chapter5

Build a Plot Layer by Layer

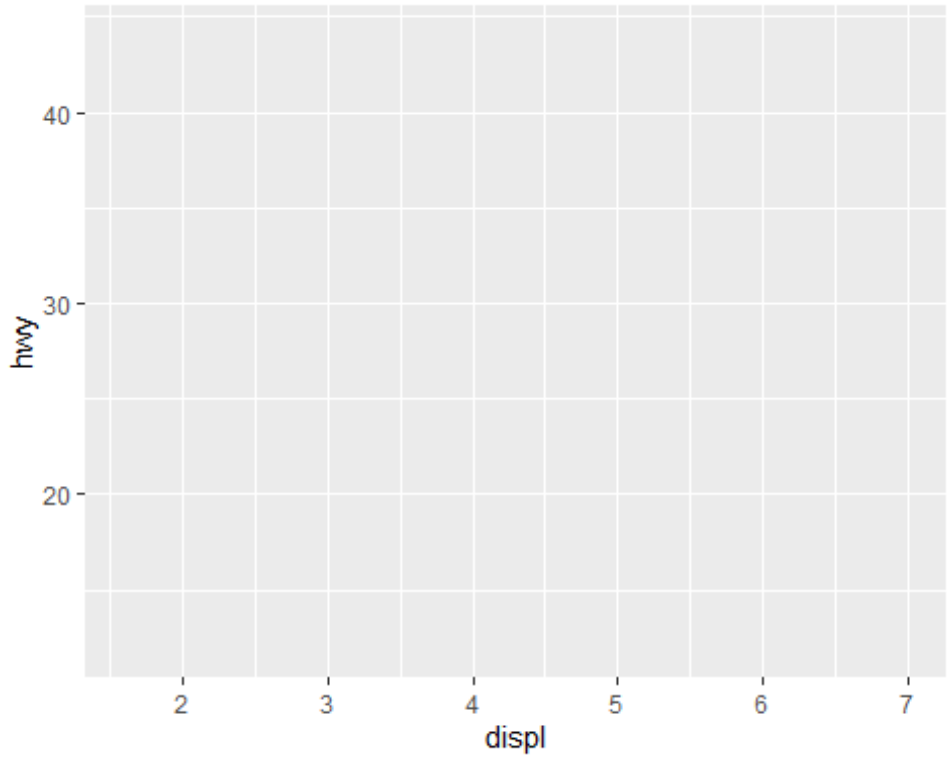
```
library(ggplot2)
library(gridExtra)
library(dplyr)
```

5.1 Introduction

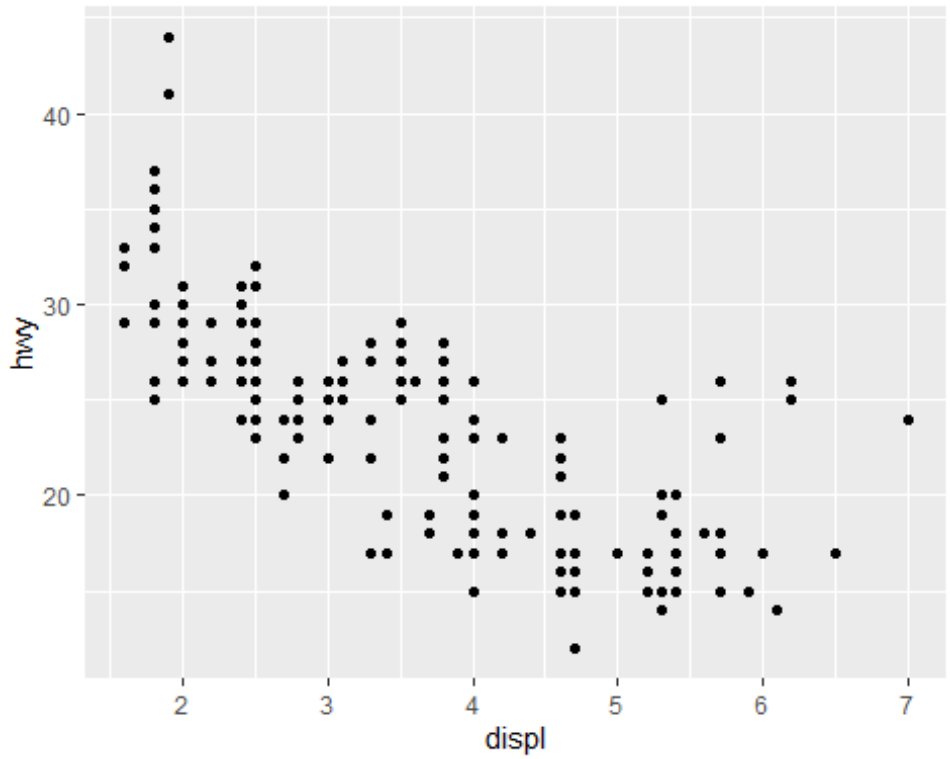
In this chapter, you'll dive into the details of a layer, and how you can control all five components: data, the aesthetic mappings, the geom, stat, and position adjustments !!

