

# Data visualization

with ggplot2

Maja Kuzman

2019-10-11

# Announcements:

1 HW1 : Good job!

# Announcements:

2 Lecture on monday is off - only consultations on  
friday at 1pm!

Announcements:

3 HW2 : Questions??

# Announcements:

subassign by reference

add multiple columns

handouts!

open a new notebook

install ggplot2

```
install.packages("ggplot2")  
install.packages("plotly")
```

# Tidy data

**Tidy data makes it analysis easier to do, faster to check, easier to plot and to reuse for other analysis. If you have a messy data set and you think that it is exactly what you need, you will most likely use it only once -for one analysis and a single graph.**

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

```
library(data.table)
tidy <- fread("http://hex.bioinfo.hr/~mfabijanic/tidyData.txt", header = TRUE)
head(tidy)
```

##	Gender	Age	Continent	Height	Weight	OTU	Count
## 1:	female	18	Asia	186	72	OTU80	55
## 2:	female	18	Europe	160	77	OTU80	2519
## 3:	female	18	Europe	156	88	OTU80	419
## 4:	female	18	Europe	189	61	OTU80	4115
## 5:	female	18	Europe	160	67	OTU80	4746
## 6:	female	18	Europe	178	65	OTU80	1053

# Graphs in R

Variable types:

- categorical :

nominal

ordinal

- quantitative :

numerical discrete

numerical continuous



# Graphs in R

Variable types:

- categorical :

nominal

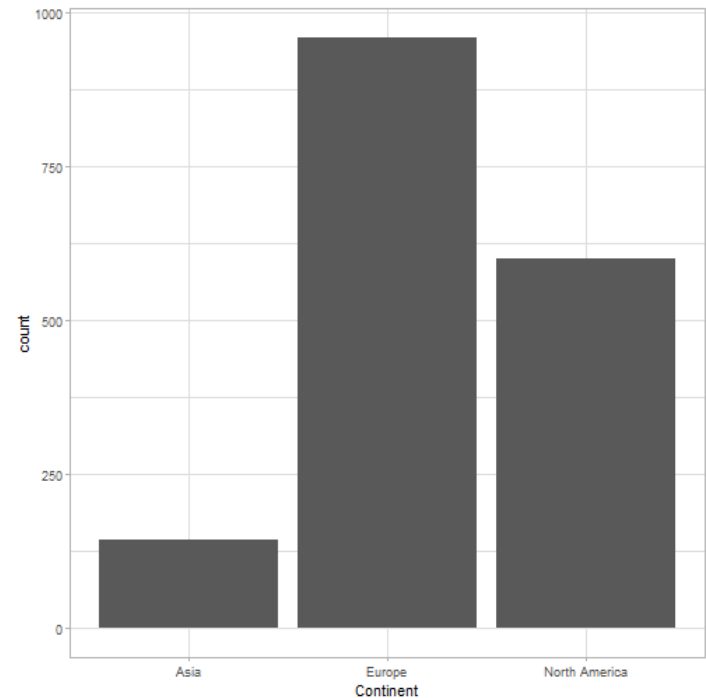
ordinal

- quantitative :

numerical discrete

numerical continuous

Categorical variables example:

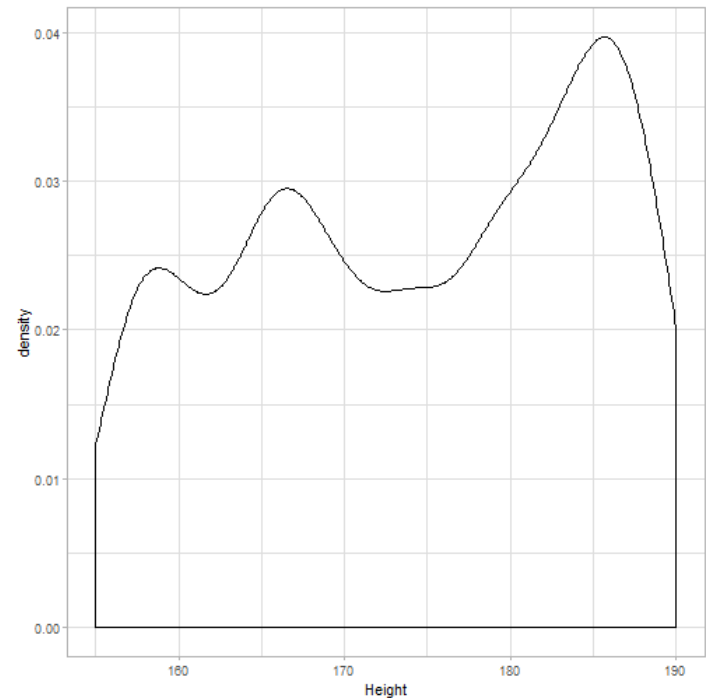


# Graphs in R

Variable types:

- categorical :  
    nominal  
    ordinal
- quantitative :  
    numerical discrete  
    numerical continuous

Quantitative variables example:

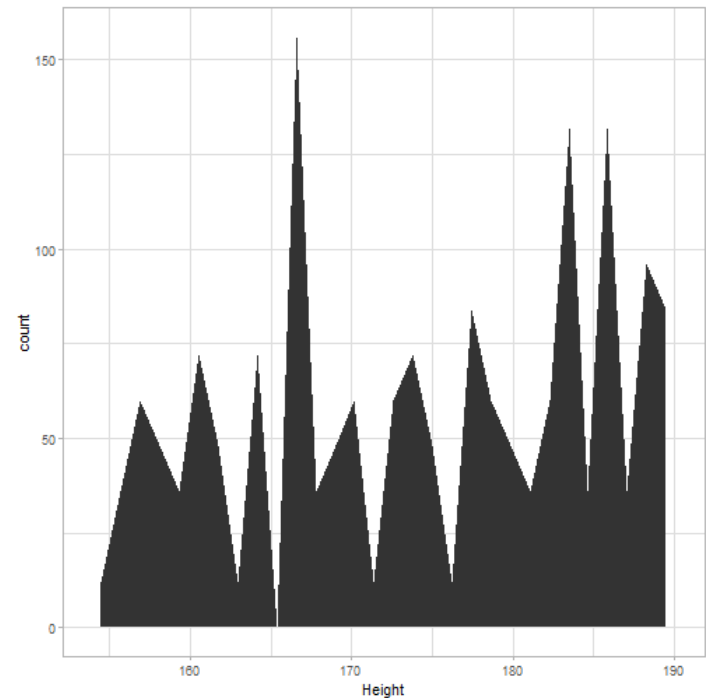


# Graphs in R

Variable types:

- categorical :  
    nominal  
    ordinal
- quantitative :  
    numerical discrete  
    numerical continuous

