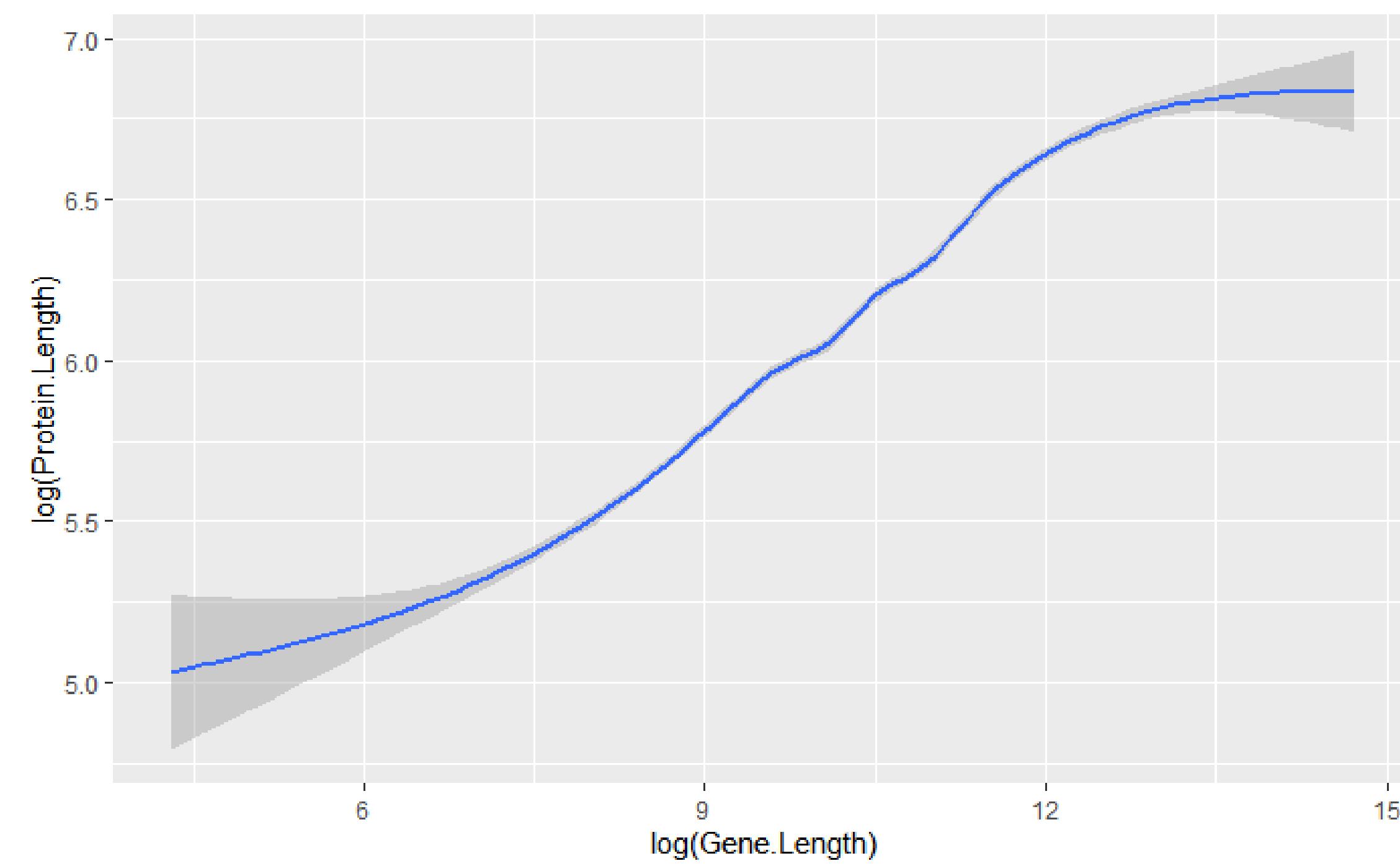


How are these plots different?

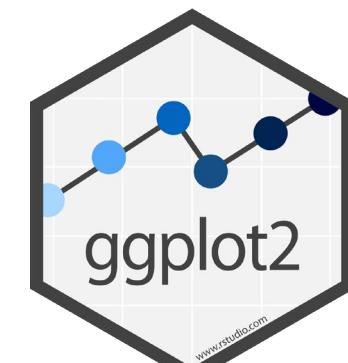
Different: geometric object (geom),
e.g. the visual object used to represent the data



```
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

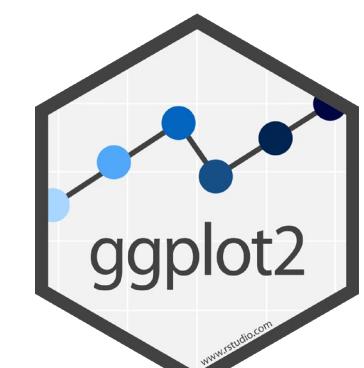
The screenshot shows the RStudio Cloud interface with the 'Cheat Sheets' page open. A red arrow points from the text 'Click CheAT Sheets in the sidebar' to the 'Cheat Sheets' link in the sidebar menu.

Deep Learning with Keras
Keras is a high-level neural networks API developed with a focus on enabling fast experimentation. Keras supports both convolution based networks and recurrent networks (as well as combinations of the two), runs seamlessly on both CPU and GPU devices, and is capable of running on top of multiple back-ends including TensorFlow, CNTK, and Theano.
Updated December 2017

Dates and Times
Lubridate makes it easier to work with dates and times in R. This lubridate cheatsheet covers how to round dates, work with time zones, extract elements of a date or time, parse dates into R and more. The back of the cheatsheet describes lubridate's three timespan classes: periods, durations, and intervals; and explains how to do math with date-times.
Updated December 2017

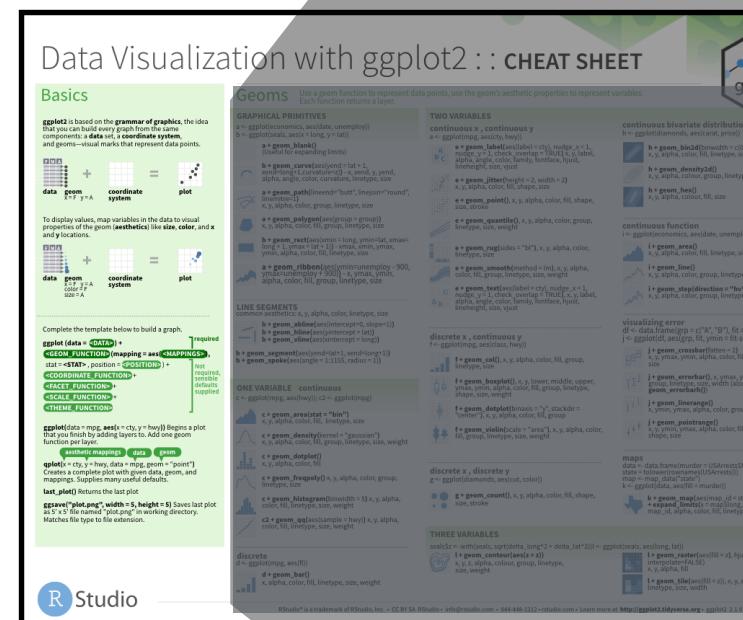
Work with Strings
The stringr package provides an easy to use toolkit for working with strings, i.e. character data, in R. This cheatsheet guides you through stringr's functions for manipulating strings. The back

Apply Functions
The purrr package makes it easy to work with lists and functions. This cheatsheet will remind you how to manipulate lists with purrr as well as how to apply functions iteratively to each element of a list



geom_ functions

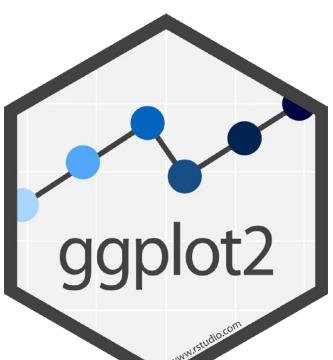
Each requires a mapping argument.



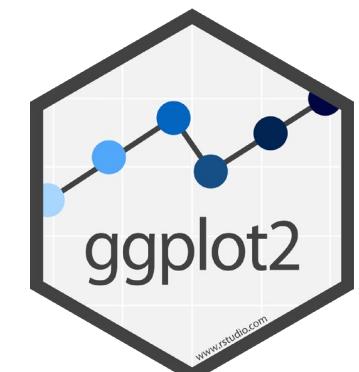
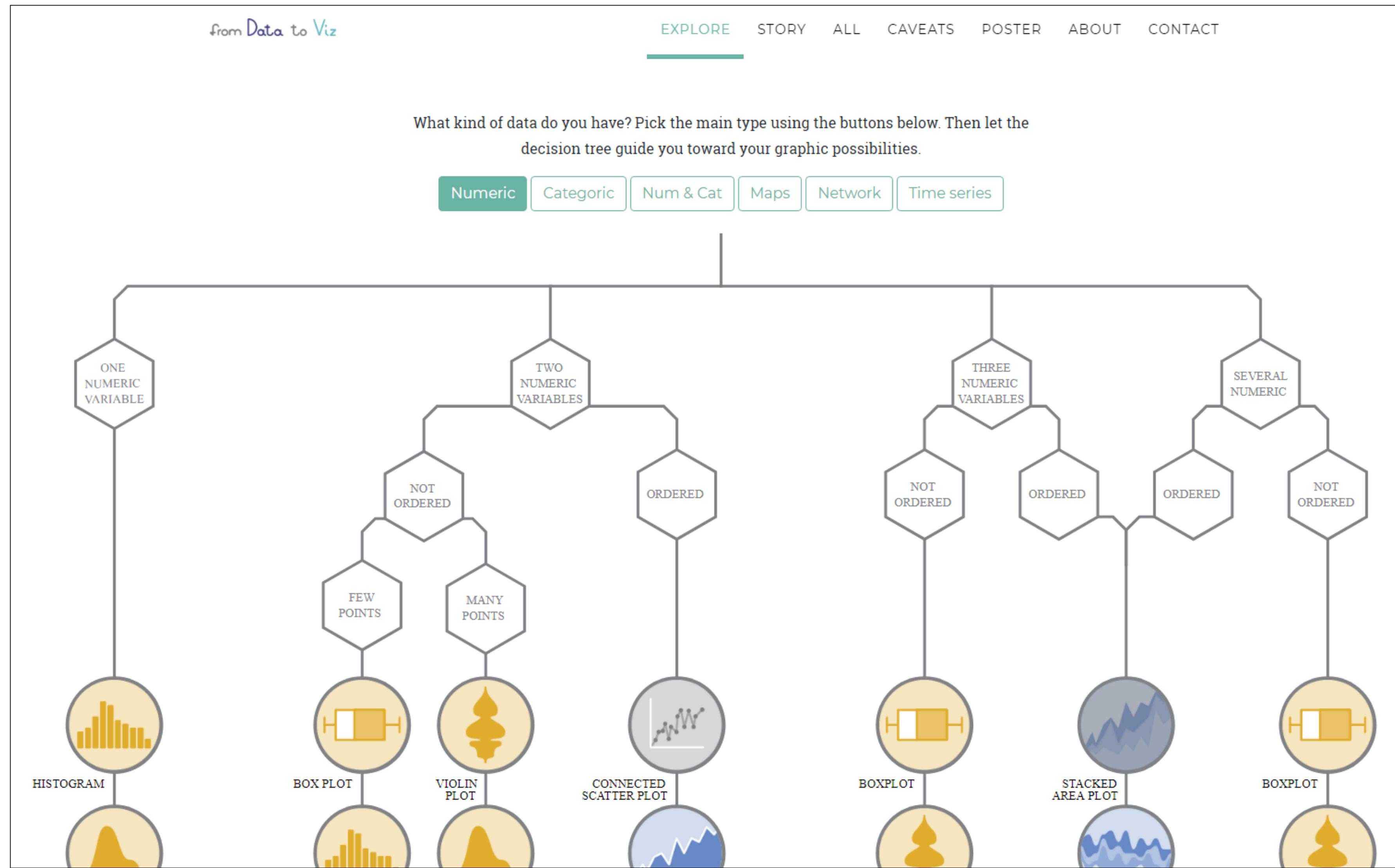
Geoms Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

ggplot2

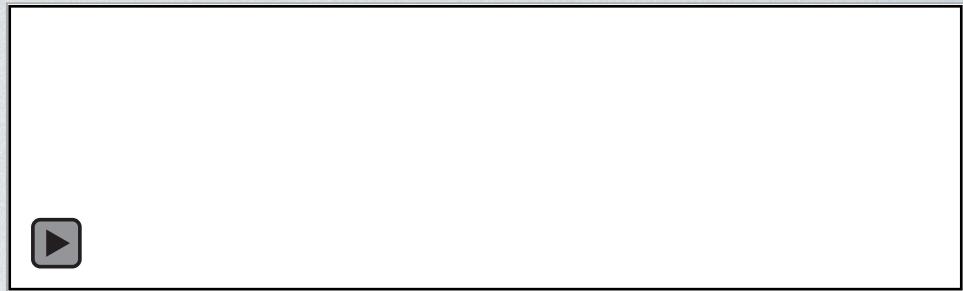
GRAPHICAL PRIMITIVES	TWO VARIABLES	continuous bivariate distribution
a + geom_blank() (Useful for expanding limits)	e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust	h + geom_bin2d(binwidth = c(0.25, 500)) x, y, alpha, color, fill, linetype, size, weight
b + geom_curve(aes(yend = lat + 1, xend = long + 1, curvature = z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size	e + geom_jitter(height = 2, width = 2) x, y, alpha, color, fill, shape, size	h + geom_density2d() x, y, alpha, colour, group, linetype, size
a + geom_path(lineend = "butt", linejoin = "round", linemetre = 1) x, y, alpha, color, group, linetype, size	e + geom_point(), x, y, alpha, color, fill, shape, size, stroke	h + geom_hex() x, y, alpha, colour, fill, size
a + geom_polygon(aes(group = group)) x, y, alpha, color, fill, group, linetype, size	e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight	
b + geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size	e + geom_rug(sides = "bl"), x, y, alpha, color, linetype, size	
a + geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, 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	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
LINE SEGMENTS common aesthetics: x, y, alpha, color, linetype, size	discrete x , continuous y	visualizing error
b + geom_abline(aes(intercept = 0, slope = 1))	f + geom_col(), x, y, alpha, color, fill, group, linetype, size	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))
b + geom_hline(aes(yintercept = lat))	f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight	j + geom_crossbar(fatten = 2) x, y, ymax, ymin, alpha, color, fill, group, linetype, size
b + geom_vline(aes(xintercept = long))	f + geom_dotplot(binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group	j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also geom_errorbarh())
b + geom_segment(aes(yend = lat + 1, xend = long + 1))	f + geom_dotplot()	j + geom_linerange() x, ymin, ymax, alpha, color, group, linetype, size
b + geom_spoke(aes(angle = 1:1155, radius = 1))	f + geom_hex() x, ymin, ymax, alpha, color, fill, group, linetype, size, weight	j + geom_pointrange() x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size
ONE VARIABLE continuous	discrete x , discrete y	maps
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)	g + geom_count(), x, y, alpha, color, fill, shape, size, stroke	data <- data.frame(murder = USArests\$Murder, state = tolower(rownames(USArests))) map <- map_data("state") k <- ggplot(data, aes(fill = murder))
c + geom_area(stat = "bin") x, y, alpha, color, fill, linetype, size		k + geom_map(aes(map_id = state), map = map) + expand_limits(x = map\$long, y = map\$lat), map_id, 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	THREE VARIABLES	
d <- ggplot(mpg, aes(fl))	seals\$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)) l <- ggplot(seals, aes(long, lat))	
d + geom_bar() x, alpha, color, fill, linetype, size, weight	l + geom_contour(aes(z = z))	
	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	



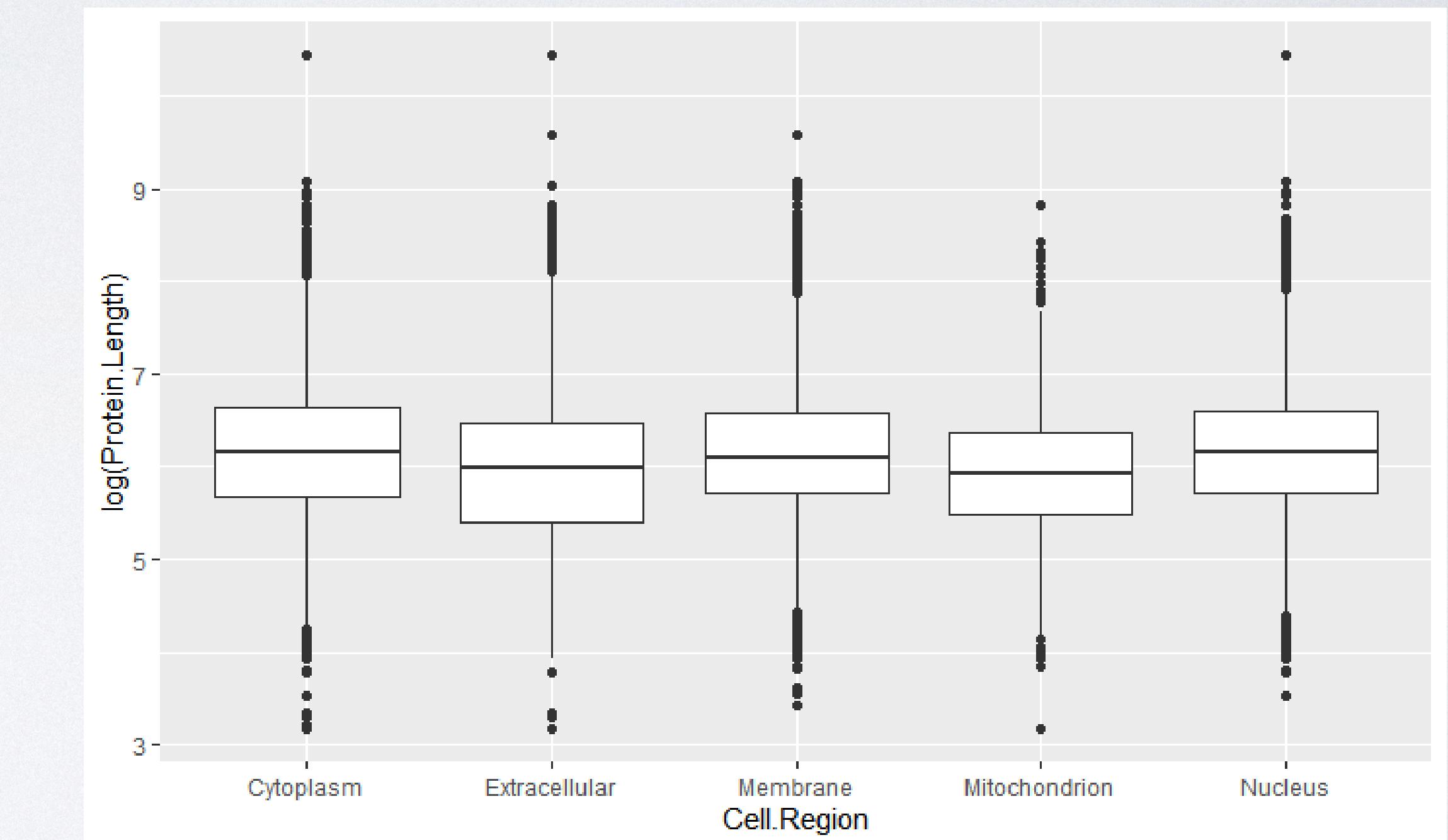
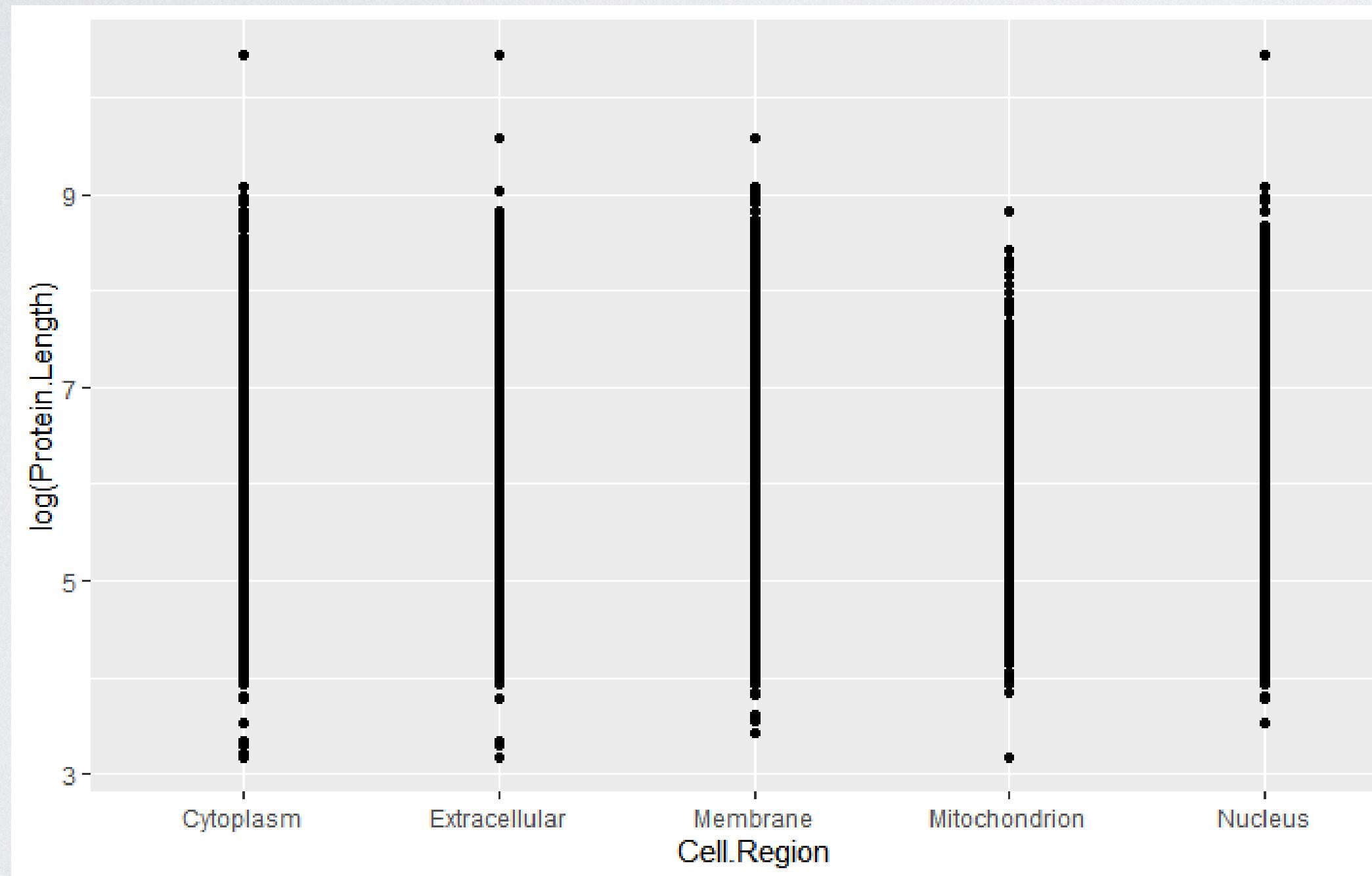
<https://www.data-to-viz.com>



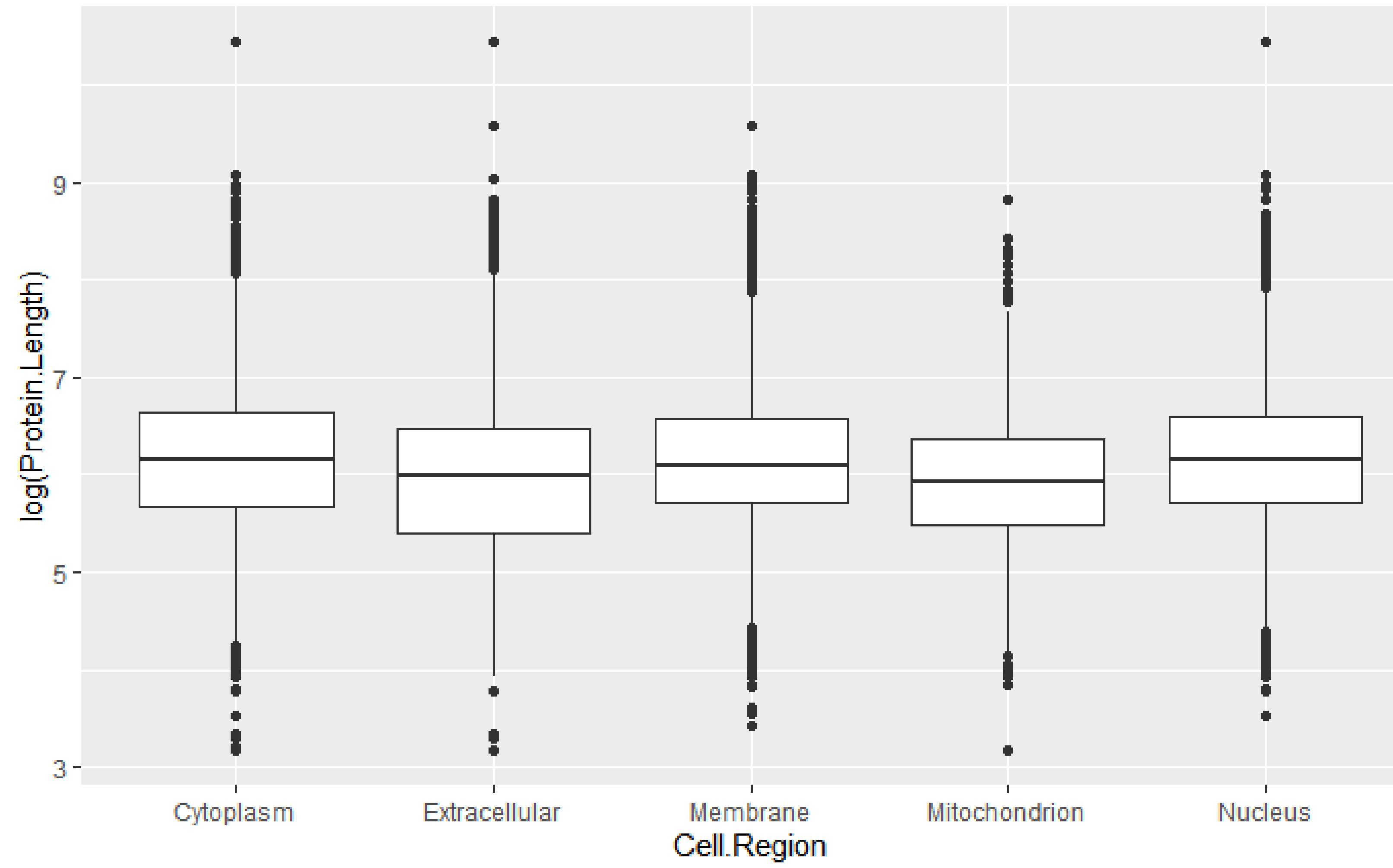
Exercise 2



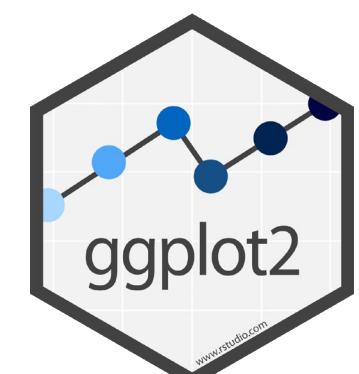
Decide how to replace this scatterplot with one that draws boxplots.
Use the cheatsheet. Try your best guess.