5.2 Building a Plot

```
p = ggplot(mpg, aes(displ, hwy))
p
```



```
p + geom_point()
```



geom_point() -> behind the scenes it calls the layer() funciton to create a new layer :

```
p + layer(
  mapping = NULL,
  data = NULL,
  geom = "point",
  stat = "identity",
  position = "identity"
)
```

- **mapping** : A set of aesthetic mappings, specified using the aes() function. If NULL, uses the default mapping set in ggplot().
- **data** : A dataset which overrides the default plot dataset. If NULL, use the default data specified in ggplot().
- **geom** : Geoms can have additional arguments. All geoms take aesthetics as parameters.
- **stat** : You only need to set one of stat and geom: every geom has a default stat, and every stat a default geom.
- **position** : method used to adjust overlapping objects, like jittering, stacking or dodging.

5.3 Data

- Every layer must have some data associated with it, and that data must be in a tidy data frame.
- The data on each layer doesn't need to be the same, and it's often useful to combine multiple datasets in a single plot.

5.4 Aesthetic Mappings

The aesthetic mappings, defined with aes(), describe how variables are mapped to visual properties or aesthetics.

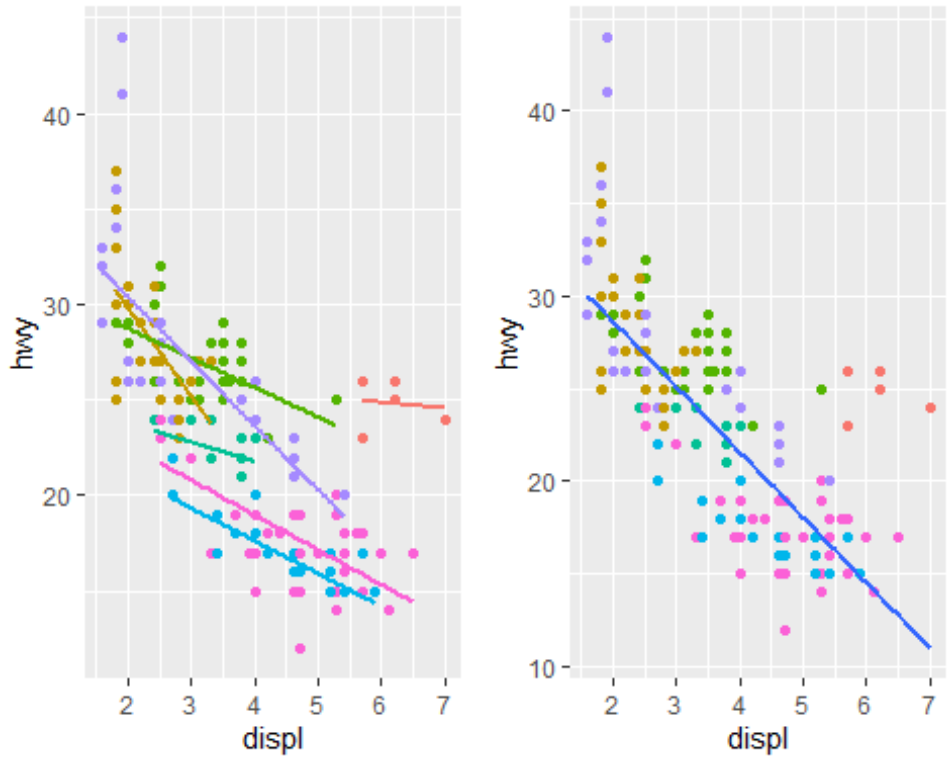
5.4.1 Speifying the aesthetics in the plot vs in the layers

