Plot() type 参数

'p'(默认，对于单个数值向量)：这是默认的绘图类型，用于绘制散点图（scatter plot）。如果 plot()函数只接受一个数值向量作为输入，那么它会将该向量视为 y 坐标，而 x 坐标则默认是 1 到 n（n 是向量的长度）。对于两个数值向量（分别代表 x 和 y 坐标），则绘制散点图。

'l'：绘制线图（line plot）。它会将数据点用线连接起来。如果数据点之间有缺失值，那么线可能会中断。

'b'：同时绘制散点图和线图（both）。每个数据点都会有一个标记，并且这些点之间会用线连接起来。

'c'：只绘制数据点的中心点（centers of points），但这通常与'p'（散点图）非常相似，除非你的数据点非常大或具有特殊的标记。

'o'：绘制线图，并在每个数据点上添加一个标记（overplot）。这类似于'b'，但通常'o'中的线条不会通过数据点的中心，而'b'会。

'h'：绘制垂直线（histogram-like），这是用于单个数值向量的情况。它会在每个数据点处绘制一个垂直线，线的长度可能由 freq 参数控制（如果提供的话）。

's'：阶梯图（stair steps）。在 x 轴方向上，它会在每个数据点处绘制一个水平阶梯，类似于阶梯函数。

'S'：另一种阶梯图，但在每个数据点之间绘制斜线，而不是水平的阶梯。

'n'：不绘制任何点或线，只创建一个空的绘图窗口或框架。这通常用于后续使用其他绘图函数（如 points(), lines(), text(), 等）在相同的坐标轴上添加内容。

Barplot vs. histogram

- Bar plot（条形图）：通常用于展示分类数据。在条形图中，不同的条形代表不同的分类，条形的高度或长度则代表每个分类的数值。
- Histogram（直方图）：通常用于展示连续数据。在直方图中，每个条形的宽度代表一个数值范围（即 bin），高度或长度则代表落在这个范围内的数据点数量。
- Bar plot：除了可以展示分类数据的数量之外，还可以展示分类之间的相对大小关系以及它们的变化趋势。
- Histogram：主要展示数据的分布情况，例如数据的集中趋势、离散程度以及分布形状等。

适用场景：

- Bar plot：适用于展示类别之间的比较，例如不同地区的销售数据、不同产品的市场份额等。
- Histogram：适用于展示数据的分布特征和变化规律，例如用户年龄分布、一段时间内用户的点击次数的分布等。

```
heights <- c(10, 15, 7, 20)
barplot(heights, main = "Basic Bar Plot", xlab = "Category", ylab = "Value")
# 如果你有分类名称，可以传递给 names.arg 参数
names <- c("A", "B", "C", "D")
barplot(heights, main = "Basic Bar Plot", xlab = "Category", ylab = "Value", names.arg = names)
ggplot(df, aes(x = Category, y = Value)) + geom_bar(stat = "identity", fill = "steelblue") + labs(title = "Bar Plot with ggplot2", x = "Category", y = "Value")
```

```
# stat = "identity"参数表示我们直接使用数据框中的值，而不是让 ggplot2 计算计数或频率

hist(data, main = "Histogram with base R", xlab = "Value", ylab = "Frequency", border = "black", col = "lightblue", breaks = 30)
# breaks 参数指定了条形的数量或边界

ggplot(data.frame(data), aes(x = data)) +
  geom_histogram(stat = "identity", binwidth = 0.5, fill = "lightblue", color = "black") +  labs(title = "Histogram with ggplot2", x = "Value", y = "Frequency")
# 由于 geom_histogram()默认计算数据的分箱（binning），所以通常不需要像 hist()那样指定 breaks。但是，你可以通过 binwidth 参数来设置每个箱子的宽度。另外，stat = "identity" 在这里是不必要的，因为 geom_histogram()默认就是计算分箱统计的。
```

Density plot

1.

```
ggplot(data=mice_data) +
  geom_line(aes(x=before, col='before'), stat='density') +
  geom_line(aes(x=after,col='after'), stat='density')
```

2.

```
ggplot(data.frame(data), aes(x = data)) +
  geom_density(fill = "lightblue", alpha = 0.5) +
  labs(title = "Density Plot with ggplot2", x = "Value", y = "Density")
```

3.

```
bw <- Hmisc::Hmisc.bw.Hn(data) # 使用 Hmisc 包来计算带宽，也可以选择其他方法或手动指定带宽
fit <- density(data, bw = bw)
plot(fit, main = "Density Plot with base R", xlab = "Value", ylab = "Density", col = "lightblue", border = "black")
# 注意，这里使用了 Hmisc 包中的 Hmisc.bw.Hn()函数来计算带宽（bandwidth），它决定了密度估计的平滑程
```