```
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)))
```



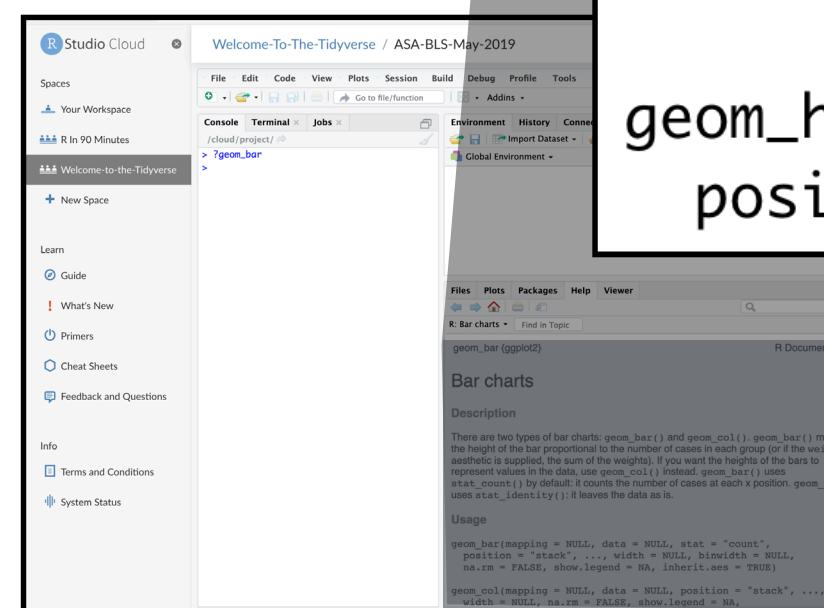
"Help" pages

To open the documentation for a function, type

```
?geom_histogram
```



function name (no parentheses)



geom_freqpoly {ggplot2}

R Documentation

Histograms and frequency polygons

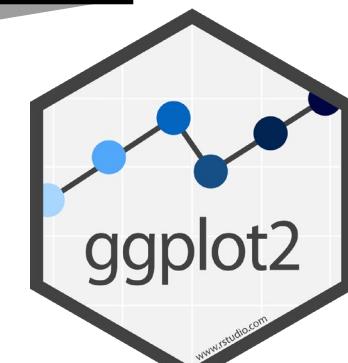
Description

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



ggplot2 part of the tidyverse
3.2.1

Reference

Articles

News

Extensions



Reference

Plot basics

All ggplot2 plots begin with a call to `ggplot()`, supplying default data and aesthetic 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()`

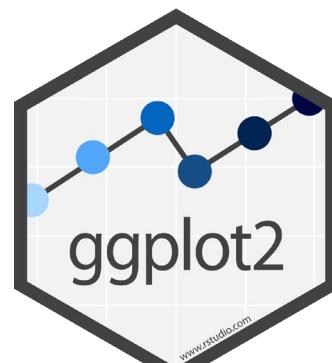
Reference lines: horizontal, vertical, and diagonal

`geom_bar() geom_col()`

Bar charts

Contents

- [Plot basics](#)
- [Layer: geoms](#)
- [Layer: stats](#)
- [Layer: position adjustment](#)
- [Layer: annotations](#)
- [Aesthetics](#)
- [Scales](#)
- [Guides: axes and legends](#)
- [Facetting](#)
- [Facetting: labels](#)
- [Coordinate systems](#)
- [Themes](#)
- [Programming with ggplot2](#)
- [Extending ggplot2](#)
- [Vector helpers](#)
- [Data](#)
- [Autoplot and fortify](#)

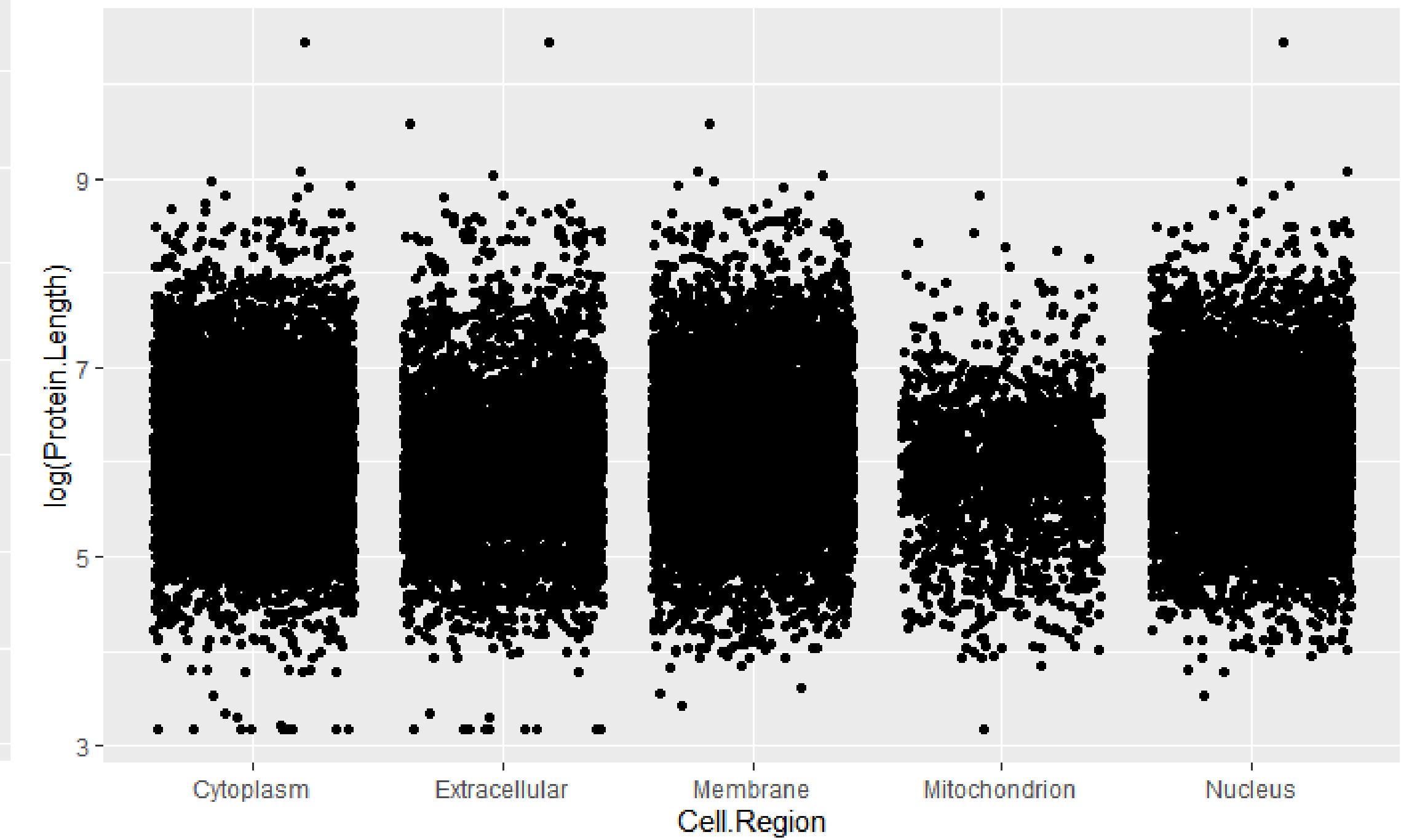
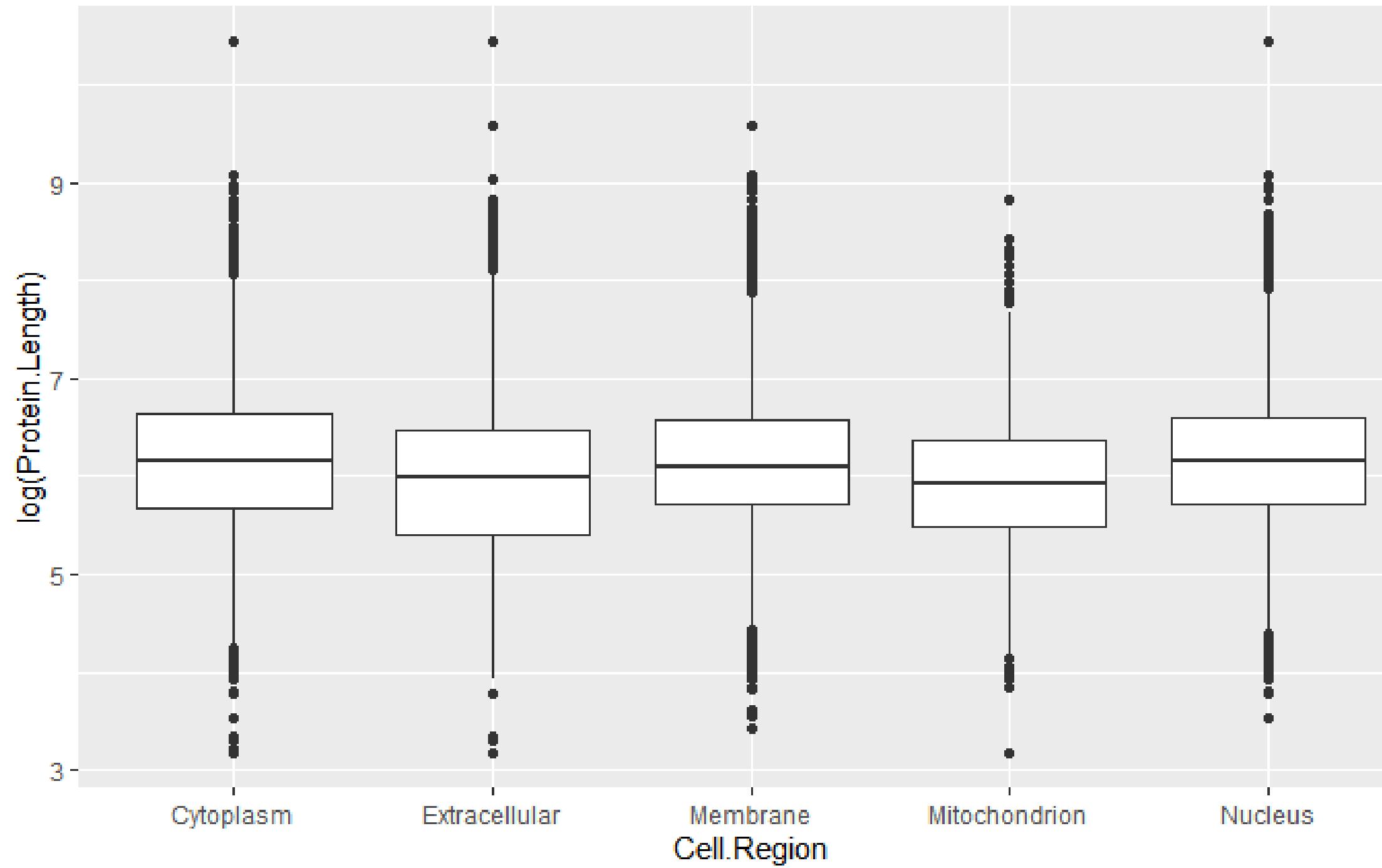


Question?

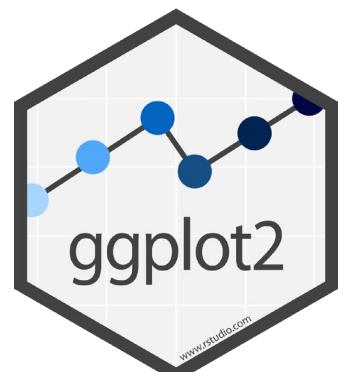
What will this code do?

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

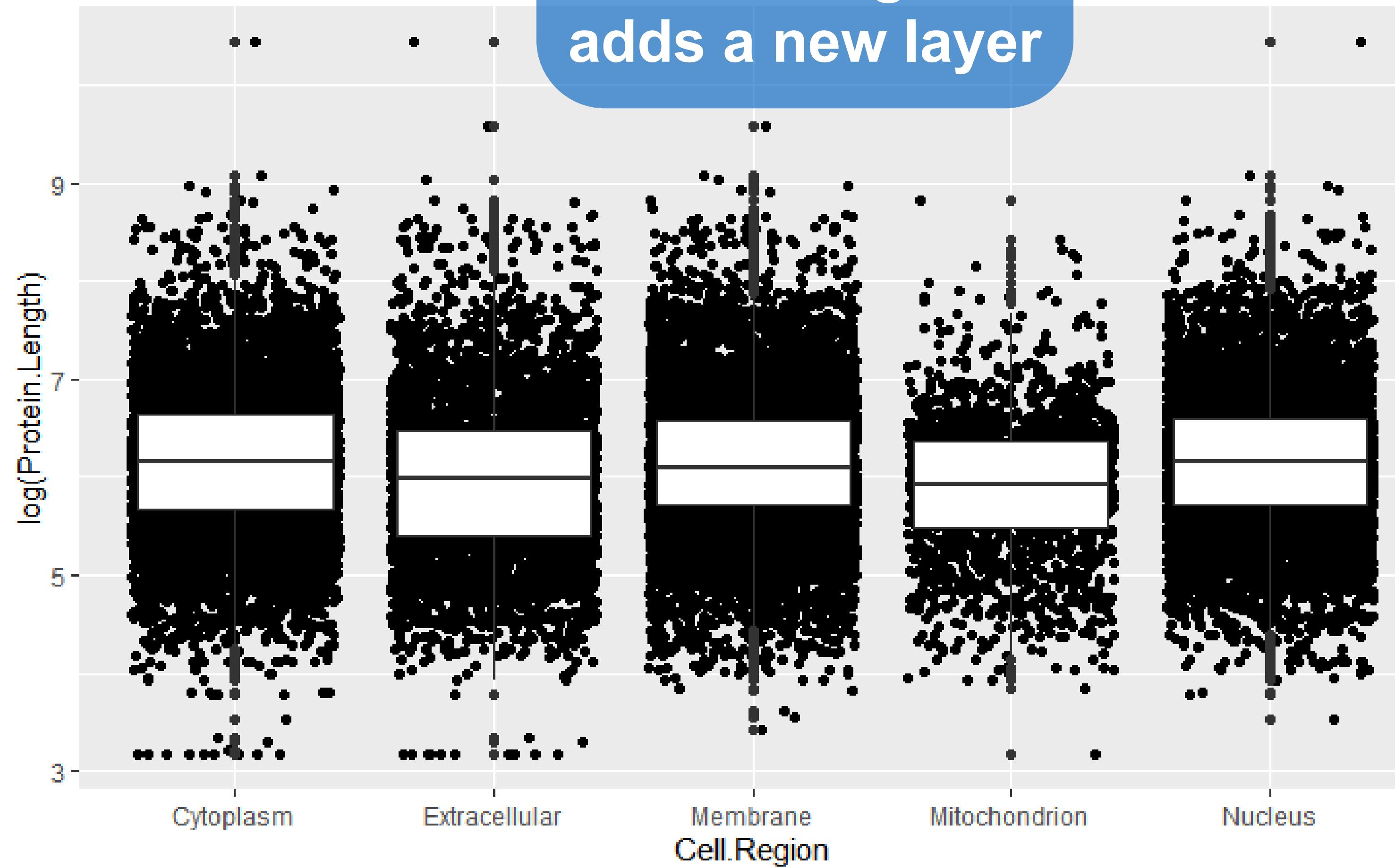
Each new geom adds a new layer



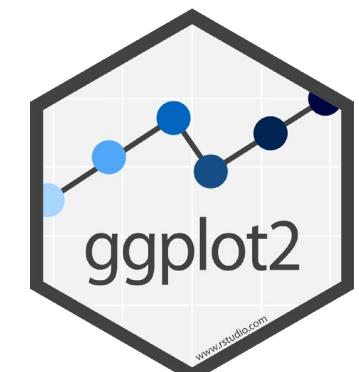
```
ggplot(uniprot_region)+  
  geom_jitter(aes(x=Cell.Region, y=log(Protein.Length)))+  
  geom_boxplot(aes(x=Cell.Region, y=log(Protein.Length)))
```



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)))
```

