R 可视化指南（初版）

黄子维

2021

目录

1 引言

本文档为所学的一次知识整理，拒绝一切盈利为目的的转载，仅供个人学习参考使用。

2 零维数据

2.1 表格的整理和呈现（例一）

表格不应只作为储存数据的工具，它也应该具有帮助演讲者阐释一种或者多种观点的功能。以加拿大安大略省各个城市水体酸度表为例:

County or District	Number and Percentage of Lakes in each Alkalinity Class										Total No. of Lakes Evaluated
	Acidic		Extreme Sensitivity		Moderate Sensitivity		Low Sensitivity		No Sensitivity		
	No.	(%)	No.	(%)	No.	(%)	No.	(%)	No.	(%)	
Algoma	68	6.3	162	15.0	425	39.2	194	17.9	234	21.6	1083
Bruce	0	0.0	0	0.0	0	0.0	0	0.0	7	100.0	7
Cochrane	2	0.7	6	2.1	10	3.5	27	9.6	237	84.0	282
Durham	0	0.0	0	0.0	0	0.0	0	0.0	1	100.0	1
Frontenac	0	0.0	0	0.0	2	2.5	12	15.0	66	82.5	80
Grey	0	0.0	0	0.0	0	0.0	0	0.0	3	100.0	3
Haliburton	9	2.4	124	32.5	167	43.7	46	12.0	36	9.4	382
Hastings	0	0.0	7	4.5	68	43.6	27	17.3	54	34.6	156
Huron	0	0.0	0	0.0	0	0.0	0	0.0	1	100.0	1
Kenora	0	0.0	1	0.3	106	27.7	128	33.5	147	38.5	382
Lanark	0	0.0	0	0.0	0	0.0	1	2.5	39	97.5	40
Leeds	0	0.0	0	0.0	0	0.0	0	0.0	32	100.0	32
Lennox & Add.	0	0.0	4	4.4	26	28.6	21	23.1	40	44.0	91
Manitoulin	27	49.1	21	38.2	2	3.6	1	1.8	4	7.3	55
Middlesex	0	0.0	0	0.0	0	0.0	0	0.0	1	100.0	1
Muskoka	11	3.4	91	28.5	193	60.5	13	4.1	11	3.4	319
Nipissing	23	2.8	187	22.5	506	60.8	103	12.4	13	1.6	832
Northumberland	0	0.0	0	0.0	0	0.0	0	0.0	1	100.0	1
Parry Sound	18	4.2	147	34.1	233	54.1	27	6.3	6	1.4	431
Peel	0	0.0	0	0.0	0	0.0	0	0.0	1	100.0	1
Peterborough	0	0.0	3	4.5	10	14.9	9	13.4	45	67.2	67
Prince Edward	0	0.0	0	0.0	0	0.0	0	0.0	9	100.0	9
Rainy River	0	0.0	15	5.4	159	57.4	67	24.2	36	13.0	277
Renfrew	2	0.5	29	8.0	174	47.8	98	26.9	61	16.8	364
Simcoe	0	0.0	0	0.0	8	38.1	2	9.5	11	52.4	21
Stormont	0	0.0	0	0.0	0	0.0	0	0.0	1	100.0	1
Sudbury	154	18.4	155	18.5	217	25.9	123	14.7	188	22.5	837
Thunder Bay	2	0.3	29	4.1	166	23.2	218	30.5	299	41.9	714
Timiskamg	20	9.5	20	9.5	45	21.3	66	31.3	60	28.4	211
Victoria	0	0.0	1	2.6	22	56.4	2	5.1	14	35.9	39
York	0	0.0	0	0.0	0	0.0	0	0.0	2	100.0	2
	336	5.0	1002	14.9	2539	37.8	1185	17.6	1660	24.7	6722

图 1: Acidity of Ontario Lakes

对上表进行**处理多余小数点、合并相似组**及**处理多余数据（每一组别湖体数量）**后得:

County or District	Percentage of Lakes in each Alkalinity Class Ordered by Region					Total No. of Lakes Evaluated
	Acidic (%)	Extreme Sensitivity (%)	Moderate Sensitivity (%)	Low Sensitivity (%)	No Sensitivity (%)	
North-western Ontario						
Rainy River	0	5	57	24	13	277
Thunder Bay	0	4	23	31	42	714
Kenora	0	0	28	34	38	382
North-eastern Ontario						
Manitoulin	49	38	4	2	7	55
Sudbury	18	19	26	15	23	837
Timiskamg	10	10	21	31	28	211
Algoma	6	15	39	18	22	1083
Parry Sound	4	34	54	6	1	431
Nipissing	3	22	61	12	2	832
Cochrane	1	2	4	10	84	282
South-eastern Ontario						
Renfrew	1	8	48	27	17	364
Hastings	0	4	44	17	35	156
Lennox & Add.	0	4	29	23	44	91
Frontenac	0	0	3	15	82	80
Lanark	0	0	0	2	98	40
Leeds	0	0	0	0	100	32
Prince Edward	0	0	0	0	100	9
Stormont	0	0	0	0	100	1
South-central Ontario						
Muskoka	3	29	61	4	3	319
Haliburton	2	33	44	12	9	382
Peterborough	0	5	15	13	67	67
Victoria	0	3	56	5	36	39
Durham	0	0	0	0	100	1
Northumberland	0	0	0	0	100	1
South-western Ontario						
Simcoe	0	0	38	10	52	21
Bruce	0	0	0	0	100	7
Grey	0	0	0	0	100	3
York	0	0	0	0	100	2
Huron	0	0	0	0	100	1
Middlesex	0	0	0	0	100	1
Peel	0	0	0	0	100	1
	5	15	38	18	25	6722

图 2: Acidity of Ontario Lakes

对上表进行**降序排列**及**数值分组**后得：

County or District	Acidic (%)	Extreme Sensitivity (%)	Moderate Sensitivity (%)	Low Sensitivity (%)	No Sensitivity (%)	Total No. of Lakes Evaluated
	Percentage of Lakes in each Alkalinity Class Ordered by Acidity					
Manitoulin	49	38	4	2	7	55
Sudbury	18	19	26	15	23	837
Timiskamg	10	10	21	31	28	211
Algoma	6	15	39	18	22	1083
Parry Sound	4	34	54	6	1	431
Muskoka	3	29	61	4	3	319
Nipissing	3	22	61	12	2	832
Haliburton	2	33	44	12	9	382
Renfrew	1	8	48	27	17	364
Cochrane	1	2	4	10	84	282
Rainy River	0	5	57	24	13	277
Peterborough	0	5	15	13	67	67
Hastings	0	4	44	17	35	156
Lennox & Add.	0	4	29	23	44	91
Thunder Bay	0	4	23	31	42	714
Victoria	0	3	56	5	36	39
Simcoe	0	0	38	10	52	21
Kenora	0	0	28	34	38	382
Frontenac	0	0	3	15	82	80
Lanark	0	0	0	2	98	40
Leeds	0	0	0	0	100	32
Prince Edward	0	0	0	0	100	9
Bruce	0	0	0	0	100	7
Grey	0	0	0	0	100	3
York	0	0	0	0	100	2
Durham	0	0	0	0	100	1
Huron	0	0	0	0	100	1
Middlesex	0	0	0	0	100	1
Northumberland	0	0	0	0	100	1
Peel	0	0	0	0	100	1
Stormont	0	0	0	0	100	1
	5	15	38	18	25	6722

图 3: Acidity of Ontario Lakes

对上表进行**处理非关注数据（最后一列）**及**添加脚注**后得:

County or District	Percentage of Lakes in some Alkalinity Classes: Acidic (%)	Extreme Sensitivity (%)	Moderate Sensitivity (%)	Low Sensitivity (%)	Total No. of Lakes Evaluated
Manitoulin[1,3]	49	38	4	2	55
Sudbury[1,2,3]	18	19	26	15	837
Timiskamg[1,2,3]	10	10	21	31	211
Algoma[3]	6	15	39	18	1083
Parry Sound[1,3]	4	34	54	6	431
Muskoka[1,3]	3	29	61	4	319
Nipissing[1]	3	22	61	12	832
Haliburton[1]	2	33	44	12	382
Renfrew[1]	1	8	48	27	364
Cochrane[2]	1	2	4	10	282
Rainy River	0	5	57	24	277
Peterborough	0	5	15	13	67
Hastings	0	4	44	17	156
Lennox & Add.	0	4	29	23	91
Thunder Bay	0	4	23	31	714
Victoria	0	3	56	5	39
Simcoe	0	0	38	10	21
Kenora	0	0	28	34	382
Frontenac	0	0	3	15	80
Lanark	0	0	0	2	40
Leeds	0	0	0	0	32
Prince Edward	0	0	0	0	9
Bruce	0	0	0	0	7
Grey	0	0	0	0	3
York[4]	0	0	0	0	2
Durham	0	0	0	0	1
Huron	0	0	0	0	1
Middlesex	0	0	0	0	1
Northumberland	0	0	0	0	1
Peel[4]	0	0	0	0	1
Stormont	0	0	0	0	1
	5	15	38	18	6722

[1]Non-ferrous smelters in Sudbury
[2]Non-ferrous smelter in Timmins
[3]Smelters in Algoma
[4]Smelters in Hamilton

图 4: Acidity of Ontario Lakes

前后对比：

County or District	Number and Percentage of Lakes in each Alkalinity Class										Total No. of Lakes Evaluated
	Acidic		Extreme Sensitivity		Moderate Sensitivity		Low Sensitivity		No Sensitivity		
	No.	(%)	No.	(%)	No.	(%)	No.	(%)	No.	(%)	
Algoma	68	6.3	162	15.0	425	39.2	194	17.9	234	21.6	1083
Bruce	0	0.0	0	0.0	0	0.0	0	0.0	7	100.0	7
Cochrane	2	0.7	6	2.1	10	3.5	27	9.6	237	84.0	282
Durham	0	0.0	0	0.0	0	0.0	0	0.0	1	100.0	1
Frontenac	0	0.0	0	0.0	2	2.5	12	15.0	66	82.5	80
Grey	0	0.0	0	0.0	0	0.0	0	0.0	3	100.0	3
Haliburton	9	2.4	124	32.5	167	43.7	46	12.0	36	9.4	382
Hastings	0	0.0	7	4.5	68	43.6	27	17.3	54	34.6	156
Huron	0	0.0	0	0.0	0	0.0	0	0.0	1	100.0	1
Kenora	0	0.0	1	0.3	106	27.7	128	33.5	147	38.5	382
Lanark	0	0.0	0	0.0	0	0.0	1	2.5	39	97.5	40
Leeds	0	0.0	0	0.0	0	0.0	0	0.0	32	100.0	32
Lennox & Add.	0	0.0	4	4.4	26	28.6	21	23.1	40	44.0	91
Manitoulin	27	49.1	21	38.2	2	3.6	1	1.8	4	7.3	55
Middlesex	0	0.0	0	0.0	0	0.0	0	0.0	1	100.0	1
Muskoka	11	3.4	91	28.5	193	60.5	13	4.1	11	3.4	319
Nipissing	23	2.8	187	22.5	506	60.8	103	12.4	13	1.6	832
Northumberland	0	0.0	0	0.0	0	0.0	0	0.0	1	100.0	1
Parry Sound	18	4.2	147	34.1	233	54.1	27	6.3	6	1.4	431
Peel	0	0.0	0	0.0	0	0.0	0	0.0	1	100.0	1
Peterborough	0	0.0	3	4.5	10	14.9	9	13.4	45	67.2	67
Prince Edward	0	0.0	0	0.0	0	0.0	0	0.0	9	100.0	9
Rainy River	0	0.0	15	5.4	159	57.4	67	24.2	36	13.0	277
Renfrew	2	0.5	29	8.0	174	47.8	98	26.9	61	16.8	364
Simcoe	0	0.0	0	0.0	8	38.1	2	9.5	11	52.4	21
Stormont	0	0.0	0	0.0	0	0.0	0	0.0	1	100.0	1
Sudbury	154	18.4	155	18.5	217	25.9	123	14.7	188	22.5	837
Thunder Bay	2	0.3	29	4.1	166	23.2	218	30.5	299	41.9	714
Timiskamg	20	9.5	20	9.5	45	21.3	66	31.3	60	28.4	211
Victoria	0	0.0	1	2.6	22	56.4	2	5.1	14	35.9	39
York	0	0.0	0	0.0	0	0.0	0	0.0	2	100.0	2
	336	5.0	1002	14.9	2539	37.8	1185	17.6	1660	24.7	6722

图 5: Acidity of Ontario Lakes

2.2 表格的整理、呈现和分析（例二）

2.2.1 表格的整理

目的：以 18 年销售数据为样本，分析 19 年销售数据是否符合 18 年规律。

表 1: 区域销售数据（所有数据以十万计）

区域\时间	18Q1	18Q2	18Q3	18Q4	19Q1	19Q2	19Q3	19Q4
华北	97.62	92.24	100.90	90.39	95.69	94.99	91.13	97.81
华西	48.29	42.31	49.98	39.09	46.38	49.74	41.74	37.39
华南	75.23	75.61	100.11	74.23	74.23	76.97	71.66	76.47
华东	49.69	57.21	80.19	51.09	52.88	49.41	59.32	52.66

提取 18 年数据：

表 2: 18 年区域销售数据（所有数据以十万计）

区域\时间	18Q1	18Q2	18Q3	18Q4
华北	97.62	92.24	100.90	90.39
华西	48.29	42.31	49.98	39.09
华南	75.23	75.61	100.11	74.23
华东	49.69	57.21	80.19	51.09

对 18 年数据进行小数点处理:

表 3: 18 年区域销售数据（所有数据以十万人民币计）

区域\时间	18Q1	18Q2	18Q3	18Q4
华北	98	92	101	90
华西	48	42	50	39
华南	75	76	100	74
华东	50	57	80	51

注：若销售数据不以十万计，则表述方法为最大的相似点再加上不同点即可。如：在脚注中添加相似点再在表格中呈现不同点。

对 18 年数据进行统计:

表 4: 18 年区域销售数据（所有数据以十万计）

区域\时间	18Q1	18Q2	18Q3	18Q4	区域总额
华北	98	92	101	90	381
华西	48	42	50	39	179
华南	75	76	100	74	324
华东	50	57	80	51	238
季度总额	271	266	331	254	1122

销售总额对比不具有实际意义，故做平均值处理:

表 5: 18 年区域销售数据（所有数据以十万计）

区域\时间	18Q1	18Q2	18Q3	18Q4	区域平均销售额
华北	98	92	101	90	95
华西	48	42	50	39	45
华南	75	76	100	74	81
华东	50	57	80	51	60
季度平均销售额	68	67	83	64	70

对于差值并不大的数据，将其放入列中，做行列转换:

表 6: 18 年区域销售数据（所有数据以十万计）

时间\区域	华北	华西	华南	华东	季度平均销售额
18Q1	98	48	75	50	68
18Q2	92	42	75	57	67
18Q3	101	50	100	80	83
18Q4	90	39	74	51	64
区域平均销售额	95	45	81	60	70

对列进行升序排列:

表 7: 18 年区域销售数据（所有数据以十万计）

时间\区域	华西	华东	华南	华北	季度平均销售额
18Q1	48	50	75	98	68
18Q2	42	57	75	92	67
18Q3	50	80	100	101	83
18Q4	39	51	74	90	64
区域平均销售额	45	60	81	95	70

对行进行降序排列:

表 8: 18 年区域销售数据（所有数据以十万计）

时间\区域	华西	华东	华南	华北	季度平均销售额
18Q3	50	80	100	101	83
18Q1	48	50	75	98	68
18Q2	42	57	75	92	67
18Q4	39	51	74	90	64
区域平均销售额	45	60	81	95	70

数据整理前后对比：

表 9: 18 年区域销售数据（所有数据以十万计）

区域\时间	18Q1	18Q2	18Q3	18Q4
华北	97.62	92.24	100.90	90.39
华西	48.29	42.31	49.98	39.09
华南	75.23	75.61	100.11	74.23
华东	49.69	57.21	80.19	51.09

表 10: 18 年区域销售数据（所有数据以十万计）

时间\区域	华西	华东	华南	华北	季度平均销售额
18Q3	50	80	100	101	83
18Q1	48	50	75	98	68
18Q2	42	57	75	92	67
18Q4	39	51	74	90	64
区域平均销售额	45	60	81	95	70

注：尽管对表格进行了一般处理，我们还可以在华东区的销售额呈现非降序排列，以及华南有非常大的数额，呈现非常大的方差。

2.2.2 呈现思路

保留

- 保留重要的位数
- 保留简短有力的标签
- 保留数据间的间隙

减少

- 减少冗余信息
- 减少单位度量到最少

概括

- 对行列进行平均值或者中位数概括
- 对不同数据进行分割

交换行列（如有必要）

- 将变化程度最小的数据组放到列中

重新排列

- 列：从上到下降序
- 行：从左到右升序

面向受众

- 按照约定俗成的季度（春夏秋冬）或者地理位置（东南西北）排序数据

2.2.3 异常值的处理

表 11: 18 年区域销售数据（所有数据以十万计）

时间\区域	华西	华东	华南	华北	季度平均销售额
18Q3	50	**80**	**100**	101	**83**
18Q1	48	50	75	98	68
18Q2	42	57	75	92	67
18Q4	39	51	74	90	64
区域平均销售额	45	53*	75*	95	67*

∗不包含 18Q3 数据

可以发现剔除异常值之后，华东华南区表现趋于稳定。

通过星号或者颜色我们可以将异常值标注出来，同时注意到华南区在剔除异常值之后表现过于稳定。

将异常值剔除后我们得到如下表格：

表 12: 18 年区域销售平均数据（所有数据以十万计）

2018	华西	华东	华南	华北
区域平均销售额	45	53*	75*	95

∗不包含 18Q3 数据

将剔除异常值的平均值带回原表:

表 13: 18 年区域销售差额表（所有数据以十万计）

时间\区域	华西	华东	华南	华北	季度平均销售额
18Q3	5	**27**	**25**	6	6^*
18Q1	3	-3	0	3	1
18Q2	-3	4	0	-3	0
18Q4	-6	-2	-1	-5	-4
区域平均销售额	0	0^*	0^*	0	0^*

$*$不包含 18Q3 数据

计算绝对平均偏离程度:

2018	华西	华东	华南	华北	平均值
区平均偏离	4	3^*	0^*	4	3

$*$不包含 18Q3 数据

可以发现 18 年的绝对平均偏离程度在剔除异常值后为三十万。

2.2.4 分析

原始销售数据如下：

表 15: 19 年区域销售数据（所有数据以十万计）

区域\时间	19Q1	19Q2	19Q3	19Q4
华北	95.69	94.99	91.13	97.81
华西	46.38	49.74	41.74	37.39
华南	74.23	76.97	71.66	76.47
华东	52.88	49.41	59.32	52.66

对上述销售数据处理后得到如下表格：

表 16: 19 年区域销售数据（所有数据以十万计）

时间\区域	华西	华东	华南	华北	季度平均销售额
19Q1	46	53	74	96	67
19Q2	50	49	77	94	68
19Q3	42	59	72	91	66
19Q4	37	53	76	98	66
区域平均销售额	44	54	75	95	67

将 18 年数据为本与 19 年数据进行比较

表 17: 18 年区域销售差额表（所有数据以十万计）

时间\区域	华西	华东	华南	华北	季度平均销售额
18Q1	3	-3	0	3	1
18Q2	-3	4	0	-3	0
18Q3	5	**27**	**25**	6	6*
18Q4	-6	-2	-1	-5	-4
区域平均销售额	0	0*	0*	0	0*

*不包含 18Q3 数据

表 18: 19 年区域销售与 18 年区销售数据对比表（所有数据以十万计）

时间\区域	华西	华东	华南	华北	季度平均销售额
19Q1	0	0	-1	1	0
19Q2	5	-4	2	-1	0
19Q3	-3	6	-3	-4	-1
19Q4	-8	0	1	3	-1
区域平均销售额	-1	1	0	0	0

结论：我们可以发现 18 年数据规律可以套用在 19 年上，建立的模型比较可靠但也说明了公司业务基本没有增长。

3 一维数据

3.1 有序数组

3.1.1 茎叶图

利用 aplpack 包进行茎叶图绘制

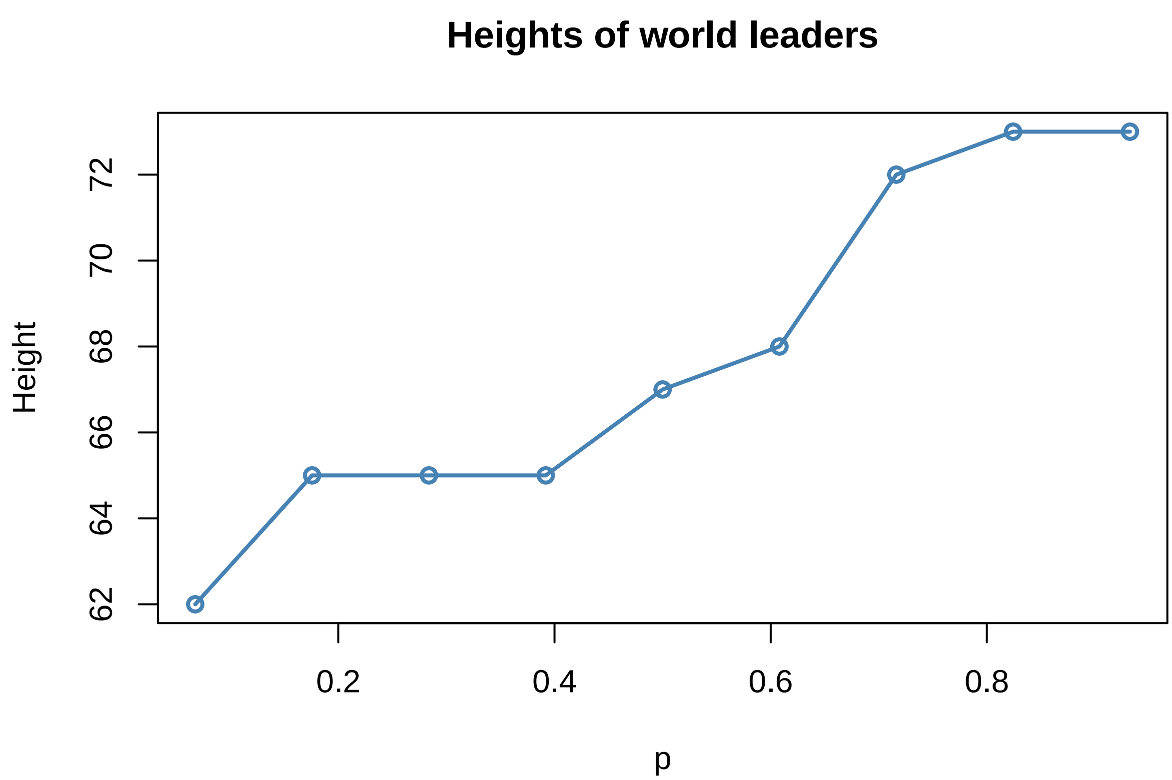
```
library(aplpack)
stem.leaf(iris$Sepal.Length)
```

```
## 1 | 2: represents 1.2
##  leaf unit: 0.1
##            n: 150
##     1     t | 3
##     5     f | 4445
##    11     s | 666677
##    22    4. | 88888999999
##    41    5* | 0000000000111111111
##    46     t | 22223
##    59     f | 4444445555555
##    73     s | 66666677777777
##   (10)   5. | 8888888999
##    67    6* | 000000111111
##    55     t | 2222333333333
##    42     f | 444444455555
##    30     s | 6677777777
##    20    6. | 8889999
##    13    7* | 01
##    11     t | 2223
##     7     f | 4
##     6     s | 67777
##     1    7. | 9
```

```
# 总共有 150 个样本，中位数出现在 78（最左侧括号处）
# 样本涵盖范围 4.3 到 7.9
```

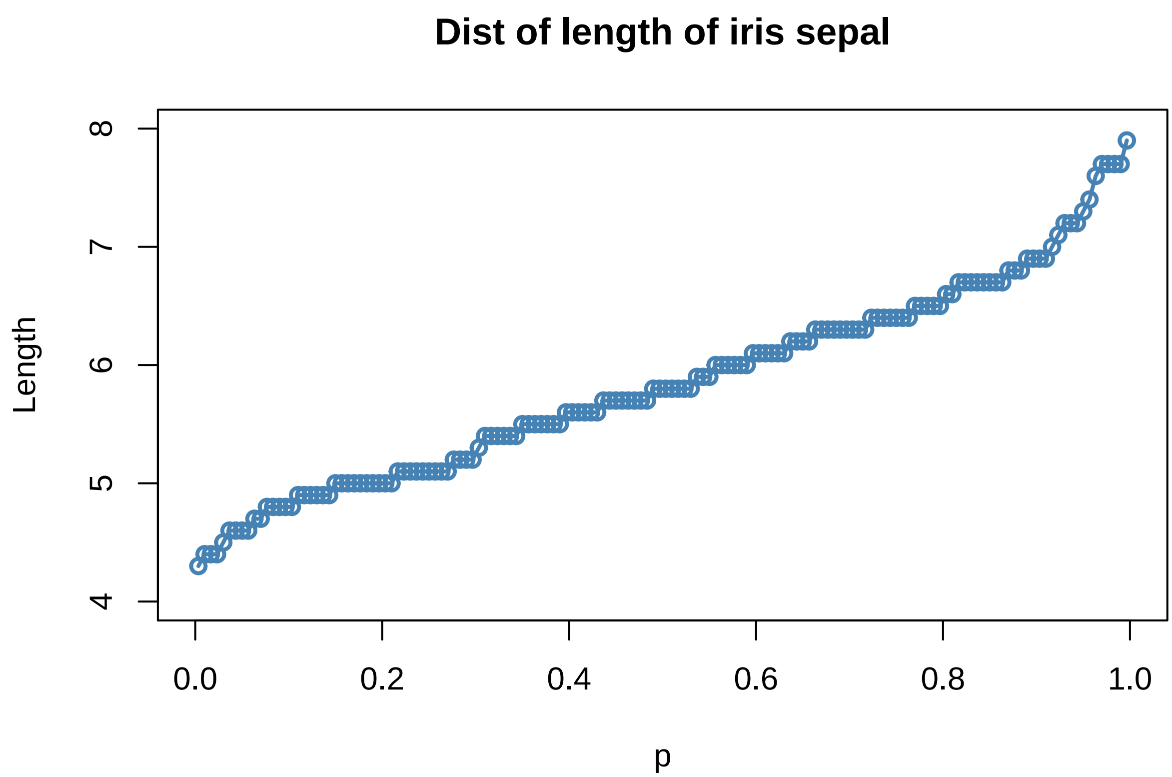
3.1.2 折线图

```
# 长度为 9 的数组为例子
# 所有身高以英尺为单位
wordLeaders <- c(62,65,65,65,67,68,72,73,73) # 输入数组
n <- length(wordLeaders) # 获取数组长度
p <- ppoints(n) # 计算百分位
plot(x = p, y = wordLeaders,
     type = "o", lwd = 2, col = "steelblue",
     xlab = "p",  # 百分比
     ylab = "Height",  # 身高
     main = "Heights of world leaders") # 世界领导人身高分布
```




```
# 长度为 150 的单位数组为例子
q <- sort(iris$Sepal.Length, decreasing = FALSE)
n <- length(q)
p <- ppoints(n) # 以数组长度计算
plot(x = p, y = q,
```

```
    type = "o", # 设定绘制图形种类
    lwd = 2, # 折线宽度
    col = "steelblue", # 绘制颜色
    xlab = "p", # 横坐标名（百分比）
    ylab = "Length", # 纵坐标名（花萼长度）
    xlim = c(0,1), # 横坐标范围
    ylim = c(4,8), # 纵坐标范围
    main = "Dist of length of iris sepal" # 图名（鸢尾花花萼长度分布）
    )
```