If you only have one layer in the plot, the way you specify aesthetics doesn't make any difference.

```
ggplot(mpg, aes(displ, hwy, colour = class)) +
geom_point()

ggplot(mpg, aes(displ, hwy)) +
geom_point(aes(colour = class))

ggplot(mpg, aes(displ)) +
geom_point(aes(y = hwy, colour = class))

ggplot(mpg) +
geom_point(aes(displ, hwy, colour = class))
```

But be careful when adding additional layers

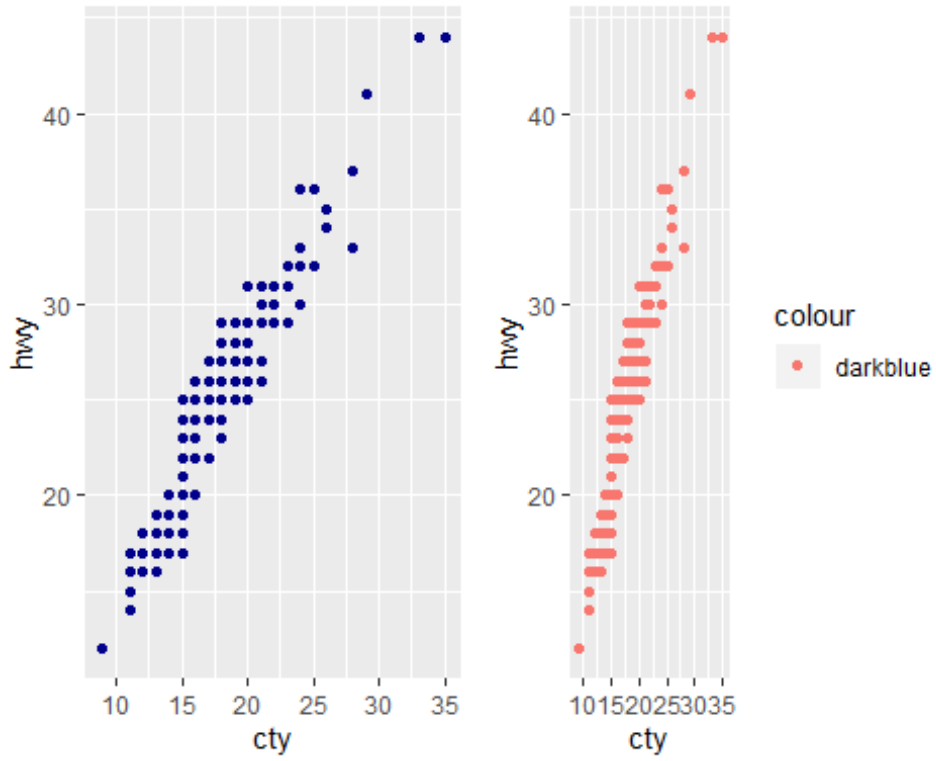
```
p1 = ggplot(mpg, aes(displ, hwy, colour = class)) +
geom_point() +
geom_smooth(method = "lm", se = FALSE) +
theme(legend.position = "none")

p2 = ggplot(mpg, aes(displ, hwy)) +
geom_point(aes(colour = class)) +
geom_smooth(method = "lm", se = FALSE) +
theme(legend.position = "none")

grid.arrange(p1,p2, ncol=2)
```