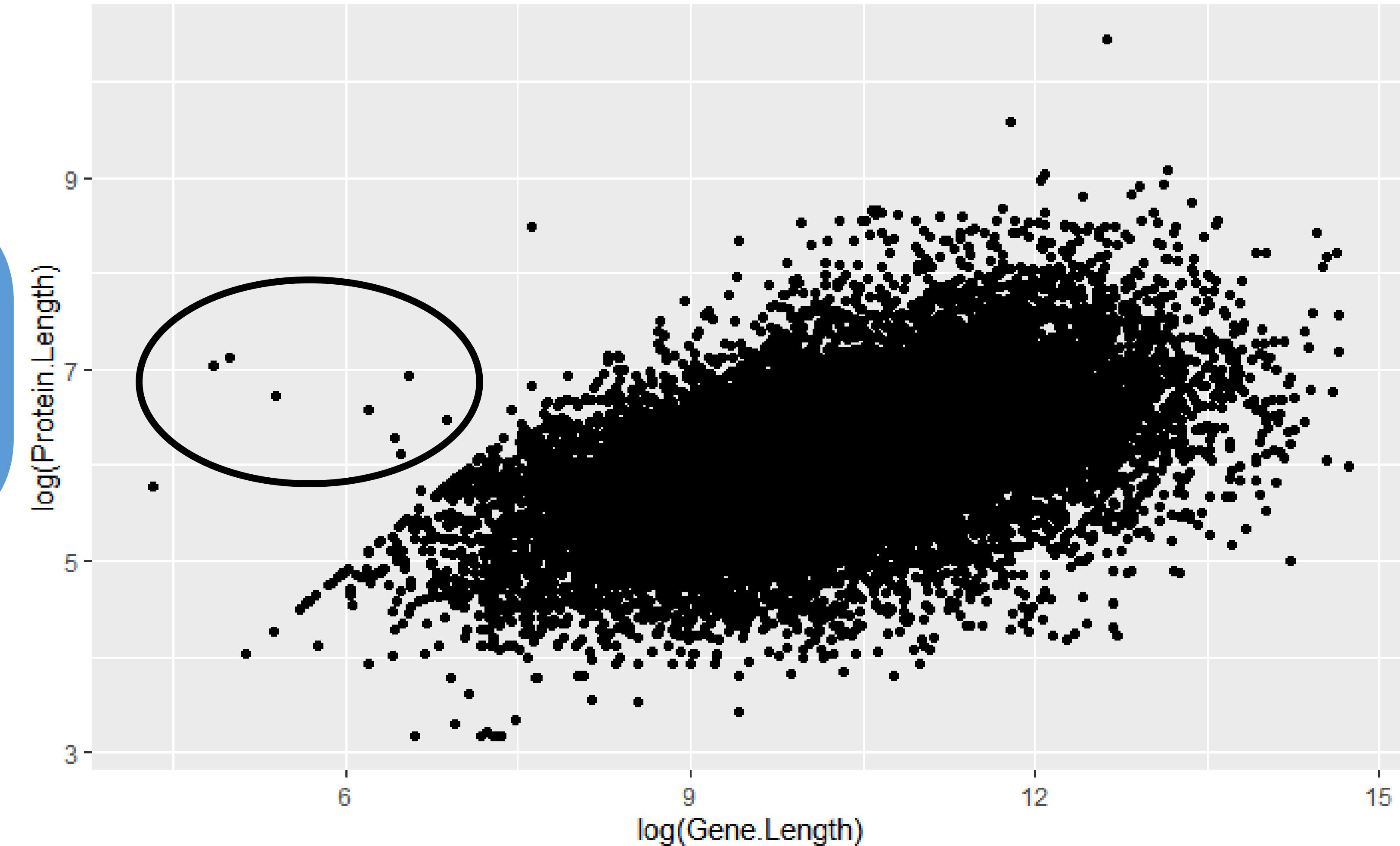


Aesthetics

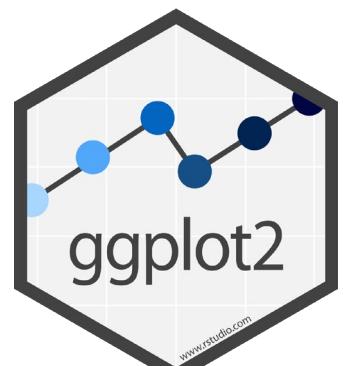
"The greatest value of a picture
is when it forces us to notice
what we never expected to see."

- John Tukey

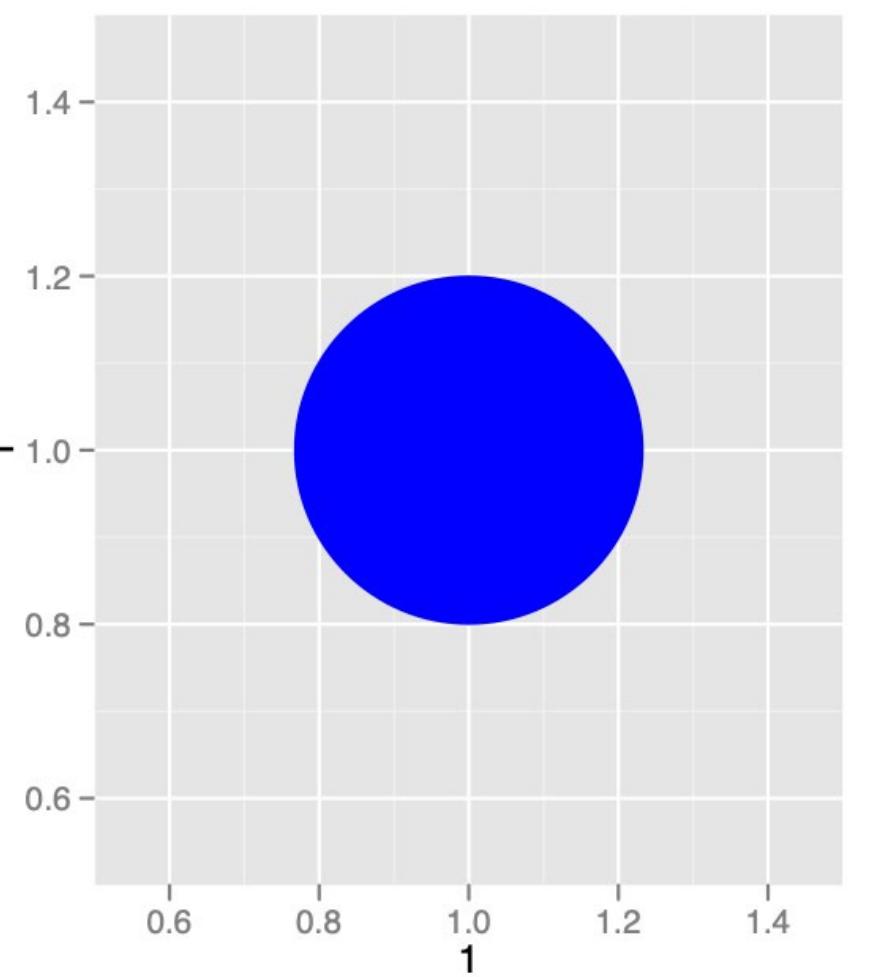
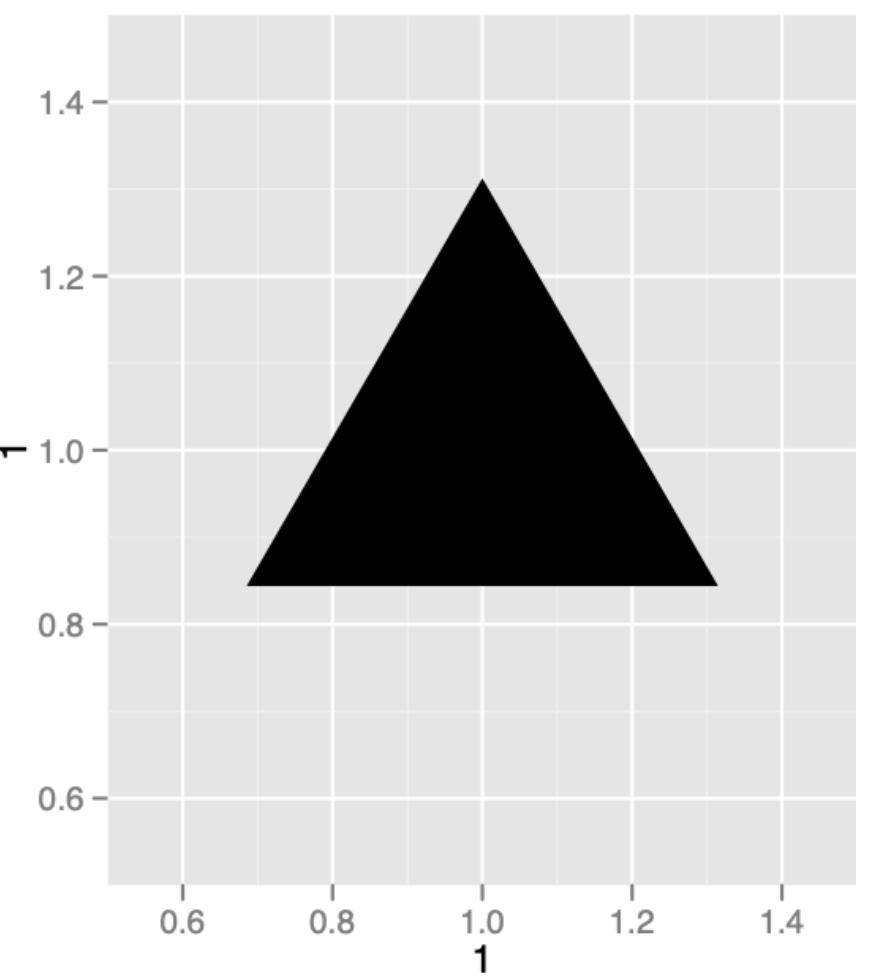
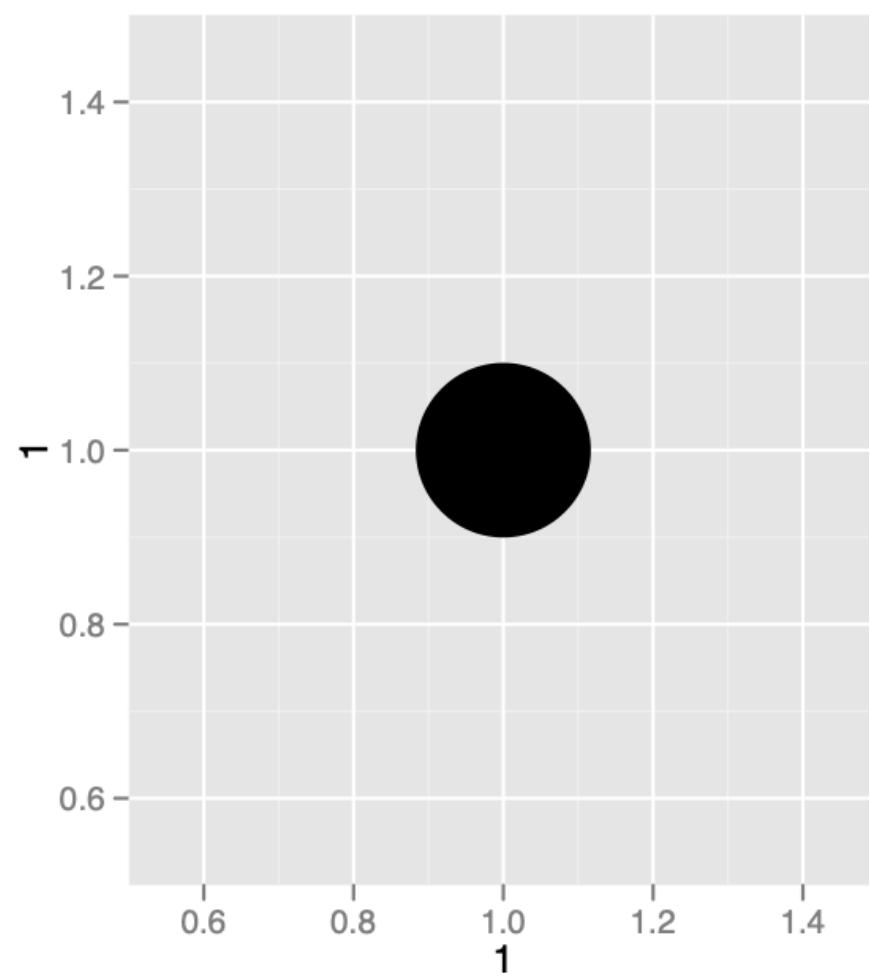
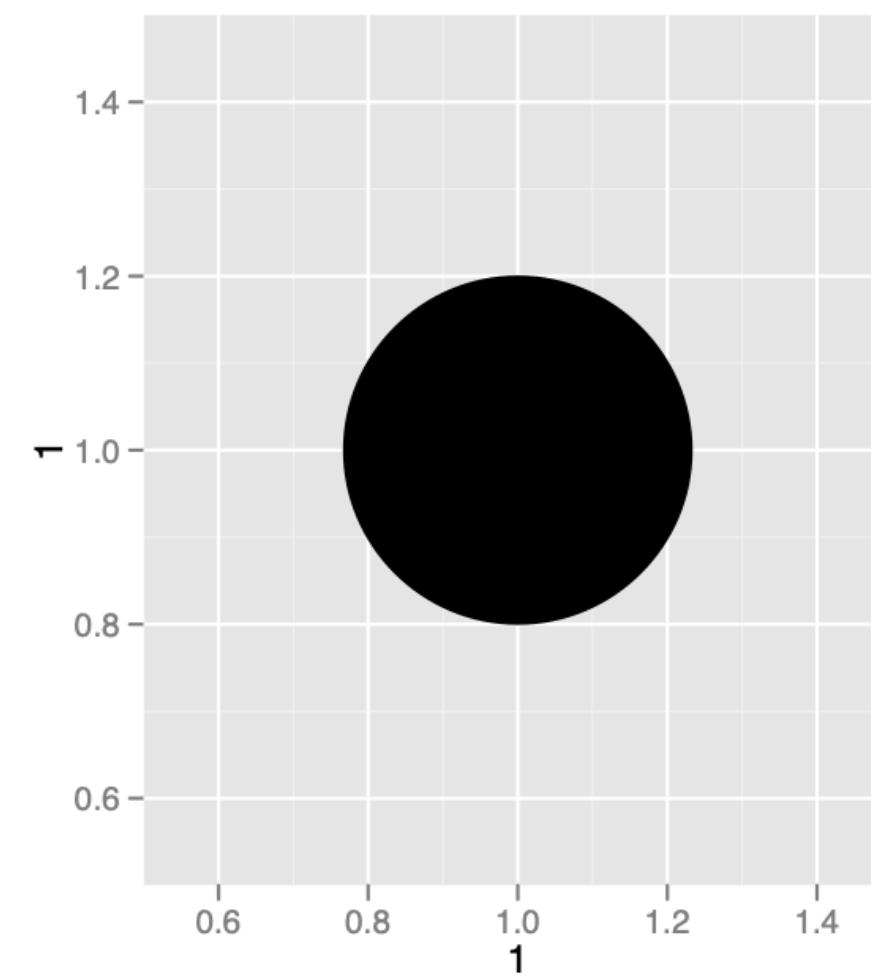
What are these
smaller genes
that make large
proteins?