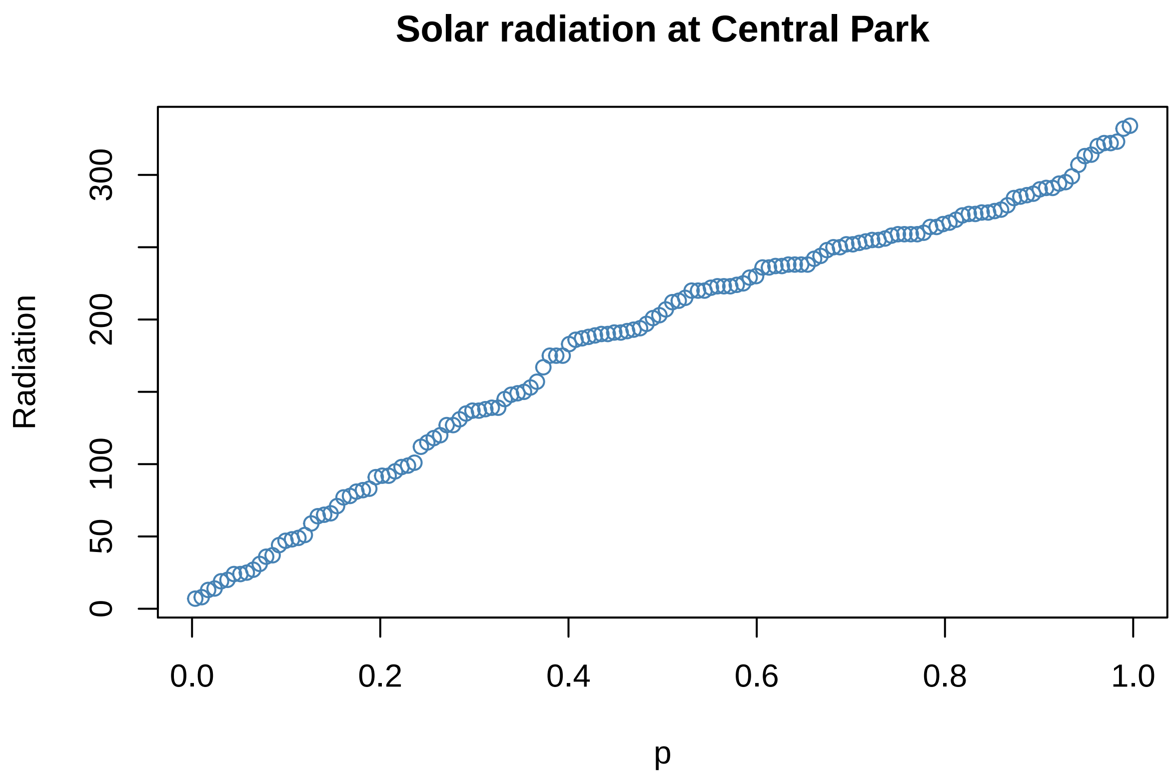


如此类图可以很轻易的得出百分位取值，如大约 75% 的取值大于 5。($x = 0.25, y \approx 5$)

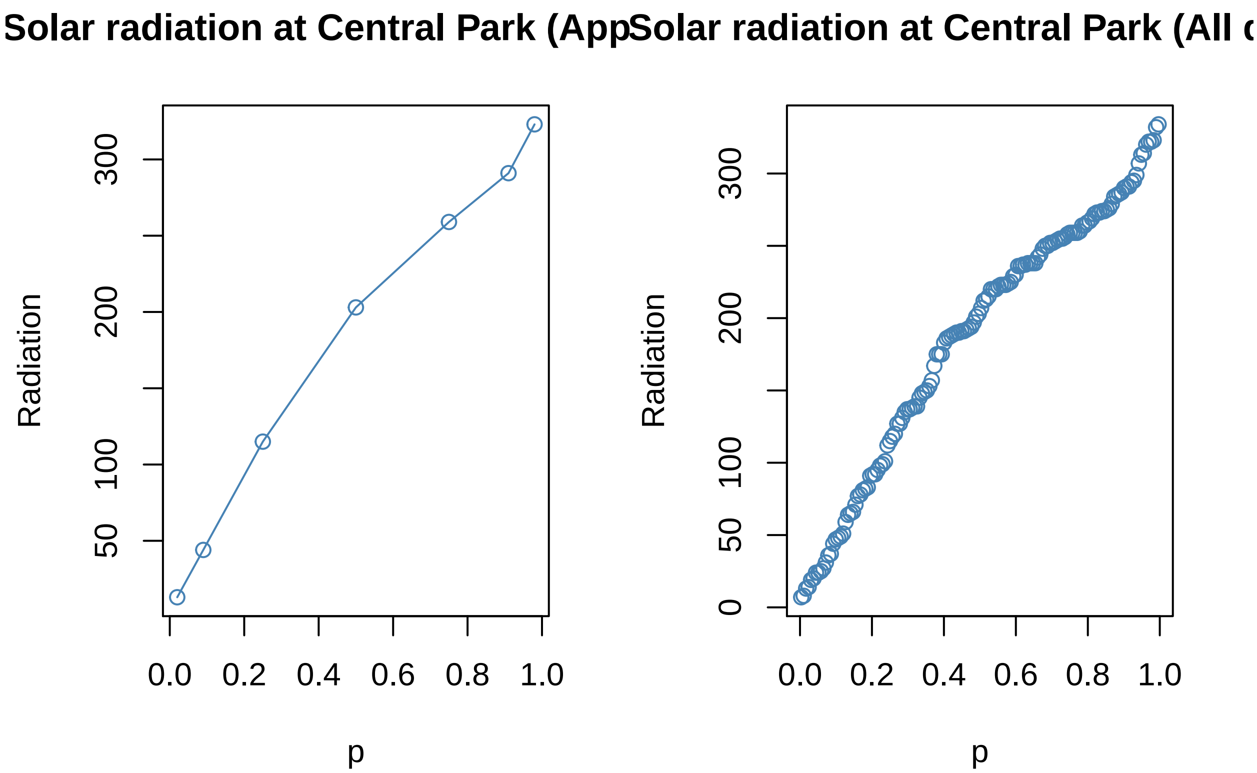
- 中位数出现在 $x = 0.5$

```
qvals <- sort(airquality$Solar.R) # 对 airquality 表格中的 Solar.R 列进行排序并赋值
n <- length(qvals)
pvals <- ppoints(n)
plot(x= pvals, y= qvals, col = "steelblue",
    xlab ="p",
```

```
     ylab = "Radiation", # 纵坐标（辐射值）
     main = "Solar radiation at Central Park") # 中央公园辐射分布
```




```
# 利用少量数据点进行图形拟合
p = c(0.02,0.09,0.25,0.50,0.75,0.91,0.98)
q = c(qvals[3],qvals[14],qvals[37],qvals[73],qvals[110],qvals[133],qvals[144])
par(mfrow = c(1,2)) # 同一区域绘制多个图形
plot(x= p, y= q, type = "o",
     col = "steelblue",
     xlab ="p",
     ylab = "Radiation",
     main = "Solar radiation at Central Park (Approx.)") # 中央公园辐射分布（拟合）
plot(x= pvals, y= qvals, col = "steelblue",
     xlab ="p",
     ylab = "Radiation",
     main = "Solar radiation at Central Park (All data)") # 中央公园辐射分布（所有数据）
```

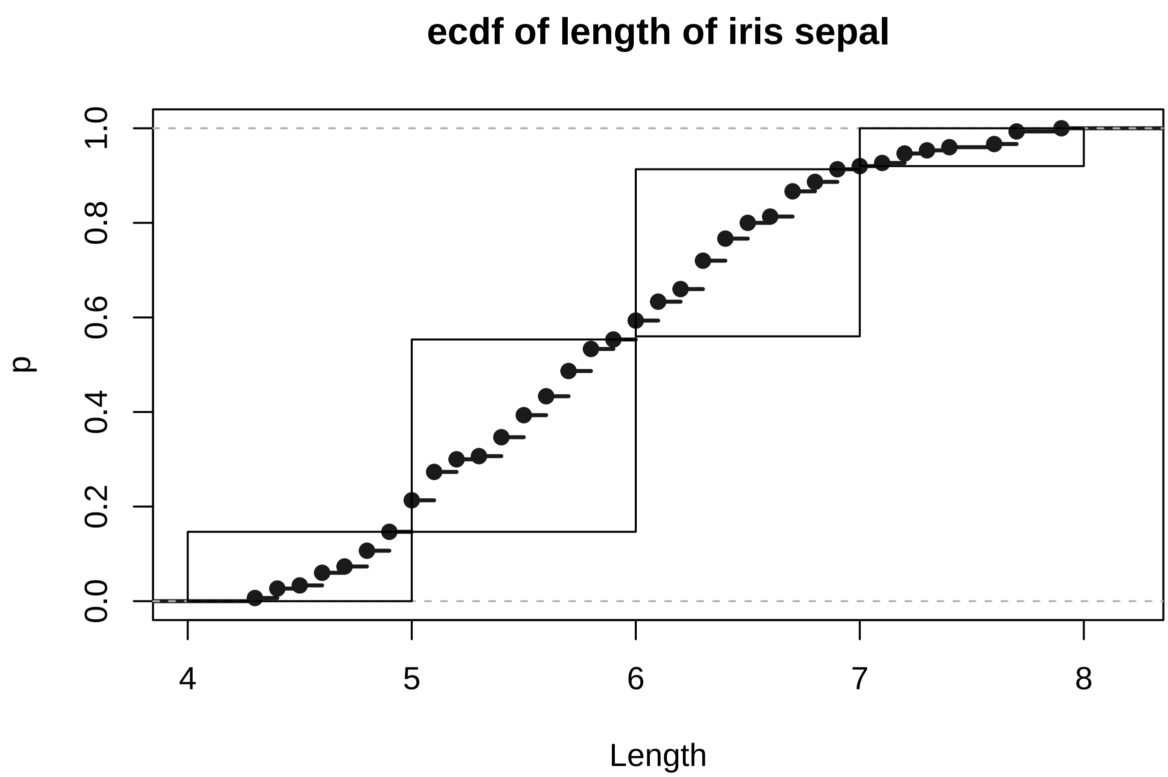


3.1.3 直方图

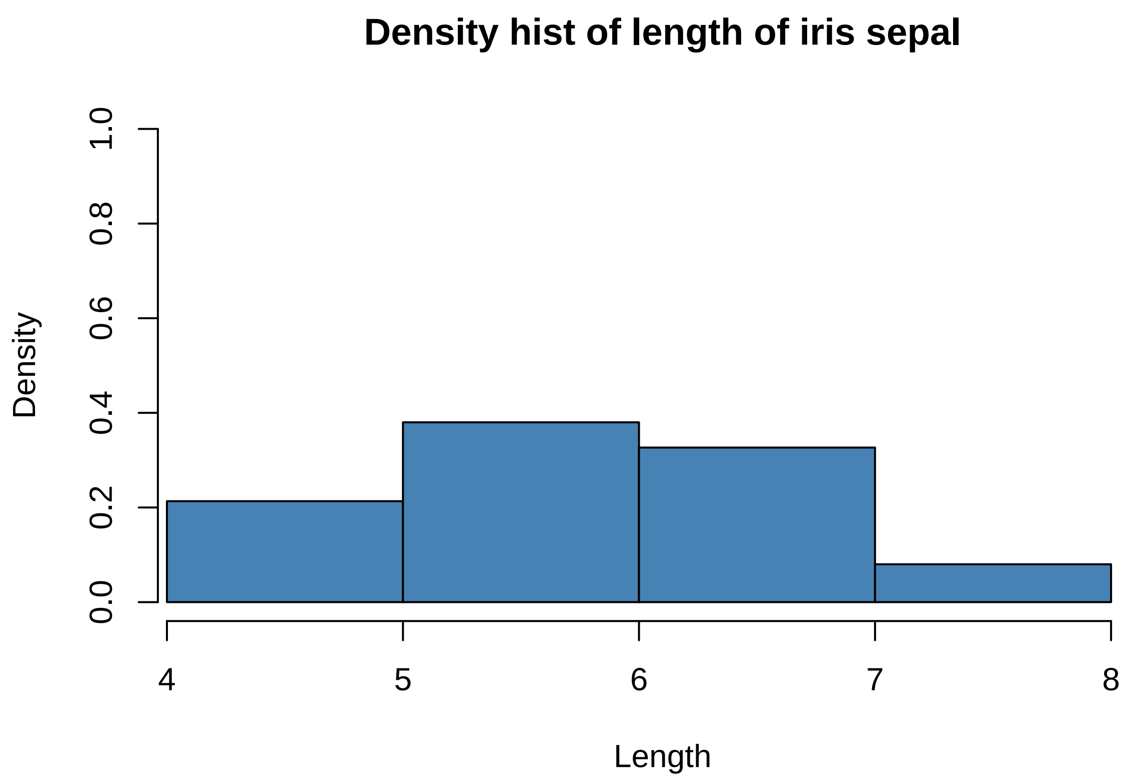
3.1.3.1 密度直方图 绘制数据在总体中占比的直方图。

```
fhat <- ecdf(iris$Sepal.Length)
plot(fhat, lwd = 2,
     col = "grey10", # 颜色
     xlab = "Length",  # 长度
     ylab = "p", # 百分比
     main = "ecdf of length of iris sepal") # 鸢尾花花萼长度经验分布函数

drawbox <- function(x,y) {
xvals <- c(x,rev(x),x[1])
yvals <- c(rep(y, each=2),y[1])
lines(xvals, yvals)
}

drawbox(c(4,5),c(0,22/150))
drawbox(c(5,6),c(22/150,83/150))
```

```
drawbox(c(6,7),c(84/150,137/150))
drawbox(c(7,8),c(138/150,1))
```

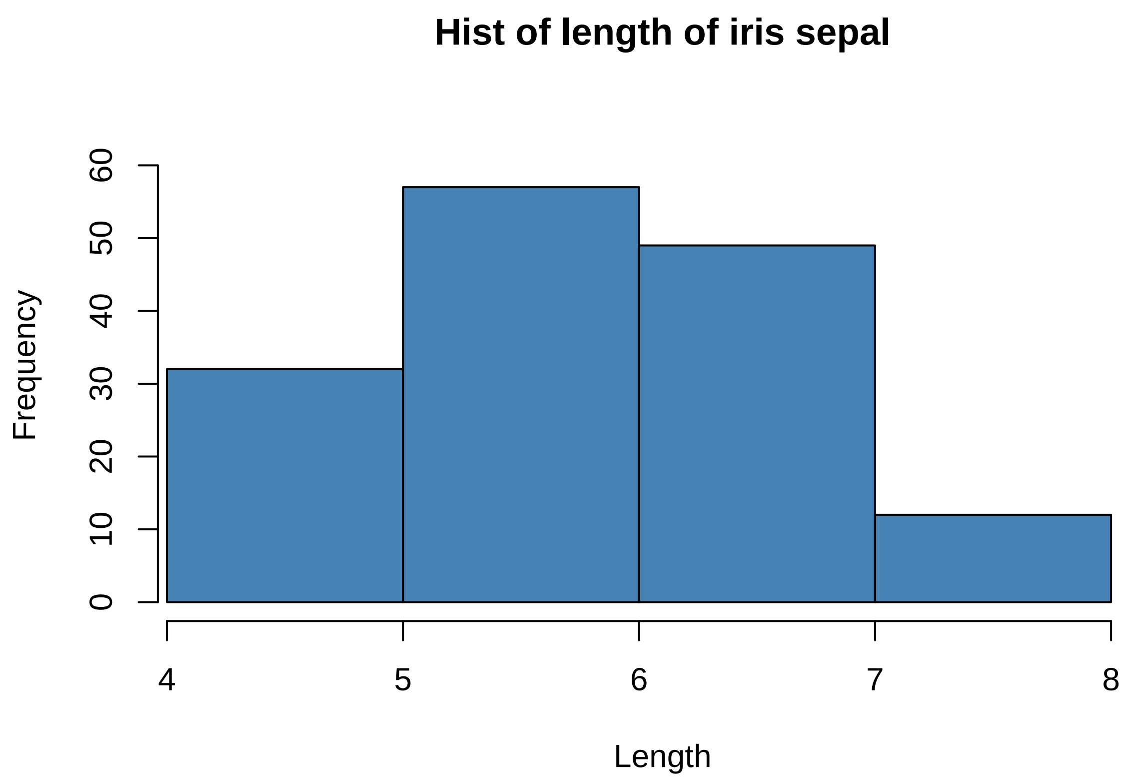



```
hist(iris$Sepal.Length, col = "steelblue",
     probability = TRUE, # 呈现密度图
     breaks = 4, # 改变直方个数
     xlim = extendrange(iris$Sepal.Length), # 略微演唱 x 轴长度
     ylim = c(0,1),
     xlab = "Length", # 花萼长度
     ylab = "Density", # 密度
     main = "Density hist of length of iris sepal" # 鸢尾花花萼长度密度直方图
     )
```



3.1.3.2 频率直方图 绘制数据在总体中出现频率的直方图。

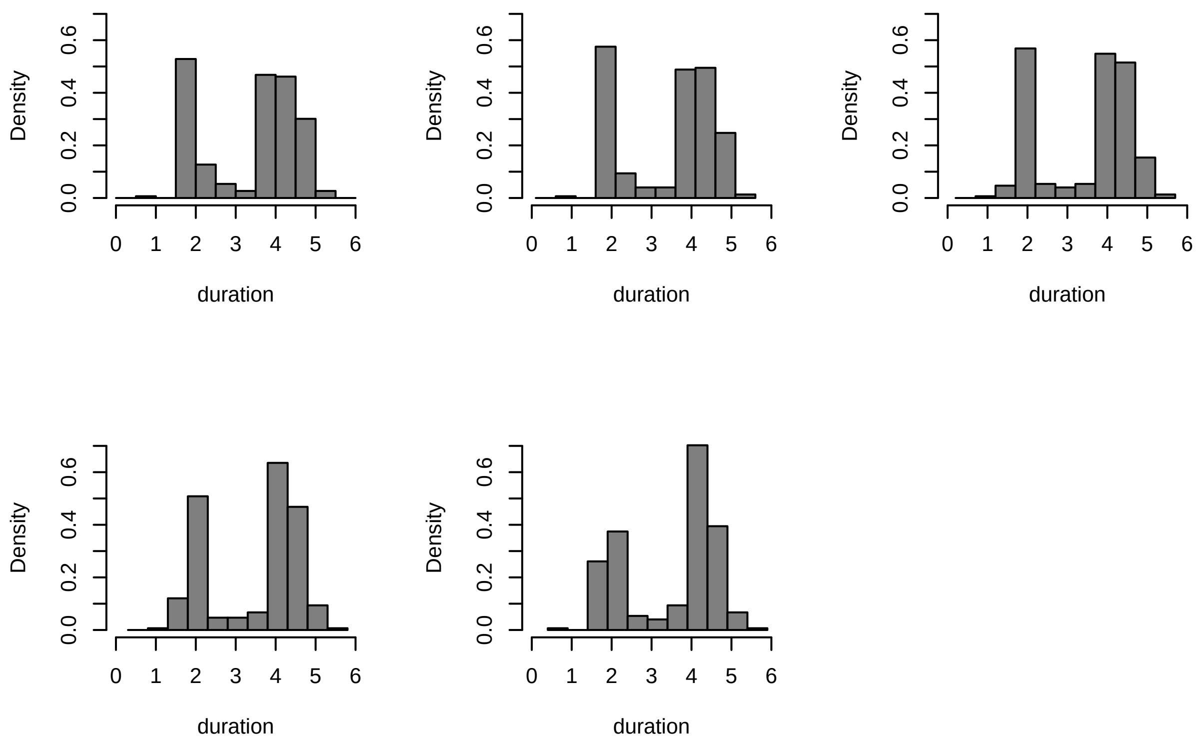
```
hist(iris$Sepal.Length, col = "steelblue",
     # probability = FALSE by default
     breaks = 4,
     xlim = extendrange(iris$Sepal.Length),
     ylim = c(0,65),
     xlab = "Length", # 花萼长度
     ylab = "Frequency", # 频率
     main = "Hist of length of iris sepal" # 鸢尾花花萼长度频率直方图
     )
```



3.1.3.3 Average shifted Histgram 对直方图提供平滑处理，是一种提供更好视觉效果和精度的呈现方法。

```
data(geyser, package="MASS") # 从 Mass 包中导入 geyser 数据
# head(geyser, n = 5) # 查看 geyser 头部五个数据
plot_colour <- "grey50" # 设定面板颜色
savePar <- par(mfrow=c(2,3)) # 设定作图行列（2 行 3 列）
with(geyser,
     for (start in seq(0, 0.4, 0.1)) # 0.0 0.1 0.2 0.3 0.4
       {hist(duration,
             # 从 start 开始到 6 结束，每 0.5 一个单位作一个 bin
             breaks=seq(start, 6.0, 0.5),
             col=plot_colour,
             probability=TRUE,
             xlim=c(0, 6),
             ylim=c(0, 0.7),
             main="",
             xlab="duration"
```

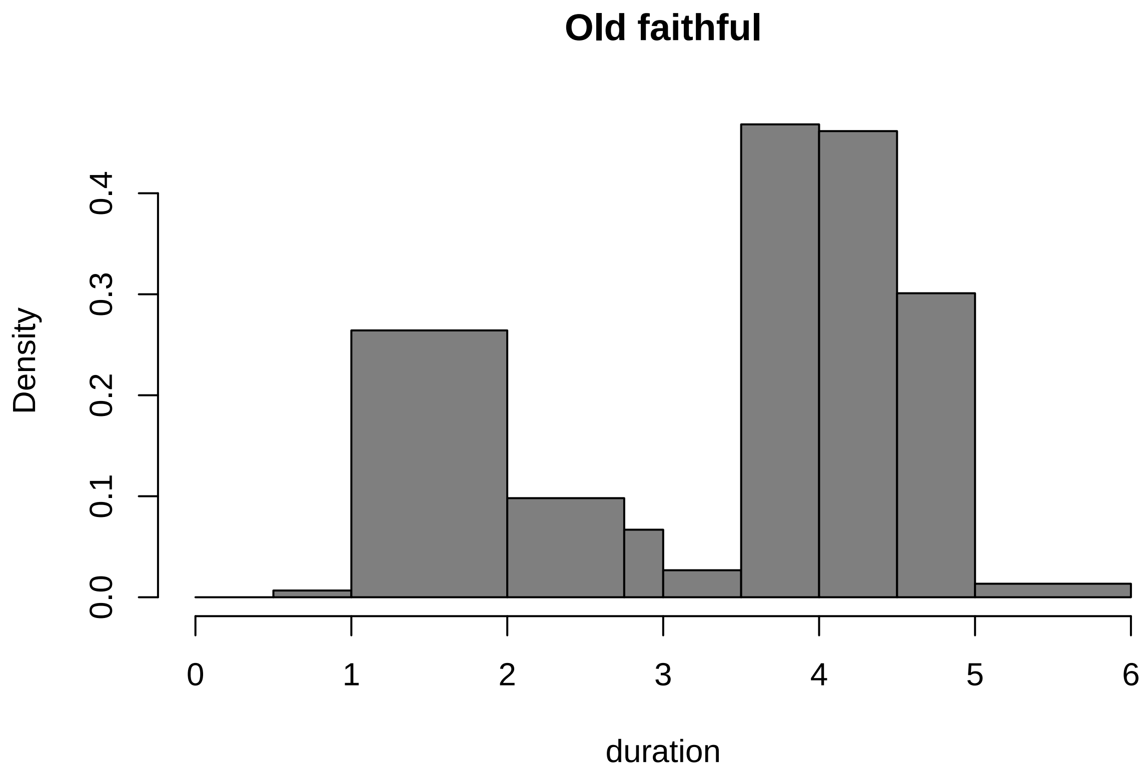
```
              )
       }
)
par(savePar) # 结束作图设定（2 行 3 列）
```



由此可见起始点不同导致的直方图效果差异很大。

另一例子为设定不同间距的直方图：

```
with(geyser,
     hist(duration, col=plot_colour,
          xlim=c(0,6),
          breaks=c(0,0.5,1.0,2,2.75,3,3.5,4,4.5,5,6.0), # 不同间距的直方图实现
          main="Old faithful",
          xlab="duration")
)
```



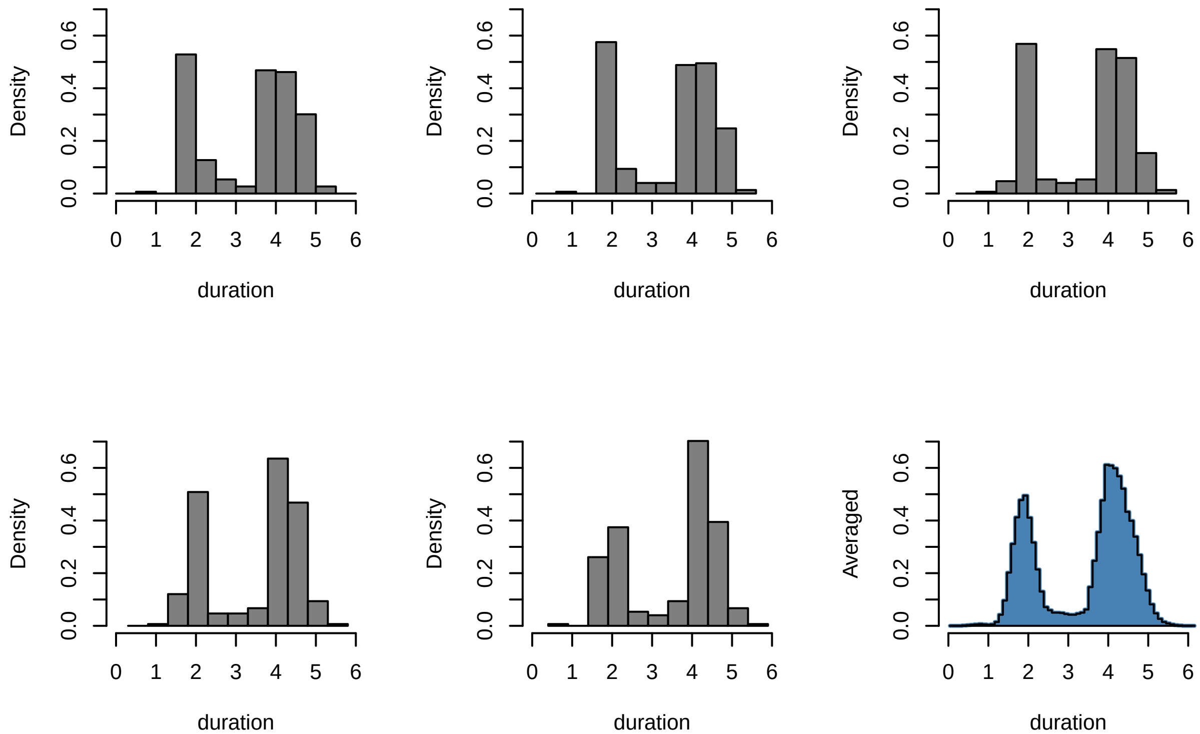
呈现效果又是不同。

```
# ash 的实现方程
ash <- function(x, n_hists = 5, binwidth = NULL, # 对最开始五个直方图的高度进行平均
                type="l",
                xlab="x",
                main ="Average shifted Histogram",
                ylab="Averaged",
                lwd=2, lty=1, xlim, ylim,
                col="steelblue", fill, ...) {
  xrange <- round(extendrange(x), 1)
  if (is.null(binwidth)) binwidth <- diff(xrange) / 10
  delta <-binwidth / n_hists
  xmin <- min(xrange) - binwidth
  xmax <- max(xrange) + binwidth
  breaks <- seq(xmin, xmax - binwidth, binwidth)
  n_breaks <- length(breaks)
  xvals <- seq(xmin, xmax-delta, delta)
  n_xvals <- length(xvals)
```

```
  density <- numeric(n_xvals)
  x_indices <- seq(1, n_xvals - n_hists)
  for (i in seq(0, n_hists - 1)) {
    cur_hist <- hist(x, breaks=breaks + delta*i,plot=FALSE)
    density[i + x_indices] <- density[i + x_indices] +
      rep(cur_hist$density, rep(n_hists, n_breaks-1))
  }
  density <- density/n_hists
  if (missing(xlim)) xlim <- xrange
  if (missing(ylim)) ylim <- extendrange(density)
  if (missing(fill)) fill <- col
  plot(rep(xvals, each=2) + rep(c(-delta/2, delta/2), length(xvals)),
       rep(density, each=2),type=type,
       xlim=xlim, ylim=ylim, xlab=xlab, ylab=ylab, main=main,
       lwd=lwd, lty=lty, col=col,
       bty="n", ...)
  polygon(rep(xvals, each=2) + rep(c(-delta/2, delta/2), length(xvals)),
          rep(density, each=2), col=fill)
}
```

```
data(geyser, package="MASS") # 从 Mass 包中导入 geyser 数据
# head(geyser, n = 5) # 查看 geyser 头部五个数据
plot_colour <- "grey50" # 设定面板颜色
savePar <- par(mfrow=c(2,3)) # 设定作图行列（2 行 3 列）
# 从 geyser 中导入 duration 数据，并开始 loop 作图
with(geyser,
     { for (start in seq(0, 0.4, 0.1)) # 0.0 0.1 0.2 0.3 0.4
       {hist(duration,
             breaks=seq(start, 6.0, 0.5), # 从 start 开始到 6 结束，每 0.5 一个单位作一个 bin
             col=plot_colour,
             probability=TRUE,
             xlim=c(0, 6),
             ylim=c(0, 0.7),
             main="",
             xlab="duration"
             )
     }
       # 开始绘制 ASH
       ash(duration, main = "", xlab = "duration",
```

```
        xlim = c(0,6), ylim = c(0,0.7))
    }
)
```