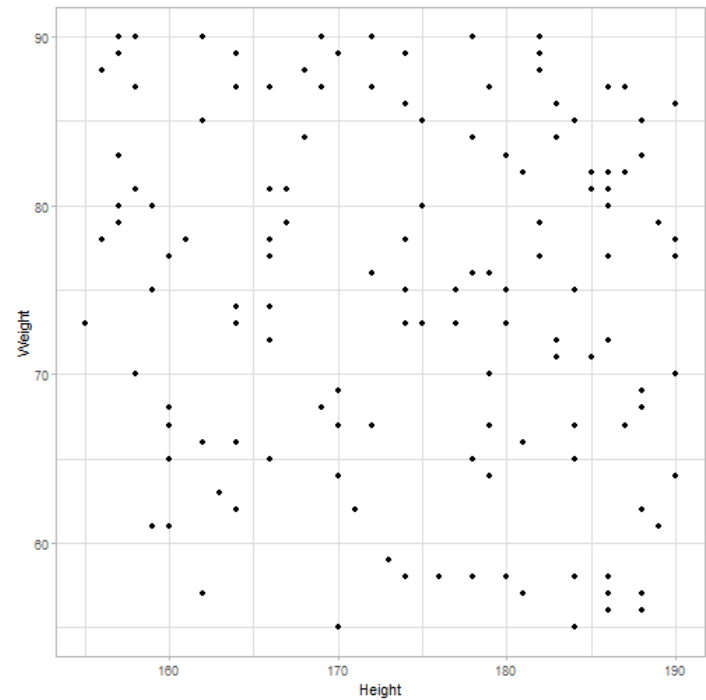
Quantitative variables example:



# Graphs in R

Graphs with two variables:  
Continuous X, Continuous Y

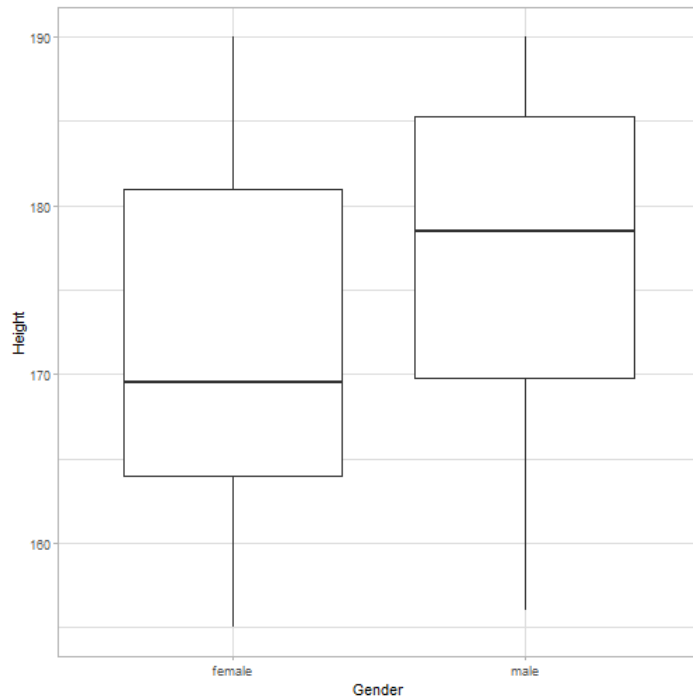
Scatterplot:



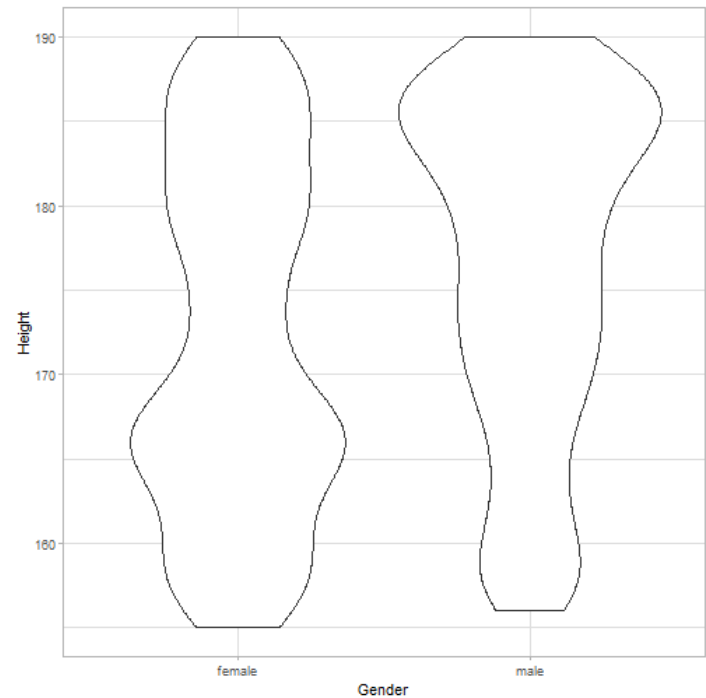
# Graphs in R

Graphs with two variables: Discrete X, Continuous Y

Boxplot:

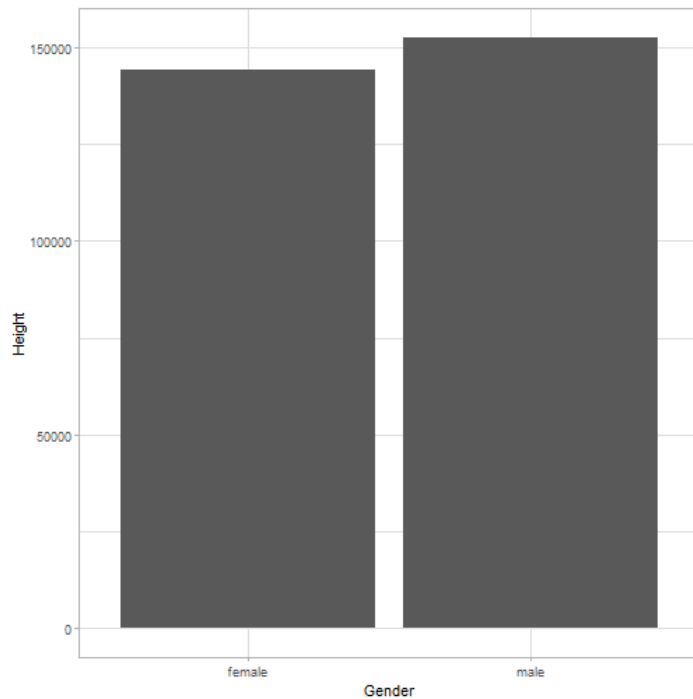


Violin plot:

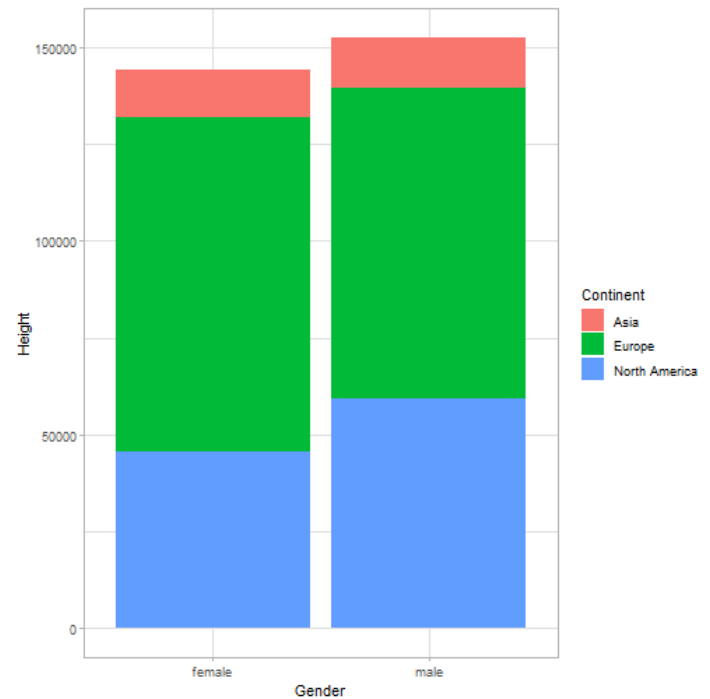


# Graphs in R

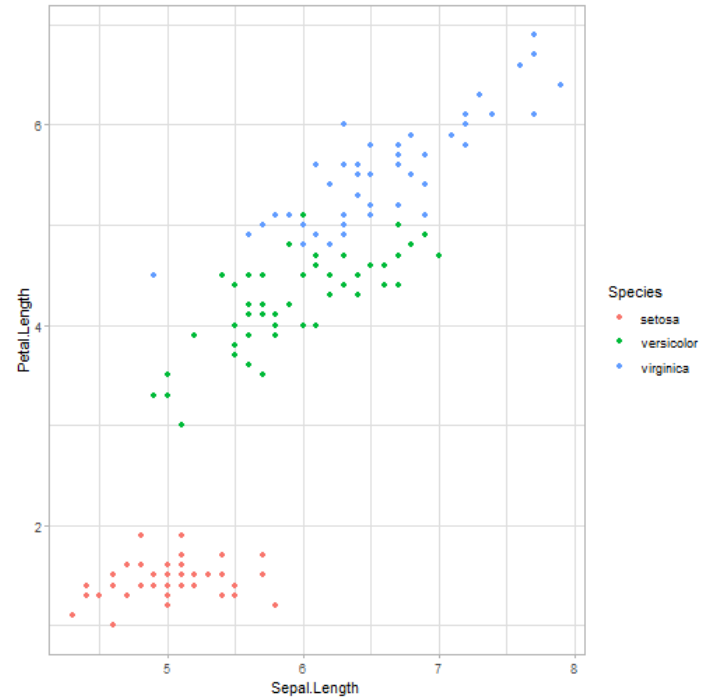
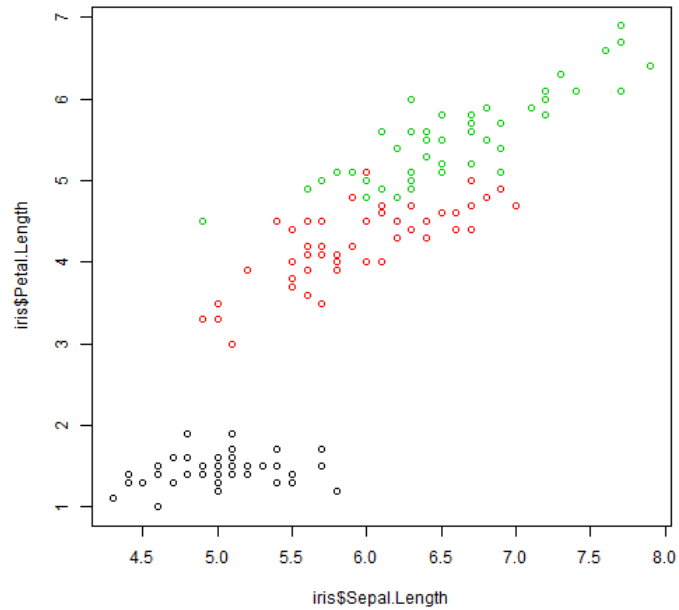
Graphs with two variables: Discrete X, Discrete Y



Barplot:



# ggplot2 VS base R plots



# ggplot2

There are different types of plots and they are used based on different data that we want to show. We will use ggplot2 package to draw graphs in R for the following reasons:

- Elegant
- Highly customisable
- Uniform
- Natural
- Expressive
- Popular
- Steep learning curve
- Slow
- Evolving pretty fast (too fast?)



## Basic ggplot logic: `ggplot(data, aes(x,y))`

The `ggplot()` object acts as a storage facility for the data. It is here where we define the data frame that houses the x and y coordinate values themselves and instructions on how to split the data. There are three ways to initialise a `ggplot()` object:

```
p ← ggplot()  
p ← ggplot(data_frame)  
p ← ggplot(data_frame, aes(x, y ))
```

Displaying the object `p` generated in the code chunk above would result in an empty plot.

# Mapping aesthetics to data

The `aes()` aesthetic mapping function lives inside a `ggplot` object and is where we specify the set of plot attributes that remain constant throughout the subsequent layers (unless overwritten more on this later).

We can consider the relationship between the `aes()` and `geoms` components as follows:

The `aes()` function is the how data is stored, how data is split  
`geoms` is the what the data looks like. These are geometrical objects stored in subsequent layers.

## Layers

We use the `+` operator to construct. By appending layers we can connect the "how" (aesthetics) to the "what" (geometric objects). Adding geometric, scale, facet and statistic layers to a `ggplot()` object is how to control virtually every visual aspect of the plot from the data contained in the object.

## Adding a geometric object layer

A geometric object is used to define the style of the plot. Common geometric objects include:

`geom_point()` which is used to draw a dot plot  
`geom_line()` used to draw a line plot  
`geom_bar()` used to draw a bar chart.

## Facets

Appending a facet layer to a ggplot generates the same plot for different subsets of data.

## Statistics

Exploratory data analysis can be done using the base packages in R, the results of which can be added to a ggplot() in the guise of a geom layer.