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

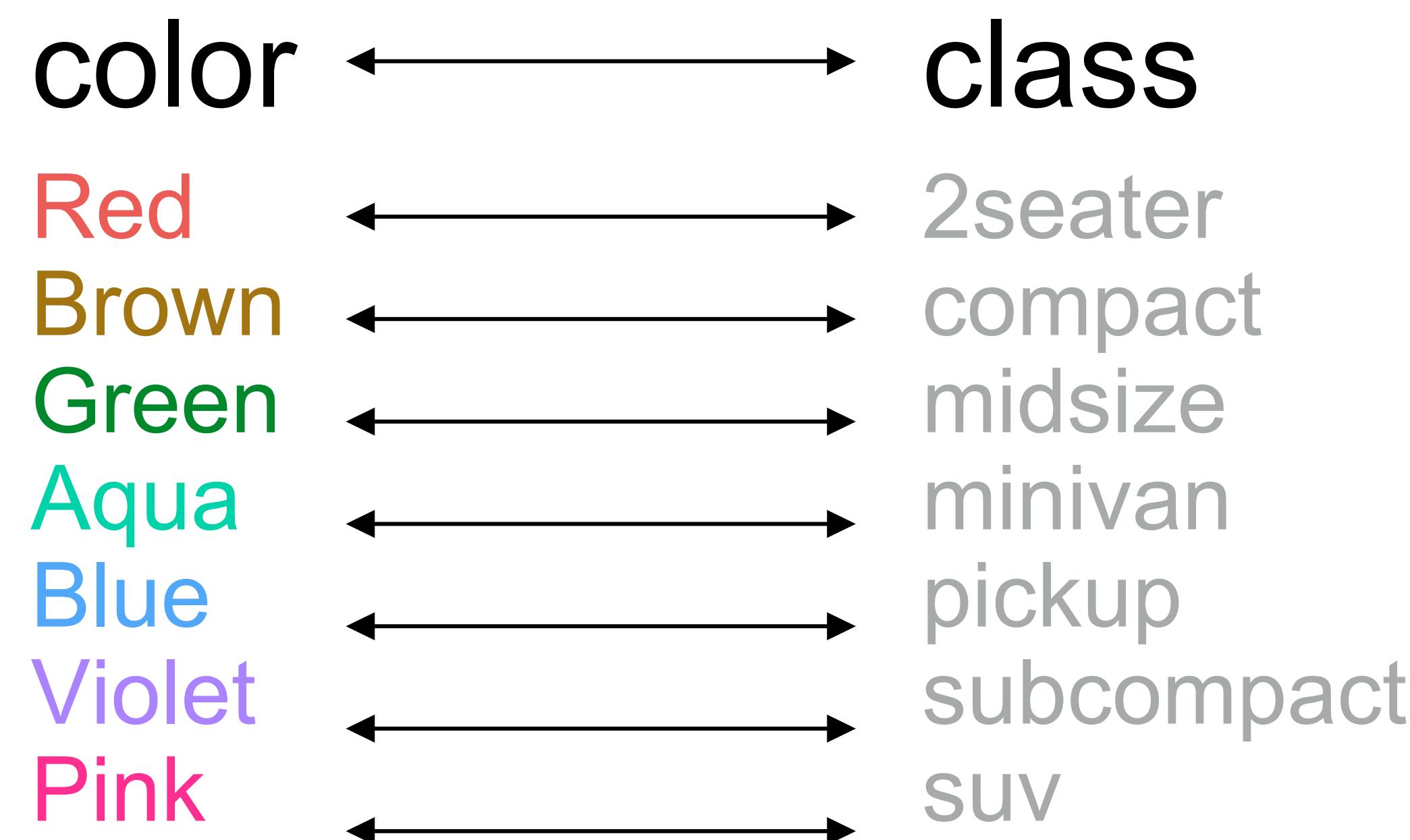


Aesthetics- Shape/Size



Aesthetics- Color

Visual Space Data Space



Aesthetics

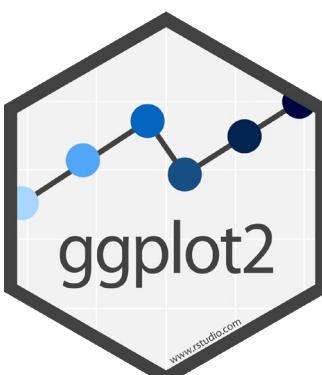
aesthetic
property

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

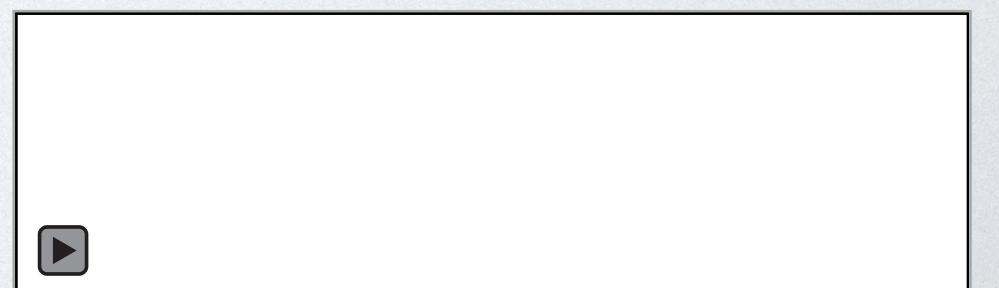


Exercise 3

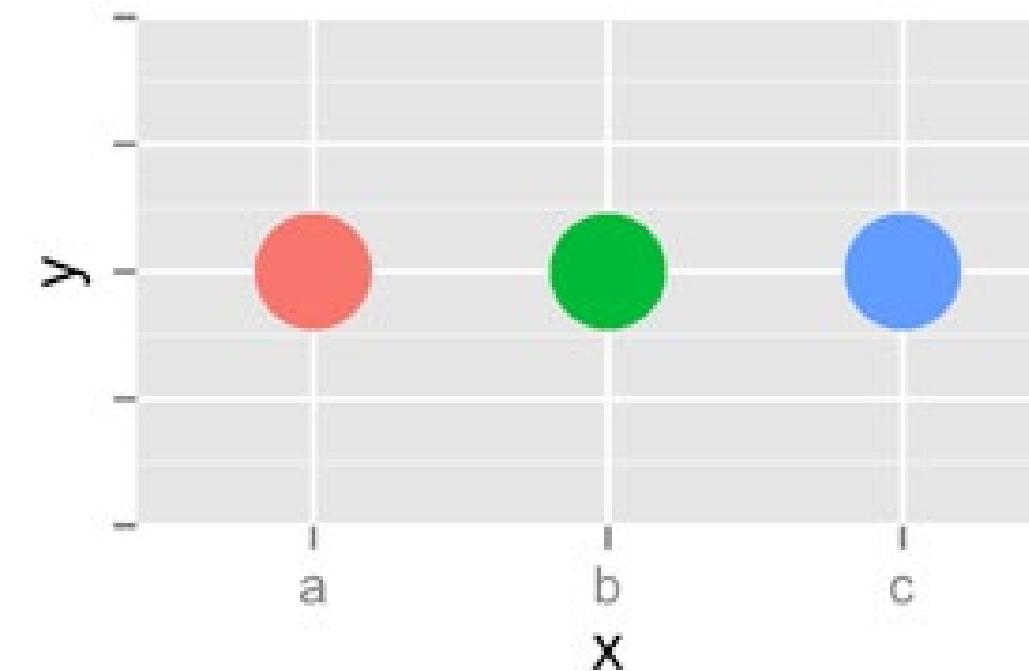
In the next chunk, add color, size, alpha, and shape aesthetics to your graph. Experiment.

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

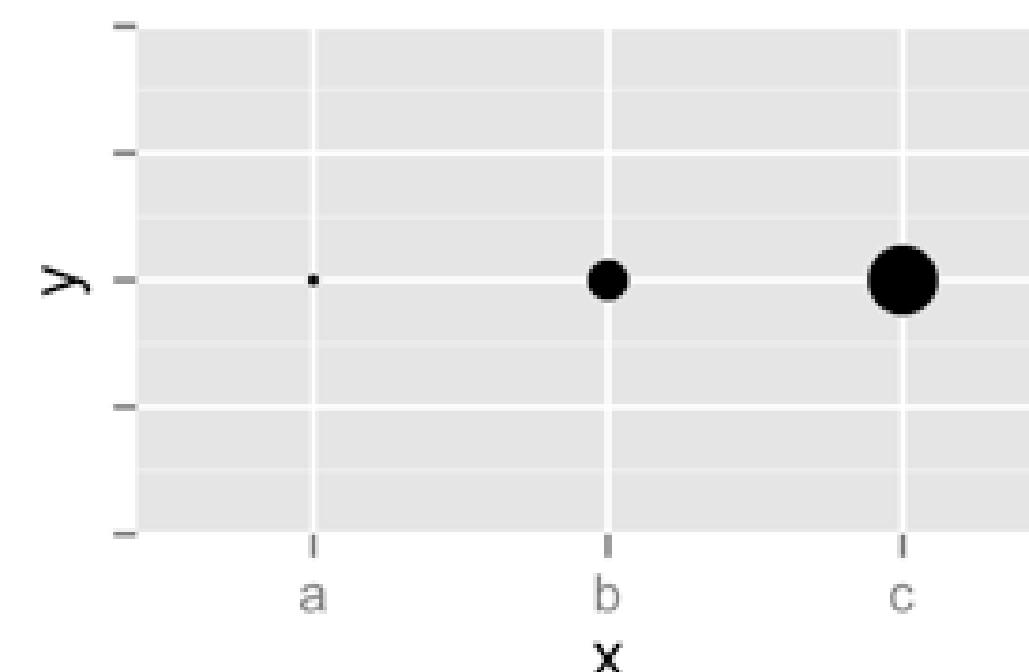
What happens when you use more than one aesthetic?



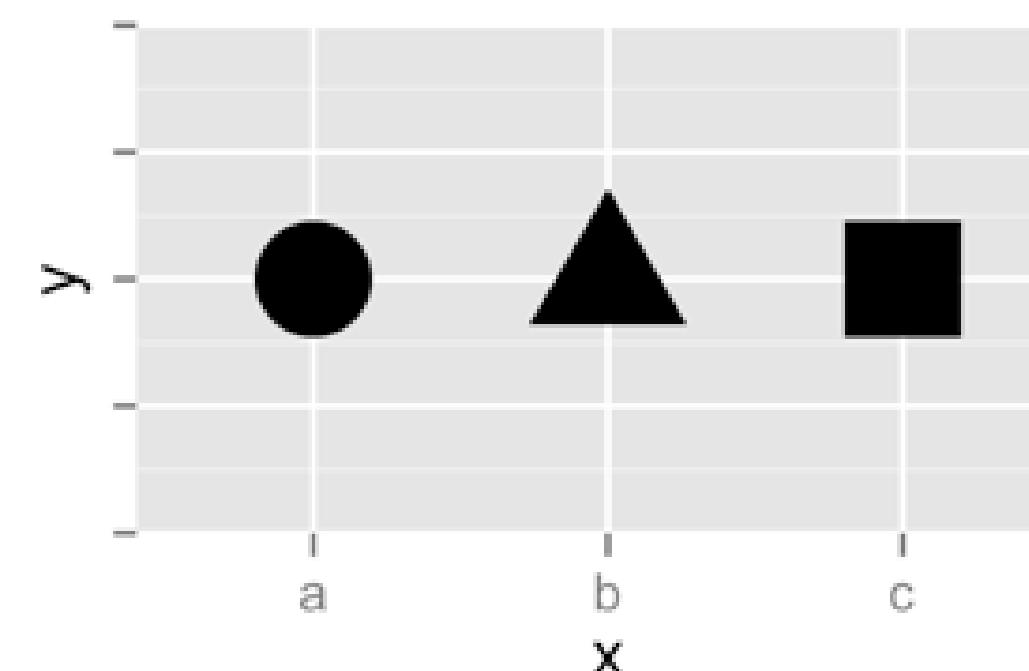
Color



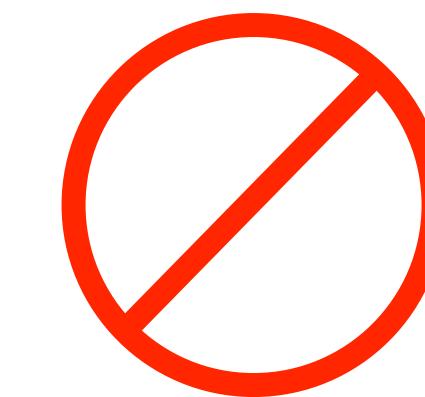
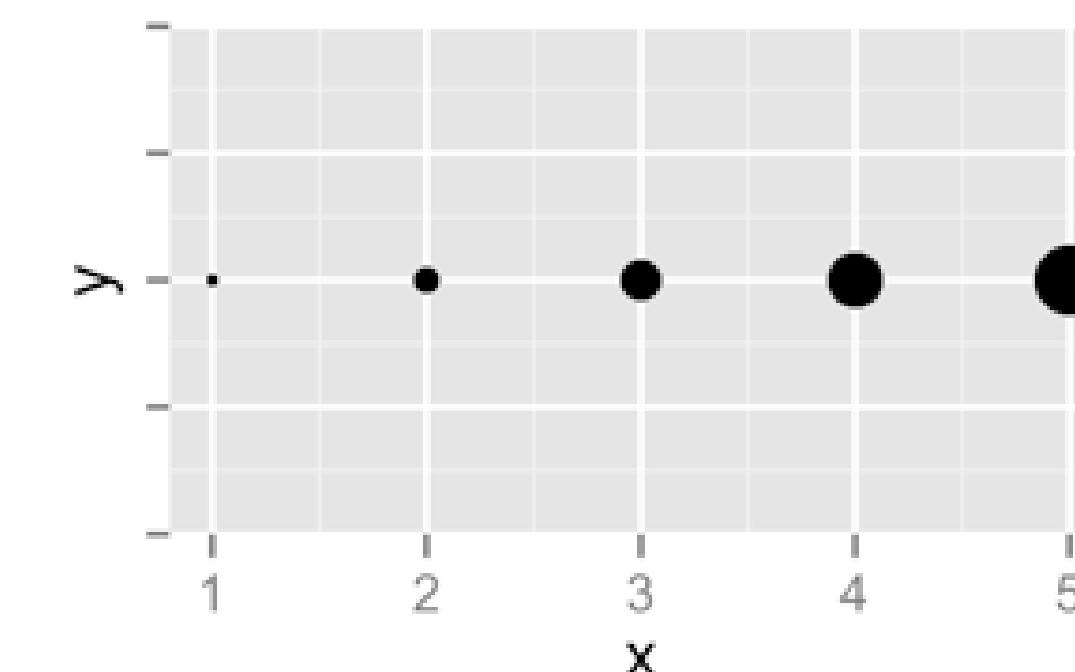
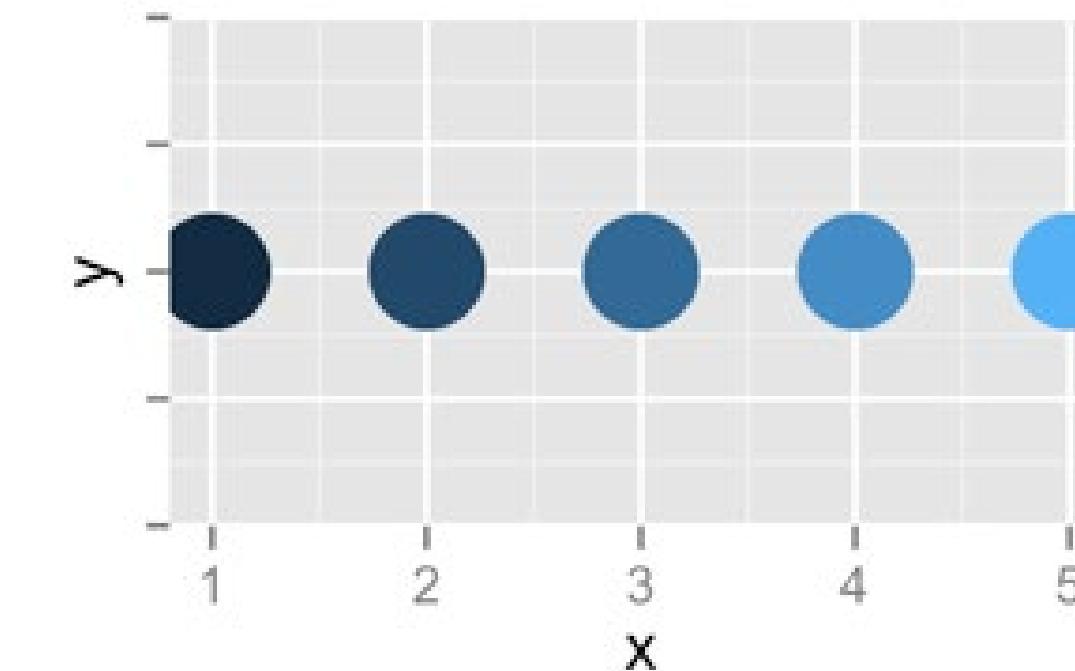
Size



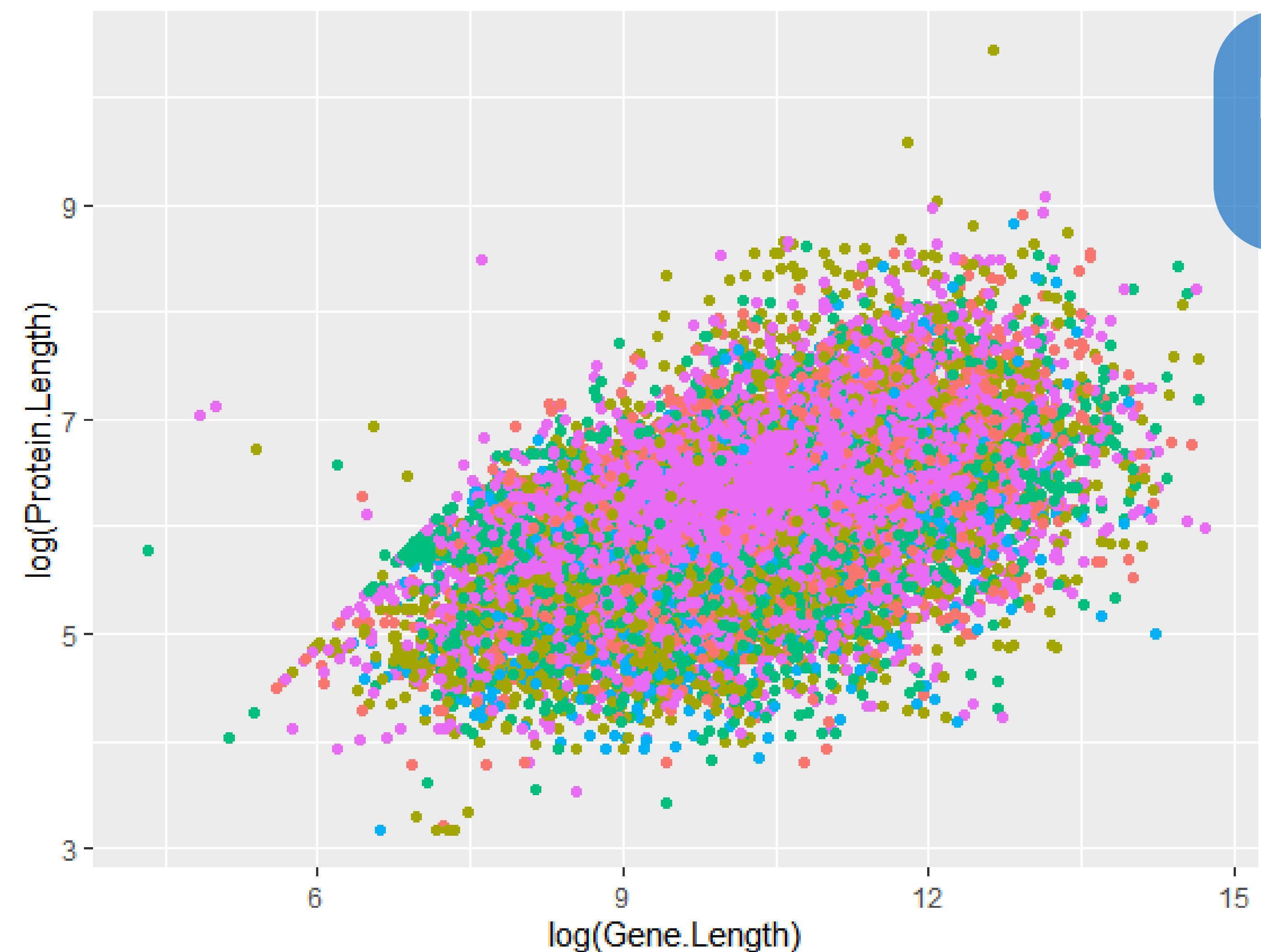
Shape



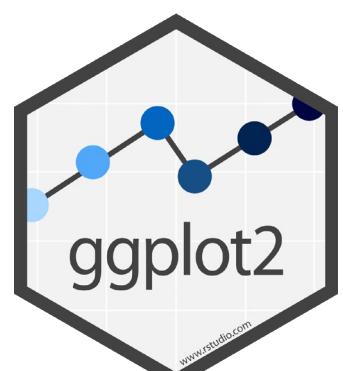
Continuous



Legend added automatically



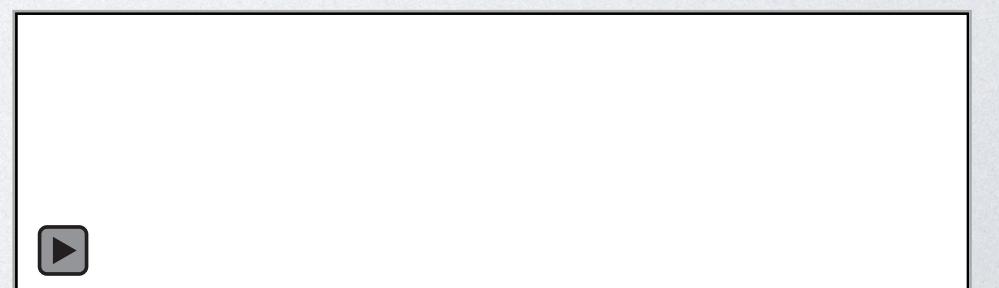
```
ggplot(uniprot_region) +  
  geom_point(aes(log(Gene.Length), log(Protein.Length), color= Cell.Region))
```