5.4.2 Setting vs Mapping

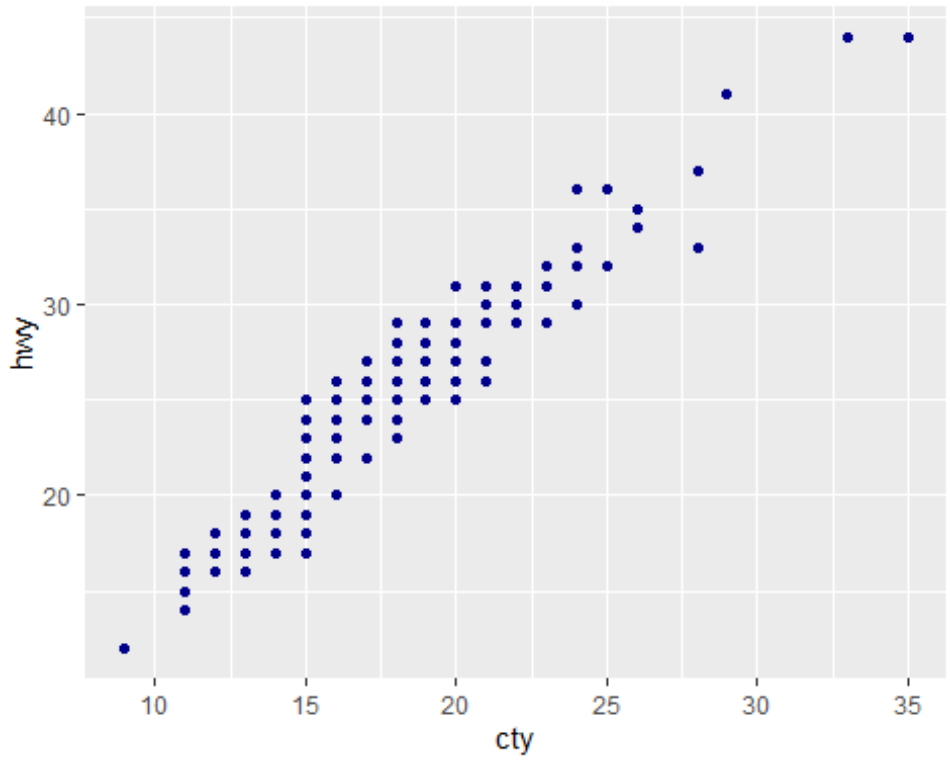
- We **map** an aesthetic to a variable (e.g., aes(colour = cut))
- or **set** it to a constant (e.g., colour = "red").

```
p1 = ggplot(mpg, aes(cty, hwy)) +
  geom_point(color = "darkblue")

p2 = ggplot(mpg, aes(cty, hwy)) +
  geom_point(aes(color = "darkblue"))

grid.arrange(p1,p2, ncol=2)
```



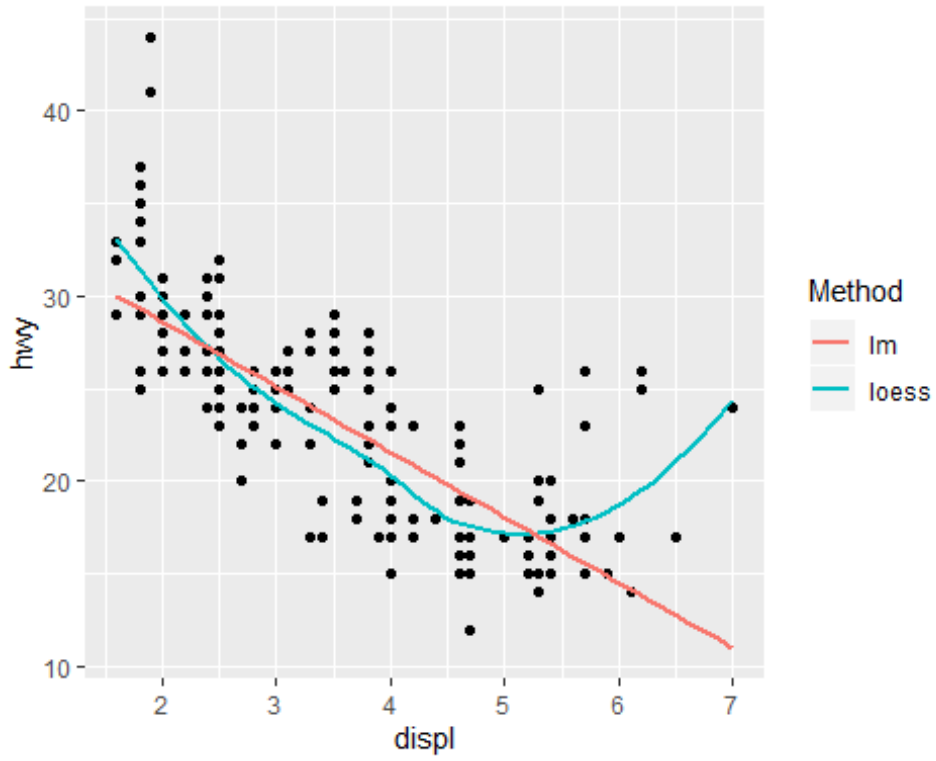
u can override by *scale_colour_identity()*

```
ggplot(mpg, aes(cty, hwy)) +
geom_point(aes(colour = "darkblue")) +
scale_colour_identity()
```



sometimes useful to map aesthetics to constants. you can “name” each layer !

```
ggplot(mpg, aes(displ, hwy)) +
geom_point() +
geom_smooth(aes(colour = "loess"), method = "loess", se = FALSE) +
geom_smooth(aes(colour = "lm"), method = "lm", se = FALSE) +
labs(colour = "Method")
```