## Example : Basic layout

x axis is categorical, y axis is numerical

1. Set the basic layout:

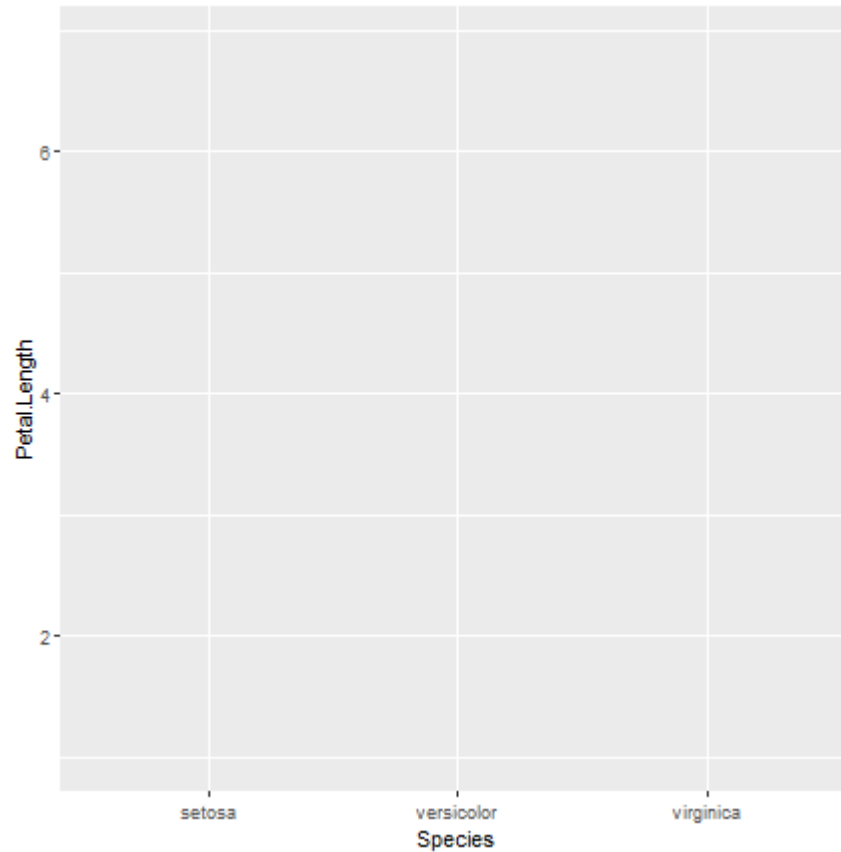
Fill the empty spaces with correct terms so you get a layout in which we use dataset iris, and we want x axis to represent Species and y to represent Petal.Length .

```
p ← ggplot( , aes( , ))  
p
```

What does p look like?

## Example : Basic Layout - solved

```
p ← ggplot( iris, aes( Species, Petal.Length ) ); p
```



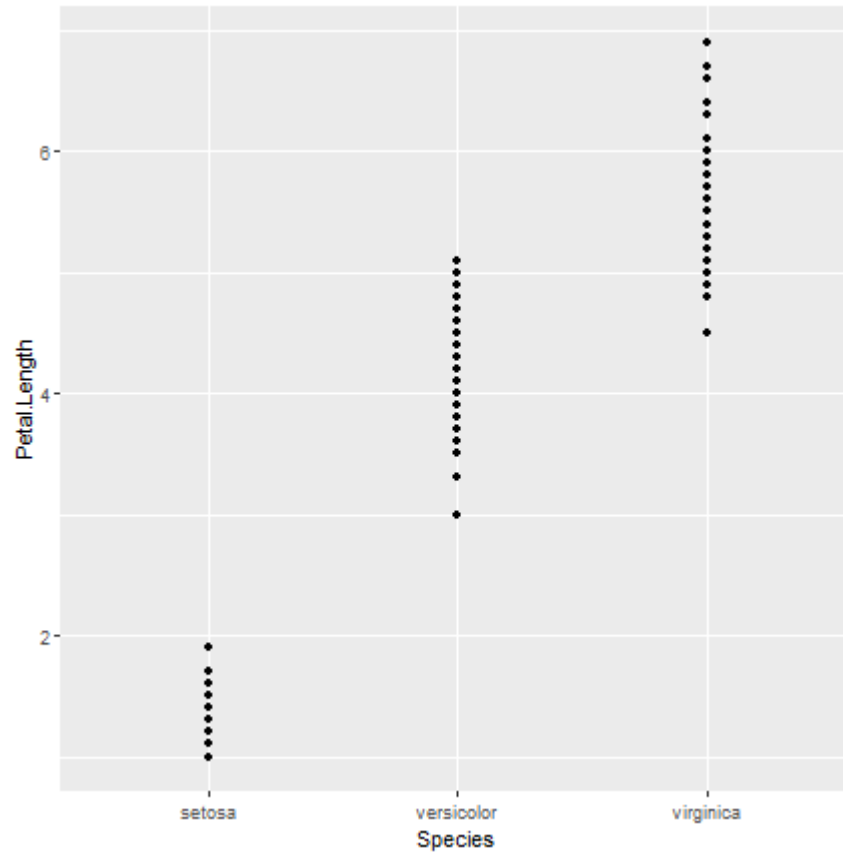
## Example : add a layer - graph type

1. Choose graph type that you want to show.. Lets say we want a scatterplot for petal length for each group. Add `geom_point()` to your layout.

```
p + ???
```

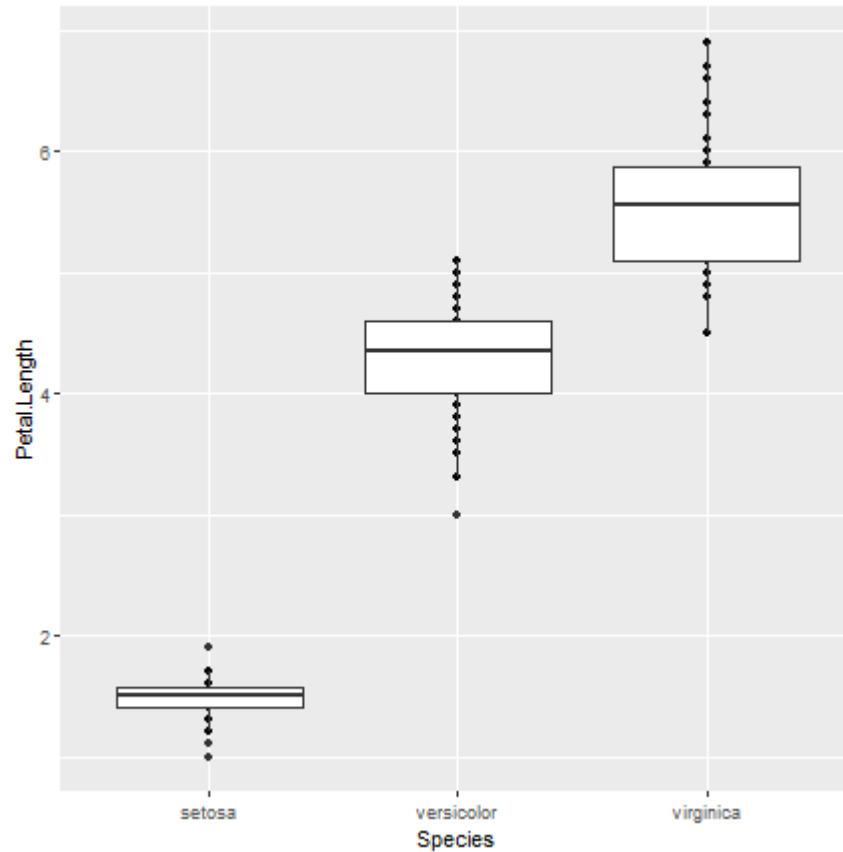
## Example : add a layer - graph type - solved

```
p + geom_point()
```