如此可得 ASH。

3.1.3.4 核密度图

3.2 数据对比

3.2.1 视觉对比

3.2.1.1 表格对比 以泰坦尼克号为例:

读取舱位及存活率用表格展现。

```
library(knitr)
classTable <- apply(Titanic, MARGIN = c(4,1), FUN = sum) # 对四行一列的数据累加
kable(classTable)
```

	1st	2nd	3rd	Crew
No	122	167	528	673
Yes	203	118	178	212

```
classTotals <- apply(classTable, MARGIN = 2, FUN = sum) # 获得舱位总人数
classSurvival <- t(classTable["Yes", ] /classTotals) # 获得存活率
rownames(classSurvival) <- c("Survived") # 重命名行名
# classSurvival <- round(classSurvival, 3) # 保留三位小数点
kable(classSurvival) # knitr 中的 kable 表格展示每个舱位存活率
```

	1st	2nd	3rd	Crew
Survived	0.6246154	0.4140351	0.2521246	0.239548

另一种展现方法:

```
newTable <- 100 * round(classSurvival, 2) # 更改列中内容
newTable <- t(newTable) # 转秩矩阵
descendingOrder <- order(newTable, decreasing = TRUE) # 降序排列，符合送上到下降序排列原则
colnames(newTable) <- c("% survived") # 更改列名
kable(newTable, caption = "Survival rates on the Titanic by class") # 加上标题
```

表 21: Survival rates on the Titanic by class

	% survived
1st	62
2nd	41

	% survived
3rd	25
Crew	24

3.2.1.2 图像对比

3.2.1.2.1 条形图

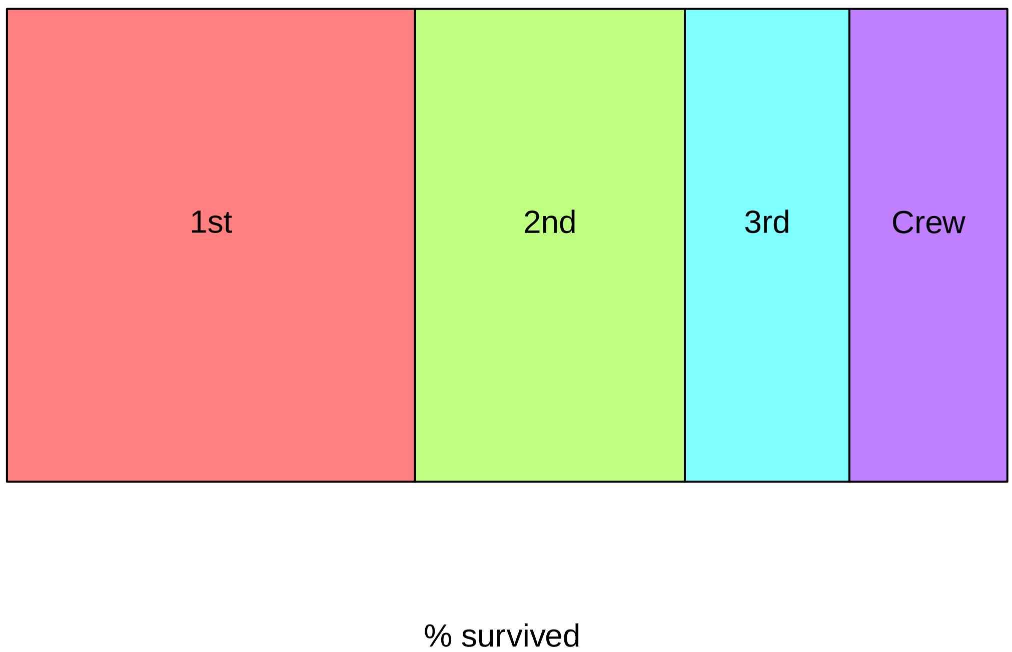
通过绘制带颜色的条形图来呈现数据大小关系。

```
kable(newTable)
```

	% survived
1st	62
2nd	41
3rd	25
Crew	24

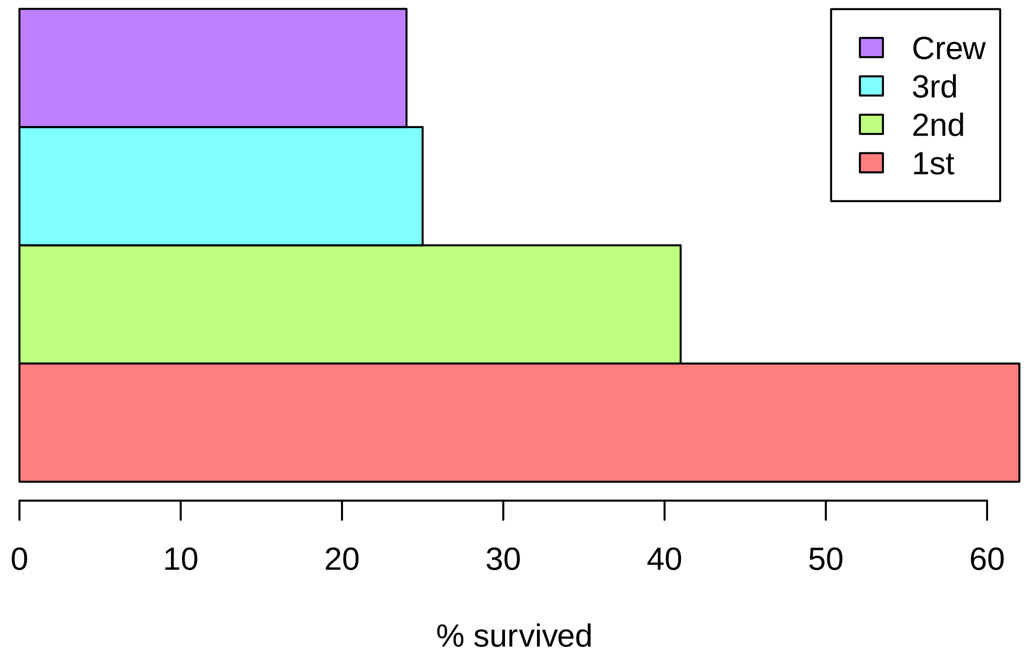
使用条形图对比不同组（舱位）内同种数据（生还率）:

```
nvals <- nrow(newTable) # newTable 的行数: 4
cols <- rainbow(nvals, alpha = 0.5) # 采用 rainbow 调色，并设定透明度为 0.5
barplot(newTable, col = cols, horiz = TRUE, axes = FALSE, # 水平方向绘制条形图并赋名
        names.arg = c(""), xlab = colnames(newTable))
xlocs <- cumsum(newTable) # 计算累加生存率
centres <- c(xlocs[1]/2, xlocs[1:(nvals -1)] + diff(xlocs)/2) # 计算每个间隔中心位置
# cumsum(newTable) -> 62 103 128 152
# xlocs[1:(nvals -1)] -> 62 103 128
# diff(xlocs)/2 -> 20.5 12.5 12.0
text(centres, 0.75, labels = rownames(newTable)) # 将 rowname 施加到图中间
```



对比组内数据（生还率）绝对大小：

```
barplot(newTable, col = cols, horiz = TRUE, beside = TRUE, # 排列绘制
        names.arg = c(""),
        xlab = colnames(newTable),
        legend.text = rownames(newTable))
```

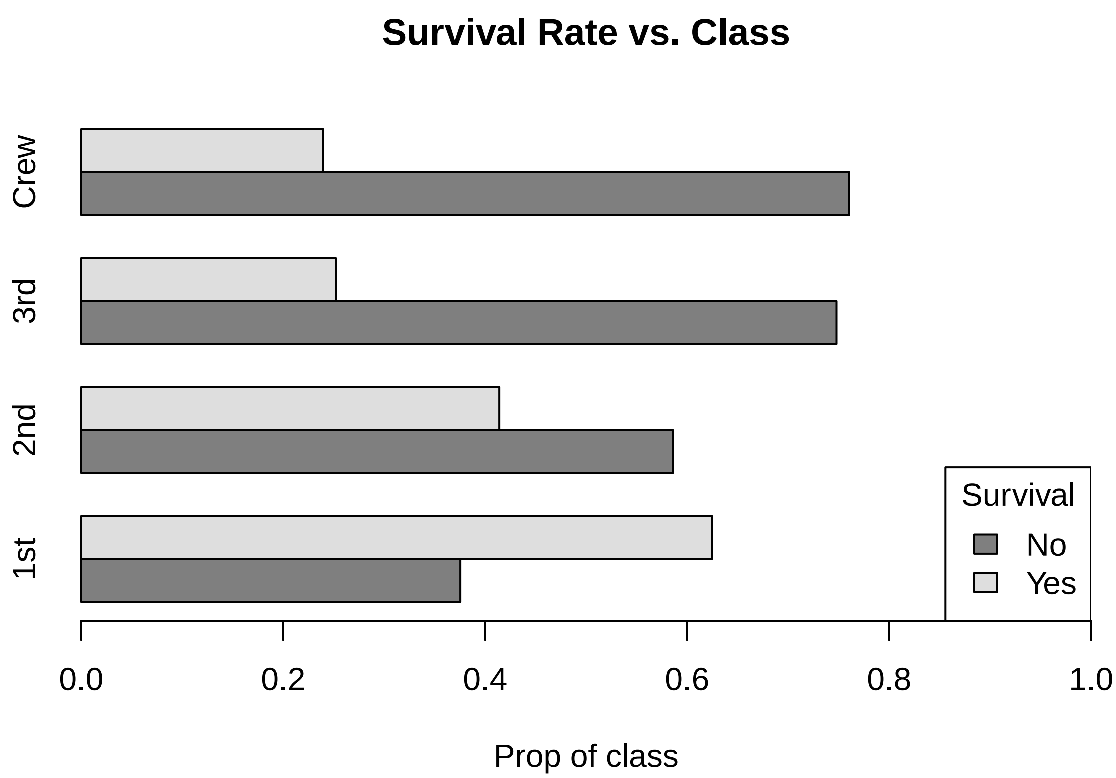


对比不同组（舱位）数据的多组数据（生还率/死亡率）绝对大小：

```
survivalProp <- classTable
survivalProp["Yes",] <- survivalProp["Yes",] /classTotals # alter 表格获得每个组的生还率
survivalProp["No",] <- survivalProp["No",] /classTotals # alter 表格获得每个组的生还率
survivalProp
```

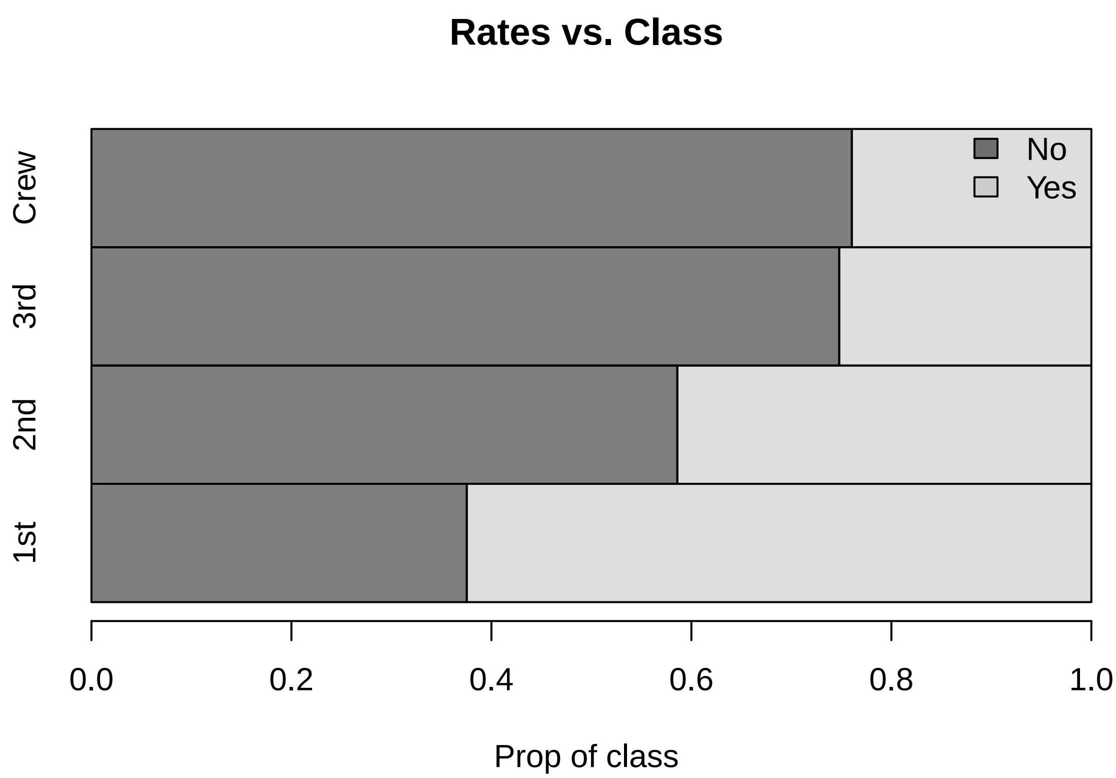
```
##          Class
## Survived       1st       2nd       3rd      Crew
##       No  0.3753846 0.5859649 0.7478754 0.760452
##       Yes 0.6246154 0.4140351 0.2521246 0.239548
```

```
survivalCols <- adjustcolor(c("black", "grey"), 0.5) # 设定组别颜色
barplot(survivalProp, col = survivalCols, # 绘制相邻条状图
        horiz = TRUE, beside = TRUE, # 水平方向, 相邻
        xlab = "Prop of class", xlim = c(0,1),
        main = "Survival Rate vs. Class")
legend("bottomright", title = "Survival", # 制作图例
       fill = survivalCols, legend = rownames(survivalProp))
```

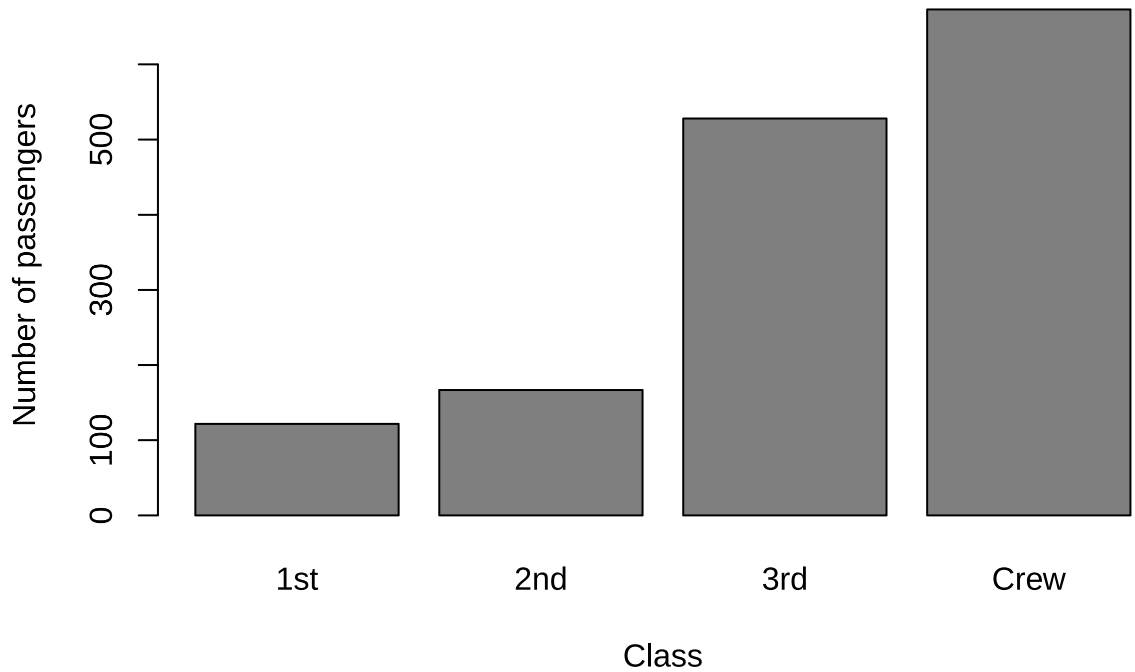


另一种对对比不同组（舱位）数据的多组数据（生还率/死亡率）绝对大小的方法:

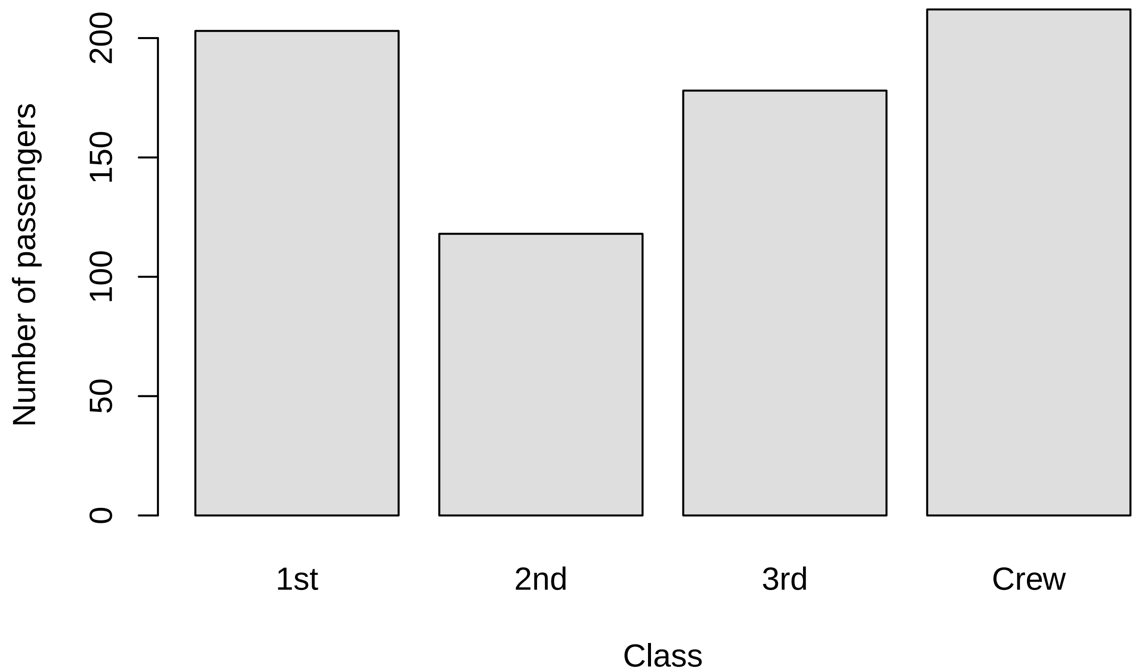
```
barplot(survivalProp, col = survivalCols,
        horiz = TRUE, beside = FALSE,
        xlab = "Prop of class", space = 0,
        main = "Rates vs. Class")
legend("topright", # 制作图例
       fill = survivalCols, # 图例分类颜色
       legend = rownames(survivalProp), # 图例名字
       bty="n") # 取消图例边框
```



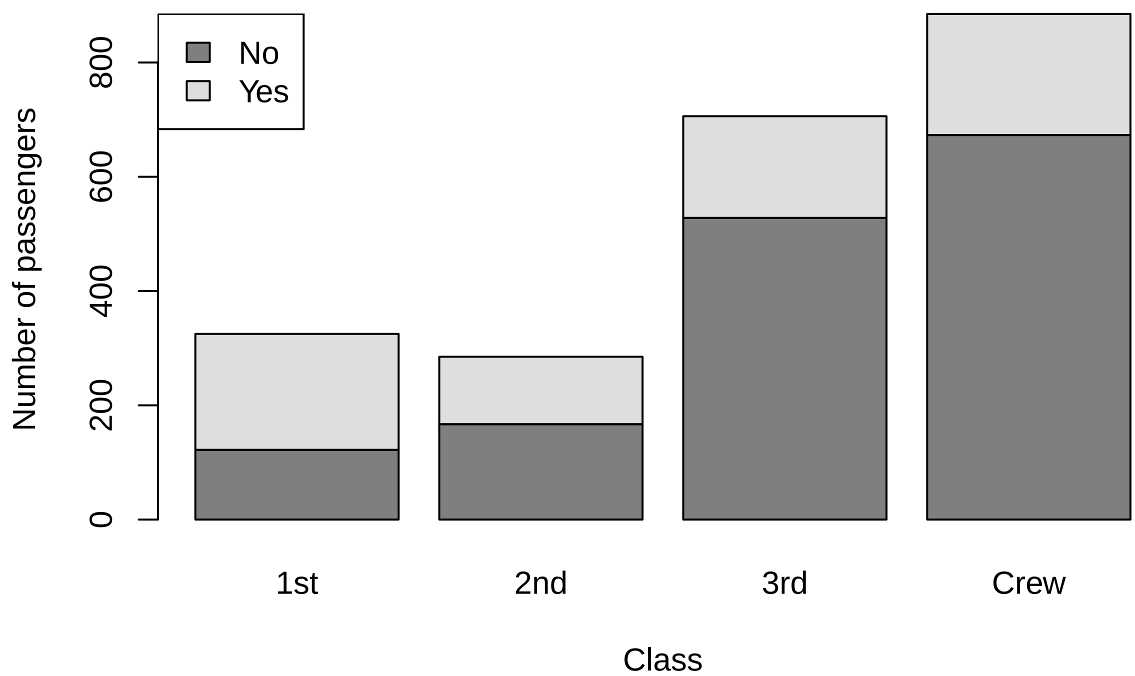

```
# 每个舱位死亡人数
barplot(classTable["No",], col = survivalCols[1], xlab="Class", ylab="Number of passengers")
```




```
# 每个舱位存活人数
barplot(classTable["Yes",], col = survivalCols[2], xlab="Class", ylab="Number of passengers")
```



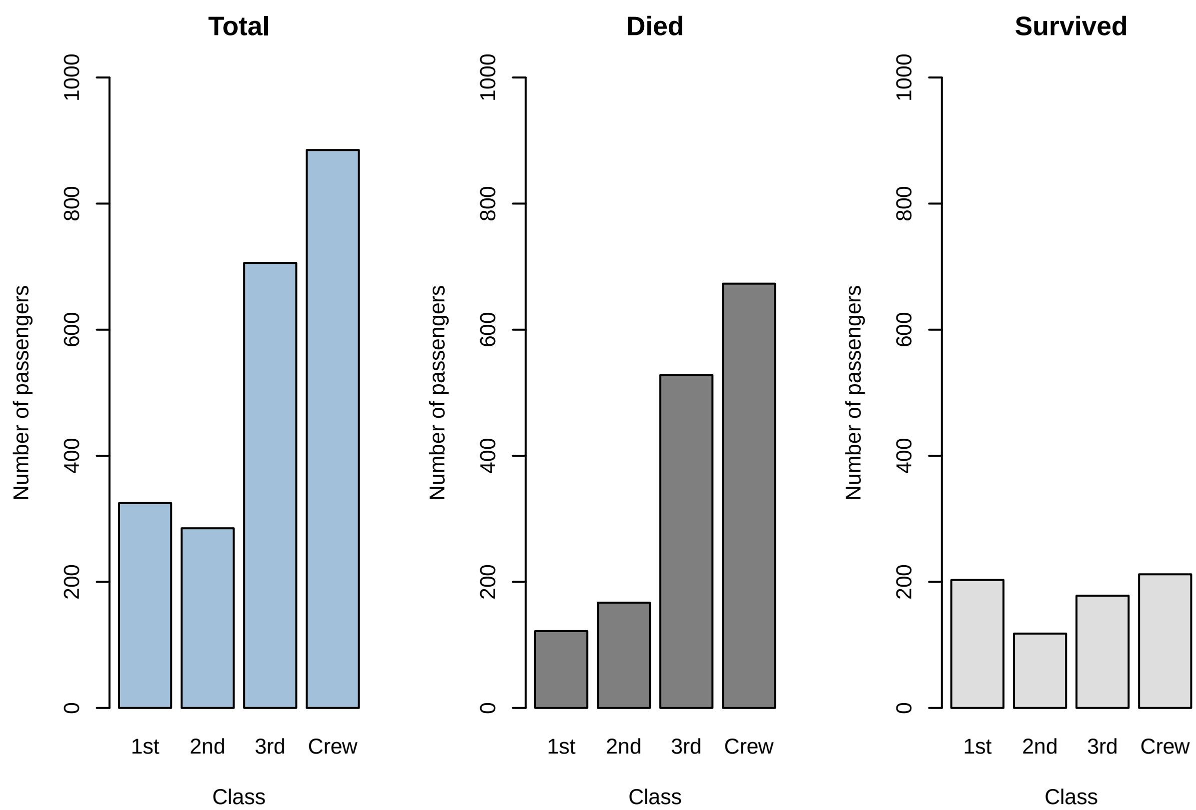

```
# 每个舱位死亡及存活人数对比
# 特别注意每个 bin 的高度为总人数
barplot(classTable, col= survivalCols, xlab="Class", ylab="Number of passengers")
legend("topleft", # 制作图例
       fill = survivalCols, legend = rownames(survivalProp))
```



多组数据相关数据集中绘图展示：

```
savePar <- par(mfrow=c(1,3)) # 一行三列绘图参数
# 三组作图代码可以单独使用
barplot(apply(classTable, MARGIN = 2, FUN = sum),
        col= adjustcolor("steelblue", 0.5),
        ylim = c(0,1000), # 确保纵坐标统一度量
        xlab="Class",
        ylab="Number of passengers",
        main = "Total")
barplot(classTable["No",], col = survivalCols[1],
        ylim = c(0,1000), # 确保纵坐标统一度量
        xlab="Class",
        ylab="Number of passengers",
        main = "Died")
barplot(classTable["Yes",], col = survivalCols[2],
        ylim = c(0,1000), # 确保纵坐标统一度量
        xlab="Class",
        ylab="Number of passengers",
```

```
      main="Survived")
```