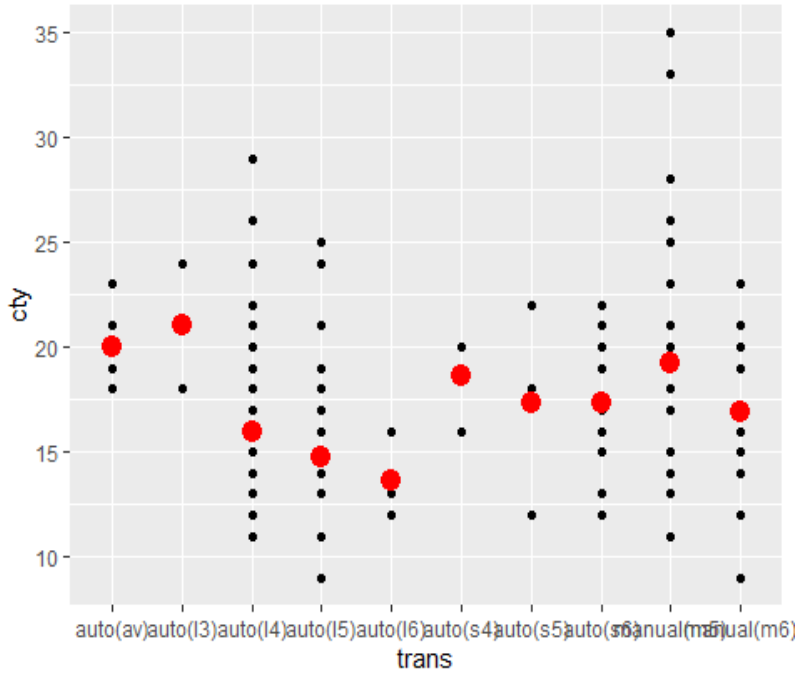
5.5 Geoms

- perform the actual rendering of the layer, controlling the type of plot that you create.
- cheatsheet

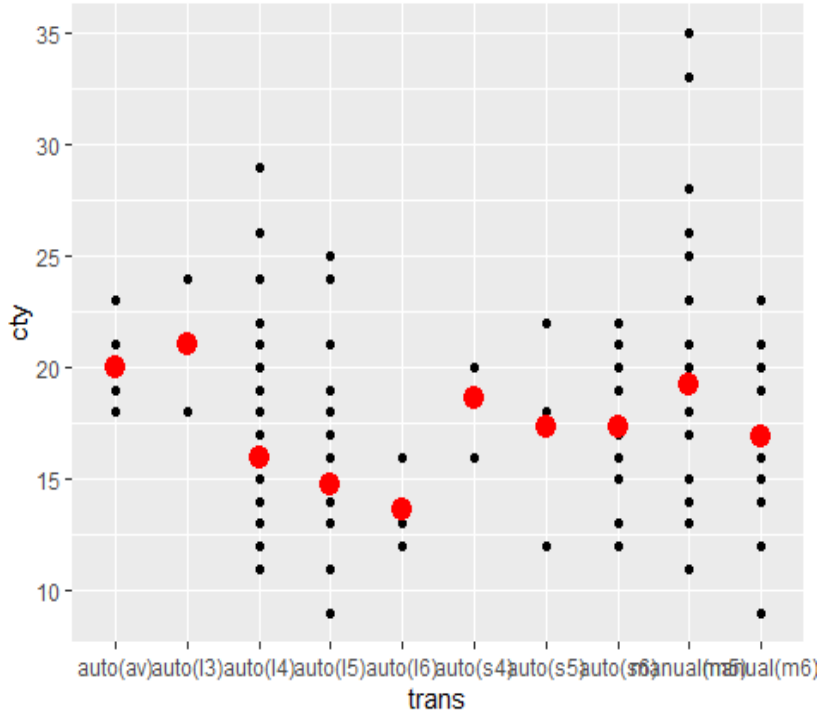
5.6 Stats

- You can either add a stat_() function and override the default geom, or
- add a geom_() function and override the default stat:

```
ggplot(mpg, aes(trans, cty)) +
  geom_point() +
  stat_summary(geom = "point", fun.y = "mean", color = "red", size = 4)
```




```
ggplot(mpg, aes(trans, cty)) +
geom_point() +
geom_point(stat = "summary", fun.y = "mean", colour = "red", size = 4)
```