## Example : add another layer- solved

```
p + geom_point() + geom_boxplot()
```





## Example : add another layer

1. Color the points by iris\$Petal.Length. - Do this inside geom\_point.
2. We wanted a scatterplot but changed our mind and now we also want boxplot on top of this scatterplot. Add + geom\_boxplot() to previous line to see what you get.

```
p + geom_point(???) +  
  ???
```

## Example : add another layer

1. Color the points by iris\$Petal.Length. - Do this inside geom\_point.
2. We wanted a scatterplot but changed our mind and now we also want boxplot on top of this scatterplot. Add + geom\_boxplot() to previous line to see what you get.

```
p + geom_point(aes(color=Petal.Length)) + geom_boxplot()
```

## Example 2: Lets build another graph!

x axis is numerical, y axis is numerical

1. Set the basic layout: We want to see if there is any connection between Petal.Length (x axis) and Sepal.Length (y) in iris dataset.

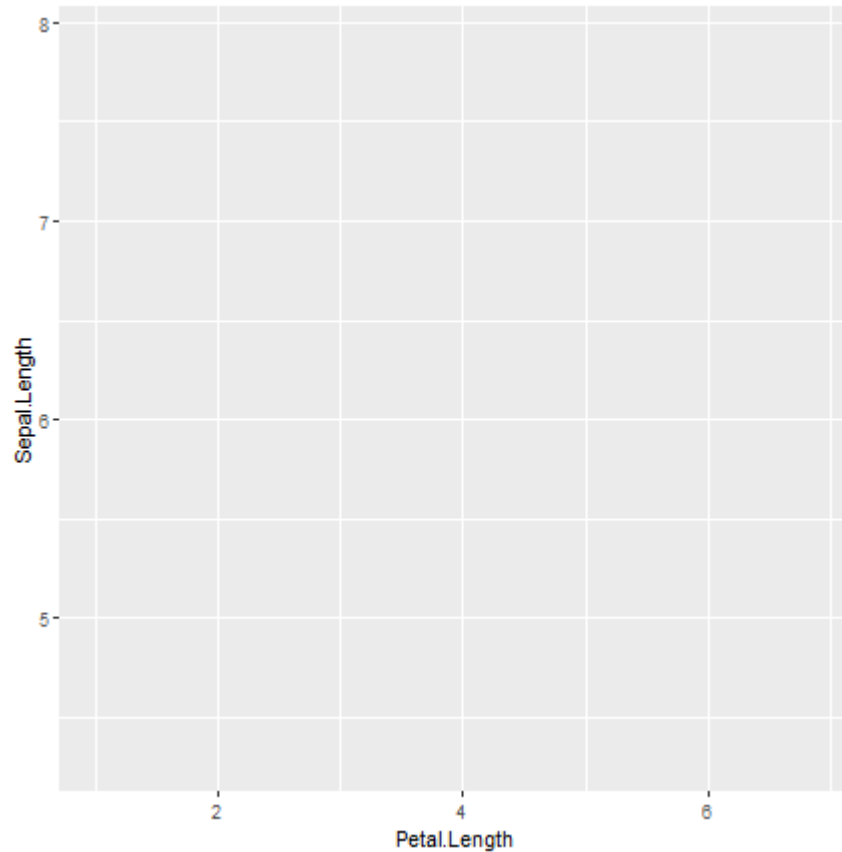
```
p2 ← ggplot(iris, aes(???, ???))  
p2
```

1. We would like to add points to the graph. Use geom\_point()

```
p2 + ???
```

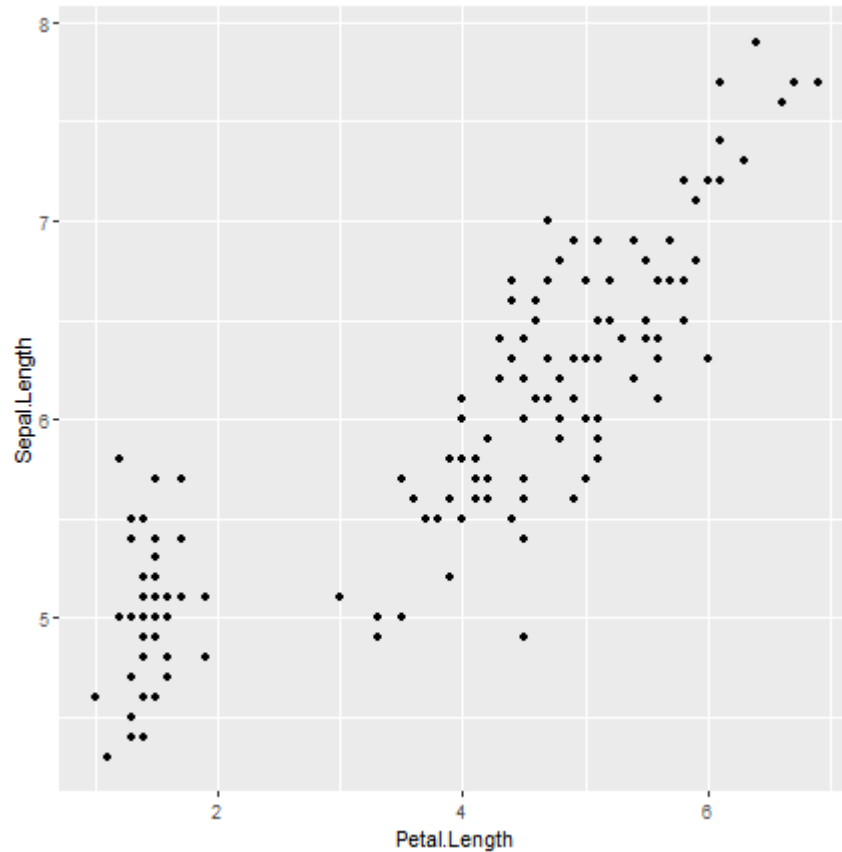
## Example 2: Lets build another graph! -SOLVED

```
p2 ←ggplot(iris, aes(Petal.Length, Sepal.Length)); p2
```