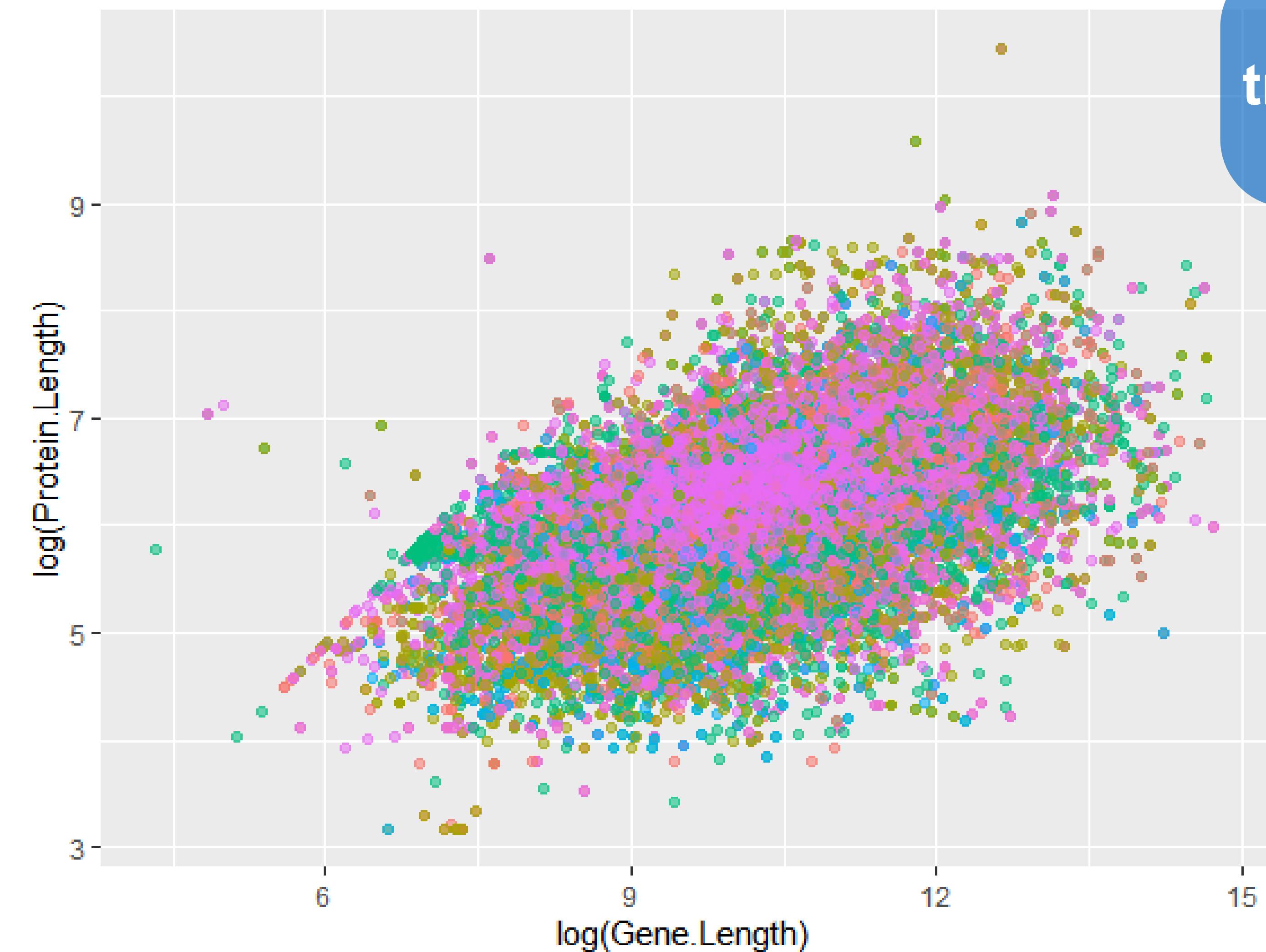
Exercise 4

What happens when you change the alpha aes?

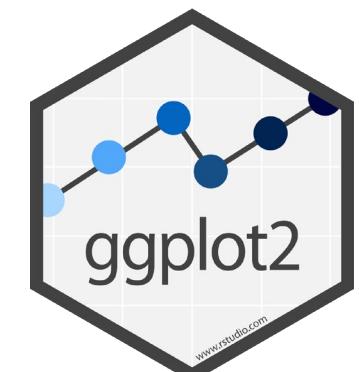
```
ggplot(uniprot_region) +  
  geom_point(aes(log(Gene.Length),log(Protein.Length), fill= cell.Region,  
alpha=0.5))
```



Changes
transparency of
points

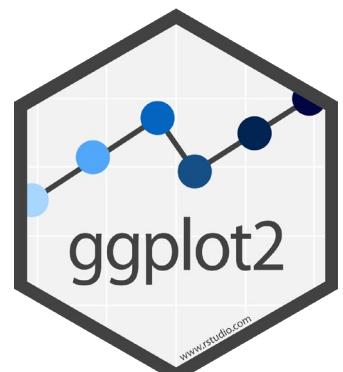


```
ggplot(uniprot_region) +  
  geom_point(aes(log(Gene.Length), log(Protein.Length), color= Cell.Region,  
alpha=0.5))
```



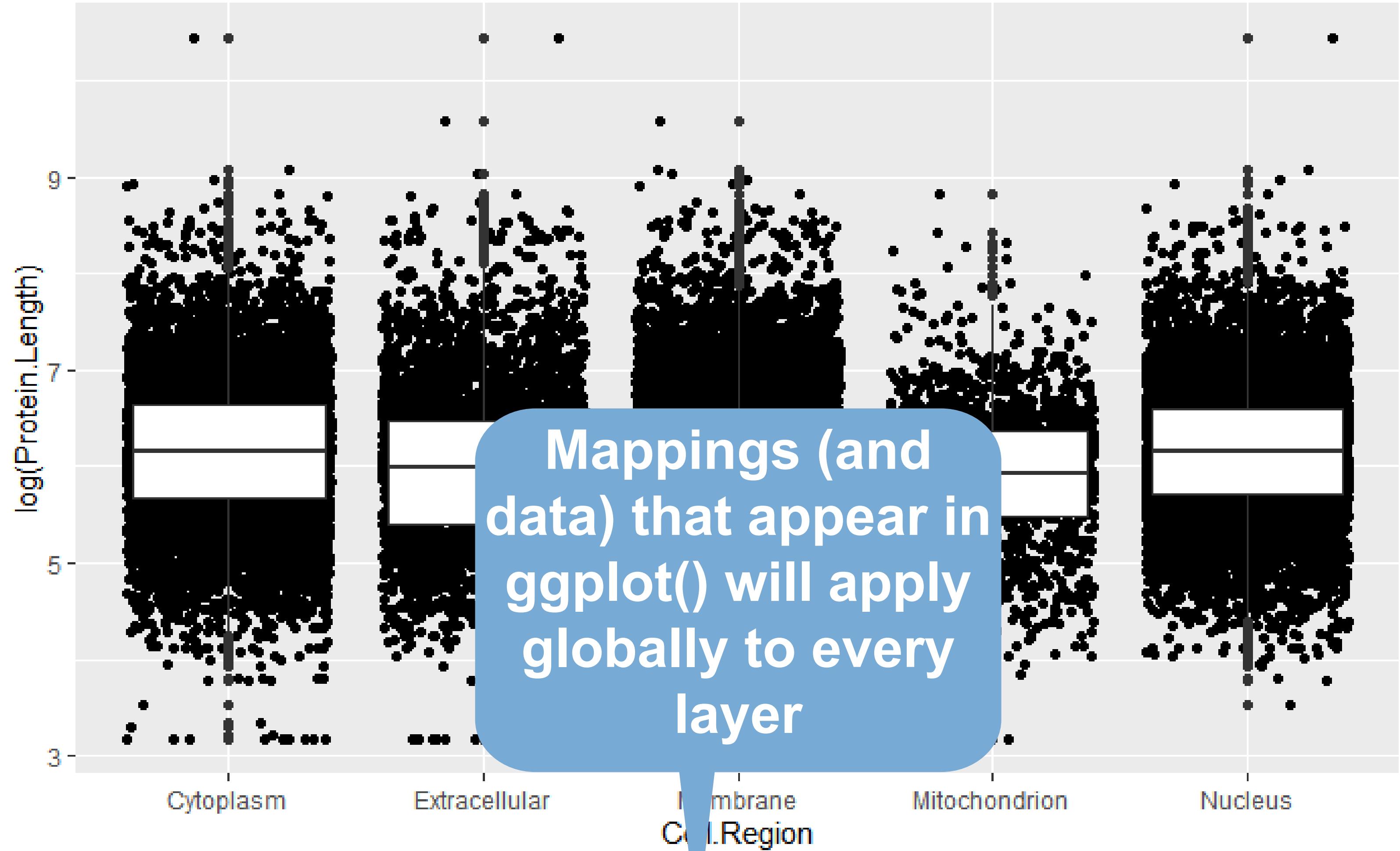
Error!

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

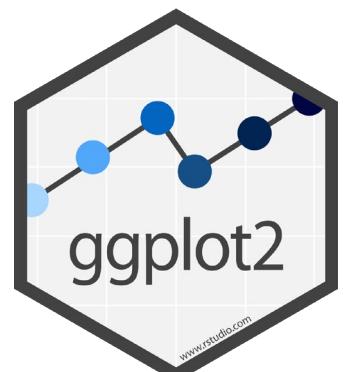


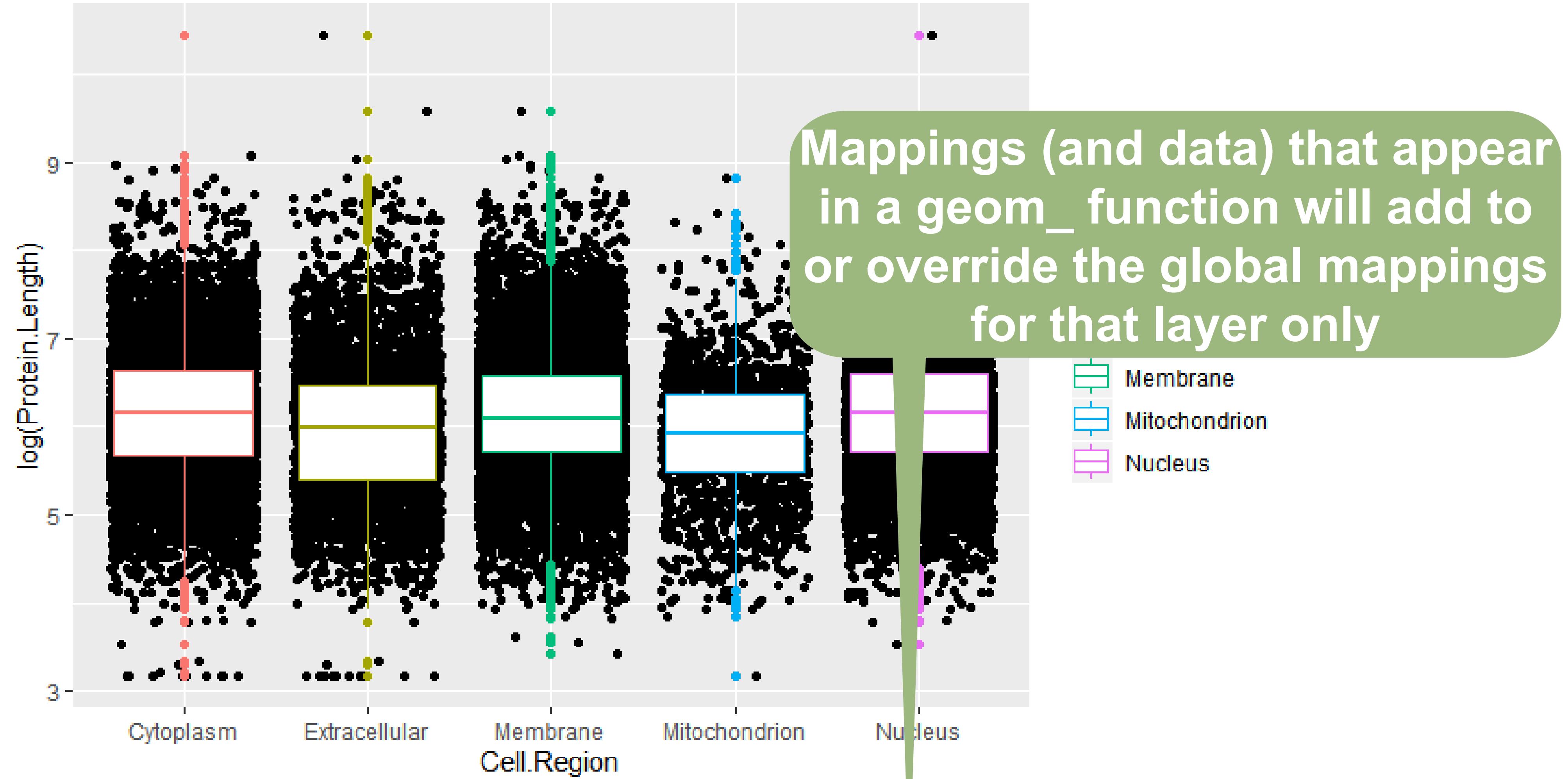
global vs. local

R

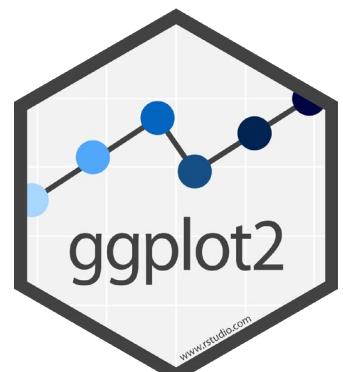


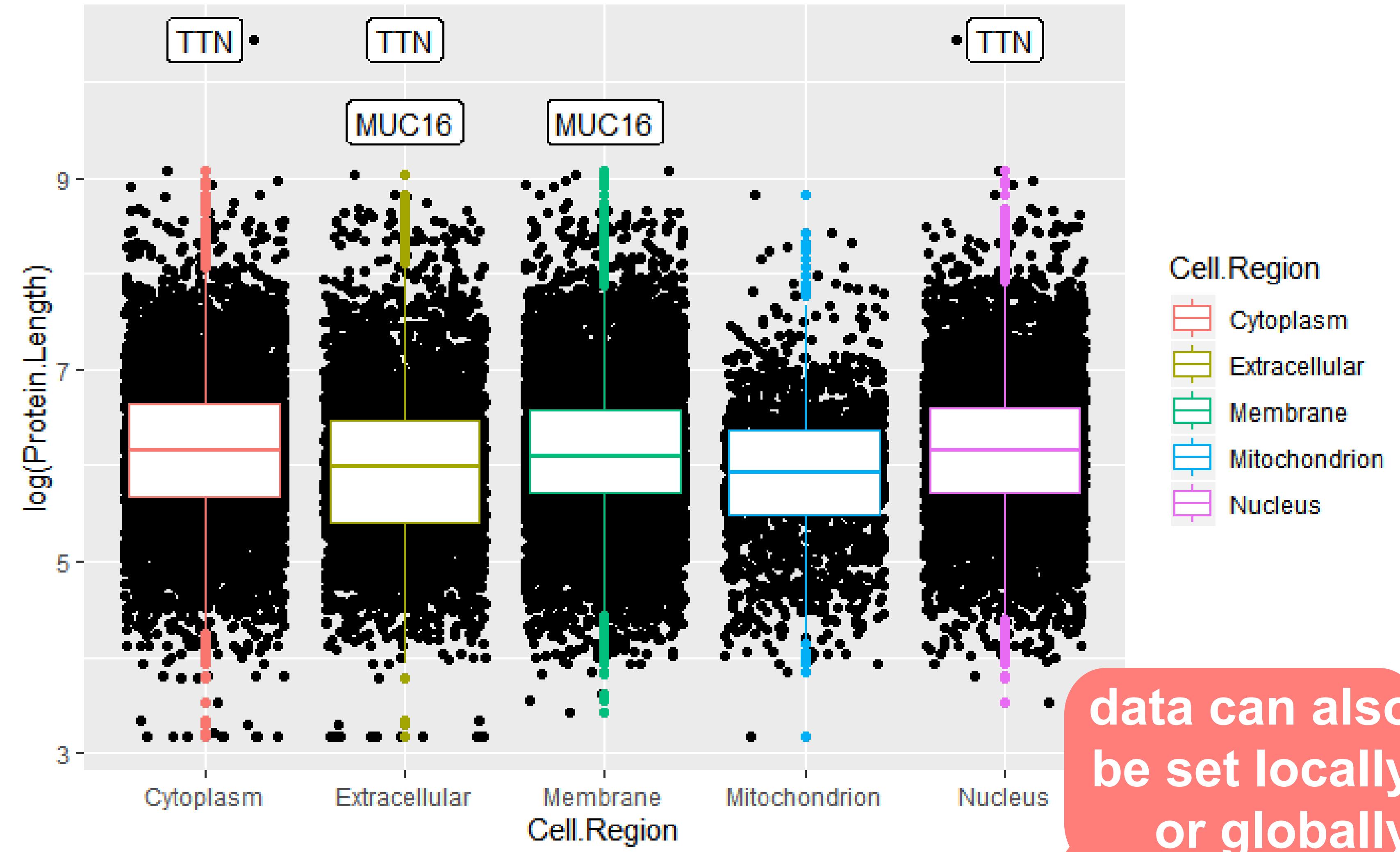
```
ggplot(uniprot_region, aes(x=Cell.Region,  
y=log(Protein.Length)))+  
  geom_jitter()+  
  geom_boxplot()
```





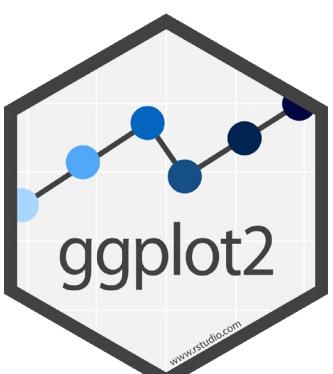
```
ggplot(uniprot_region, aes(x=Cell.Region,
y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=Cell.Region))
```