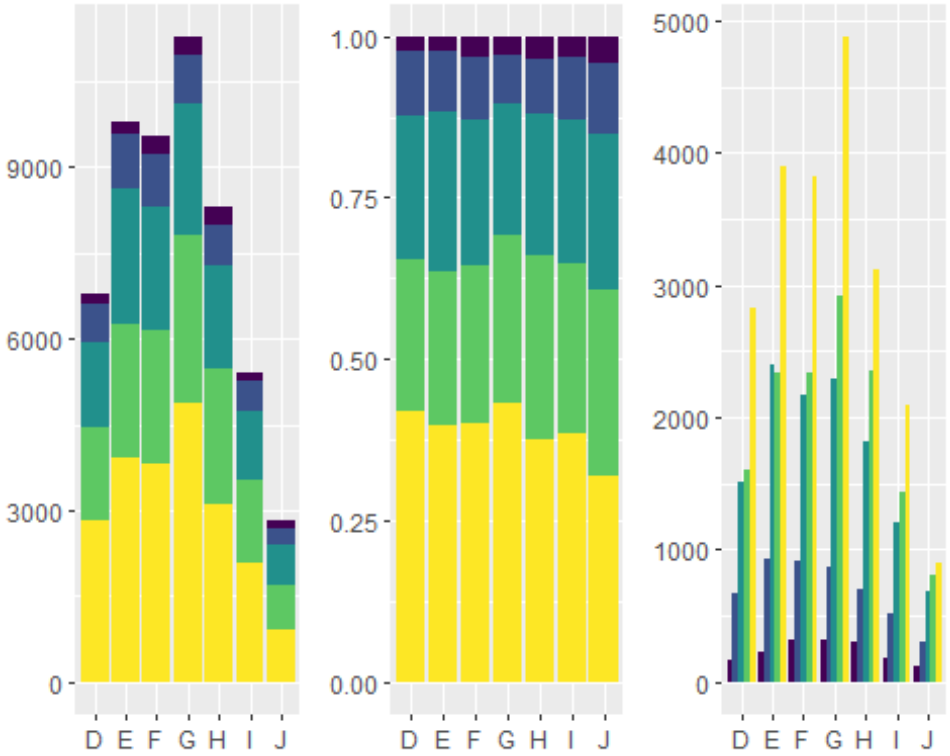
5.7 Position Adjustments

Position adjustments apply minor tweaks to the position of elements within a layer.

- Three adjustments apply primarily to *bars* :

1. position stack(): stack overlapping bars (or areas) on top of each other.
2. position fill(): stack overlapping bars, scaling so the top is always at 1.
3. position dodge(): place overlapping bars (or boxplots) side-by-side.

```
dplot = ggplot(diamonds, aes(color, fill = cut))+
  xlab(NULL) + ylab(NULL) + theme(legend.position = "none")

a1 = dplot + geom_bar() #default : stack
a2 = dplot + geom_bar(position = "fill")
a3 = dplot + geom_bar(position = "dodge")

grid.arrange(a1,a2,a3, ncol=3)
```



- There are three position adjustments that are primarily useful for *points*:

1. position nudge(): move points by a fixed offset.
2. position jitter(): add a little random noise to every position.
3. position jitterdodge(): dodge points within groups, then add a little random noise.

```
a1 = ggplot(mpg, aes(displ, hwy)) +
  geom_point(position = "jitter")

a2 = ggplot(mpg, aes(displ, hwy)) +
  geom_point(position = position_jitter(width = 0.05, height = 0.5))

# geom_jitter is convenient !
a3 = ggplot(mpg, aes(displ, hwy)) +
  geom_jitter(width = 0.05, height = 0.5)

grid.arrange(a1,a2,a3, ncol=3)
```