## Example 2: Lets build another graph! -SOLVED

```
p2 + geom_point()
```



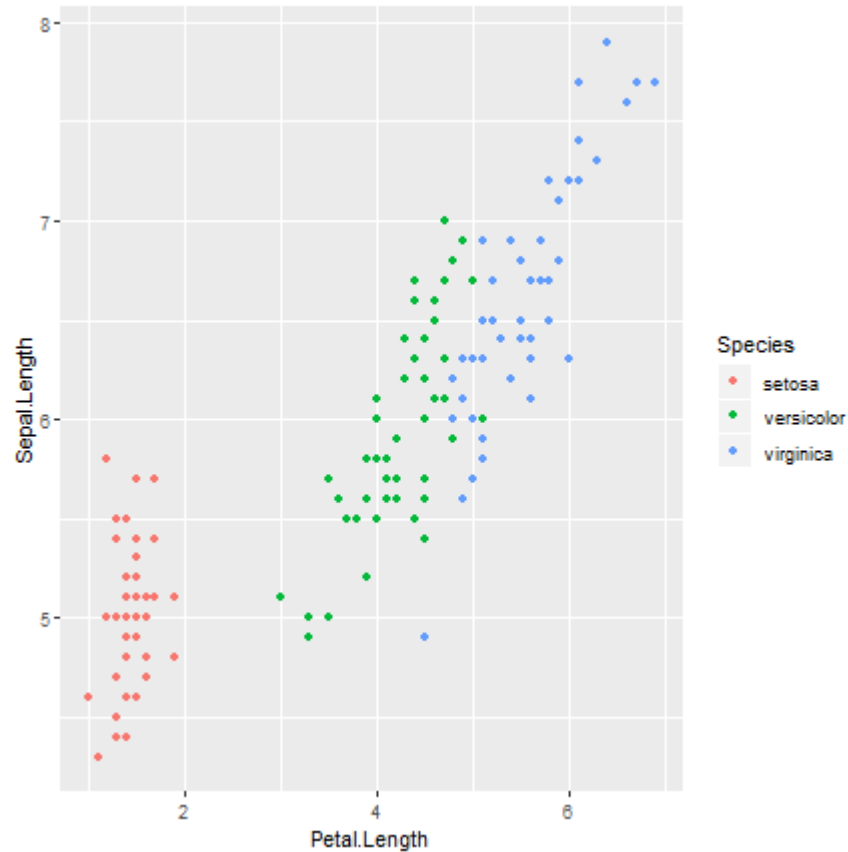
## change it

1. We don't really like the graph that much. From it we can't conclude if dependencies are species related or not.. It would look better if points could be colored by Species. But for this we need to change the aesthetics and do everything again because we set aesthetics in the first step... no worries, just reset the aesthetics inside of `geom_point()` function by `aes(color=Species)`

```
p3←p2 + geom_point(???)  
p3
```

## Example 2: change it -SOLVED

```
p3←p2 + geom_point(aes(color=Species)); p3
```



## Example 2: do some magic!

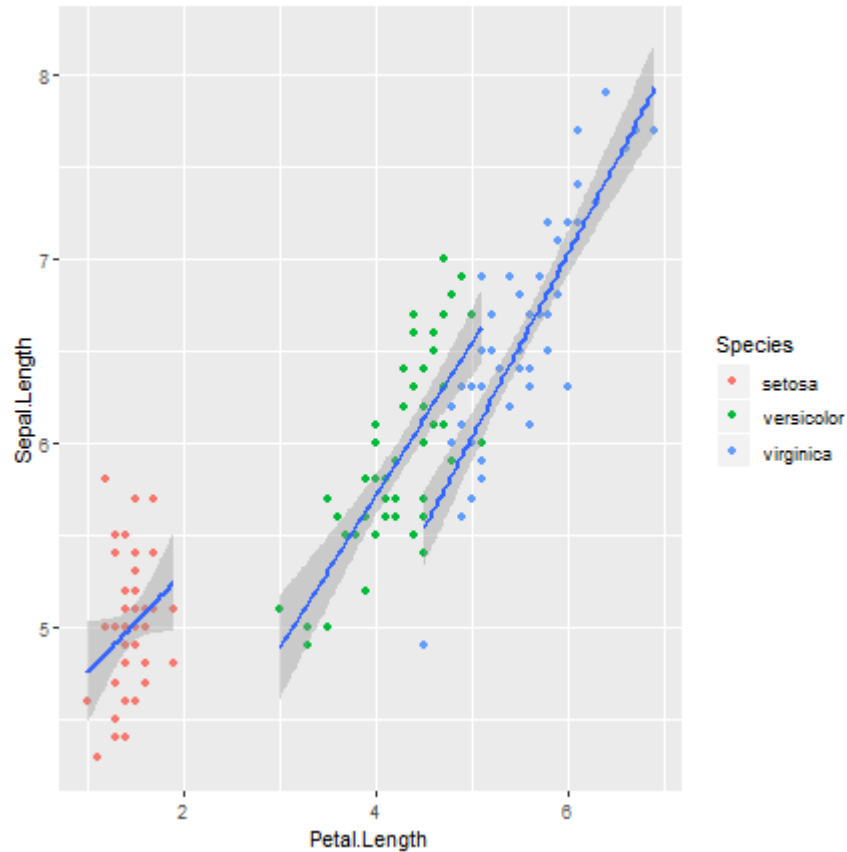
It looks to us now that if we put linear regression lines through each group of points, maybe lines would be the same for blue and green points! Lets check this by adding `geom_smooth()` to plot. Again you might need to set the aesthetics for `geom_smooth` also, but this time we want to group it by species, not color by species.

```
p4 ← p3 + ???  
p4
```



## Example 2: do some magic! - solved

```
p4 ← p3 + geom_smooth(aes(group=Species), method="lm"); p4
```



## Example 2: do some more magic!

This looks ok but now we would like for each group to appear in its own graph. For this use `facet_wrap()`. Parameter to `facet_wrap` is variable by which you would want to separate the graphs (`~variable2`). If you put `~variable2`, then the graph will be separated into as many columns as there are levels in `variable2`. Lets separate it to columns by Species variable.

```
p4 + ???
```