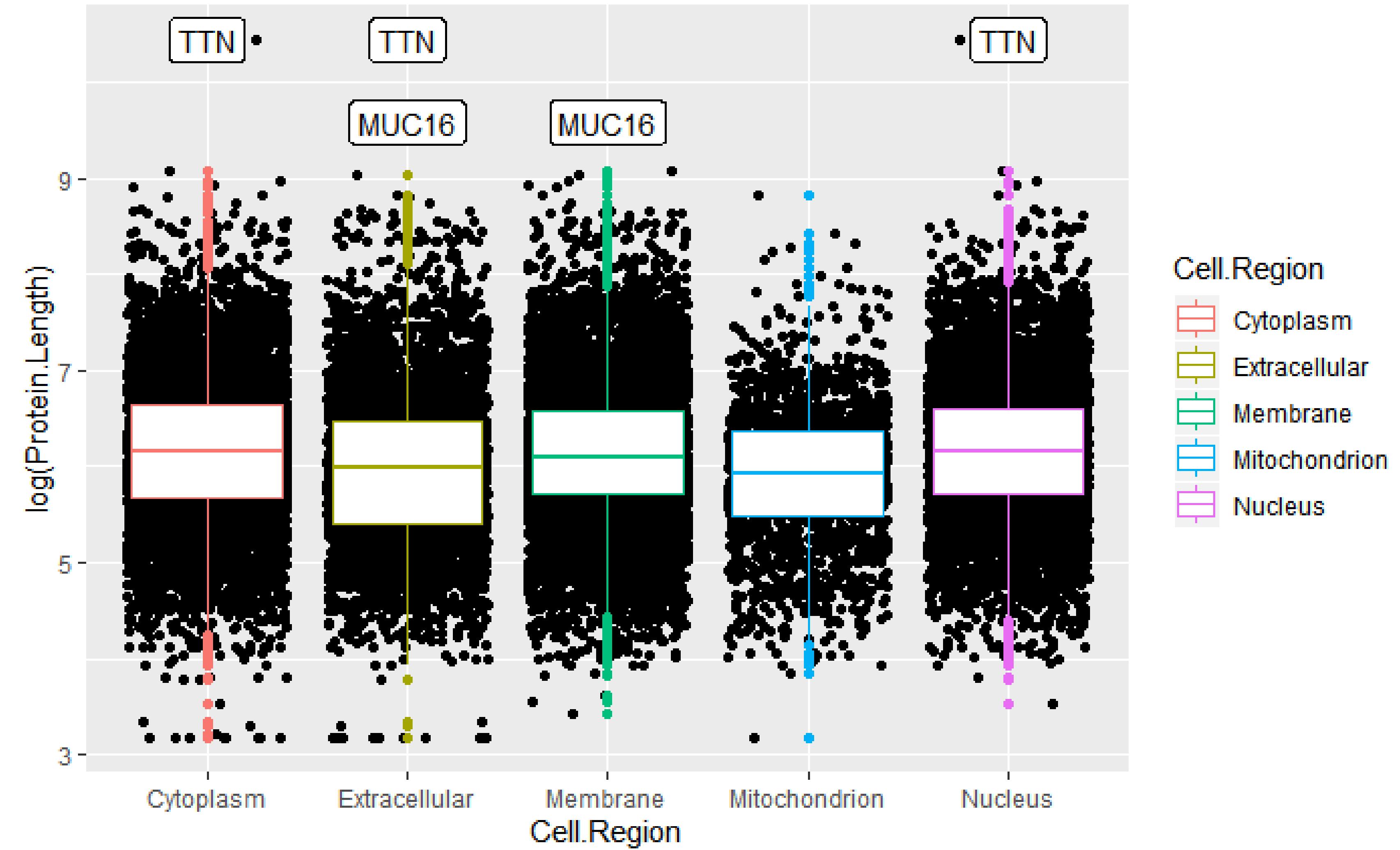


data can also
be set locally
or globally

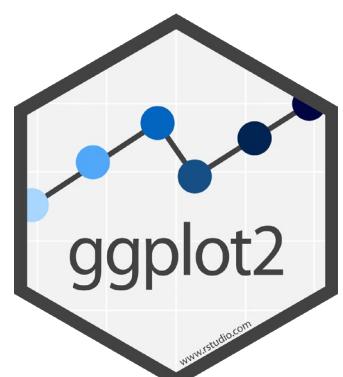
```
ggplot(uniprot_region, aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=Cell.Region))+
  geom_label(data=large_prot, aes(label=Gene.Name))
```

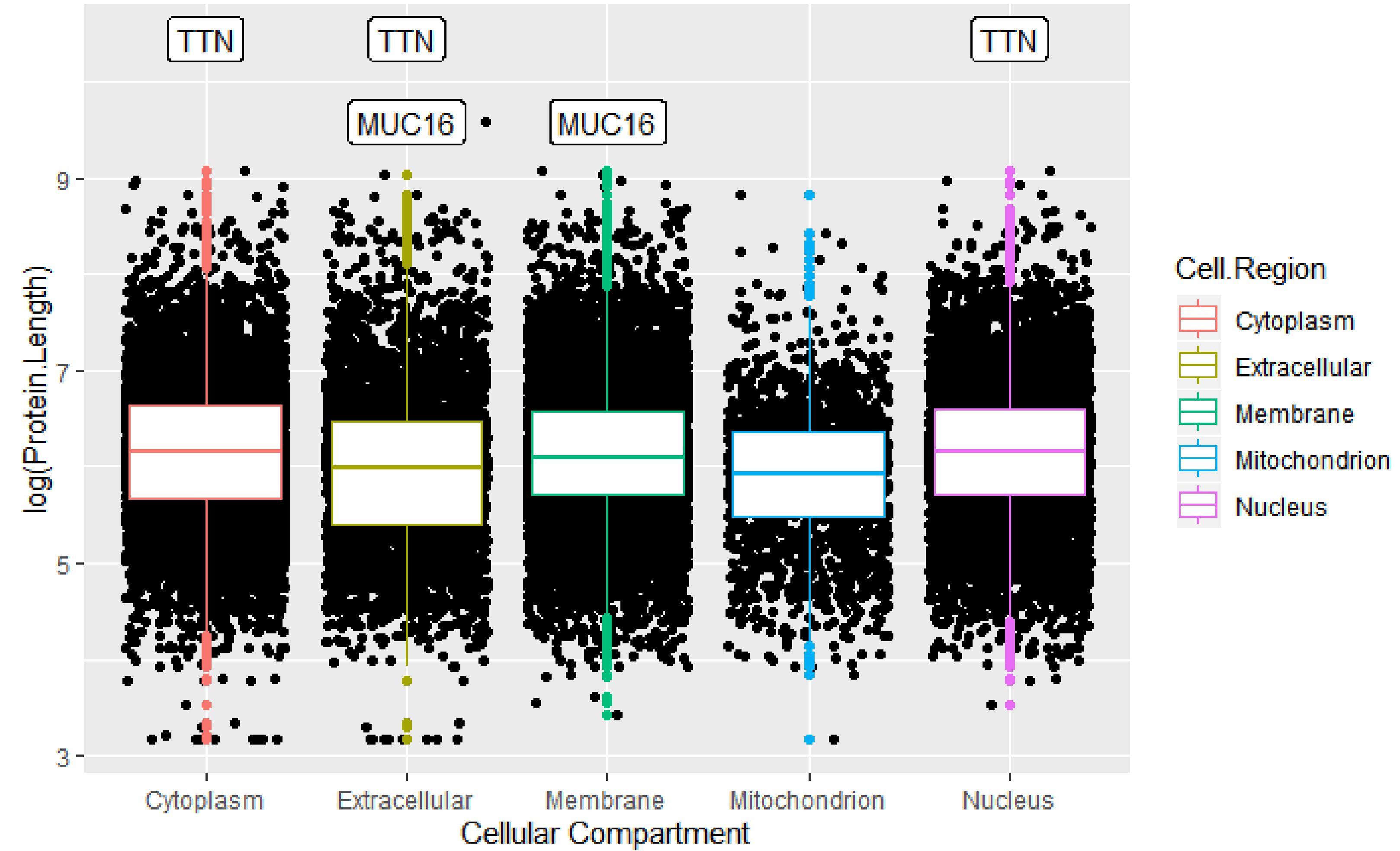


Labels and Legends

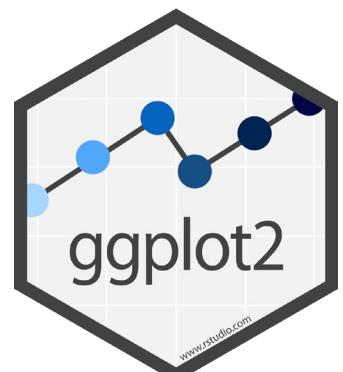


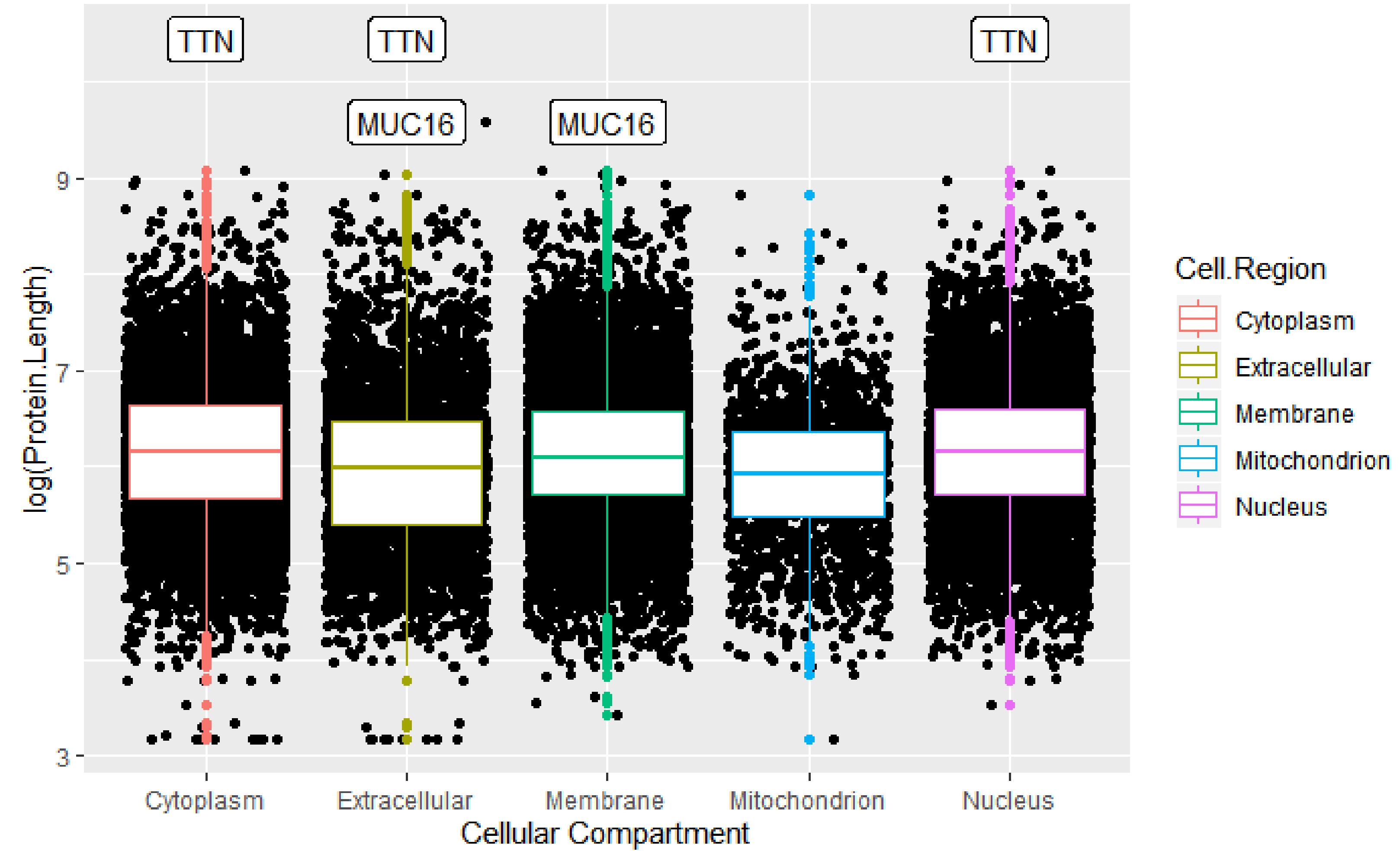
```
ggplot(uniprot_region, aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=Cell.Region))+
  geom_label(data=large_prot, aes(label= Gene.Name))
```



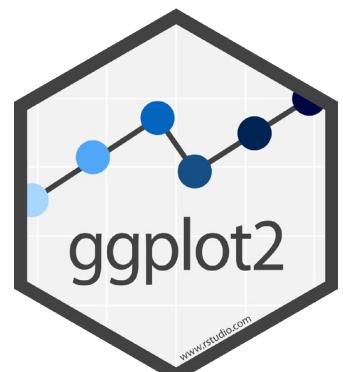


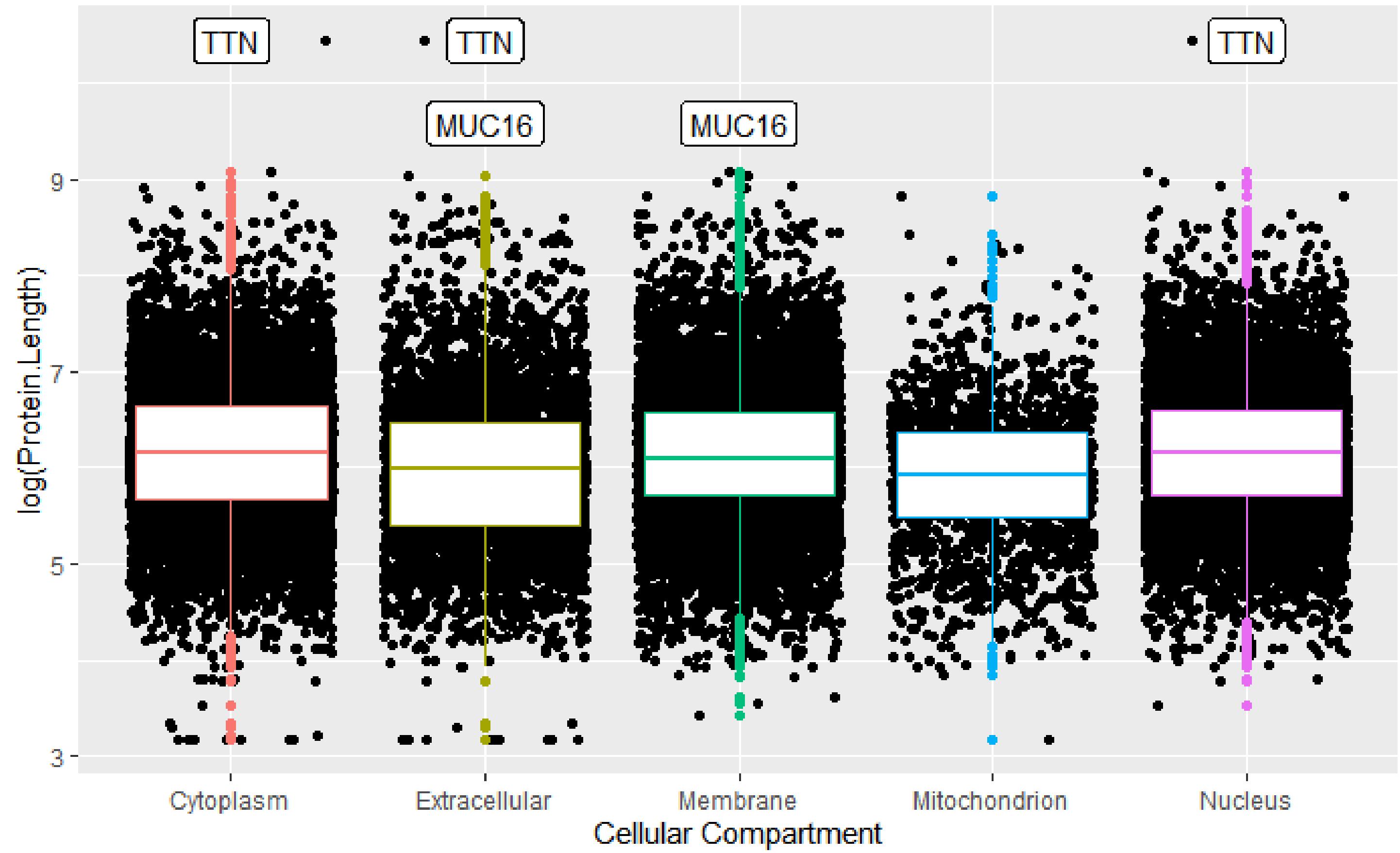
```
ggplot(uniprot_region, aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=Cell.Region))+
  geom_label(data=large_prot, aes(label= Gene.Name))+
  labs(x="Cellular Compartment")
```



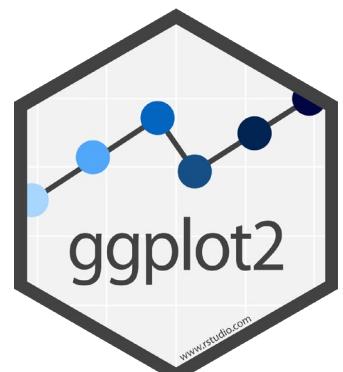


```
ggplot(uniprot_region, aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=Cell.Region))+
  geom_label(data=large_prot, aes(label= Gene.Name))+
  labs(x="Cellular Compartment")
```





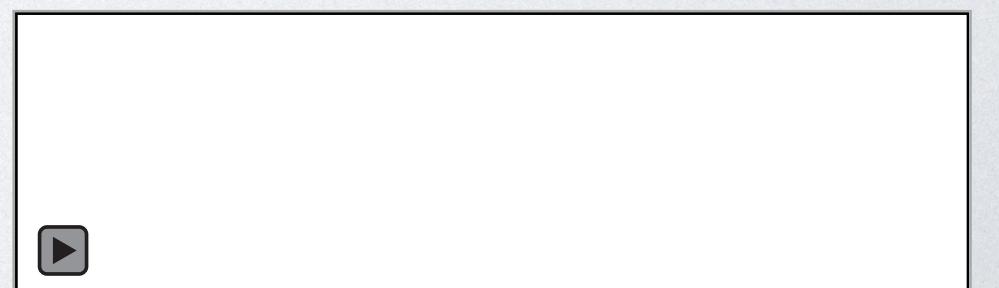
```
ggplot(uniprot_region, aes(x=cell.Region, y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=cell.Region))+
  geom_label(data=large_prot, aes(label= Gene.Name))+
  labs(x="Cellular Compartment")+
  guides(fill="none")
```