Q-Q plot

1.

```
library(ggpubr)
ggqqplot(data)
```

2.

```
qqnorm(data) # 绘制正态 QQ 图
qqline(data, col = "red", lwd = 2) # 添加通过原点的线，用于比较
```

abline()
1.
```
plot(1:10, 1:10, type = "n") # 创建一个空的图形窗口
abline(a = 0, b = 1, col = "red", lwd = 2) # 绘制通过原点的直线，斜率为 1
```
2.
```
plot(1:10, rnorm(10), type = "p") # 绘制散点图
abline(h = c(-1, 0, 1, 2), col = "blue", lty = 2) # 绘制多条水平线
```
3.
```
set.seed(123) # 设置随机种子以便结果可复现
x <- 1:100
y <- 2 * x + rnorm(100, sd = 10) # 生成一些带有噪声的数据
fit <- lm(y ~ x) # 拟合线性模型
plot(x, y, pch = 19) # 绘制散点图并添加回归线
abline(fit, col = "green", lwd = 2) # 使用模型的系数绘制回归线
```

画图出错
```
# First, let's clean up the environment.
rm(list=ls())
dev.off()
```

比如说有一维坐标是 id 或者是年份，这一维可能是废的，如果问的问题不是同年份相关的变化的话，所以这个时候对原始数据画 bar plot 就不合适，比如说 Semester 1 考试中的第二题
那么要进行的相当于是进行一个桶的操作，就是我们只关心 patient 数量的 count 的分布
```
g2.3 = ggplot(hosp1, mapping = aes(x = patients, fill = weekday))
g2.3 = g2.3 + geom_histogram(position = "identity", alpha = 0.2, bins = 10)
g2.3
```

boxplot()
1.
boxplot 凹陷下去的是 median 而不是 mean
```
boxplot(hosp1$patients ~ hosp1$weekday, notch = T, ylab = "patients per day")
```
2.
绘制 boxplot 并把 primary points 标出来并加上一些 jitter
实际上同时绘制出 outlier 更加方便
在 R 中，geom_boxplot() 是 ggplot2 包中用于绘制箱型图（boxplot）的函数。默认情况下，箱型图显示了中位数、四分位数以及被视为“异常值”或“离群点”（outliers）的点（这些点通常位于四分位距的 1.5 倍之外）。

然而，geom_boxplot()本身并不直接支持在箱型图上标记非离群点（即位于箱型图内部或须线范围内的点）并添加 jitter（抖动）。但是，你可以通过结合使用 geom_point()（或 geom_jitter()）来达到这个效果。

```
# 假设你有一个名为 df 的数据框，它有一个名为 group 的分类变量和一个名为 value 的数值变量
# df <- data.frame(group = ..., value = ...)

# 计算 IQR（四分位距）和离群点的阈值
IQR <- IQR(df$value)
upper_limit <- quantile(df$value, probs = 0.75) + 1.5 * IQR
lower_limit <- quantile(df$value, probs = 0.25) - 1.5 * IQR

# 创建一个逻辑向量来标记非离群点
non_outliers <- df$value >= lower_limit & df$value <= upper_limit

# 绘制箱型图并叠加非离群点（使用 geom_jitter 添加抖动）
ggplot(df, aes(x = group, y = value)) +
geom_boxplot(width = 0.5) +
geom_jitter(data = df[non_outliers, ], aes(x = group, y = value), width = 0.2, height = 0)
```

注意：在上面的代码中，geom_jitter()的 width 和 height 参数用于控制抖动的大小。将 height 设置为 0 意味着只在 x 轴方向上添加抖动（这对于分类的 x 轴变量很有用）。如果你想要在两个方向上都有抖动，可以调整这两个参数。

对 data 这个一维 vector 画 histogram
可以直接 hist(data)
如果要用 ggplot 的话，他只接受 data frame,实际上这也算是一个缺陷

```
ggplot(data.frame(value = data), aes(x = value)) +
  geom_histogram()
```

你千万不能在 geom_histogram()中设置 stat=identity，那样会报错，因为 stat = "identity" 模式下，geom_histogram()不会进行数据的分箱（binning）操作，因为它期望数据已经分箱好了。在 stat = "identity"时，你需要提供 y 美学映射，因为 geom_histogram()在默认情况下（即不使用 stat = "identity"）会自动计算每个箱（bin）中的频数作为 y 轴的值。
如果你确实有一个已经分箱好的数据框，并且想要使用 stat = "identity"，你应该确保你的数据框有一个表示频数的列，并且将这个列映射到 y 美学上。但是，在大多数情况下，你不需要这样做来绘制一个直方图。

```
ggplot(data.frame(value = data), aes(x = value)) +
  geom_histogram() +
  geom_line(aes(color = "red"), stat = "density")
```

上面这个是错的，因为 y 轴代表的是 count，然后把 density 加上去之后虽然 y lab 变成 density 了，但是标度没有变，所以 density 非常非常小，在[0,1]之间，看起来就

和 x axis 重合了
所以很明显 density plot 和 histogram 不能画在一张图上，y 轴不同肯定会出问题！