## Example 2: do some more magic! - solved

```
p4 + facet_wrap(~Species)
```

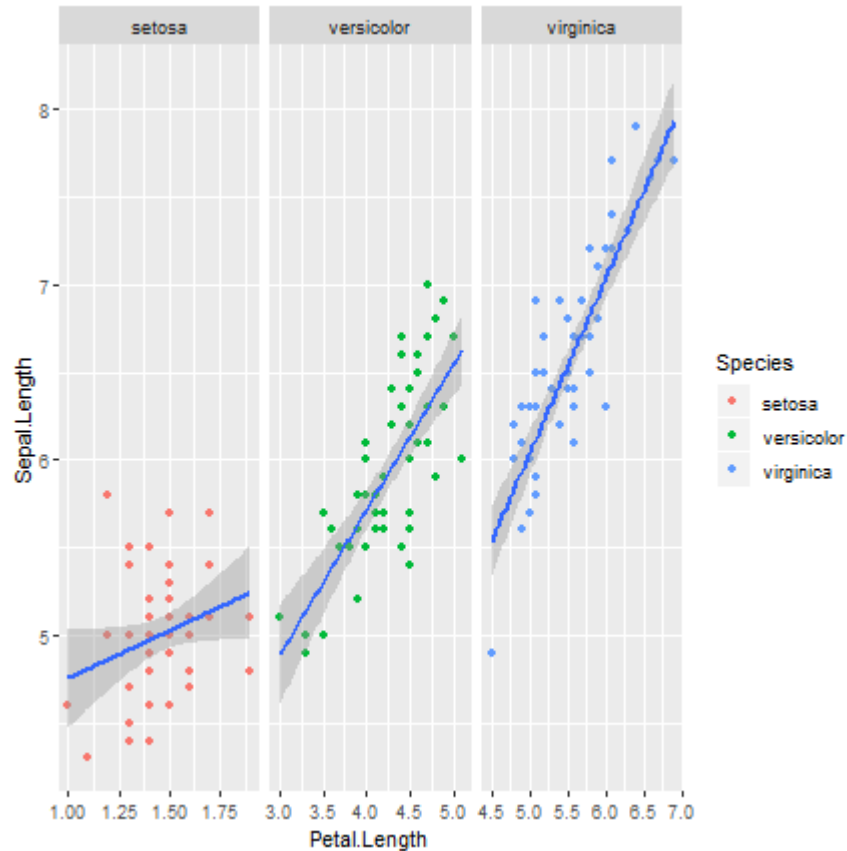
## Example 2: do some more magic again! - solved

Hmm.. graph looks kind of funny.. It is because x and y axis are automatically chosen to be the same everywhere (how smart!). but in this picture, you don't want that because each group occupies their own part and it just might look better if x axis was "free". No problem, just add `scales="free_x"` as a parameter to `facet_wrap`.

```
p4 + facet_wrap(~Species, ???)
```

## Example 2: do some more magic again! - solved

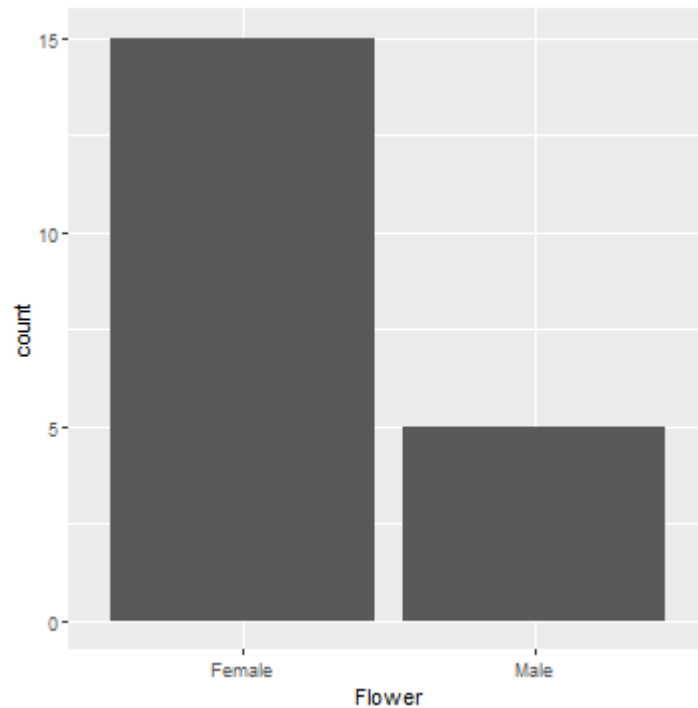
```
p4 + facet_wrap(~Species, scales="free_x")
```



## Example 3: back to basics - FILL

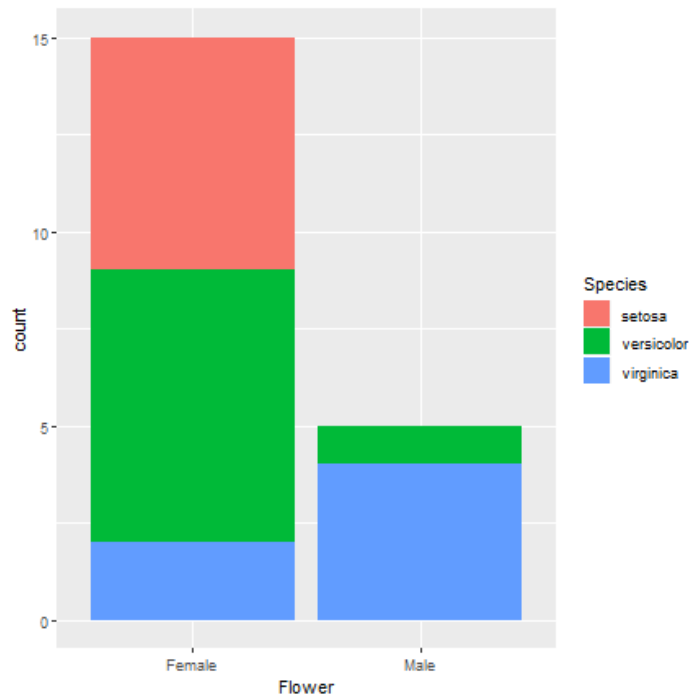
```
set.seed(200)
smallIris <- iris[sample(1:150,20),]
smallIris$Flower <- rep(c("Female","Male"),c(15,5))
```

```
ggplot(smallIris, aes(Flower))+
  geom_bar()
```