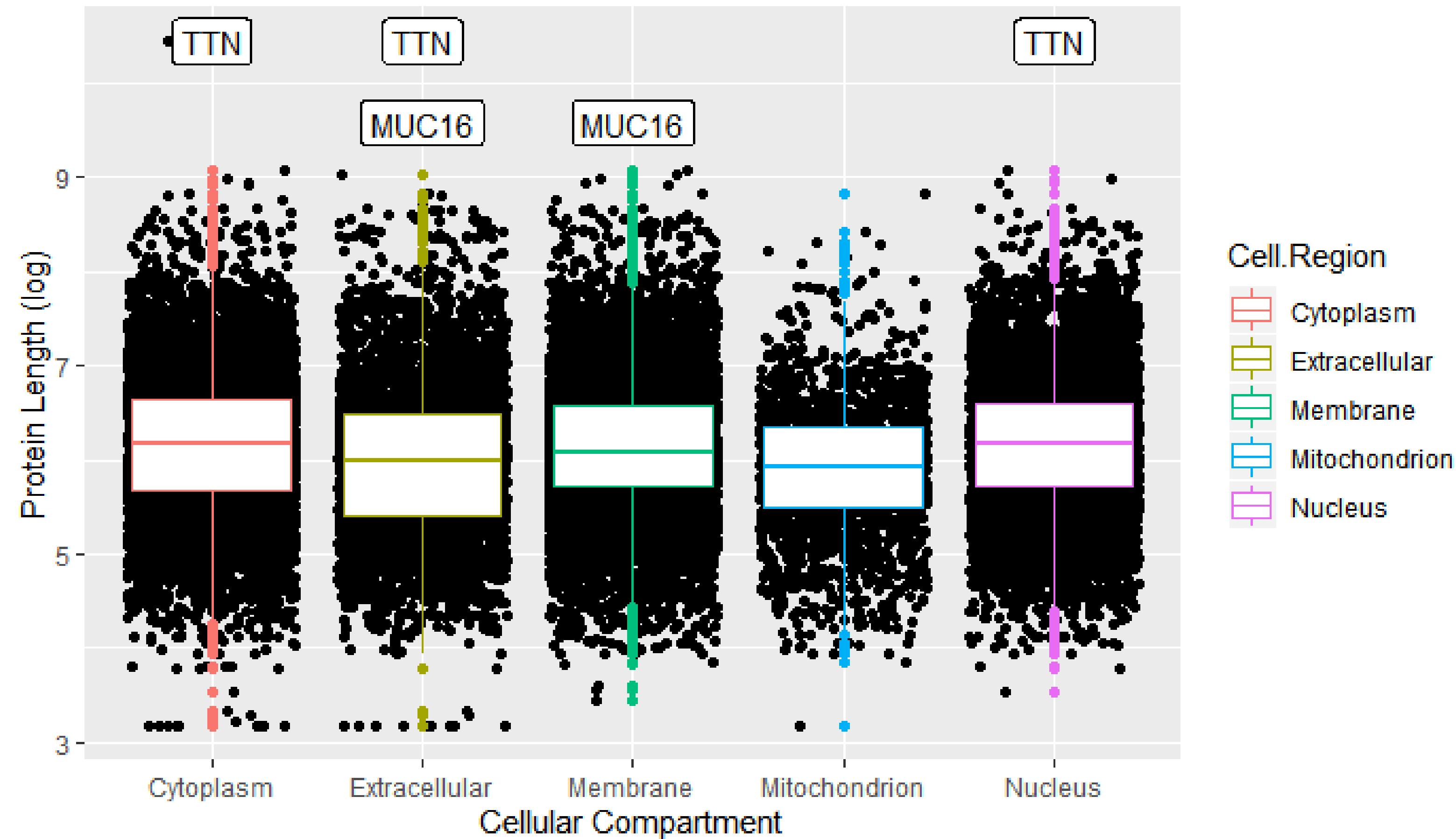
Exercise 5

Change the x and y axis labels and add a title

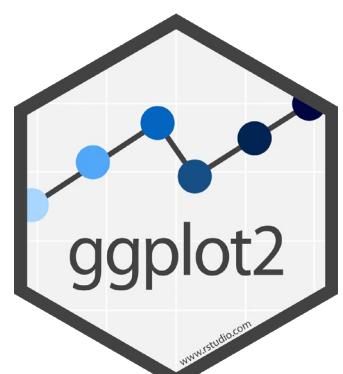
```
ggplot(uniprot_region, aes(x=cell.Region, y=log(Protein.Length)))+  
  geom_jitter()  
  geom_boxplot(aes(color=cell.Region))+  
  geom_label(data=large_prot, aes(label= Gene.Name))
```



Lengths of proteins found in different cellular compartments



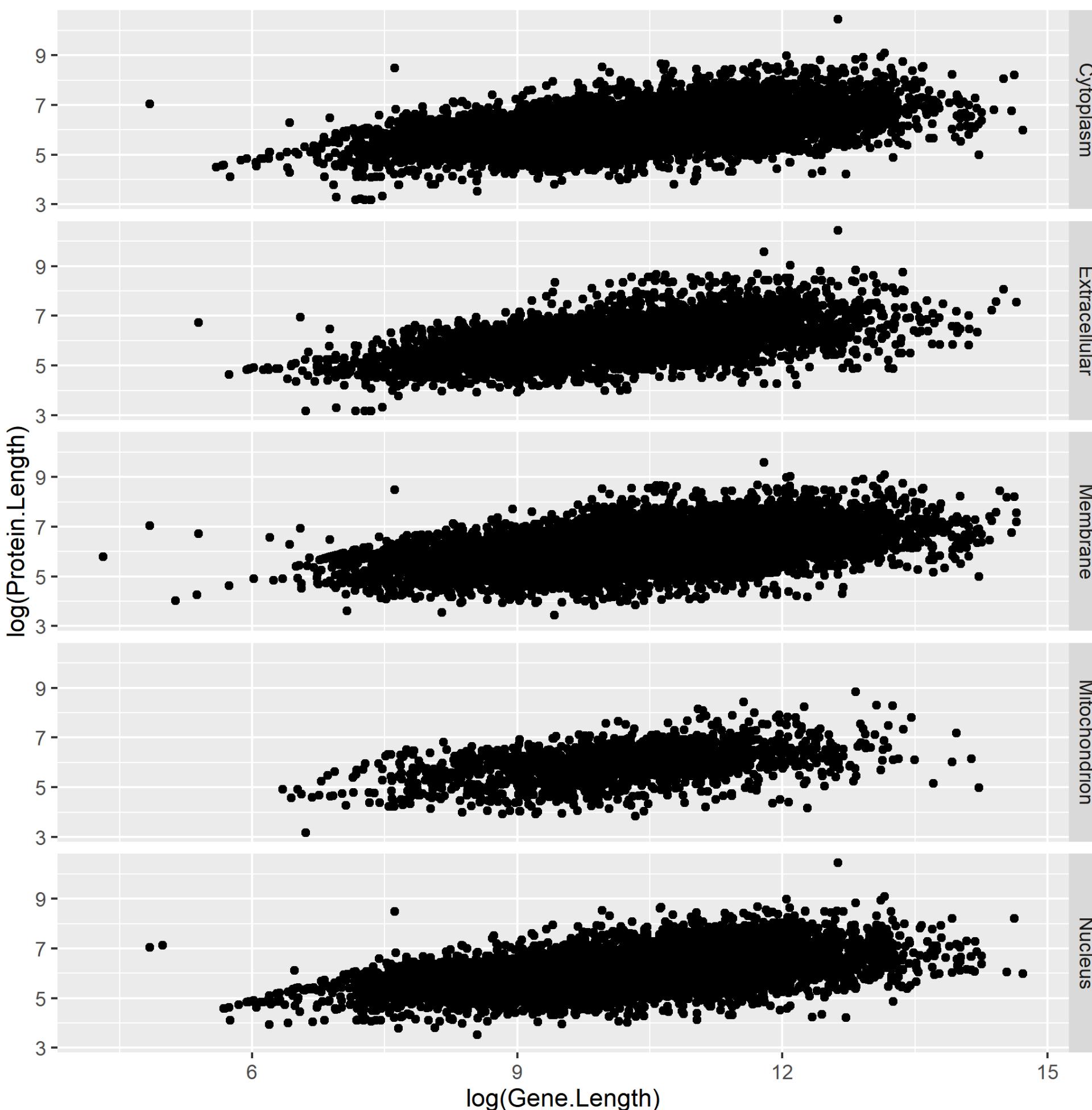
```
ggplot(uniprot_region, aes(x=Cell.Region, y=log(Protein.Length)))+
  geom_jitter()+
  geom_boxplot(aes(color=Cell.Region))+
  geom_label(data=large_prot, aes(label= Gene.Name))+
  labs(x="Cellular Compartment", y="Protein Length (log)", title= "Lengths of proteins
found in different cellular compartments")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



Facets

Facets

Subplots that display subsets of the data.



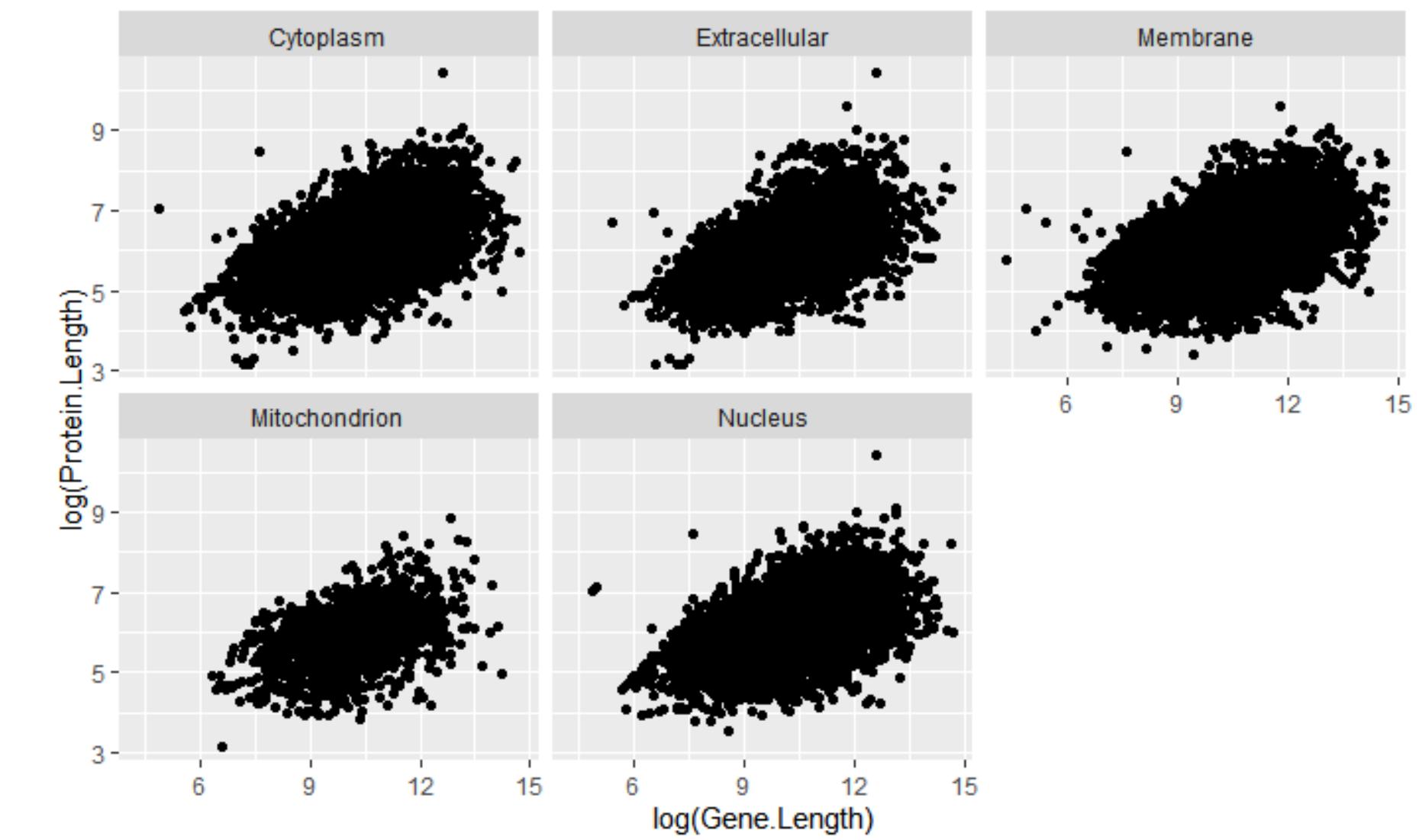
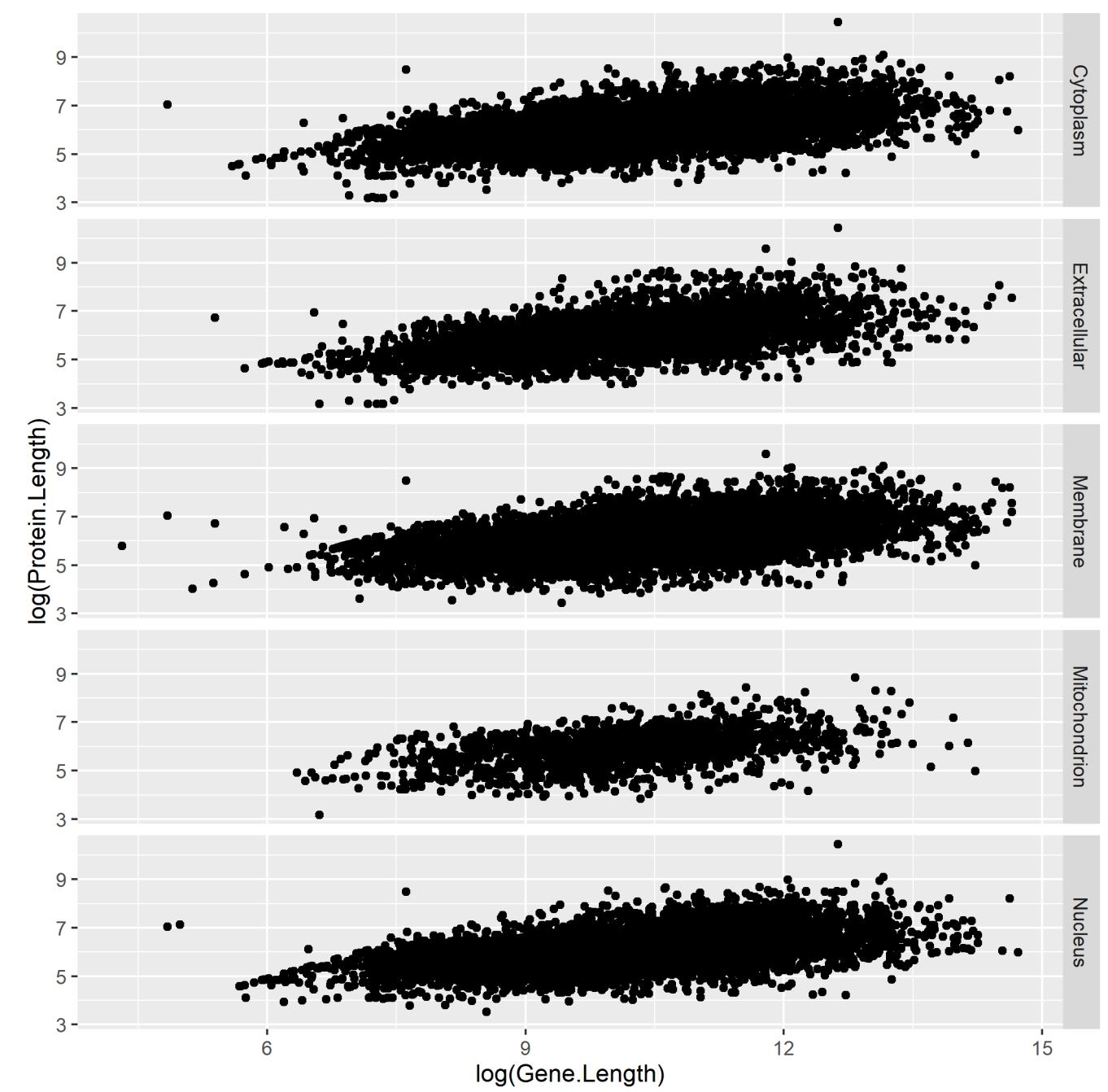
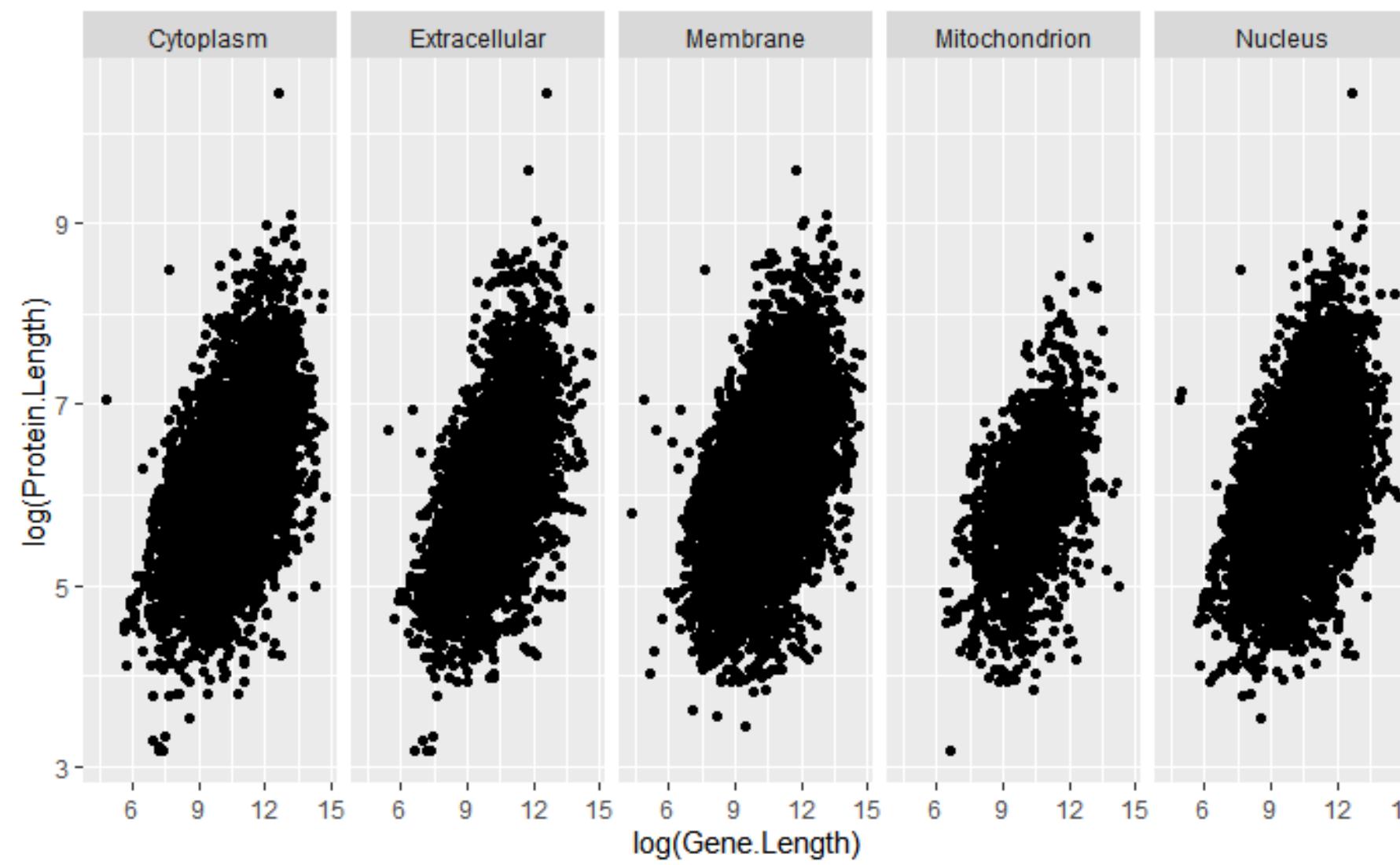
```
ggplot(uniprot_region)+  
  geom_point(aes(x=log(Gene.Length), y=log(Protein.Length)))+  
  facet_grid(rows = vars(Cell.Region))
```

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)
```