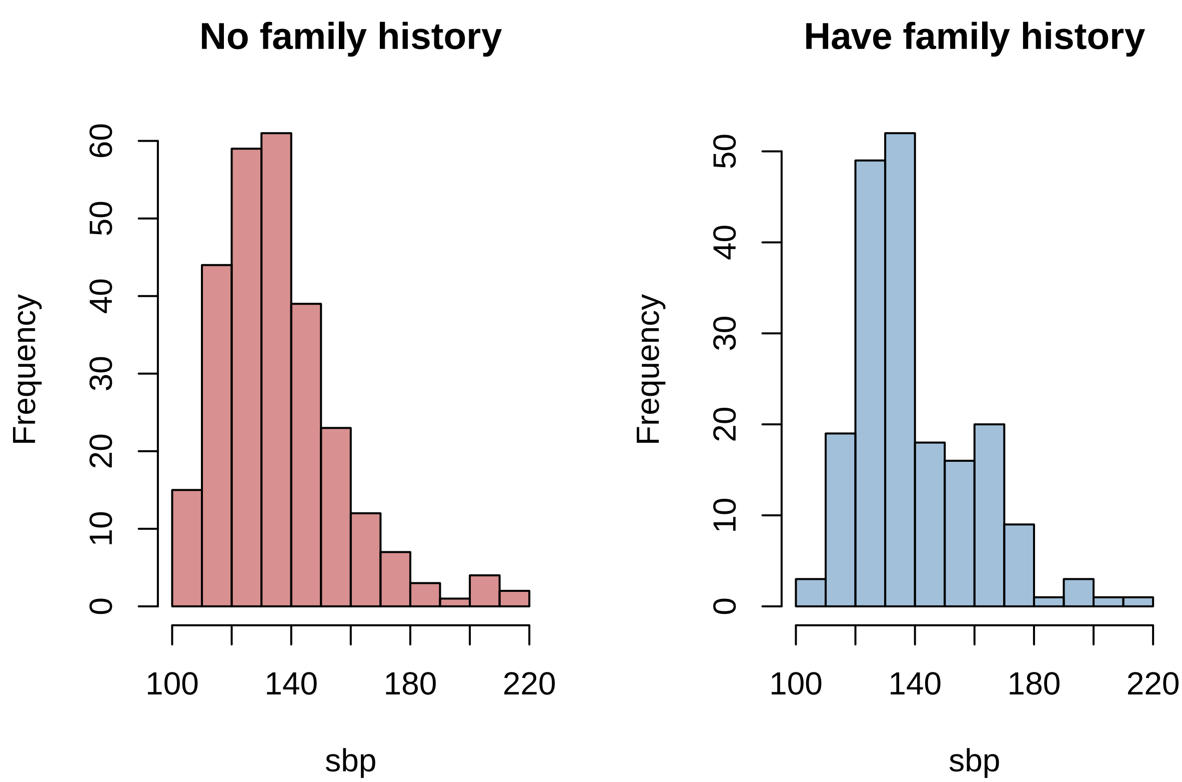

```
par(savePar) # 退出作图设定
```

3.2.1.2.2 直方图 下面将展示使用直方图对比不同组内同种数据：

```
# install.packages("loon") # 安装 loon 包
library(loon.data) # 使用 loon 包中的数据
data("SAheart", package = "loon.data") # 导入南非心脏病数据集
kable(head(SAheart)) # 检查南非心脏病头部数据
```

sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
160	12.00	5.73	23.11	Present	49	25.30	97.20	52	Yes
144	0.01	4.41	28.61	Absent	55	28.87	2.06	63	Yes
118	0.08	3.48	32.28	Present	52	29.14	3.81	46	No
170	7.50	6.41	38.03	Present	51	31.99	24.26	58	Yes
134	13.60	3.50	27.78	Present	60	25.99	57.34	49	Yes
132	6.20	6.47	36.21	Present	62	30.77	14.14	45	No

```r
# 提取 SAheart 数据集中所有 famhist 为 Absent 项目，产生一个 false/true list,
# 并储存到 noFamilyHistory（没有家族病史）
noFamilyHistory <- SAheart[, "famhist"] == "Absent"
FamilyHistory <- SAheart[, "famhist"] == "Present"
# 调整配色
famHistoryCol <- adjustcolor("steelblue", 0.5)
noHistoryCol <- adjustcolor("firebrick", 0.5)
savePar = par(mfrow=c(1,2)) # 设定作图参数
# 截取 SAheart 中所有 noFamilyHistory 为 true 的位置的 sbp 值
hist(SAheart[noFamilyHistory,"sbp"],
     col=noHistoryCol,
     xlab = "sbp",
     main="No family history")
# 截取 SAheart 中所有 FamilyHistory 为 true 的位置的 sbp 值
hist(SAheart[FamilyHistory,"sbp"],
     col=famHistoryCol,
     xlab = "sbp",
     main="Have family history")
```

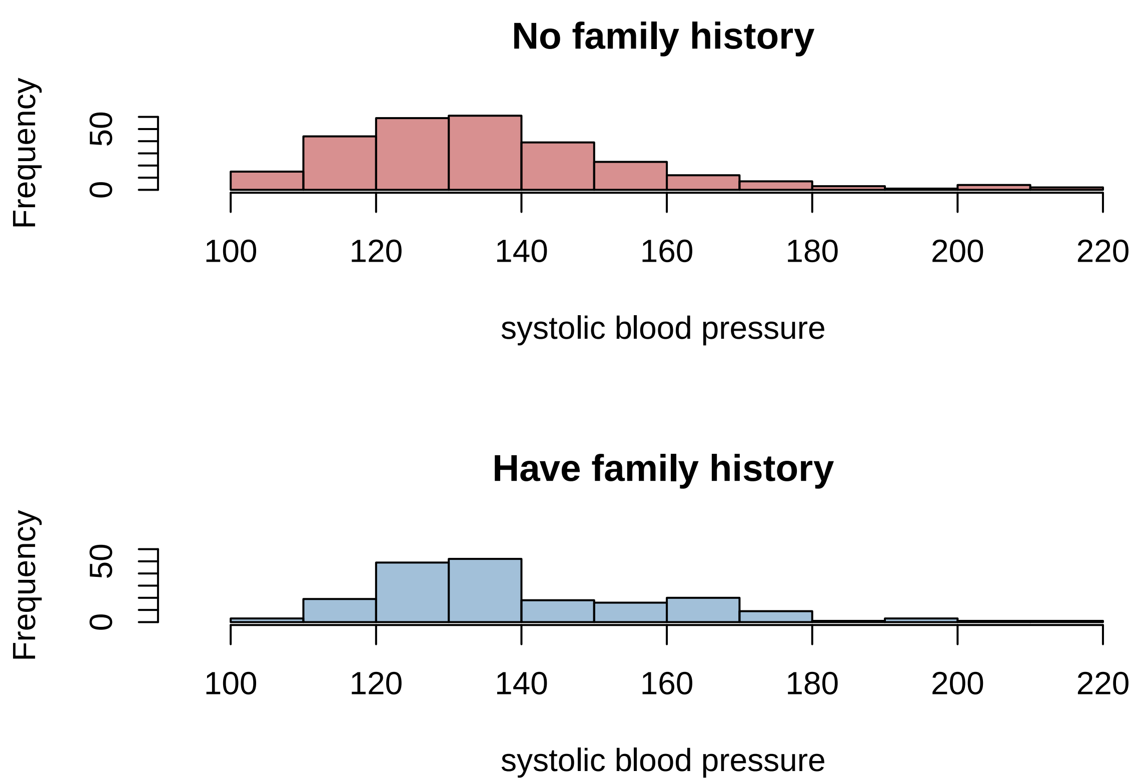


```
par(savePar)
```

注意纵坐标不相同。

以下为横纵坐标相同的绘制方法:

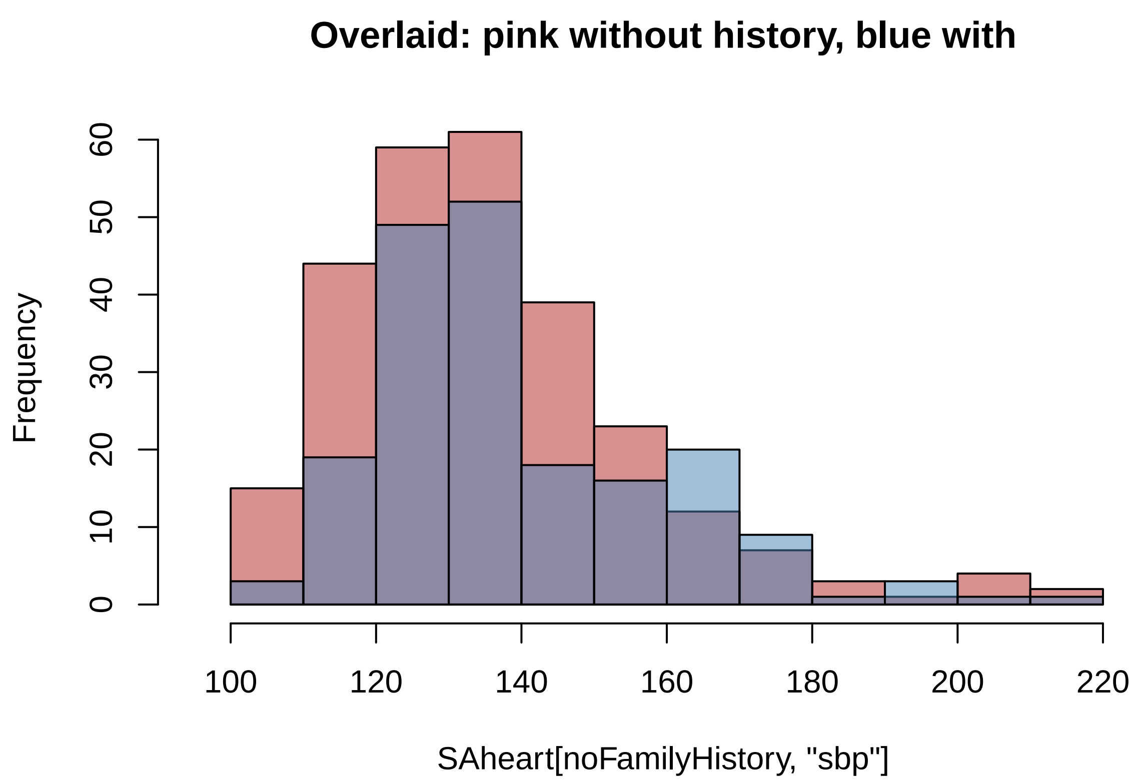
```
savePar = par(mfrow = c(2,1))
hist(SAheart[noFamilyHistory,"sbp"],
     col = noHistoryCol,
     main = "No family history",
     xlab = "systolic blood pressure",
     xlim = extendrange(SAheart[,"sbp"]),
     ylim = c(0, 60))
hist(SAheart[FamilyHistory,"sbp"],
     col = famHistoryCol,
     main = "Have family history",
     xlab = "systolic blood pressure",
     xlim = extendrange(SAheart[,"sbp"]),
     ylim = c(0, 60))
```



```
par(savePar)
```

重叠绘制：

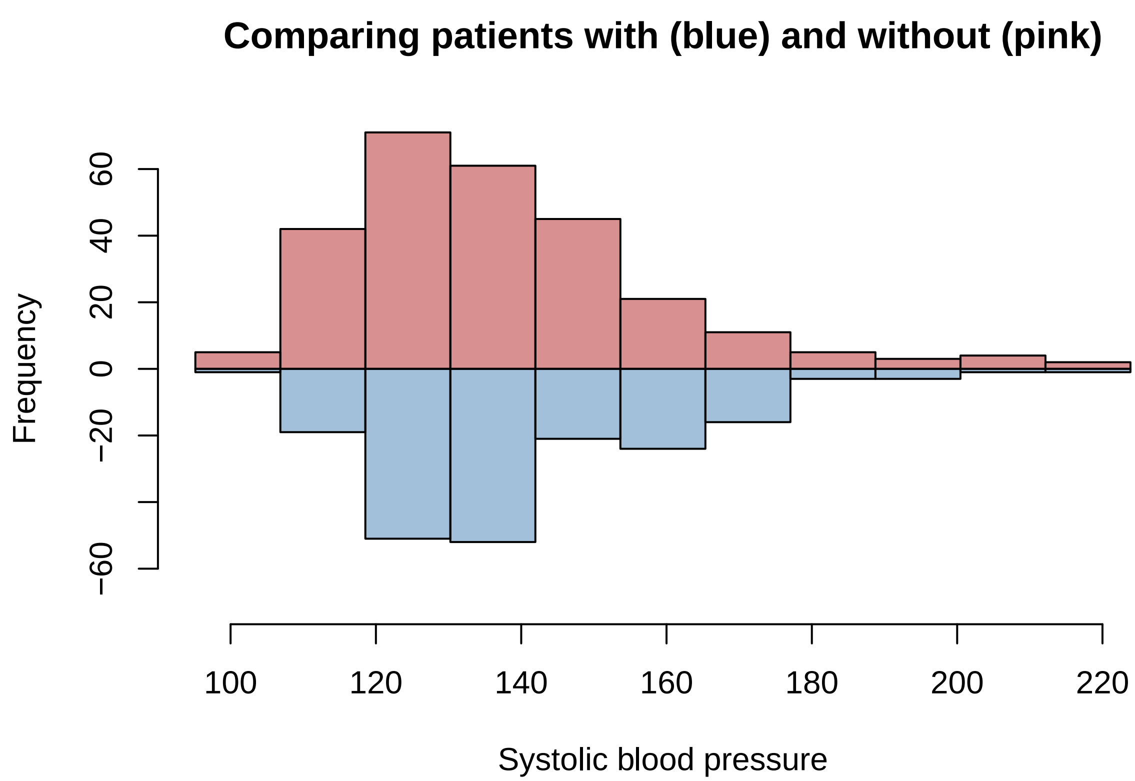
```
# 利用标题充当图例
hist(SAheart[noFamilyHistory,"sbp"], col=noHistoryCol,
     main="Overlaid: pink without history, blue with", # 粉色为没有家族病史，蓝色为有家族病史
     xlim=extendrange(SAheart[,"sbp"]))
hist(SAheart[FamilyHistory,"sbp"],
     col=famHistoryCol,
     xlim=extendrange(SAheart[,"sbp"]),
     add=TRUE)
```



反向绘制：

```
xrange <- extendrange(SAheart[,"sbp"]) # 根据 sbp 的取值范围略微拓展并统一横坐标
breaks <- seq(xrange[1], xrange[2], length.out = 12) # 分割 12 块，返还一个 list of float
h1 = hist(SAheart[noFamilyHistory,"sbp"], # 储存图像并不绘制
          breaks= breaks, plot=FALSE)
```

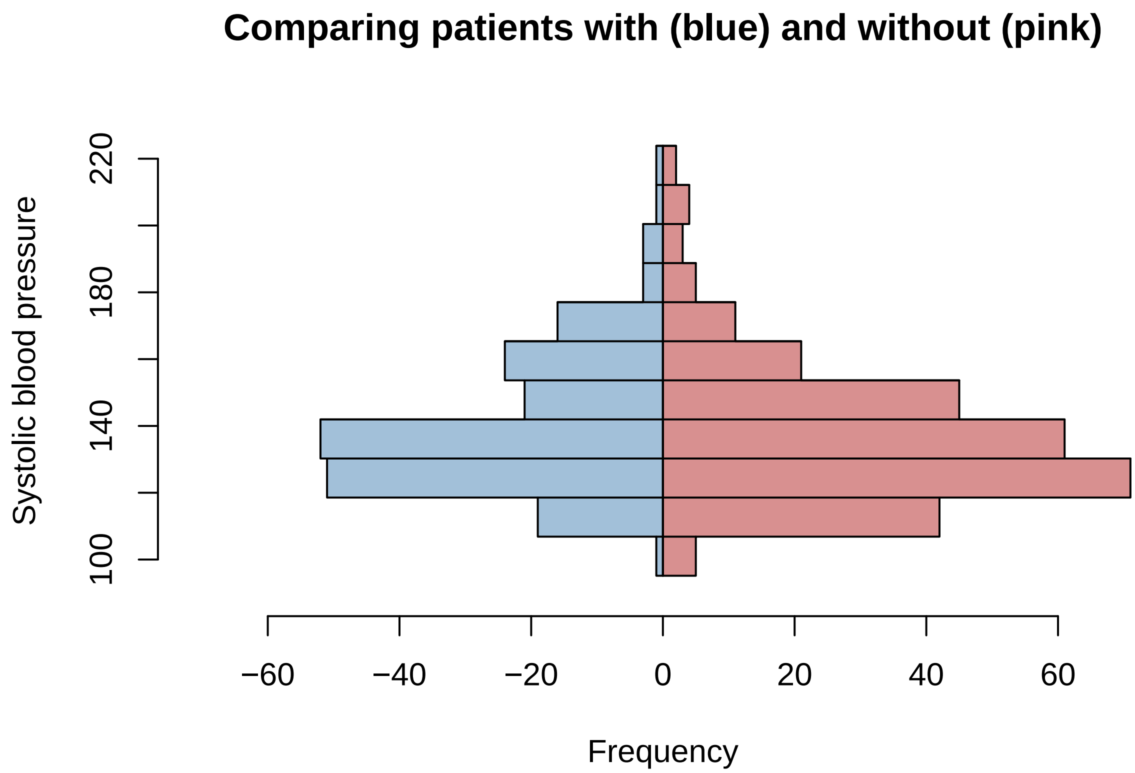
```
h2 = hist(SAheart[FamilyHistory,"sbp"],
          breaks= breaks, plot=FALSE)
hmax = max(c(h1$counts, h2$counts)) # 取 ylim 上限
h2$counts = - h2$counts # 确定 h2 为反向图像
hmin = -hmax # 堆成 ylim
X = c(h1$breaks, h2$breaks) # 合成一个 24 位数的 list
xmax = max(X) # 取 24 中最大值
xmin = min(X) # 取 24 中最小值
plot(h1, xlab="Systolic blood pressure",
     main="Comparing patients with (blue) and without (pink)", # 标题充当图例
     ylim=c(hmin, hmax), # 堆成纵坐标
     xlim=c(xmin, xmax), # 将 24 中最小最大值赋予横坐标范围
     col=noHistoryCol)
lines(h2, col=famHistoryCol) # 添加 h2 到图中
```



类人口分布的纵向直方图绘制方法:

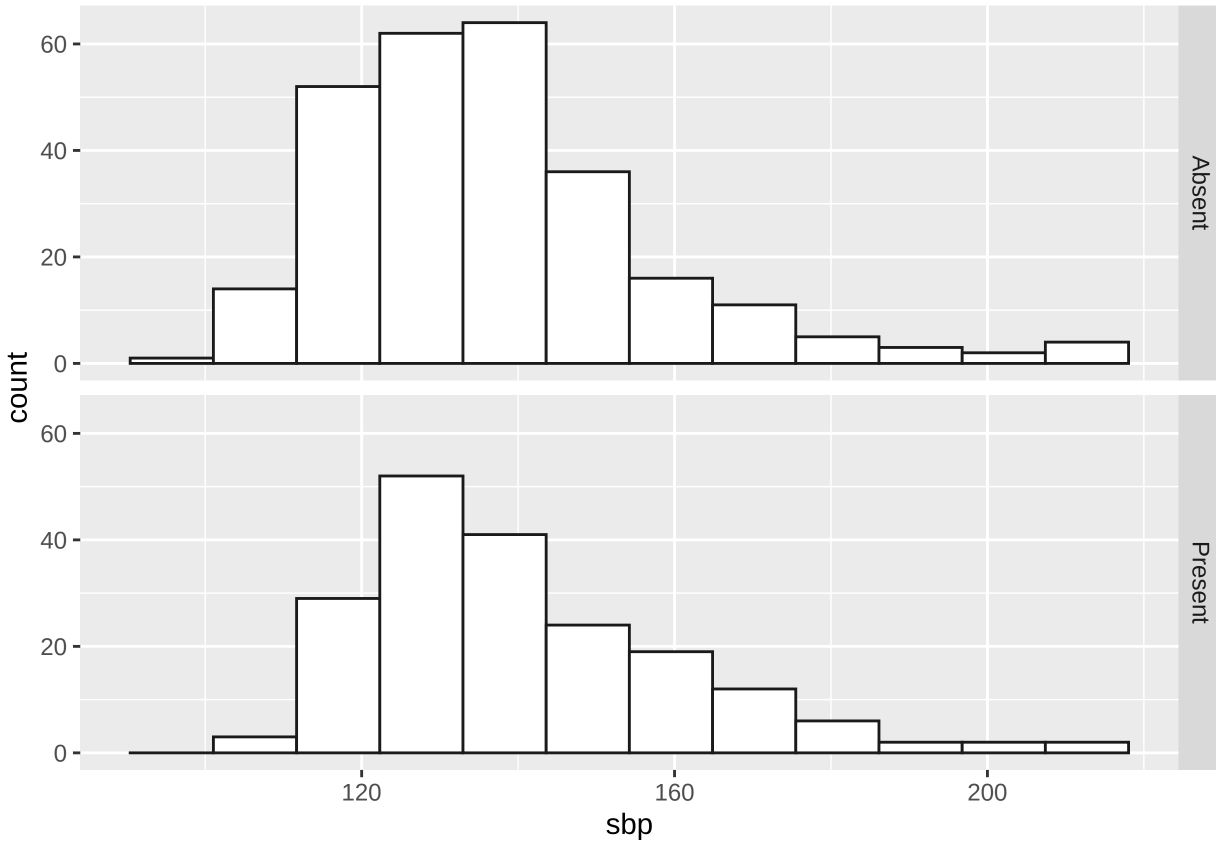
```
yrange <- extendrange(SAheart[,"sbp"])
breaks <- seq(yrange[1], yrange[2], length.out = 12)
```

```
h1 = hist(SAheart[noFamilyHistory,"sbp"], breaks= breaks, plot=FALSE)
h2 = hist(SAheart[FamilyHistory,"sbp"], breaks= breaks, plot=FALSE)
nbreaks <- length(breaks)
hmax = max(c(h1$counts, h2$counts))
h2$counts = - h2$counts
hmin = -hmax
Y <-rep(h1$breaks, each=2) # 所有 h1 中元素在原元素后重复一次并构成新的 list
X <-c(0, rep(h1$counts, each=2), 0)
# rep(0,2) 和 range(Y) 的对应元素分别为线段的起点终点
# type = "l" 为绘制线段
plot(rep(0,2), range(Y), type = "l",
     col="black",
     xlim = c(hmin, hmax),
     ylim = extendrange(Y),
     bty="n",
     xlab="Frequency",
     ylab="Systolic blood pressure",
     main="Comparing patients with (blue) and without (pink)")
polygon(X, Y, col=noHistoryCol) # 绘制多边形并填充颜色
for (i in 1:nbreaks) {lines(c(0, h1$counts[i]), c(rep(h1$breaks[i+1],2)))}

Y <-rep(h2$breaks, each=2) # 所有 h2 中元素在原元素后重复一次并构成新的 list
X <-c(0, rep(h2$counts, each=2), 0)
polygon(X, Y, col=famHistoryCol)
for (i in 1:nbreaks) {lines(c(0, h2$counts[i]), c(rep(h2$breaks[i+1],2)))}
```



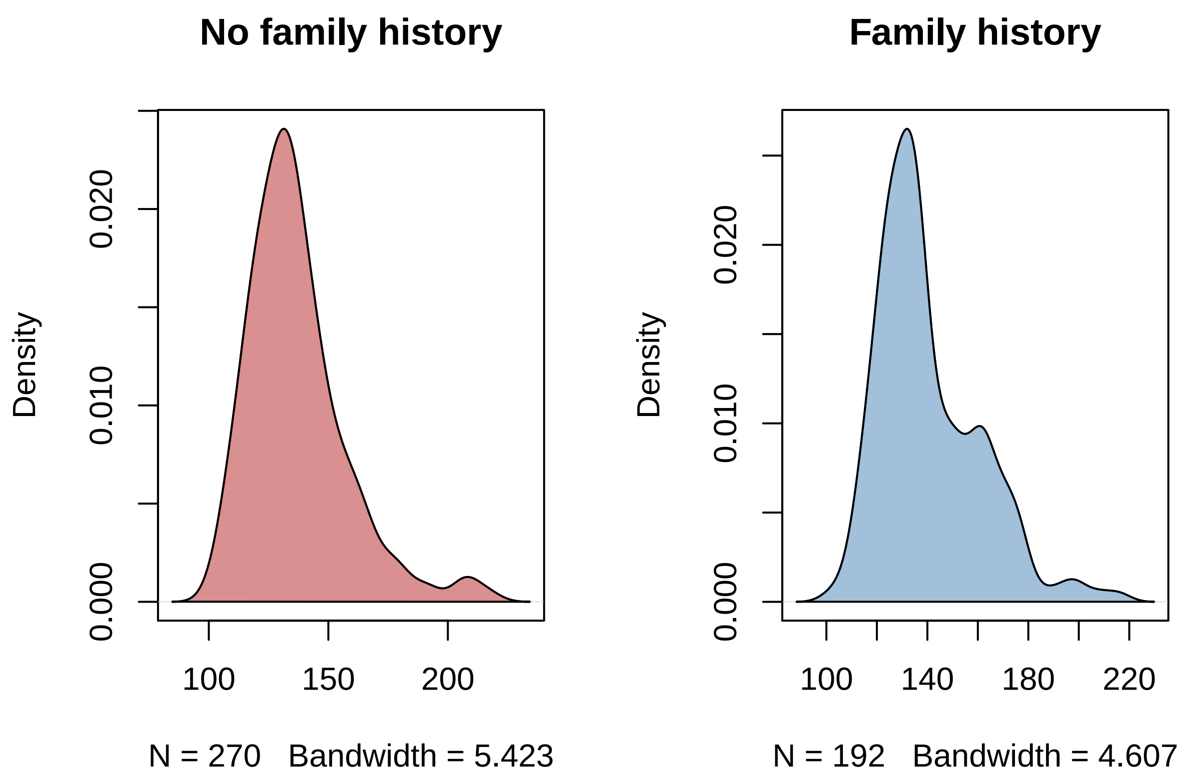
使用 ggplot2 绘制：

```
library(ggplot2) # 导入 ggplot2 包
ggplot(data = SAheart, mapping = aes(x=sbp)) + # 建立图床
  geom_histogram(bins = 12, # 绘制直方图并设定 bins 数量
                 colour = "grey10", # 网格颜色
                 fill = "white") + # 网格内颜色
  facet_grid(famhist ~.) # 根据 famhist 变量来对比绘制
```



3.2.1.2.3 密度曲线 使用密度曲线比较数据。

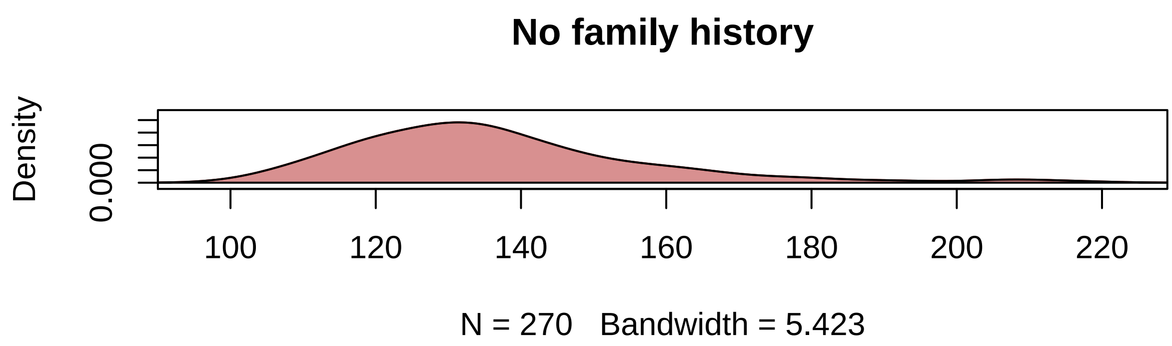
```
savePar = par(mfrow=c(1,2))
densAbsent <- density(SAheart[noFamilyHistory,"sbp"], bw="SJ") # 储存密度图
densPresent <- density(SAheart[FamilyHistory,"sbp"], bw="SJ")

plot(densAbsent, col="firebrick", main="No family history") # 绘制密度图
polygon(densAbsent, col=noHistoryCol) # 多边形填色
plot(densPresent, col="steelblue", main="Family history")
polygon(densPresent, col=famHistoryCol)
```