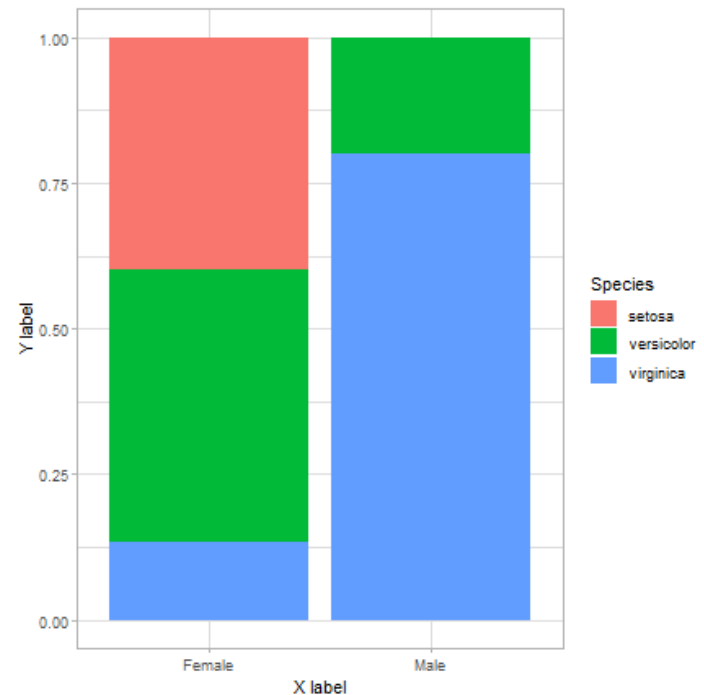


## Example 3: back to basics - FILL

```
ggplot(smallIris,  
       aes(Flower, fill=Species))  
geom_bar()
```



```
ggplot(smallIris,  
       aes(Flower, fill=Species))  
geom_bar(position="fill")+  
xlab("X label") +  
ylab("Y label") +  
theme_light()
```



Announcements:

ggplot2 cheat sheet!

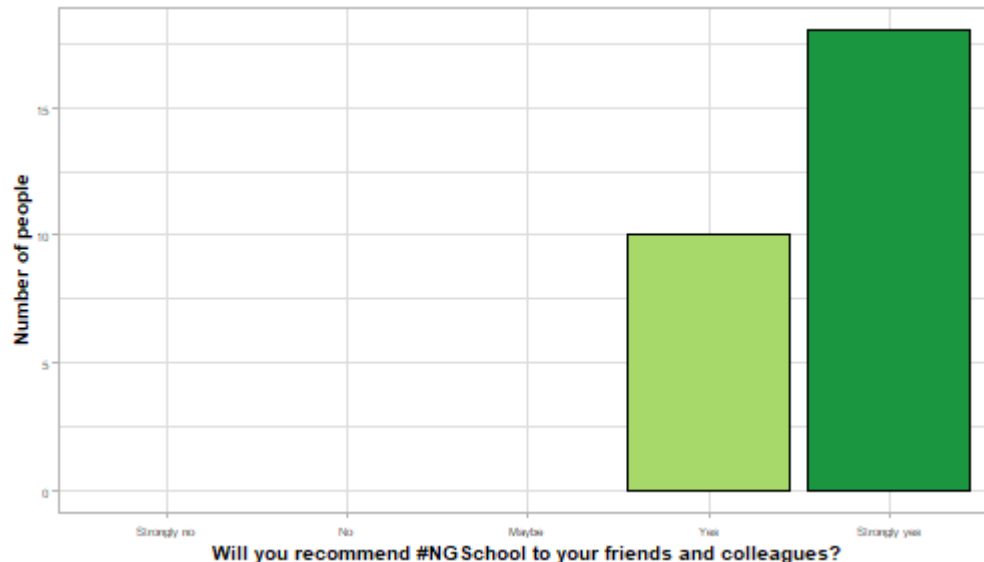


## Exercise: Recreate the following plot

```
df <- read.table("http://hex.bioinfo.hr/~mfabijanic/df.txt", header=T)
```

These are the colors from Strongly yes to Strongly no: c("#1a9641", "#a6d96a", "#ffffbf", "#fdae61", "#d7191c")

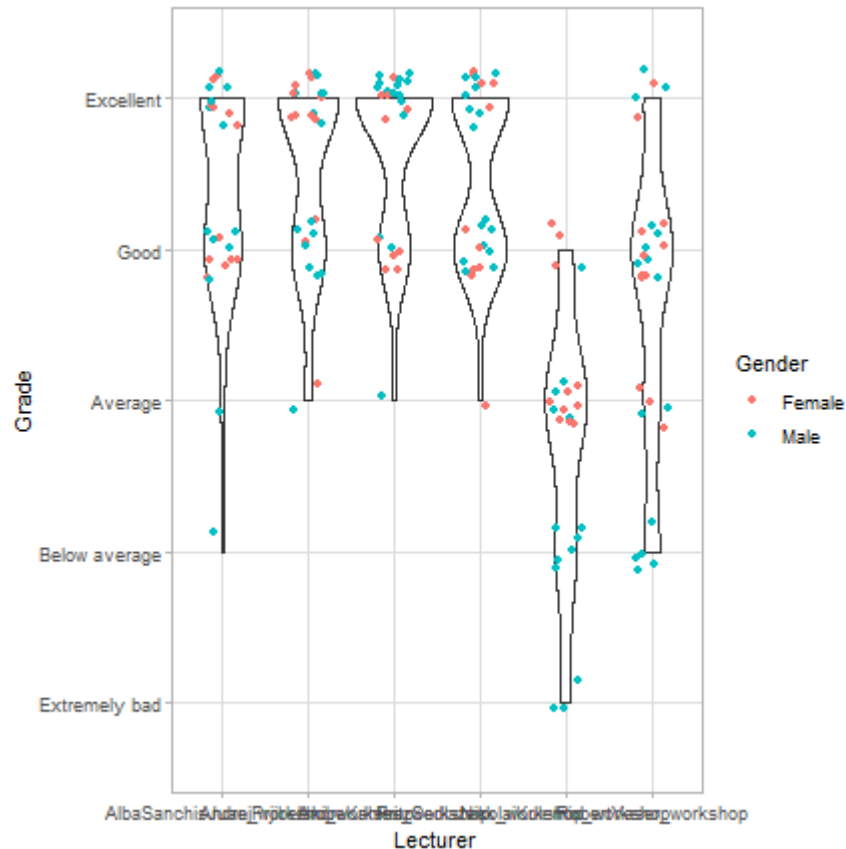
```
## <ScaleContinuousPosition>  
## Range:  
## Limits:    0 --    18
```



## Exercise: Recreate the following plot

```
works <- fread("http://hex.bioinfo.hr/~mfabijanec/works.txt", header
```

Hint: Jitter the points!



## Final magic

```
plot2 ← ggplot(works, aes(x=Lecturer, group=Lecturer, y=Grade))+  
  geom_violin()+geom_jitter(width = 0.2, height = 0.2, aes(color=Ge  
  theme(axis.text.x = element_text(angle=90))  
plot2
```