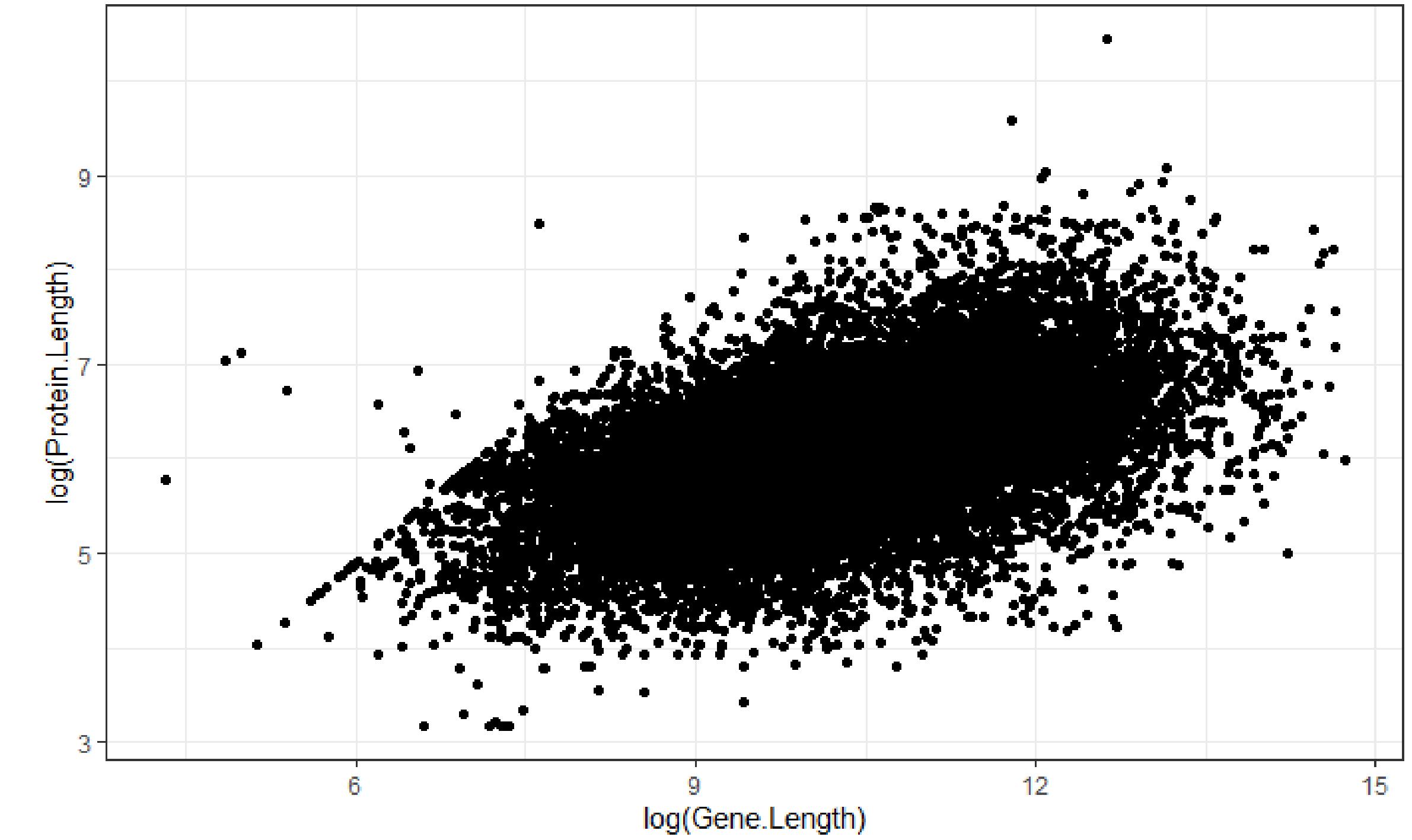
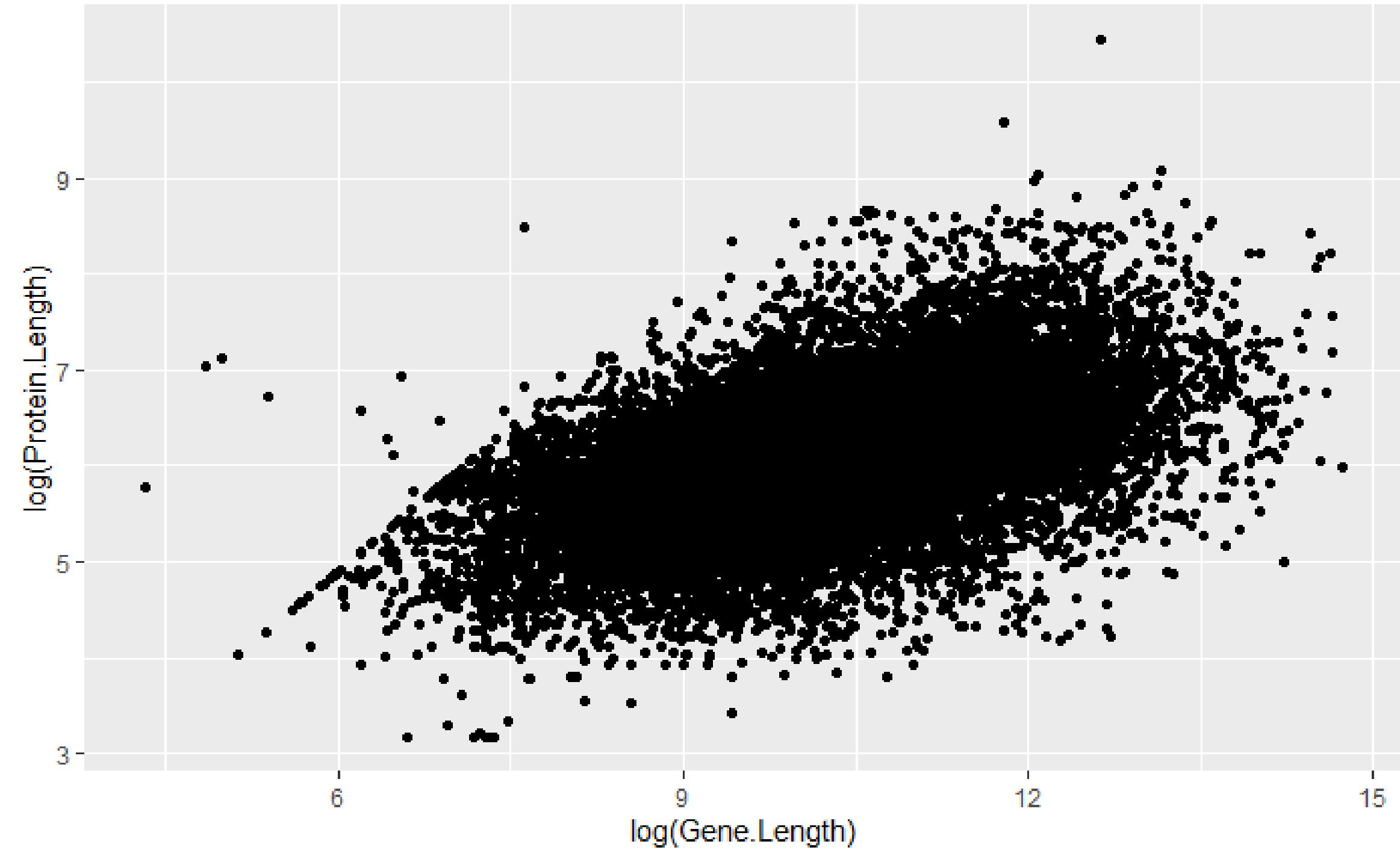
Facets



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

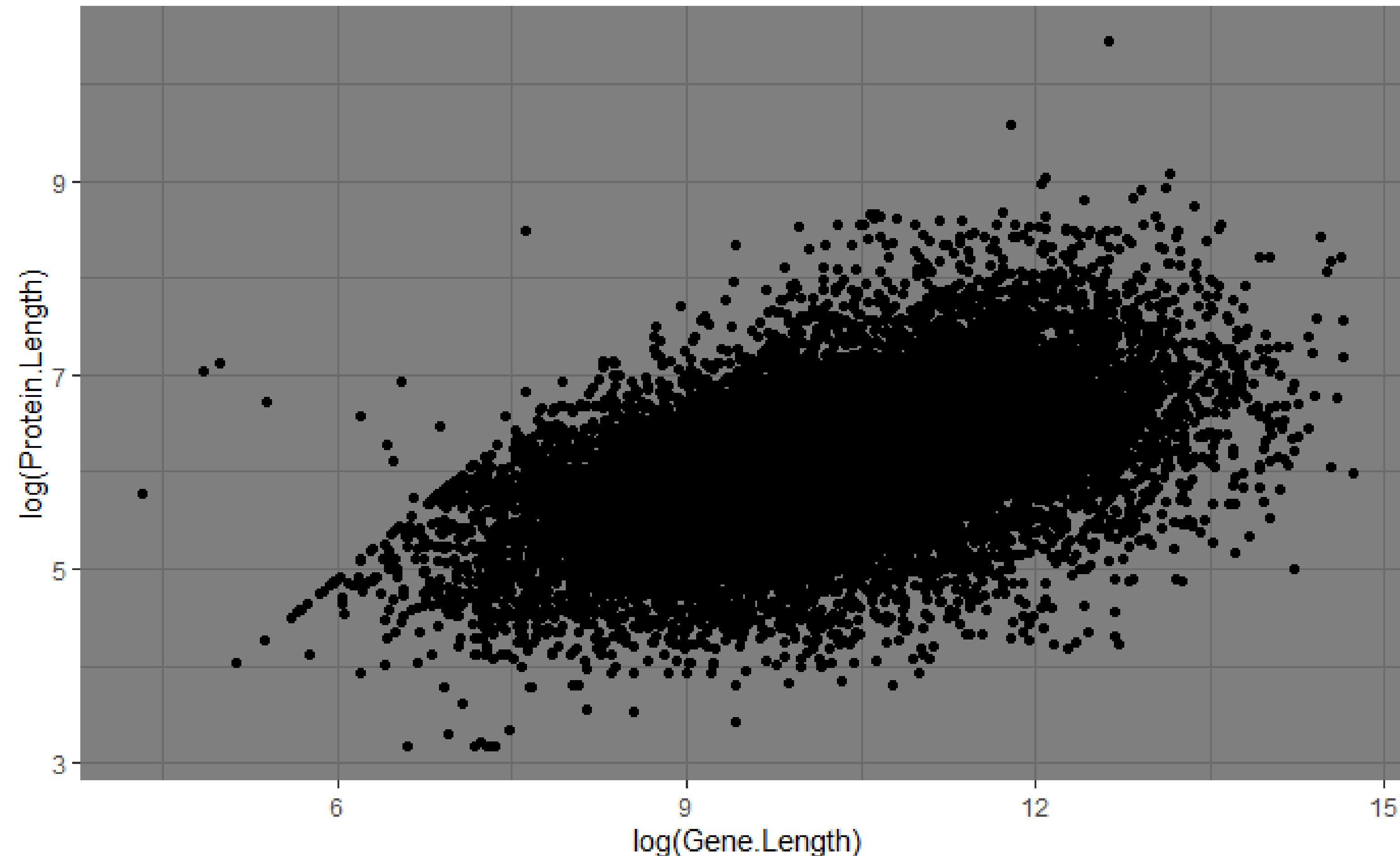
Themes

Themes



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

Themes

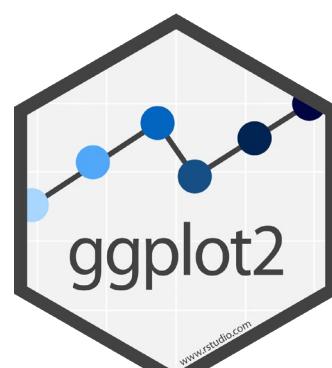
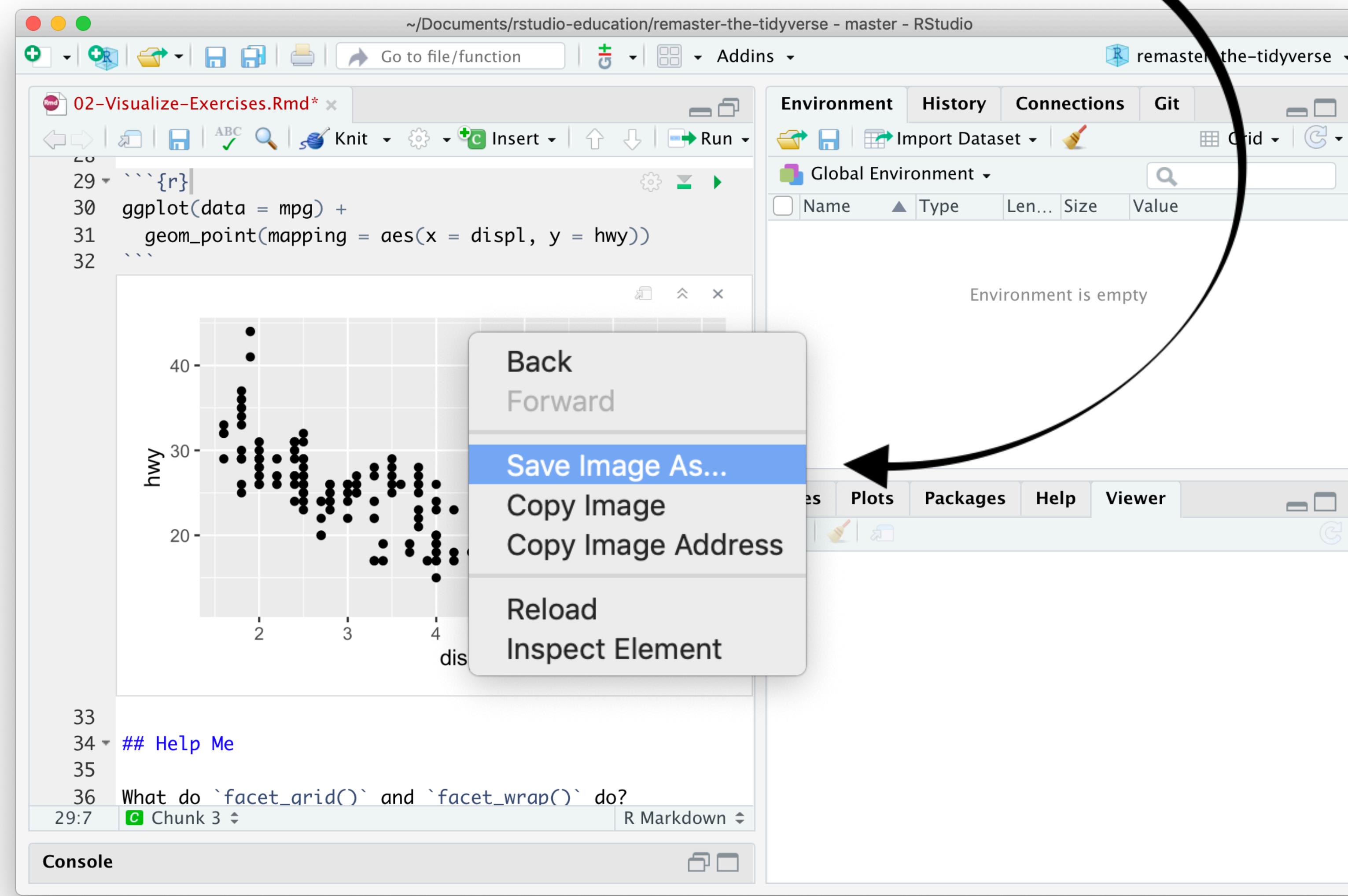


```
ggplot(uniprot_region)+  
  geom_point(aes(x=log(Gene.Length), y=log(Protein.Length)))+  
  theme_dark()
```

Saving graphs

GUI method

Right click on the plot



Code method

ggsave() saves the last plot.

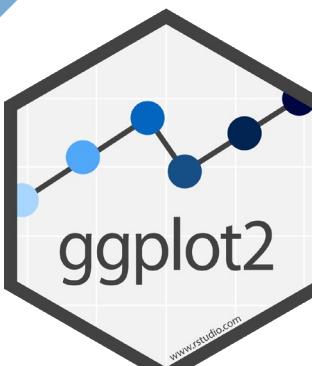
Uses size on screen:

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

Specify size in inches

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

Q: But where will it save it?
A: Alongside your .Rmd



A ggplot2 template

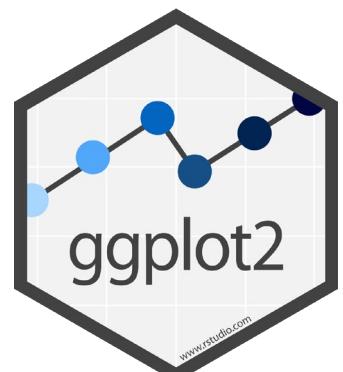
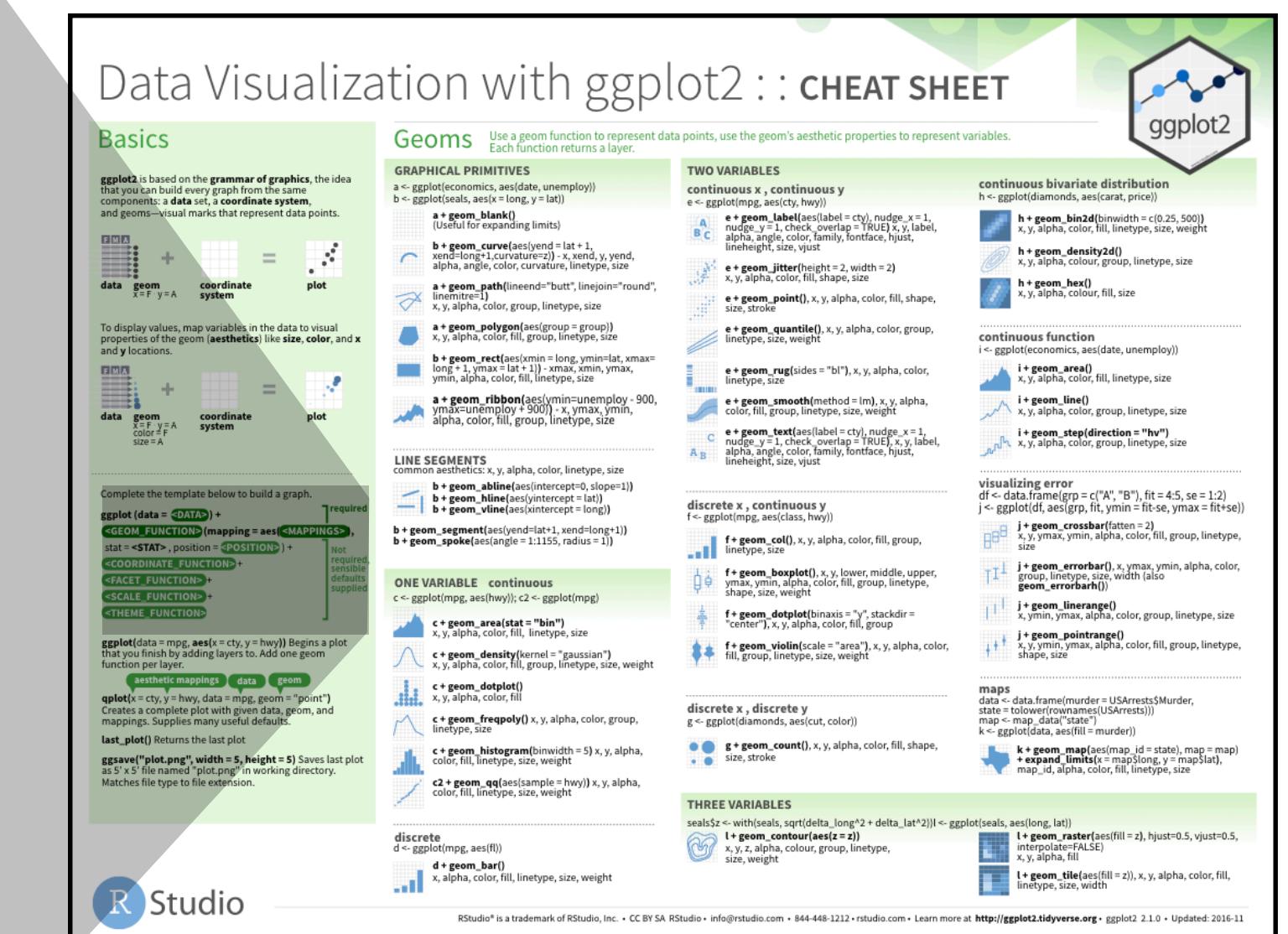
Make any plot by filling in the parameters of this template

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