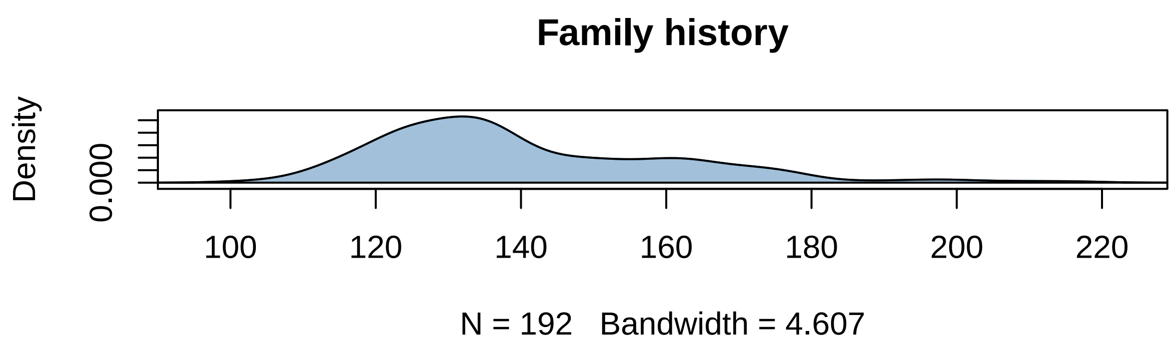



```
par(savePar) # 结束绘制
```

注意纵坐标不相同。

```
savePar = par(mfrow = c(2,1))
xlim <- extendrange(SAheart[,"sbp"]) # 统一横坐标
ylim <- extendrange(c(densAbsent$y, densPresent$y)) # 统一纵坐标
densAbsent <- density(SAheart[noFamilyHistory,"sbp"], bw = "SJ")
densPresent <- density(SAheart[FamilyHistory,"sbp"], bw = "SJ")

plot(densAbsent, col = "firebrick", main = "No family history",
     xlim = xlim, ylim = ylim)
polygon(densAbsent, col = noHistoryCol)
plot(densPresent, col = "steelblue", main = "Family history",
     xlim = xlim, ylim = ylim)
polygon(densPresent, col = famHistoryCol)
```

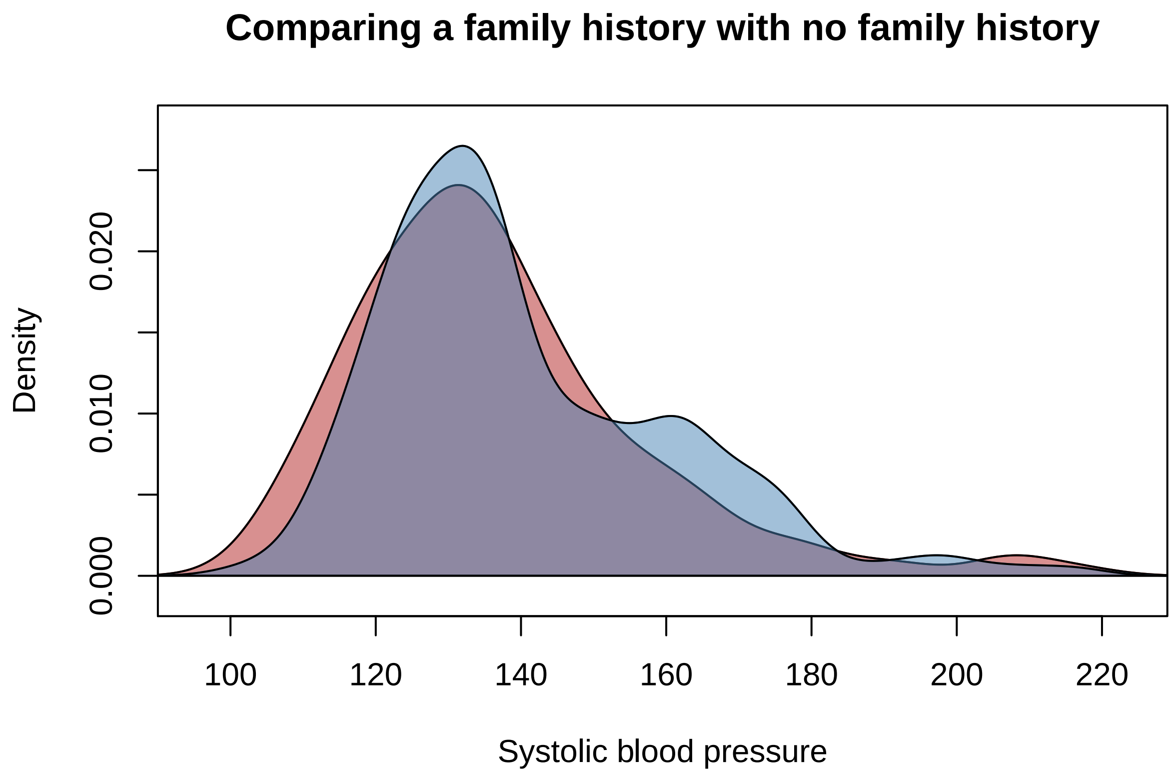





```
par(savePar)
```

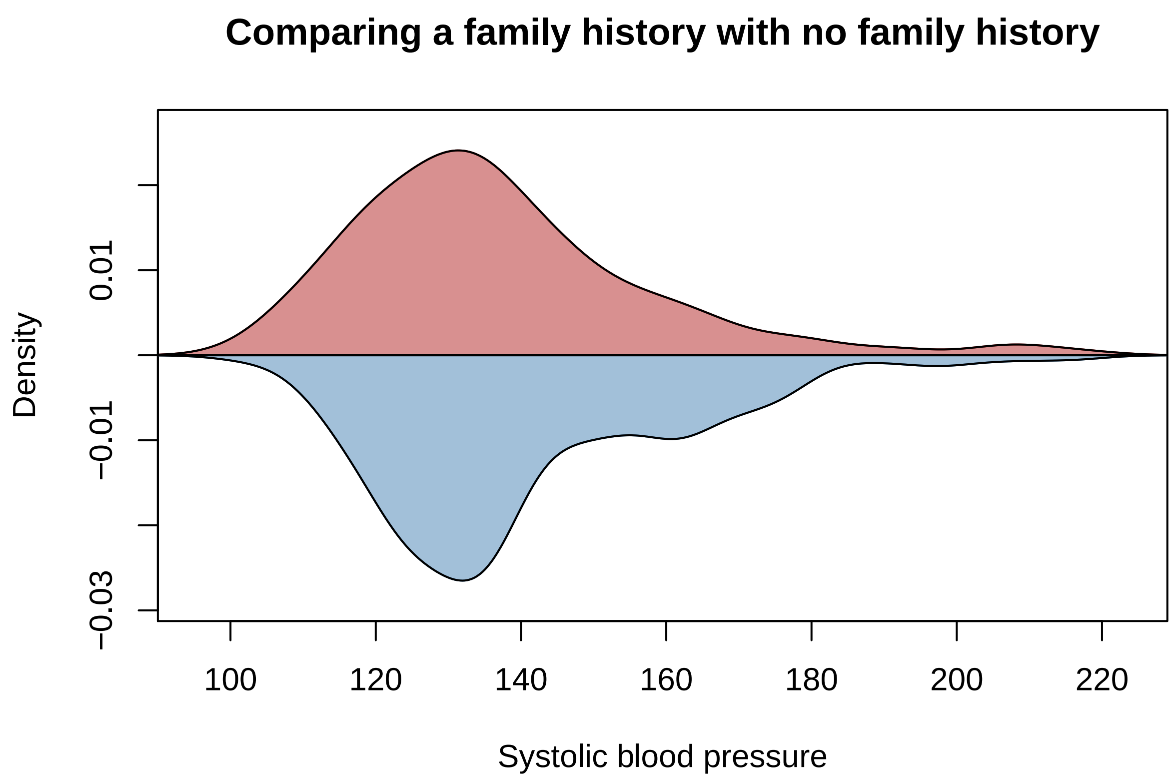
重叠绘制：

```
xlim <- extendrange(SAheart[,"sbp"]) # 统一横坐标
ylim <- extendrange(c(densAbsent$y, densPresent$y)) # 统一纵坐标
densAbsent <- density(SAheart[noFamilyHistory,"sbp"], bw = "SJ")
densPresent <- density(SAheart[FamilyHistory,"sbp"], bw = "SJ")

plot(densAbsent, col="firebrick", # 绘制主图 (没有家族史)
     xlab="Systolic blood pressure",
     main="Comparing a family history with no family history",
     xlim=xlim, ylim=ylim)
polygon(densAbsent, col=noHistoryCol)

lines(densPresent, col="steelblue")
polygon(densPresent, col=famHistoryCol)
```



反向绘制：

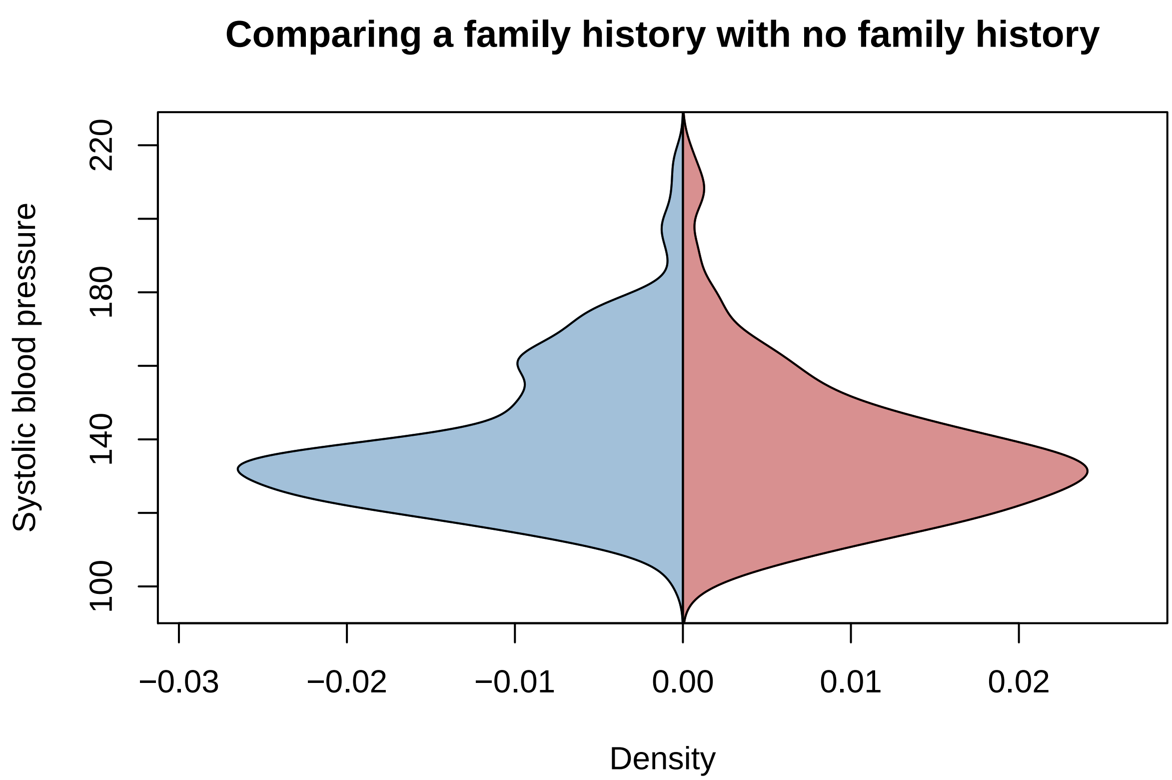
```
xlim <- extendrange(SAheart[,"sbp"]) # 统一横坐标
densAbsent <- density(SAheart[noFamilyHistory,"sbp"], bw = "SJ")
densPresent <- density(SAheart[FamilyHistory,"sbp"], bw = "SJ")
densPresent$y <- -densPresent$y # 改变有家族史的数据的纵坐标
ylim <- extendrange(c(densAbsent$y, densPresent$y)) # 设定纵坐标

plot(densAbsent, col="firebrick", # 绘制主图（没有家族史）
     xlab="Systolic blood pressure",
     main="Comparing a family history with no family history",
     xlim=xlim, ylim=ylim)
polygon(densAbsent, col=noHistoryCol)

lines(densPresent, col="steelblue")
polygon(densPresent, col=famHistoryCol)
```



纵向绘制方法:

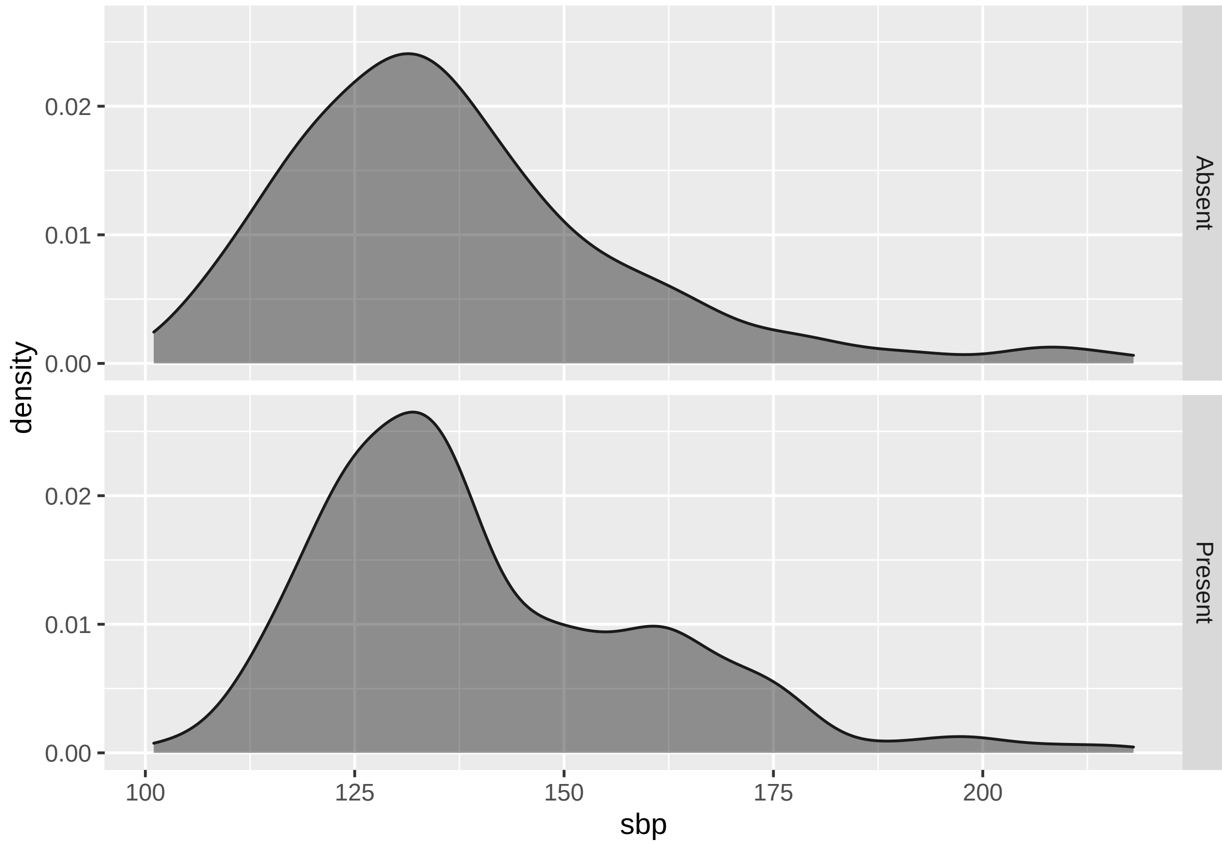
```
densAbsent <- density(SAheart[noFamilyHistory,"sbp"], bw = "SJ")
densPresent <- density(SAheart[FamilyHistory,"sbp"], bw = "SJ")
ylim <- extendrange(SAheart[,"sbp"]) # 设定纵坐标范围
densPresent$y <- - densPresent$y
xlim <- extendrange(c(densAbsent$y, densPresent$y))

xyswitch <- function(xy_thing) { # 颠倒 xy 顺序的函数
  yx_thing <- xy_thing
  yx_thing$x <- xy_thing$y
  yx_thing$y <- xy_thing$x
  yx_thing
  }

plot(xyswitch(densAbsent),
     col="firebrick",
     xlab="Density",
     ylab="Systolic blood pressure",
```

```
     main="Comparing a family history with no family history",
     xlim=xlim, ylim=ylim)

polygon(xyswitch(densAbsent), col=noHistoryCol)
lines(xyswitch(densPresent), col="steelblue")
polygon(xyswitch(densPresent), col=famHistoryCol)
```



使用 ggplot2 绘制：

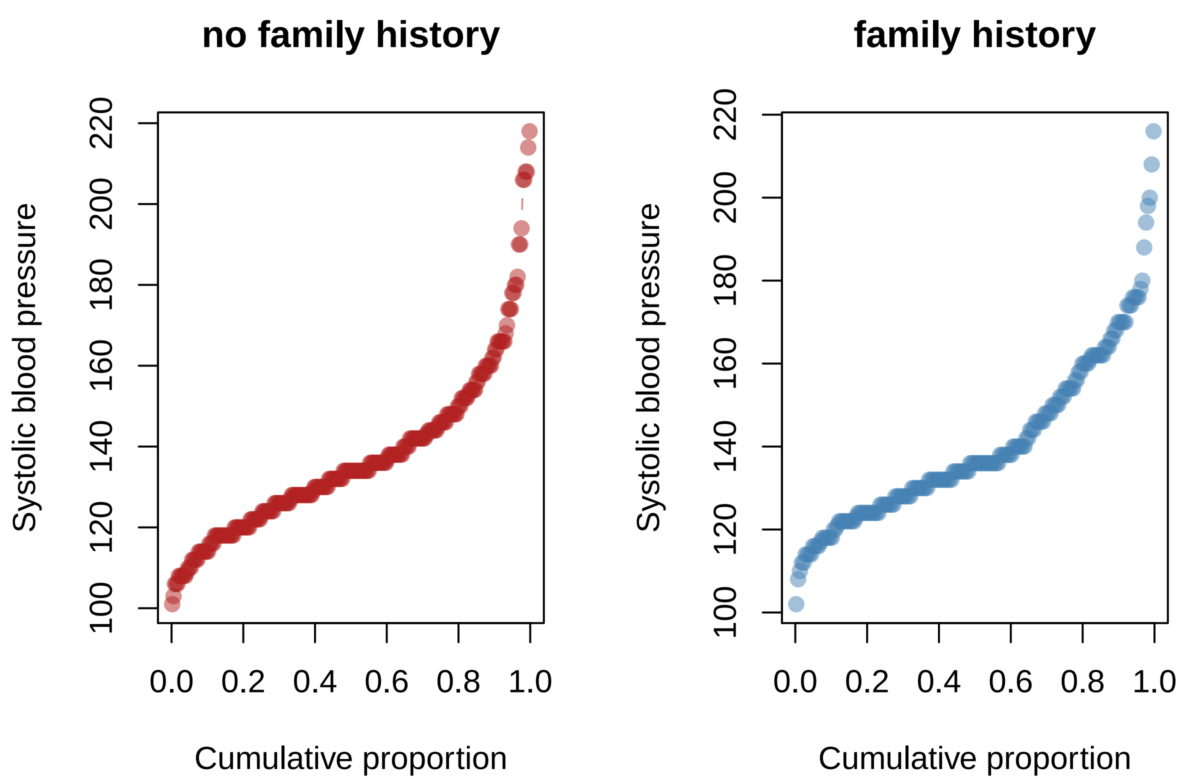
```
library(ggplot2)
ggplot(data = SAheart, mapping = aes(x=sbp, col=famhist)) +
  geom_density(colour = "grey10",
               fill = "black",
               alpha = 0.4,
               bw = "SJ") +
  facet_grid(famhist ~.)
```



3.2.1.2.4 百分位图 并排绘制：

```
savePar <- par(mfrow=c(1,2))
nAbsent <- sum(noFamilyHistory)
nPresent <- sum(FamilyHistory)
pAbsent <- ppoints(nAbsent) # list of 百分位（有家族病史）
pPresent <- ppoints(nPresent) # list of 百分位（无家族病史）

plot(pAbsent, sort(SAheart[noFamilyHistory,"sbp"]),
     type="b", # "b" for both points and lines
     col=noHistoryCol,
     pch=19, # 点的大小
     xlab="Cumulative proportion",
     ylab = "Systolic blood pressure",
     main="no family history")

plot(pPresent, sort(SAheart[FamilyHistory,"sbp"]),
     type="b",
```

```
     col=famHistoryCol,
     pch=19,
     xlab="Cumulative proportion",
     ylab = "Systolic blood pressure",
     main="family history")
```

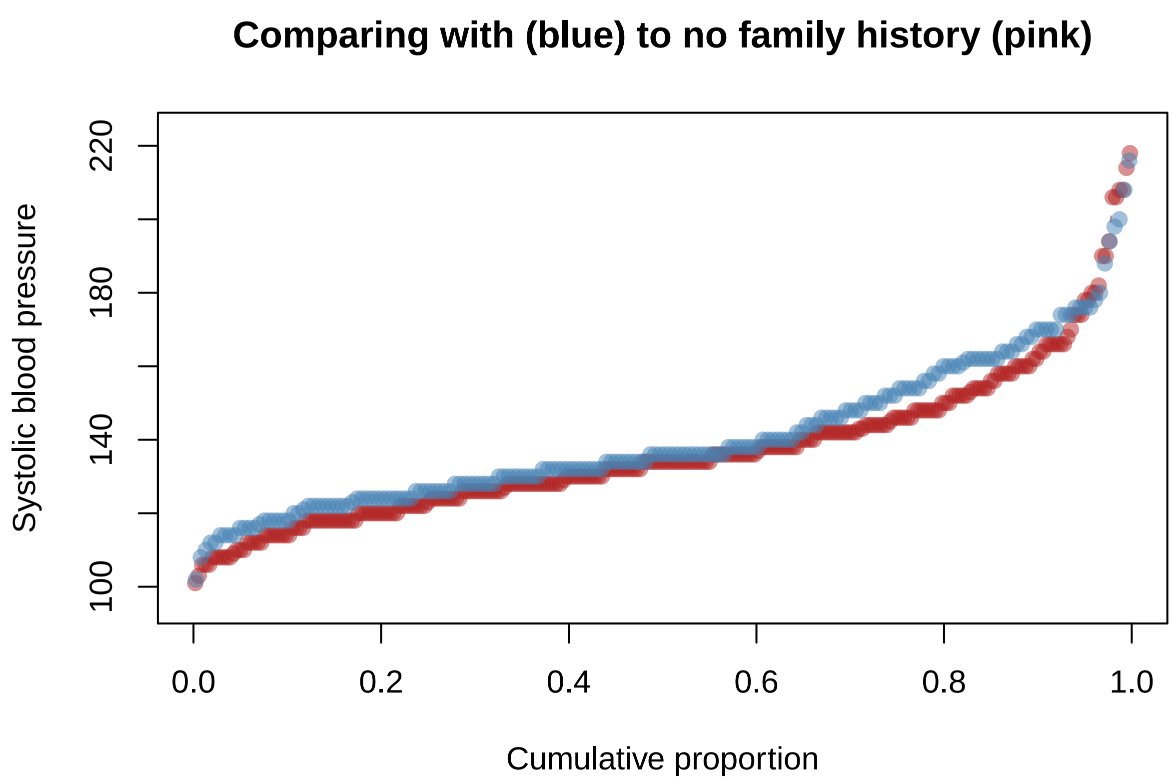



```
par(savePar)
```

重叠绘制：

```
ylim <- extendrange(SAheart[,"sbp"])
nAbsent <- sum(noFamilyHistory)
nPresent <- sum(FamilyHistory)
pAbsent <- ppoints(nAbsent)
pPresent <- ppoints(nPresent)

plot(pAbsent, sort(SAheart[noFamilyHistory,"sbp"]), type = "b",
     col = noHistoryCol,  pch = 19,
     ylim = ylim,
```

```
     xlab = "Cumulative proportion",
     ylab = "Systolic blood pressure",
     main = "Comparing with (blue) to no family history (pink)")

points(pPresent, sort(SAheart[FamilyHistory,"sbp"]),
       type = "b",
       col = famHistoryCol,
       pch=19)
```



3.2.2 数据关系

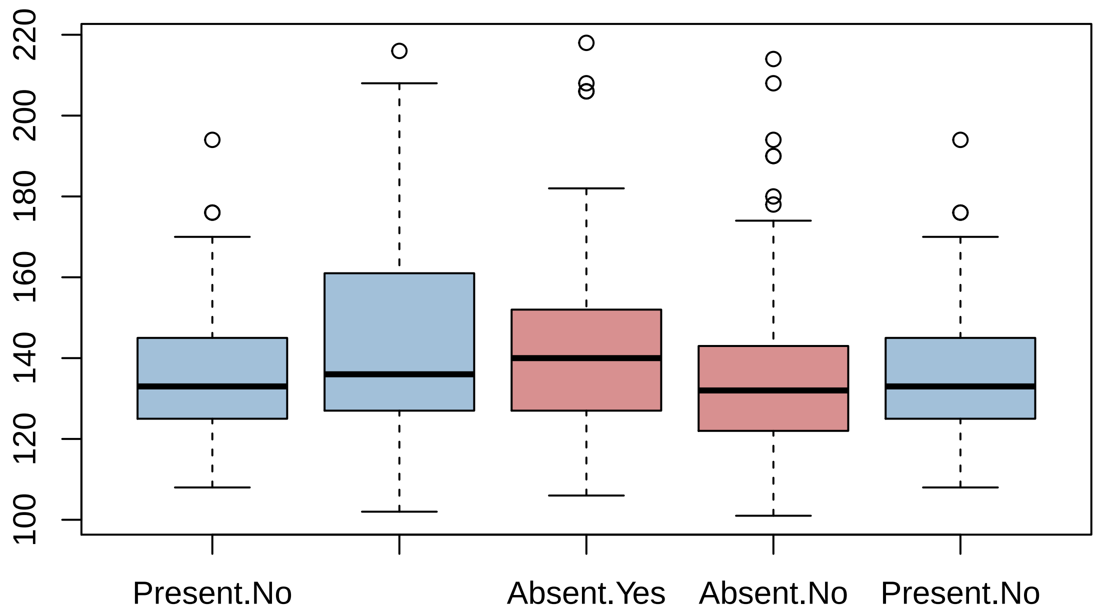
```
kable(head(SAheart))
```

sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
160	12.00	5.73	23.11	Present	49	25.30	97.20	52	Yes
144	0.01	4.41	28.61	Absent	55	28.87	2.06	63	Yes
118	0.08	3.48	32.28	Present	52	29.14	3.81	46	No
170	7.50	6.41	38.03	Present	51	31.99	24.26	58	Yes
134	13.60	3.50	27.78	Present	60	25.99	57.34	49	Yes
132	6.20	6.47	36.21	Present	62	30.77	14.14	45	No

```
# 假设我们需要探究多变量对单一变量的影响
# 以南非心脏病数据集为例子
# 家族病史（famhist）的有无以及冠状动脉疾病（chd）的有无对心脏收缩压（sbp）的影响
groups <- with(SAheart, split(sbp, list(famhist, chd))) # 对原始数据集进行分组
kable(t(names(groups)))
```

Absent.No	Present.No	Absent.Yes	Present.Yes

```
# 以下绘制两两配对箱型图（历遍）
ord <- c("Present.No", "Present.Yes", "Absent.Yes", "Absent.No", "Present.No")
cols <- adjustcolor(c("steelblue", "steelblue", "firebrick", "firebrick", "steelblue"), 0.5)
boxplot(groups[ord], col=cols)
```



一个更多配对的例子：

```
# 安装依赖包
# if (!requireNamespace("BiocManager", quietly = TRUE))
#     install.packages("BiocManager")
# BiocManager::install(version = "3.12")
# BiocManager::install("BiocGenerics")
```

```
library("PairViz")
data(cancer) # 维他命 C 治疗五种不同部位的癌症和生存时间的数据集
str(cancer)
```

```
## 'data.frame':    64 obs. of  2 variables:
##  $ Survival: int  124 42 25 45 412 51 1112 46 103 876 ...
##  $ Organ   : Factor w/ 5 levels "Breast","Bronchus",..: 5 5 5 5 5 5 5 5 5 5 ...
```

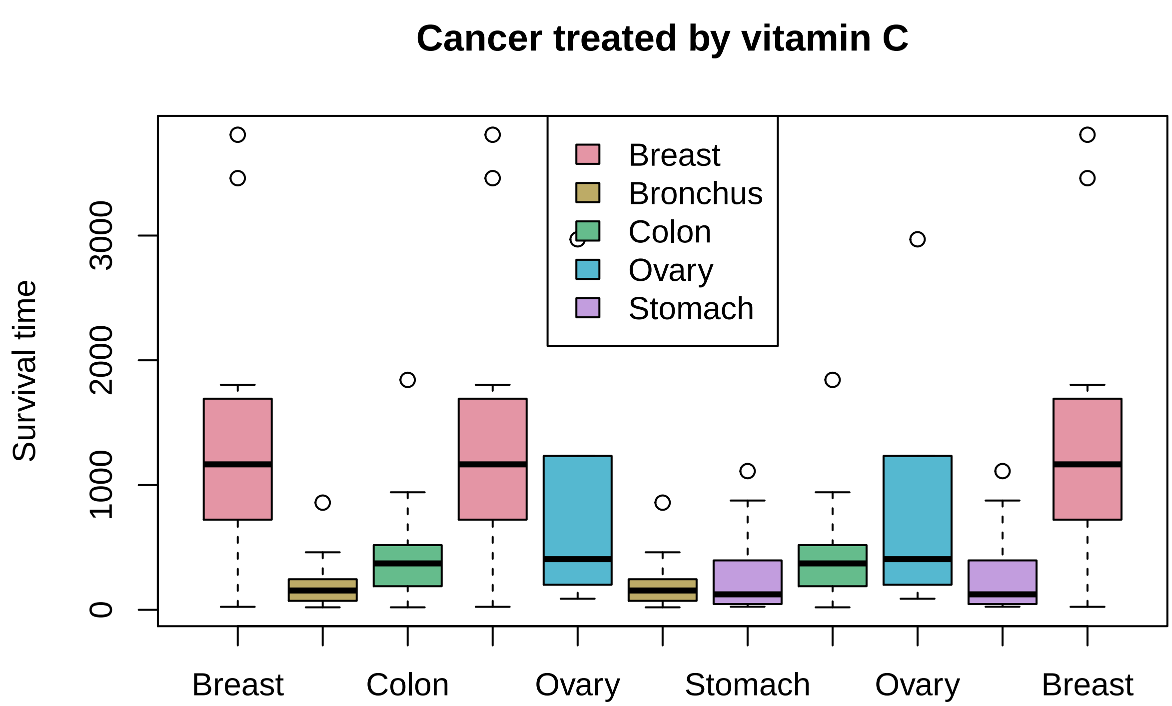
```
organs <- with(cancer, split(Survival, Organ))
organNames <- names(organs)
str(organs)
```

```
## List of 5
##  $ Breast  : int [1:11] 1235 24 1581 1166 40 727 3808 791 1804 3460 ...
##  $ Bronchus: int [1:17] 81 461 20 450 246 166 63 64 155 859 ...
##  $ Colon   : int [1:17] 248 377 189 1843 180 537 519 455 406 365 ...
##  $ Ovary   : int [1:6] 1234 89 201 356 2970 456
##  $ Stomach : int [1:13] 124 42 25 45 412 51 1112 46 103 876 ...
```

```r
ord <- eulerian(5) # 计算两两配对的序号排列
ord
```

```
##  [1] 1 2 3 1 4 2 5 3 4 5 1
```

```r
library(colorspace)
cols <- rainbow_hcl(5, c = 50) # 选取颜色
boxplot(organs[ord], # 根据序号 Loop 数据
        col=cols[ord], # 根据序号给数据上色
        ylab="Survival time",
        main="Cancer treated by vitamin C")
legend("top",fill = cols, legend = organNames)
```



4 二维数据

4.1 时间数据

4.1.1 图像绘制

以内建太阳黑子数据集为例子：

```
str(sunspot.month) # 以月份为单位查看 sunspot 中的所有时间数据
```

```
##  Time-Series [1:3177] from 1749 to 2014: 58 62.6 70 55.7 85 83.5 94.8 66.3 75.9 75.5 ...
```

```
sunspot.month[1:3] # 查看以月份为单位的前三个数据
```

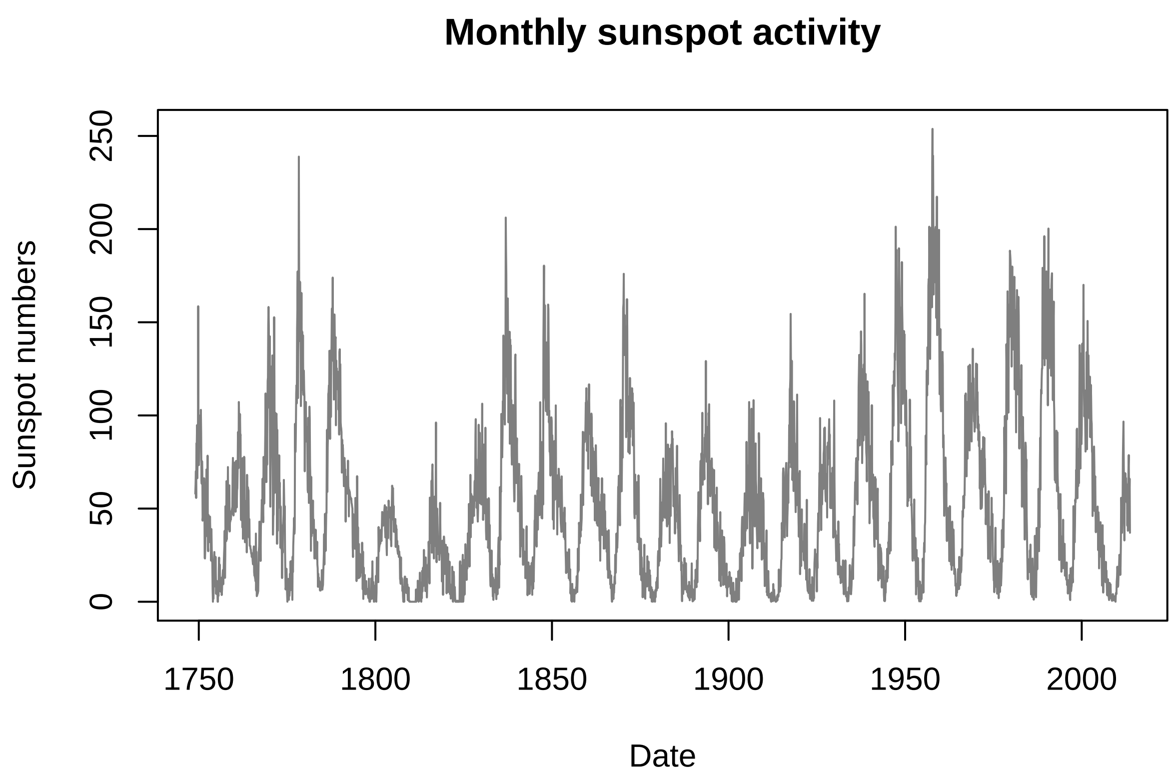
```
## [1] 58.0 62.6 70.0
```

```
time(sunspot.month)[1:6] # 月份的储存其实为 float 型
```

```
## [1] 1749.000 1749.083 1749.167 1749.250 1749.333 1749.417
```

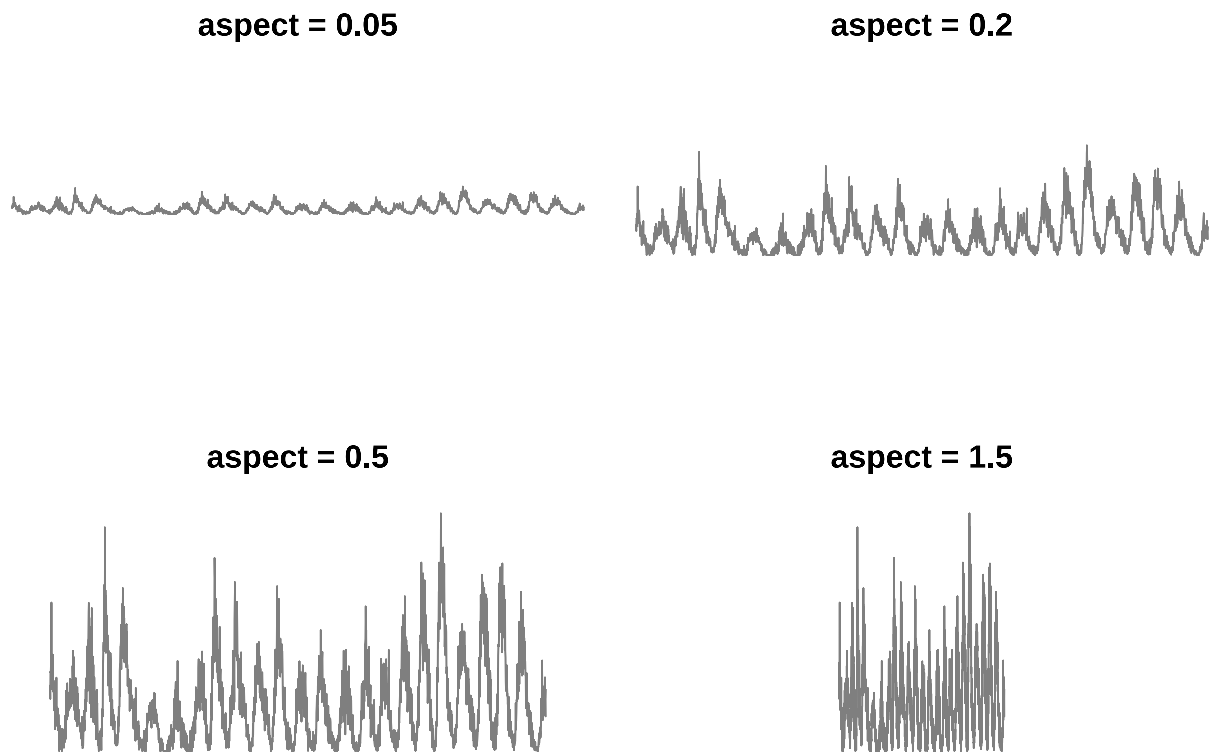
以时间为顺序绘制所有数据：

```
plot(sunspot.month,
     type = "l", # 折线型
     col = "grey50", # 上色
     main = "Monthly sunspot activity", # 太阳黑子活动（月份）
     xlab = "Date", # 时间（月份）
     ylab = "Sunspot numbers") # 太阳黑子数
```



改变图像纵横比:

```
grey <- adjustcolor("grey", 0.35)
savePar <- par(mfrow = c(2,2),mar = c(2.5, 0.1, 3, 0.1), oma = rep(0,4))
for (aspect in c(0.05, 0.2, 0.5, 1.5)) {
  plot(sunspot.month,
       type = "l",
       asp = aspect, # 设定纵横比
       col = "grey50",
       main = paste("aspect =", aspect),
       xlab = "", axes=FALSE,
       ylab = "")
}
```




```
par(savePar)
```

以月份为单位查看时间段内数据:

```
# 1960 年 1 月开始, 1961 年 12 月结束
window(sunspot.month, start=c(1960,1), end=c(1961, 12))
```

```
##         Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
## 1960 146.3 106.0 102.2 122.0 119.6 110.2 121.7 134.1 127.2  82.8  89.6  85.6
## 1961  57.9  46.1  53.0  61.4  51.0  77.4  70.2  55.8  63.6  37.7  32.6  39.9
```

查看数据集起始和结束:

```
start(sunspot.month)
```

```
## [1] 1749    1
```

```
end(sunspot.month)
```

```
## [1] 2013    9
```

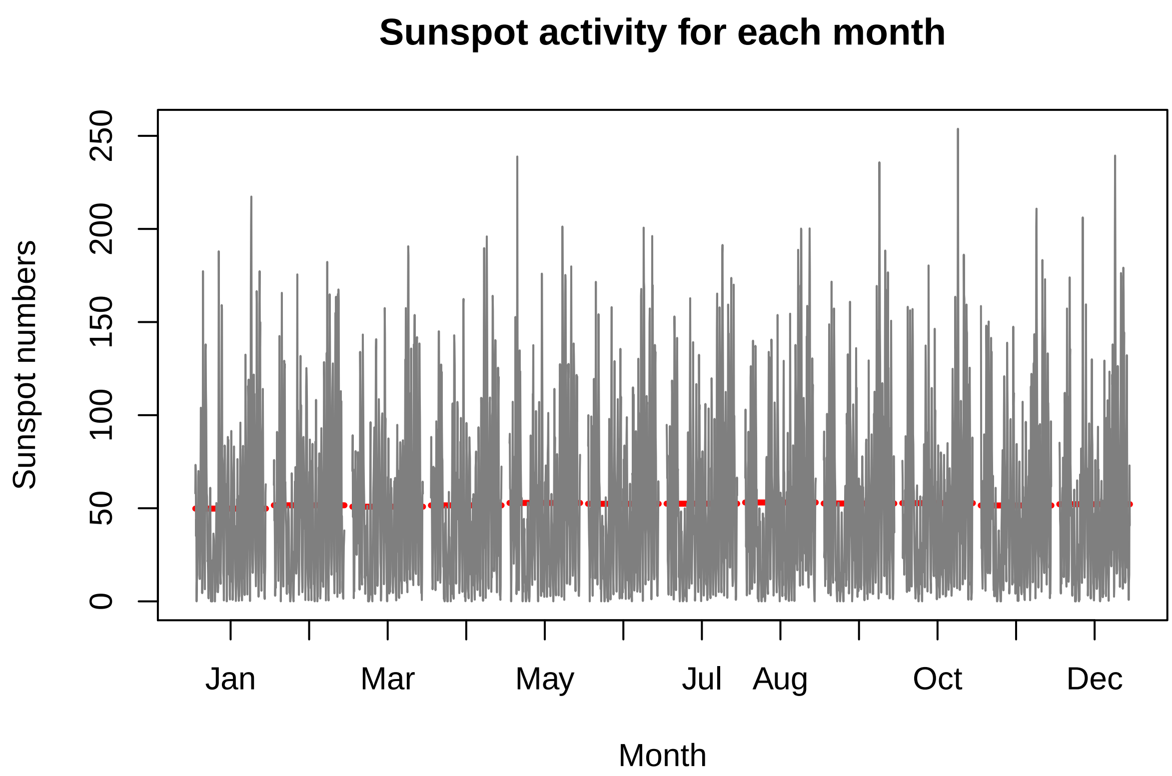
查看数据集循环

```
window(cycle(sunspot.month), start=c(1960,1), end=c(1960,12))
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1960   1   2   3   4   5   6   7   8   9  10  11  12
```

以月份为单位绘制所有数据:

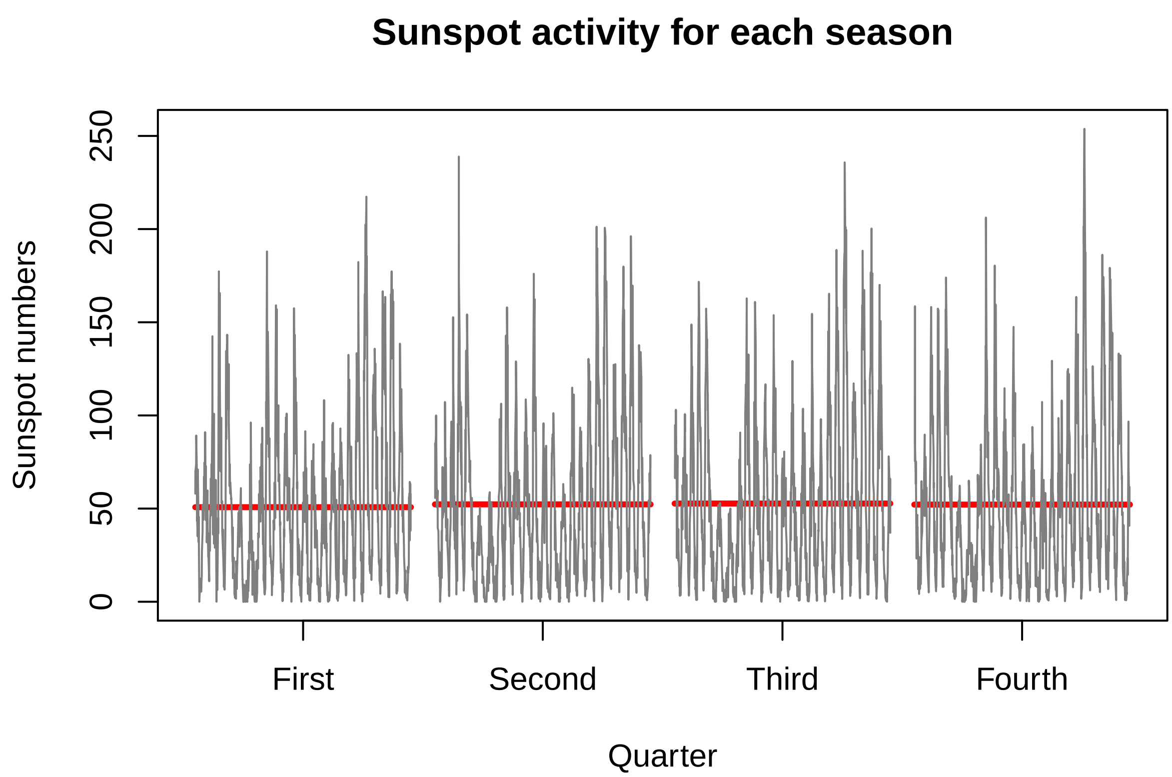
```
monthplot(sunspot.month, # 注意为 monthplot
          main = "Sunspot activity for each month", # 每月的太阳活动情况
          xlab = "Month", # 月份
          ylab = "Sunspot numbers", # 太阳黑子数量
          col = "grey50",
          labels = month.abb, # 每个月份的标签
          col.base = "red", # 参考线（均值）的颜色
          lwd.base = 3 # 线的宽度
          )
```



以季度为单位绘制所有数据:

```
# cycle(sunspot.month) 获得每年的 cycle 序号
quarter <- 1 + ((cycle(sunspot.month) - 1) %/% 3) # 注意为%/% 不是%%
# 获得新的循环
#       Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
# 1749   1   1   1   2   2   2   3   3   3   4   4   4
```

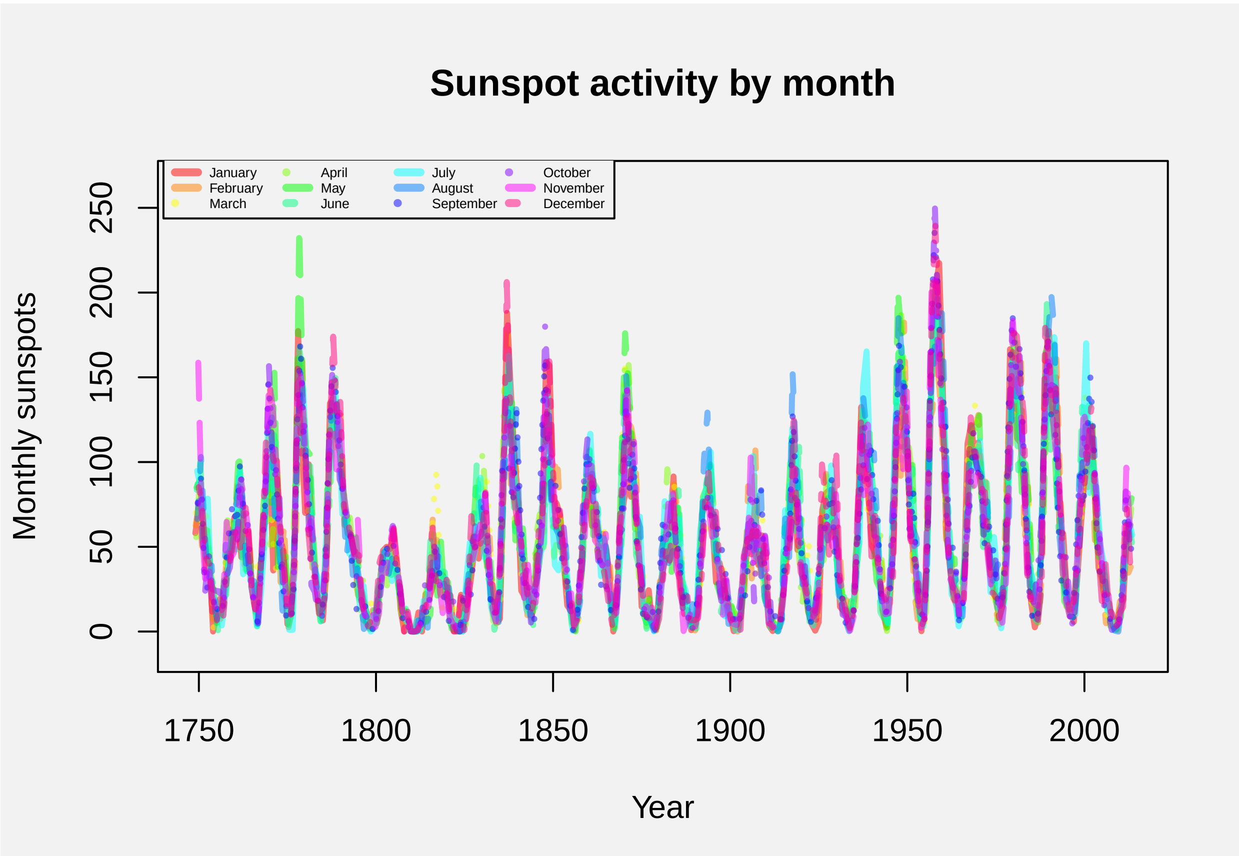
```
monthplot(sunspot.month, # 注意为 monthplot
          main = "Sunspot activity for each season", # 每季度的太阳活动情况
          xlab = "Quarter", # 月份
          ylab = "Sunspot numbers", # 太阳黑子数量
          phase = quarter,
          col = "grey50",
          labels = c("First", "Second", "Third", "Fourth"), # 每个季度的标签
          col.base = "red", # 参考线 (均值) 的颜色
          lwd.base = 3 # 线的宽度
          )
```



以每一年中的相同月份为单位绘制所有数据:

```
savePar <- par(bg="grey95") # 设定作图背景颜色
cols <- rainbow(12, alpha = 0.5) # 选取十二种颜色, 透明度设置为 0.5

# 从 1749 年 1 月开始, 间隔一年 (deltat = 1) 绘制以月份的图
plot(window(sunspot.month, start = c(1749,1), deltat = 1),
     ylim = extendrange(sunspot.month),
     col = cols[1],
     ylab = "Monthly sunspots",
     xlab = "Year",
     lwd = 3,
     main = "Sunspot activity by month")

# 绘制二到十二月的图
for (i in 2:12) {
  lines(window(sunspot.month,start= c(1749,i), deltat=1),
        lty=i, lwd=3, col=cols[i]) } # 2 到 12 种折线类型, 2 到 12 种颜色
```

```
# 制作图例
legend(1740, 280,  # 图例空间位置
       legend = c("January", "February",
                  "March", "April",
                  "May", "June",
                  "July", "August",
                  "September", "October",
                  "November", "December"),
       lty = 1:12, col = cols, # 十二种折线类型，分别配色
       lwd = 4, ncol = 4, # 线宽四单位，四列图例
       cex = 0.4)  # 设置图例大小
```



以十一年为单位循环绘制所有数据：

```
timeseries <- sunspot.month
# 间隔十一年
startYears <- seq(start(timeseries)[1], end(timeseries)[1], 11 )
# start(timeseries) -> 1749 1
# end(timeseries)[1] -> 2013 9
```

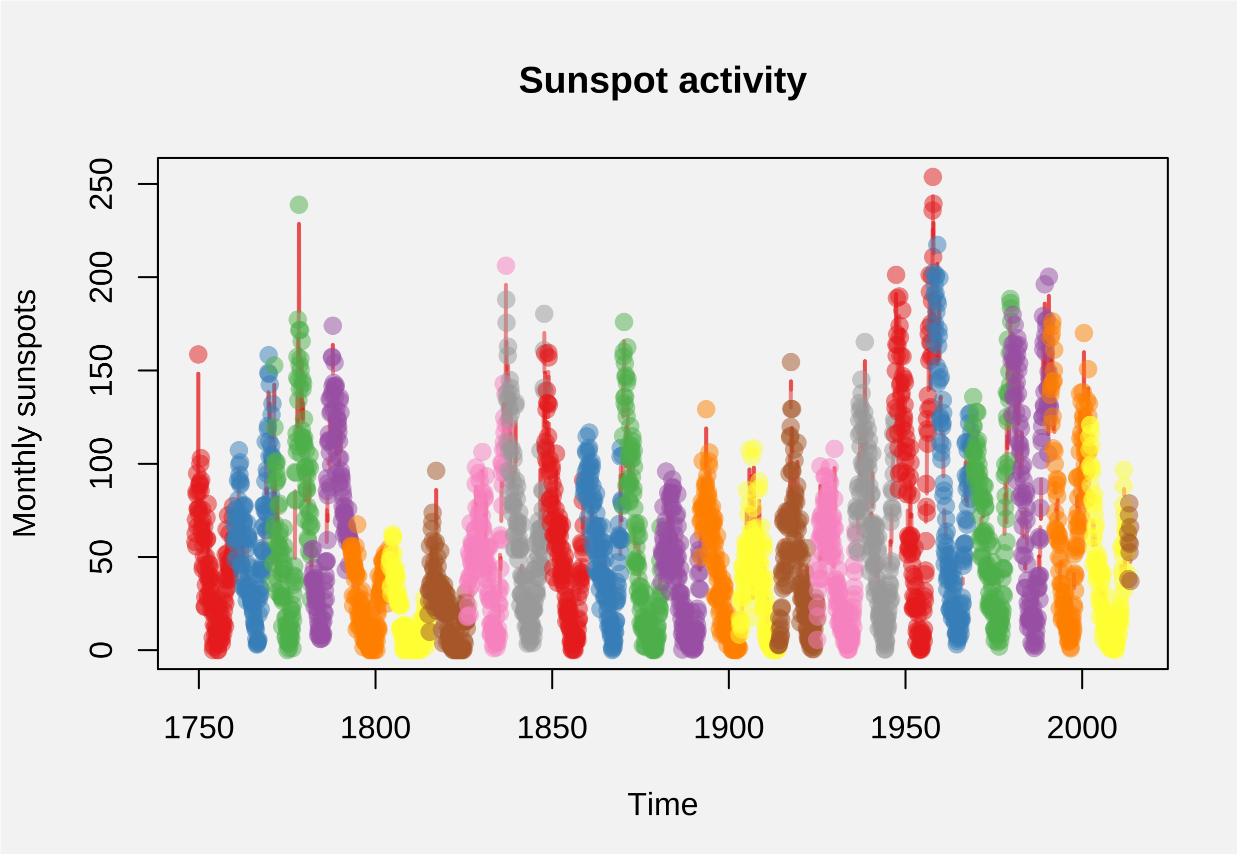
```
startYears
```

```
##  [1] 1749 1760 1771 1782 1793 1804 1815 1826 1837 1848 1859 1870 1881 1892 1903
## [16] 1914 1925 1936 1947 1958 1969 1980 1991 2002 2013
```

```
# 有 25 个循环
ncycles <- length(startYears)
# 给每个月份以 132 为单位打上循环标签 1749 1 和 1749 2 都属于 1 循环
cycleNums <- rep(1:ncycles, each=132)
# 最后一个循环不完整，截取需要的长度
cycleNums <- cycleNums[1:length(timeseries)]
# 循环位置
cyclePositions <- rep(1:132, ncycles)
# 截取需要长度
cyclePositions <- cyclePositions[1:length(timeseries)]
```

```
library(RColorBrewer)
cols <- adjustcolor(rep(brewer.pal(9, name = "Set1"),
                        # 从 brewer.pal 的 Set1 中选取 9 种颜色
                        length.out = ncycles), # 循环 25 次
                    alpha.f = 0.5) # 设定透明度
```

```
opt <- par(bg="grey95") # 设定背景颜色
plot(x = time(timeseries), y = as.vector(timeseries),
     type = "b",
     col = cols[cycleNums],
     lwd = 2, # 线宽
     pch = 19, # 点大小
     xlab = "Time", # 时间
     ylab = "Monthly sunspots", # 太阳黑子月度活动
     main = "Sunspot activity" # 太阳黑子活动
)
```

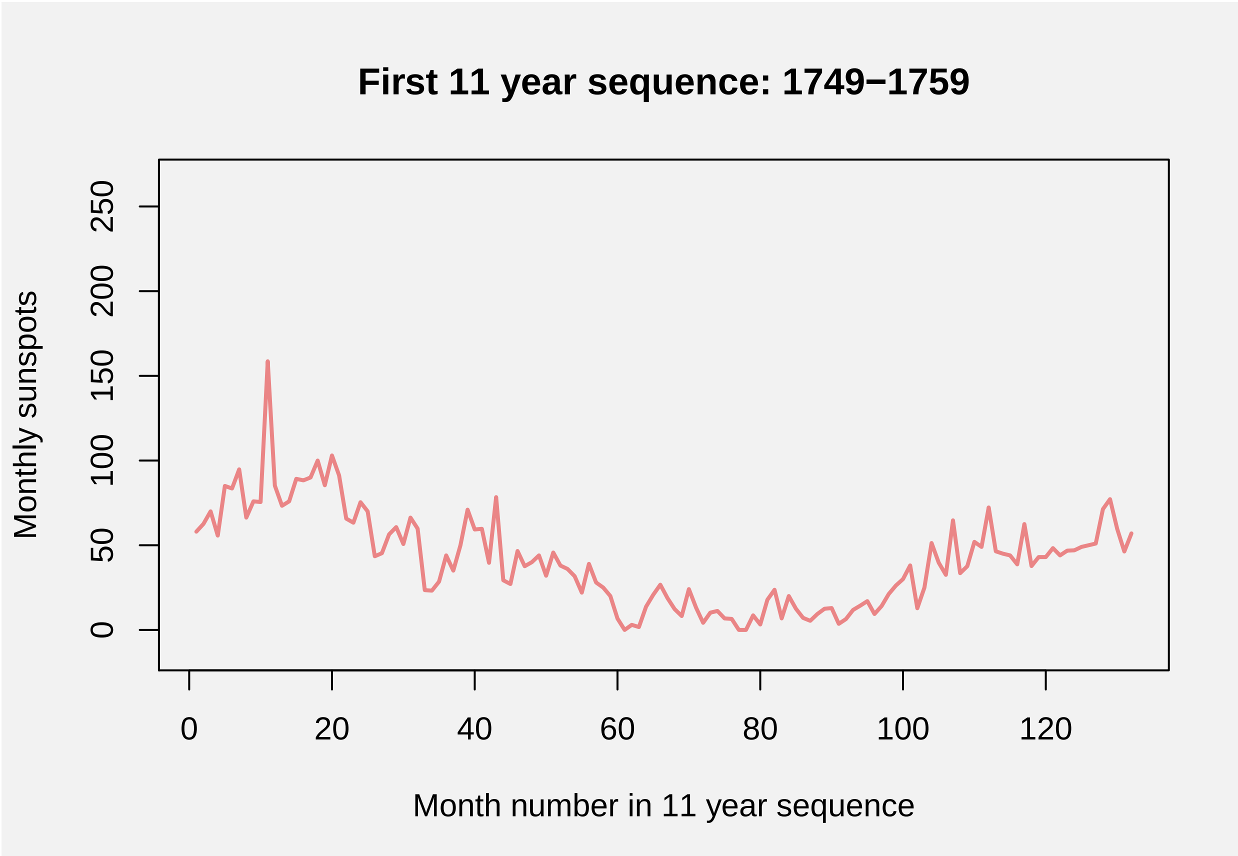



```
par(opt)
```

绘制第一个十一年循环内的详细数据:

```
opt <- par(bg="grey95")
data <- window(timeseries,
               # 第一个循环第一年的一月
               start = c(startYears[1],1),
               # 第二个循环开始前一年的十二月
               end = c(startYears[2]-1, 12),
               frequency = 12) # 月份
plot(1:length(data), data, # 横纵数据
     type = "l", # 折线型
     ylim = extendrange(timeseries), # 略微扩展纵坐标
     col = cols[1], # 从先前设定的调色板中提取第一个颜色
     ylab = "Monthly sunspots",
     xlab = "Month number in 11 year sequence", # 十一年中的月份序数
     lwd = 2,
     # 1749 年到 1759 年中每个月份的太阳黑子活动情况
```

```
     main = "First 11 year sequence: 1749-1759")
```

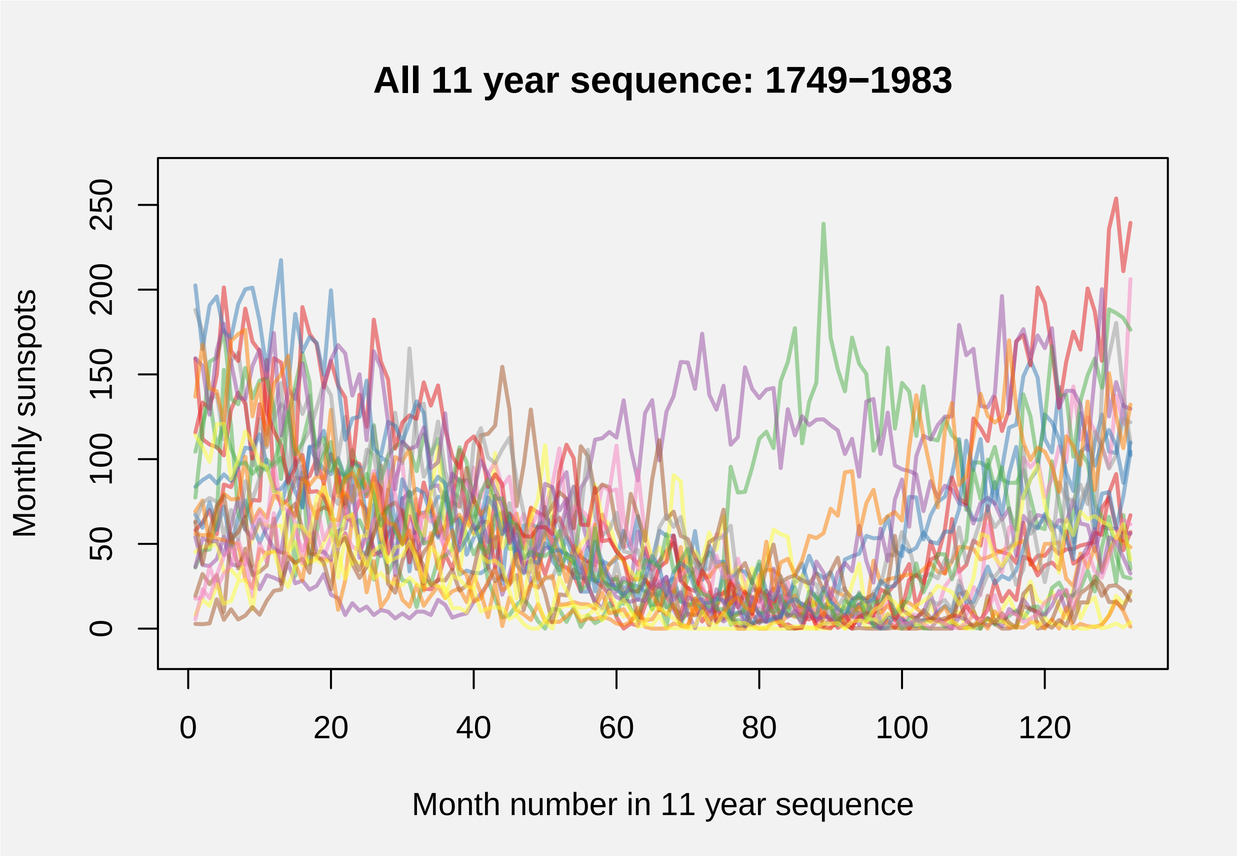



```
par(opt)
```

绘制所有十一年循环内的详细数据:

```
opt <- par(bg="grey95")
# 绘制第一个循环
data <- window(timeseries,
               # 第一个循环第一年的一月
               start = c(startYears[1],1),
               # 第二个循环开始前一年的十二月
               end = c(startYears[2]-1, 12),
               frequency = 12) # 月份
plot(1:length(data), data, # 横纵数据
     type = "l", # 折线型
     ylim = extendrange(timeseries), # 略微扩展纵坐标
     col = cols[1], # 从先前设定的调色板中提取第一个颜色
     ylab = "Monthly sunspots",
```

```
    xlab = "Month number in 11 year sequence", # 十一年中的月份序数
    lwd = 2,
    # 1749 年到 1983 年中每个月份的太阳黑子活动情况
    main = "All 11 year sequence: 1749-1983")

# 绘制第二个到倒数第二个循环
for (i in 2:(ncycles-1)) {
  data <- window(timeseries,
                 start = c(startYears[i],1),
                 end = c(startYears[i+1]-1, 12),
                 frequency =12)
  lines(1:length(data), data, col = cols[i], lwd = 2)
}

# 绘制最后一个循环
data <- window(timeseries,
               start =c(startYears[ncycles],1),
               end = end(timeseries),
               frequency = 12)
lines(1:length(data), data, col = cols[ncycles], lwd = 2)
```

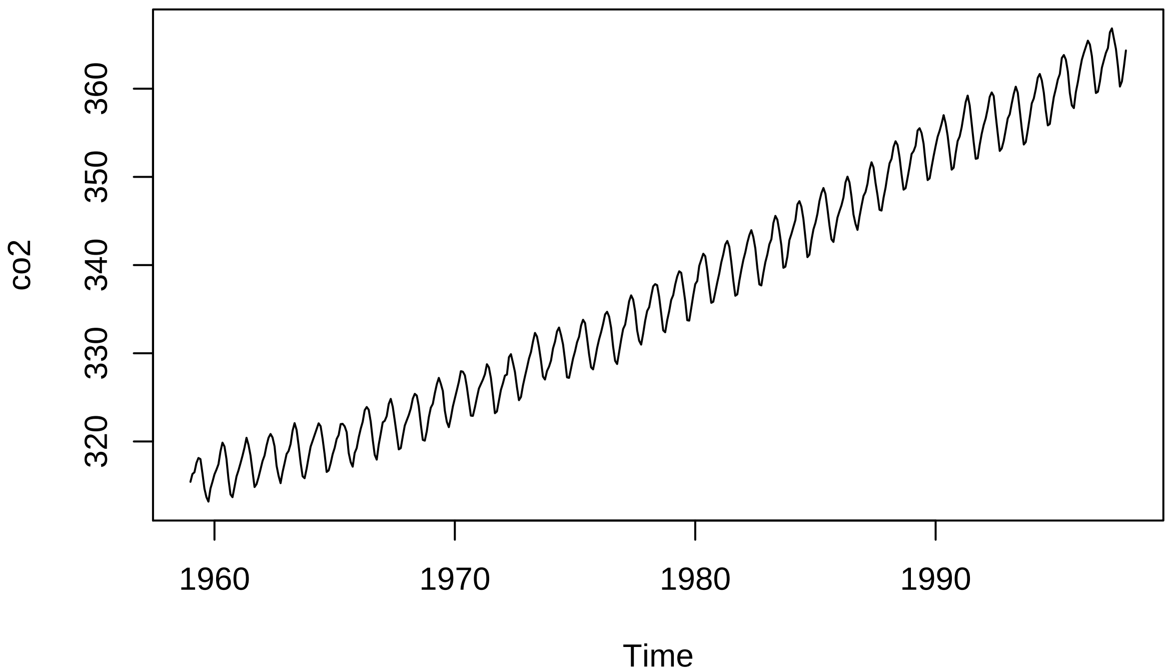



```
par(opt)
```

4.1.2 数据分解

以 R 自带碳排放数据为例:

```
# co2: 从 1959 到 1997 的碳排放记录
plot(co2)
```




```
head(time(co2))
```

```
## [1] 1959.000 1959.083 1959.167 1959.250 1959.333 1959.417
```

```
# 定义一个从时间序列中取得所有年份排列的函数
getYears <- function(ts) {unique(floor(time(ts)))}
# 取得 co2 数据集的年份
getYears(co2)
```

```
##  [1] 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973
## [16] 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988
## [31] 1989 1990 1991 1992 1993 1994 1995 1996 1997
```

```
# 定义一个从时间序列中取得循环序号的函数
getMonthNos <- function(ts) {1:frequency(ts)}
# 取得 co2 数据集的循环序号
getMonthNos(co2)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

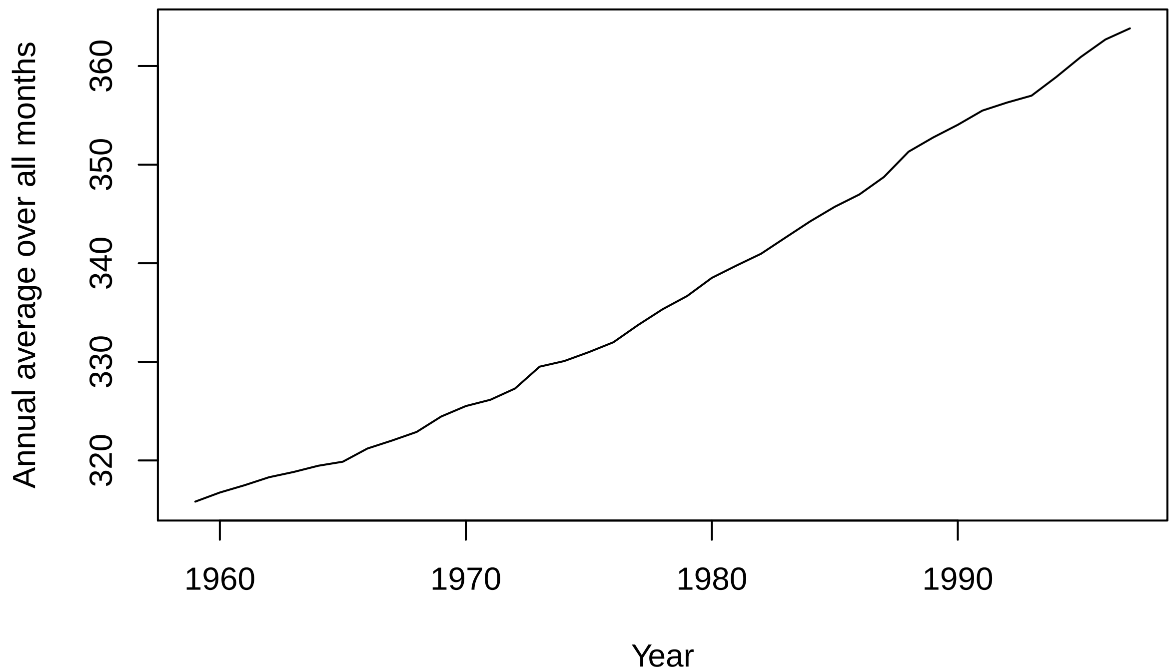
```
# 定义平均函数
getAves <- function(x, by = "year" ){
  years <- getYears(x)
  monthNos <- getMonthNos(x)
  nyears <- length(years)
  nmonths <- length(monthNos)
  if (! (by %in% c("year", "month"))) {
    stop("unknown value for by = ",
         by,
         "; by must be one of {\"year\", \"month\"}")
  }
  if(by == "year"){
    aves <- sapply(1:nyears, FUN=function(i) {
      mean(window(x,
                  start = c(years[i], monthNos[1]),
                  end = c(years[i], monthNos[nmonths])))})
    aves <- data.frame(aves = aves, row.names = years)
  } else {
    aves <- sapply(1:nmonths,
                   FUN=function(i) { mean(window(x,
                                                  start = c(years[1], monthNos[i]),
                                                  end = c(years[nyears], monthNos[i]),
                                                  frequency=1))})
    aves <- data.frame(aves = aves, row.names = month.abb[monthNos])
  }
  aves
}
```

```
# 取得每年平均 co2 水平
t(head(getAves(co2, by = "year"), n = 3))
```

```
##          1959     1960    1961
## aves 315.8258 316.7475 317.485
```

绘制年均曲线:

```
plot(getYears(co2), getAves(co2, by = "year")$aves,
     type = "l",
     xlab = "Year",
     ylab = "Annual average over all months")
```

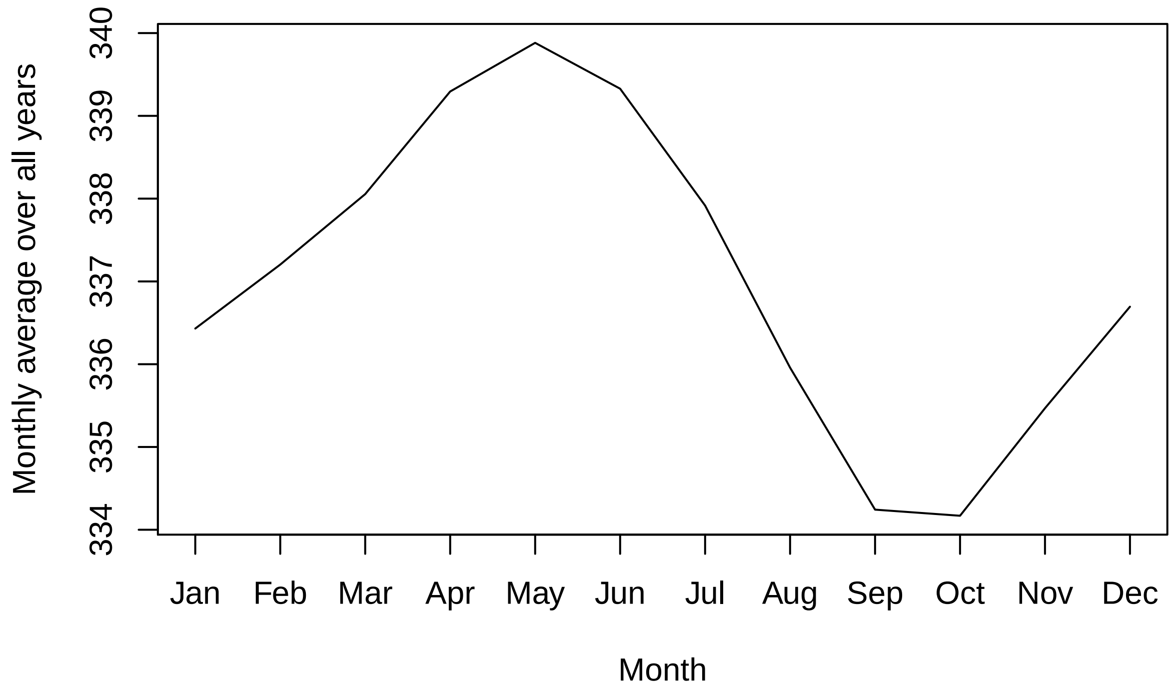



```
# 取得每月平均 co2 水平
t(head(getAves(co2, by = "month"), n = Inf)) # Inf 列举所有数据
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## aves 336.4308 337.2033 338.0546 339.2944 339.8821 339.3282 337.9164 335.9579
##           Sep      Oct      Nov      Dec
## aves 334.2428 334.1692 335.4679 336.6946
```

绘制月均曲线:

```
plot(getAves(co2, by = "month")$aves, type="l",
     xlab="Month",
     xaxt="n", # 给横轴坐标添加名字
     ylab="Monthly average over all years")
# 添加横坐标名字
axis(1, at=getMonthNos(co2), labels=month.abb)
```

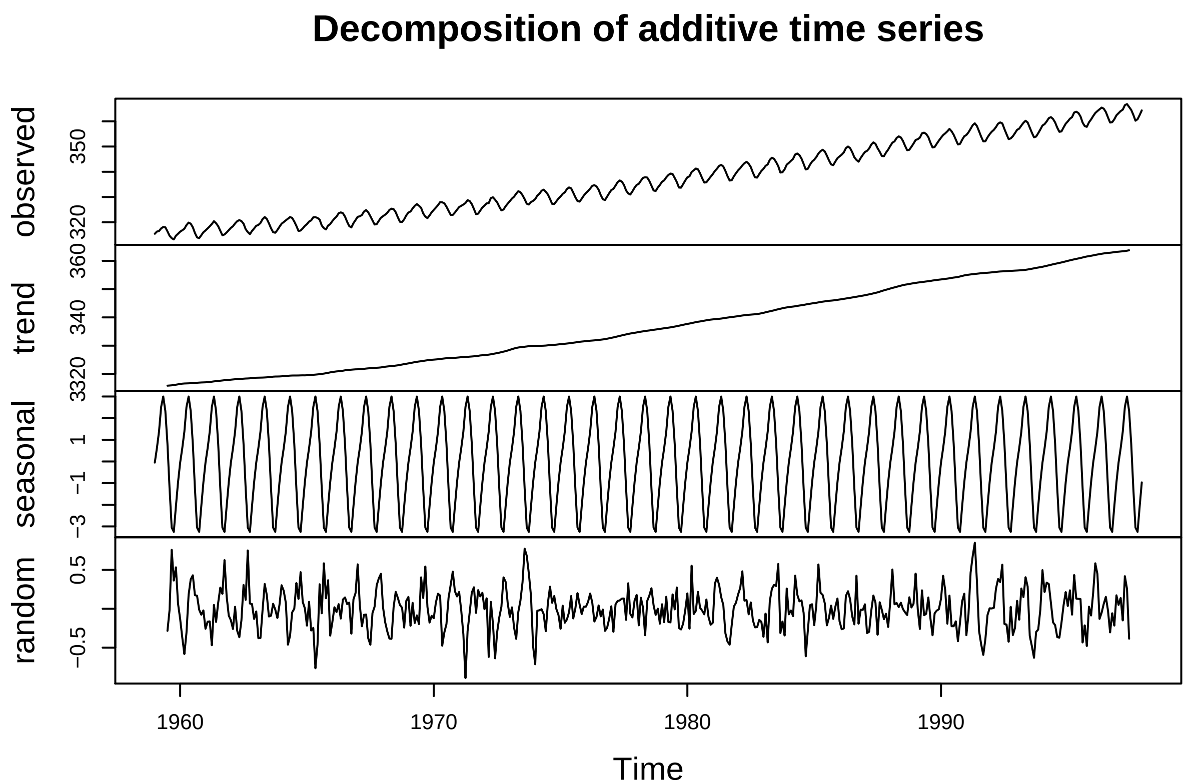


分解有潜在规律的时间序列:

```
# 分解 co2 时间序列
co2_decomp <- decompose(co2)
# 列出所含信息
str(co2_decomp)
```

```
## List of 6
##  $ x       : Time-Series [1:468] from 1959 to 1998: 315 316 316 318 318 ...
##  $ seasonal: Time-Series [1:468] from 1959 to 1998: -0.0536 0.6106 1.3756 2.5168 3.0003 ...
##  $ trend   : Time-Series [1:468] from 1959 to 1998: NA NA NA NA NA ...
##  $ random  : Time-Series [1:468] from 1959 to 1998: NA NA NA NA NA ...
##  $ figure  : num [1:12] -0.0536 0.6106 1.3756 2.5168 3.0003 ...
##  $ type    : chr "additive"
##  - attr(*, "class")= chr "decomposed.ts"
```

```
# 绘制图像
plot(co2_decomp)
```

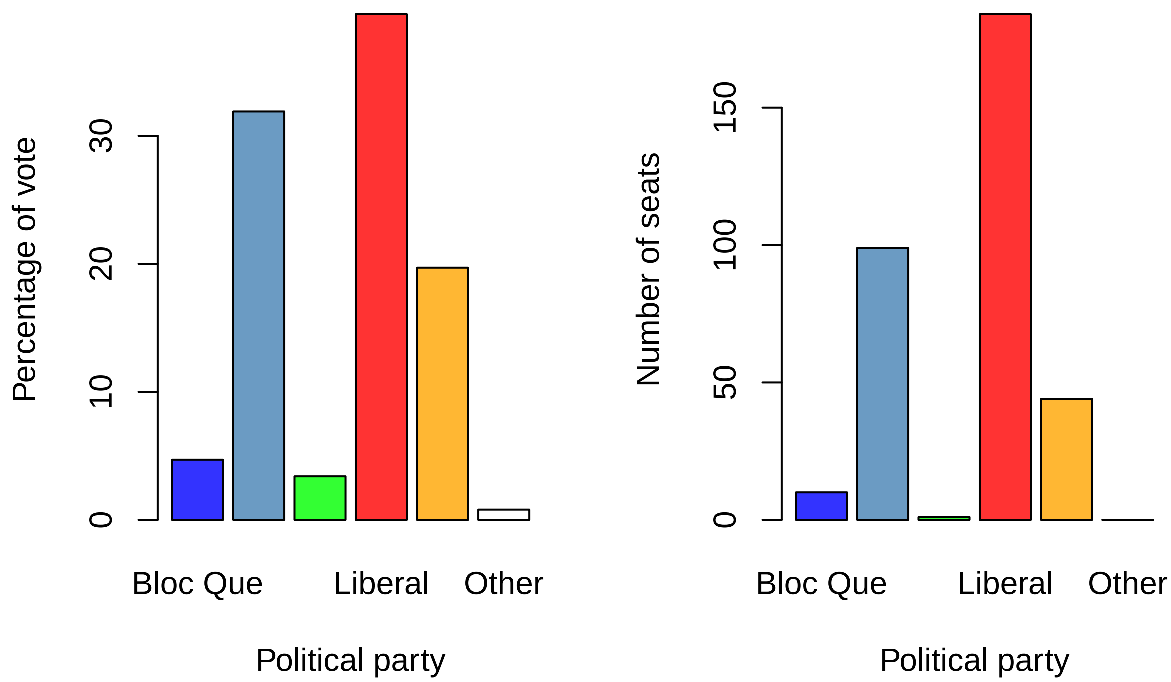

Decomposition of additive time series
observed
320
350
trend
320
340
360
seasonal
−3
−1
1
3
random
−0.5
0.5
1960
1970
1980
1990
Time

4.2 分类数据

4.2.1 条形图和饼状图

条形图绘制:

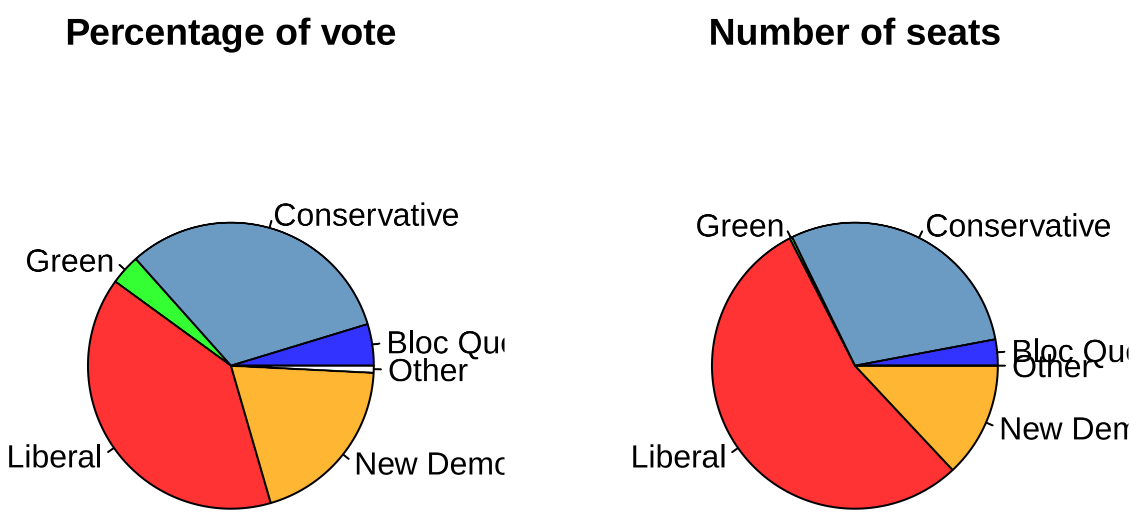
```
parties <- c("Bloc Que", "Conservative", "Green", "Liberal", "New Democrats", "Other")
votes2015 <- c(4.7, 31.9, 3.4, 39.5, 19.7, 0.8)
seats2015 <- c(10, 99, 1, 184, 44, 0)
cols <- adjustcolor(c("blue", "steelblue", "green",
                      "red","orange", "white"), alpha.f = 0.8)
savePar <- par(mfrow = c(1,2))
barplot(votes2015,
        ylab = "Percentage of vote",
        col = cols,
        names.arg = parties,
        xlab = "Political party")
barplot(seats2015,
        ylab = "Number of seats",
        col = cols,
        names.arg = parties,
        xlab = "Political party")
```




```
par(savePar)
```

饼状图绘制:

```
savePar <- par(mfrow=c(1,2))
pie(votes2015,
    main="Percentage of vote",
    col=cols,
    labels=parties)
pie(seats2015,
    main="Number of seats",
    col=cols,
    labels=parties)
```




```r
par(savePar)
```

4.2.2 树形图（Treemap）

适用于嵌套数据关系。

```r
library(loon) # 以 loon 包中橄榄油生产数据集为例子
```

```
## Loading required package: tcltk

## loon Version 1.3.4.
## To learn more about loon, see l_web().

##
## Attaching package: 'loon'

## The following object is masked from 'package:graph':
##
##     complement
```

```
head(olive, n =5) # 查看头部信息
```

```
##   Region          Area palmitic palmitoleic stearic oleic linoleic linolenic
## 1  South North-Apulia     1075          75     226  7823      672        36
## 2  South North-Apulia     1088          73     224  7709      781        31
## 3  South North-Apulia      911          54     246  8113      549        31
## 4  South North-Apulia      966          57     240  7952      619        50
## 5  South North-Apulia     1051          67     259  7771      672        50
##   arachidic eicosenoic
## 1        60         29
## 2        61         29
## 3        63         29
## 4        78         35
## 5        80         46
```

```
# 提取产地信息
areas <- unique(olive[,"Area"])
areas
```

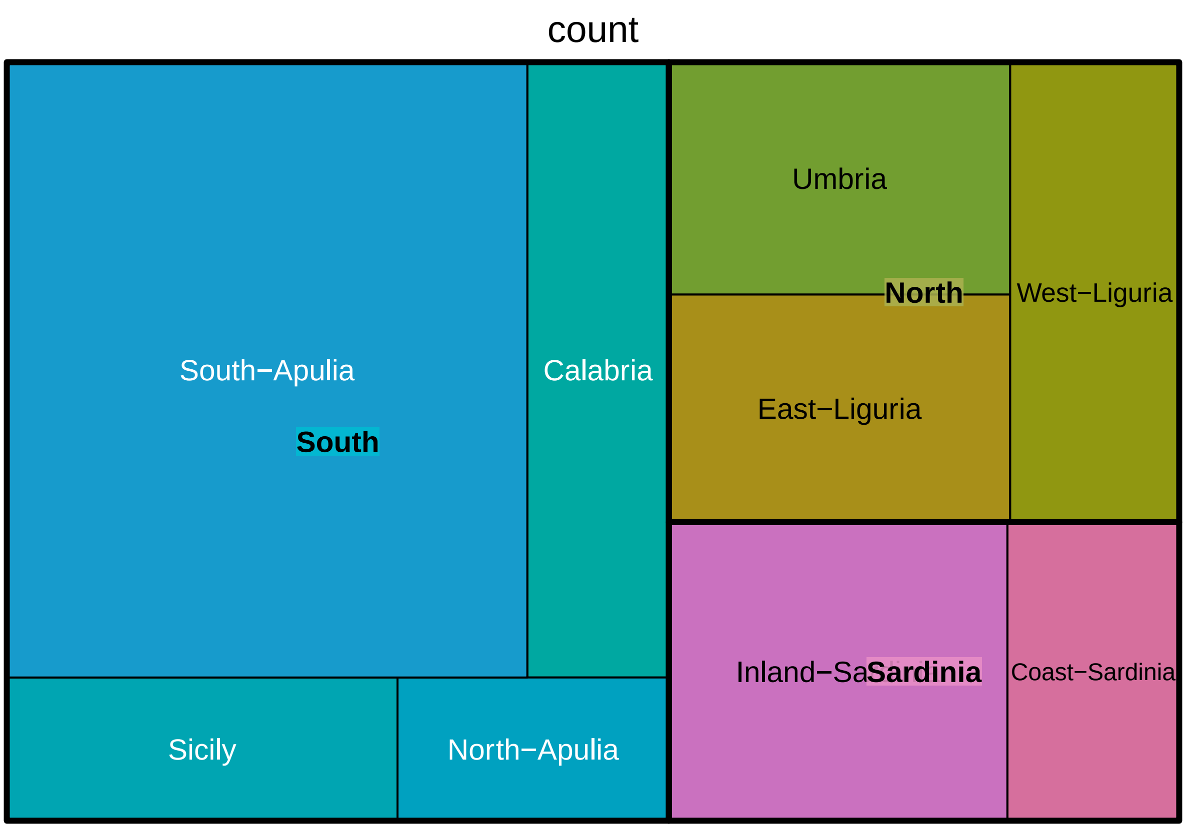
```
## [1] North-Apulia    Calabria        South-Apulia    Sicily
## [5] Inland-Sardinia Coast-Sardinia  Umbria          East-Liguria
## [9] West-Liguria
## 9 Levels: North-Apulia Calabria South-Apulia Sicily ... Umbria
```

```
nAreas <- length(areas) # 产地数目
counts <- rep(0, nAreas) # 初始化计数序列
regions <- c() # 初始化产地的区域 (South, Sardinia, North)
for (i in 1:nAreas) {
  counts[i] <- sum(olive[,"Area"] == areas[i])
  regions <- c(regions, unique(olive[olive[,"Area"]==areas[i],"Region"]))
}
regions <- as.factor(levels(olive[,"Region"])[regions])
# 创建定制数据集
oliveOilCounts <- data.frame(Region = regions, Area = areas, count = counts)
oliveOilCounts
```

```
##     Region            Area count
## 1    South    North-Apulia    25
## 2    South        Calabria    56
```

```
## 3    South    South-Apulia   206
## 4    South         Sicily    36
## 5 Sardinia Inland-Sardinia    65
## 6 Sardinia  Coast-Sardinia    33
## 7    North         Umbria    51
## 8    North    East-Liguria    50
## 9    North    West-Liguria    50
```

```
# install.packages("treemap")
# 加载 treemap 包
library(treemap)
treemap(oliveOilCounts, index=c("Region", "Area"), vSize = "count") # vsize 视觉大小依据
```



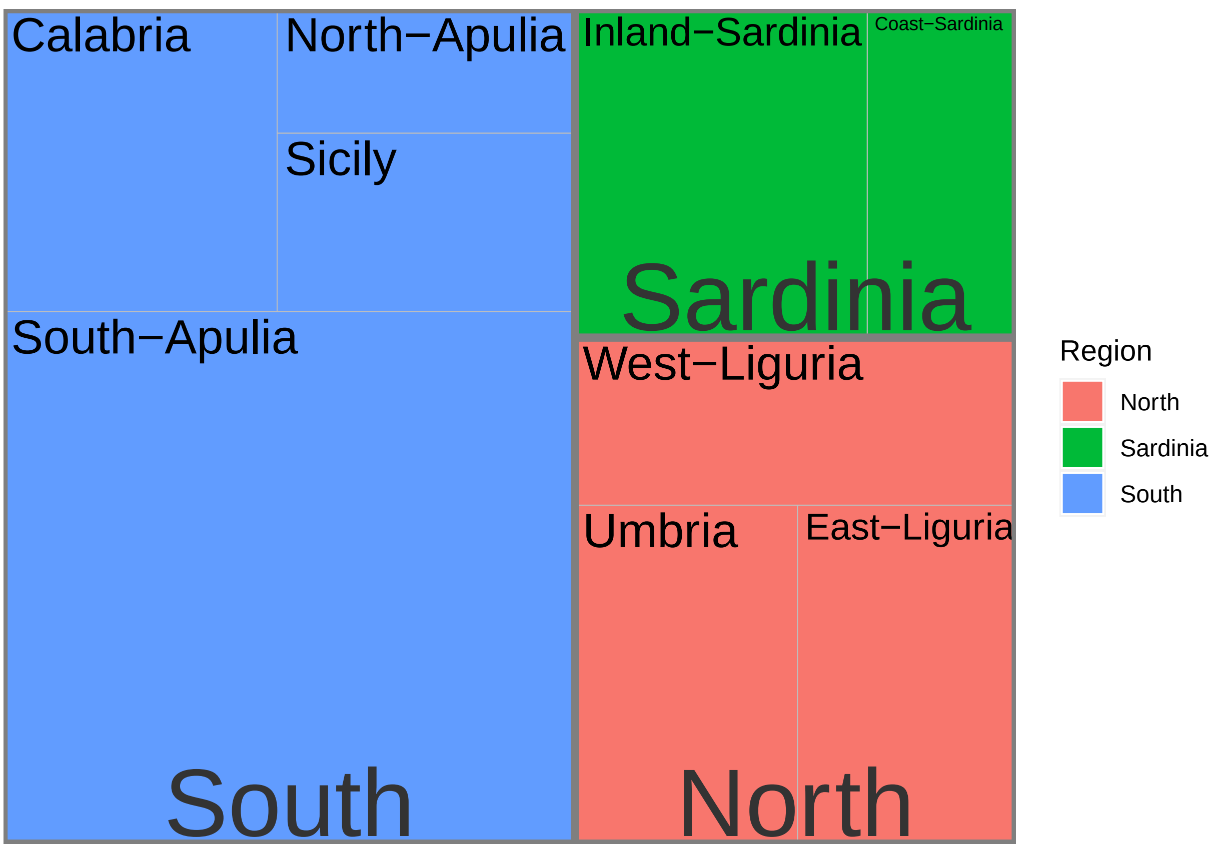
或者迭代动态绘图：

```
# itreemap(oliveOilCounts, index=c("Region"), vSize = "count")
```

使用 ggplot2 绘制：

```r
# 安装加载依赖包
# install.packages("treemapify")
library(ggplot2)
library(treemapify)
```

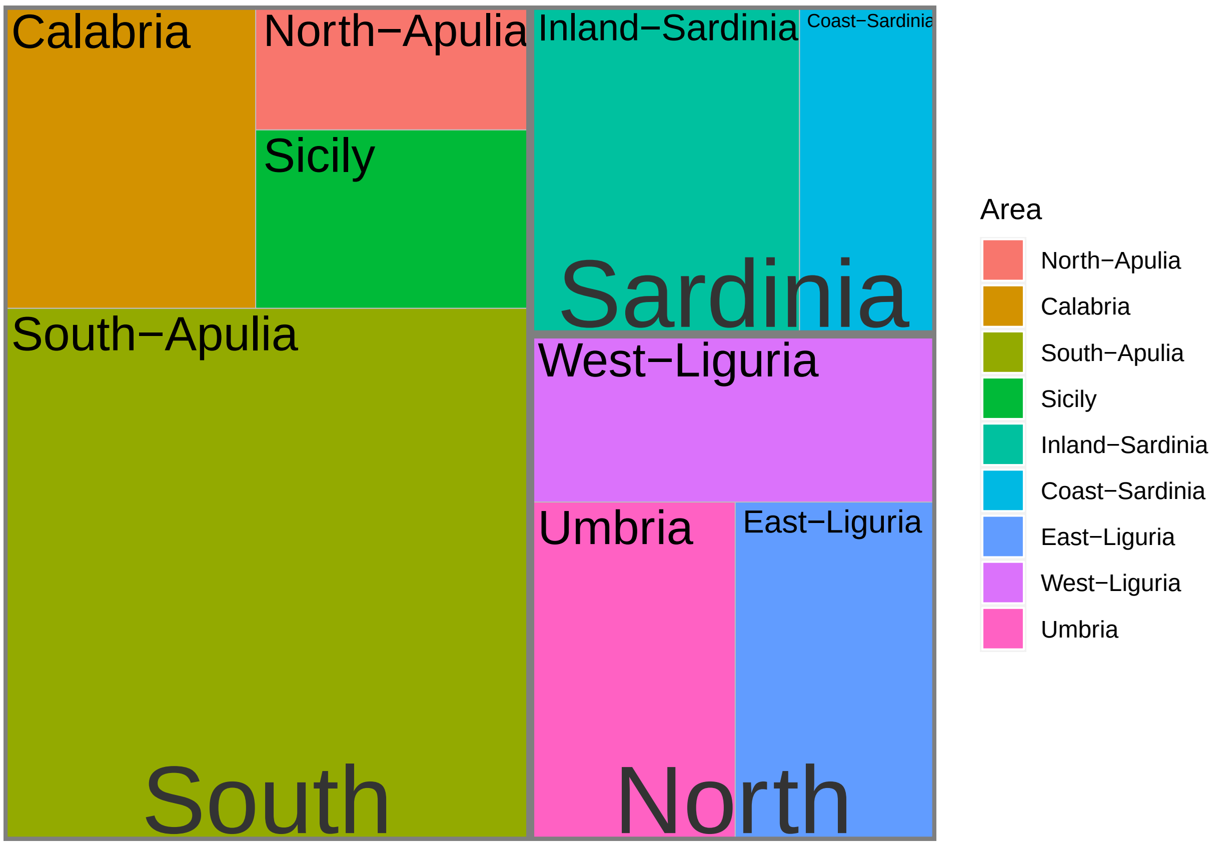
```r
treeMapPlot <- ggplot(oliveOilCounts, aes(area = count,
                                          fill = Region,
                                          label = Area,
                                          subgroup = Region))
treeMapPlot + geom_treemap() + geom_treemap_text() +
  geom_treemap_subgroup_border() + geom_treemap_subgroup_text()
```



对地域进行颜色细分:

```r
treeMapPlot <- ggplot(oliveOilCounts, aes(area = count,
                                          fill = Area, # 代码变化
                                          label = Area,
                                          subgroup = Region))
treeMapPlot + geom_treemap() + geom_treemap_text() +
```

```
geom_treemap_subgroup_border() + geom_treemap_subgroup_text()
```



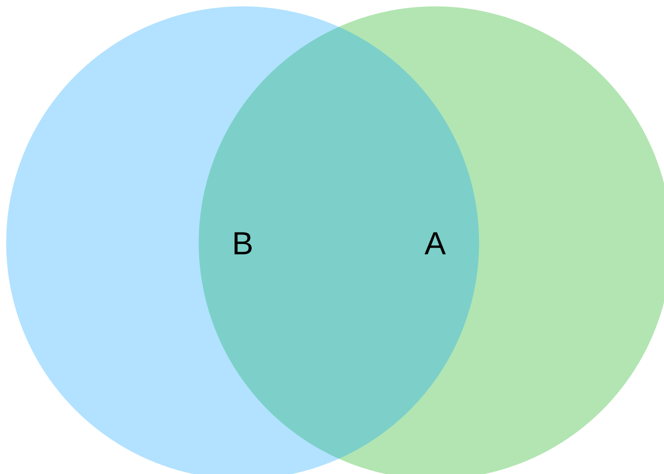
4.2.3 Venn 图

适用于非嵌套数据关系。

```
# 安装加载所需依赖包
# install.packages("venneuler")
# 如提示 error: JVM could not be found, 下载最新 java 即可
library(venneuler)
```

```
## Loading required package: rJava
```

```
# 如下三块位置面积相等
ve <- venneuler(c(A = 1, B = 1, "A&B" = 1))
plot(ve)
```



4.2.4 Eikosogram, Mosaic & Spine

适用于交叉数据展现条件概率关系。

```
# 使用 R 语言自带 HairEyeColor 数据集为例
HairEyeColor
```

4.2.4.1 二分类

```
## , , Sex = Male
##
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    32   11    10     3
##   Brown    53   50    25    15
##   Red      10   10     7     7
##   Blond     3   30     5     8
```

```
##
## , , Sex = Female
##
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    36    9     5     2
##   Brown    66   34    29    14
##   Red      16    7     7     7
##   Blond     4   64     5     8
```

```
# 忽略性别差异
HairEye <- apply(HairEyeColor,1:2 ,sum)
knitr::kable(HairEye)
```

	Brown	Blue	Hazel	Green
Black	68	20	15	5
Brown	119	84	54	29
Red	26	17	14	14
Blond	7	94	10	16

```
# 安装依赖包
# install.packages("eikosograms")
library(eikosograms)
```

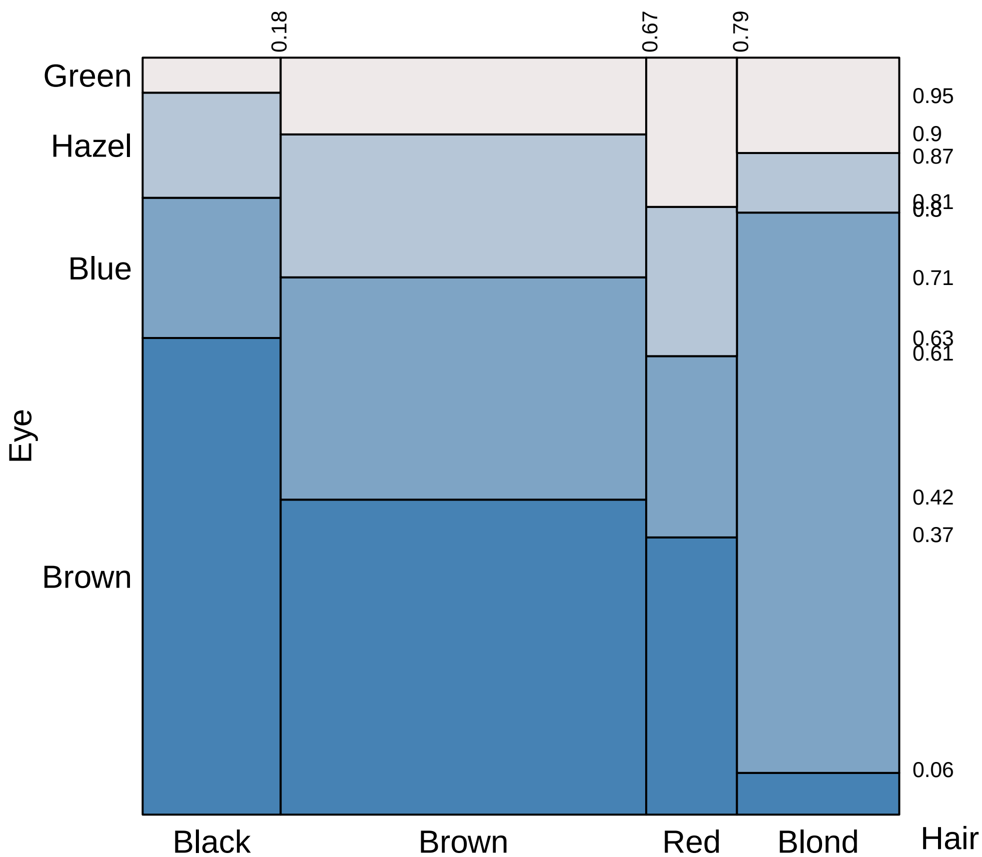
```
rownames(HairEye) # 头发颜色
```

```
## [1] "Black" "Brown" "Red"   "Blond"
```

```
colnames(HairEye) # 眼睛颜色
```

```
## [1] "Brown" "Blue"  "Hazel" "Green"
```

```
# 两个 TRUE 给出概率关系
eikos(x = "Hair", y = "Eye", data=HairEye, xaxs=TRUE, yaxs=TRUE)
```

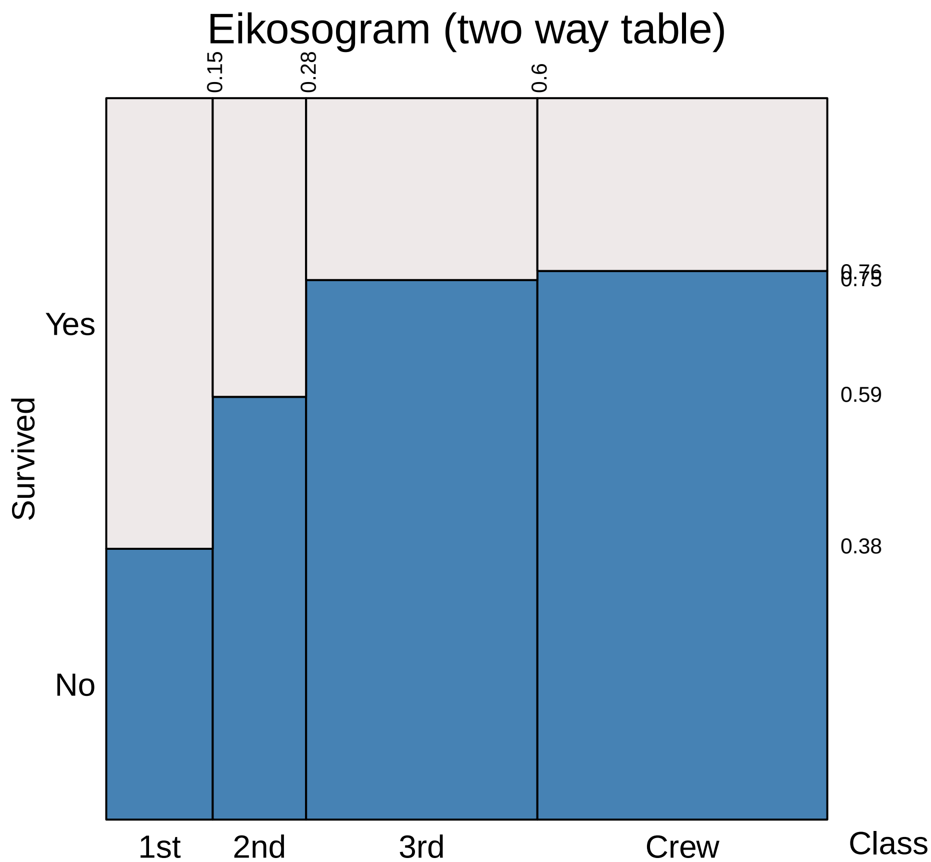


重新回顾泰坦尼克号数据集:

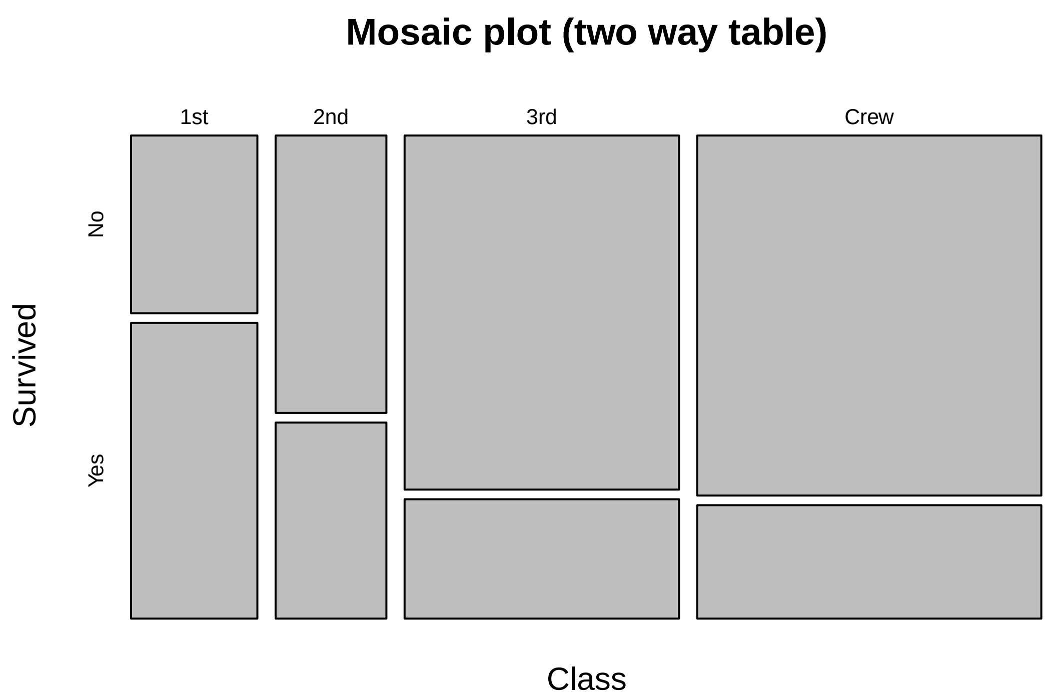
```
# 对已有数据进行重新排列（生存与否 vs. 舱位）并赋值
TitanicSurvClass <- margin.table(Titanic, margin=c(1,4))
TitanicSurvClass
```

```
##       Survived
## Class   No Yes
##   1st  122 203
##   2nd  167 118
##   3rd  528 178
##   Crew 673 212
```

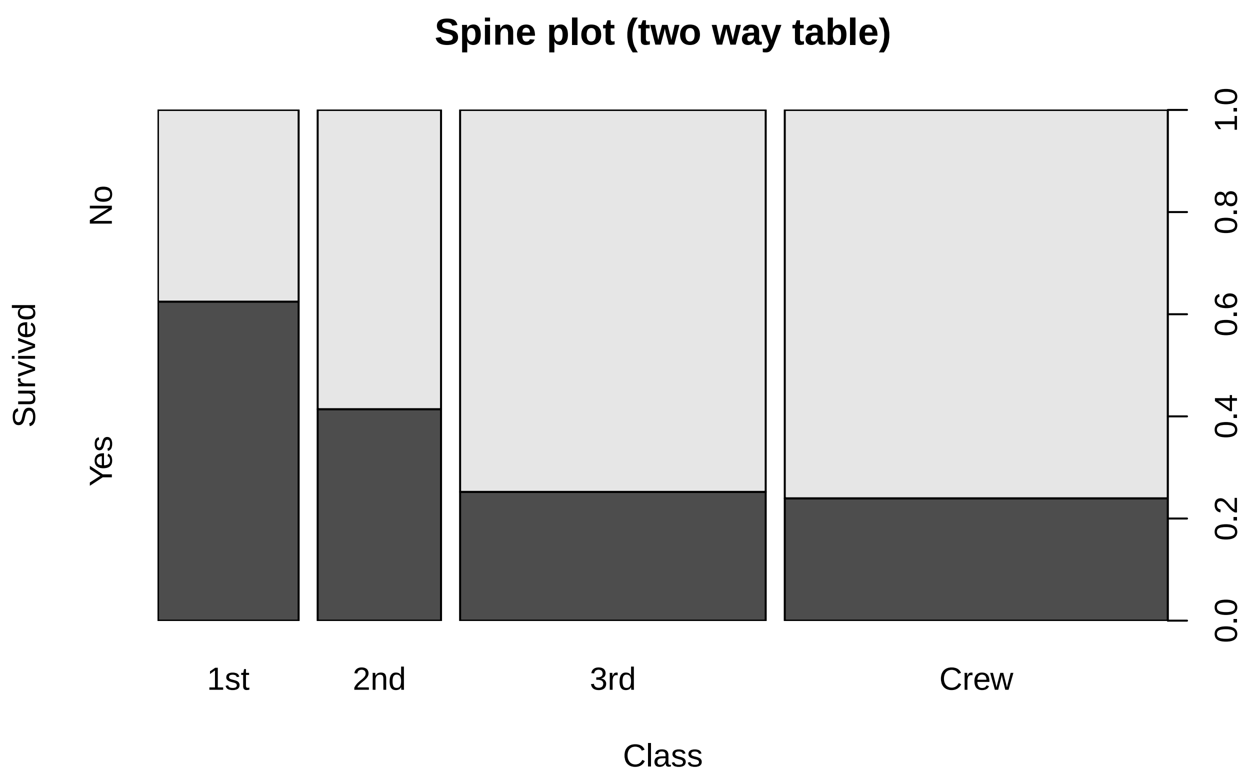
```
eikos(y = "Survived", x = "Class", data = Titanic, main = "Eikosogram (two way table)")
```




```
mosaicplot(TitanicSurvClass, main = "Mosaic plot (two way table)")
```

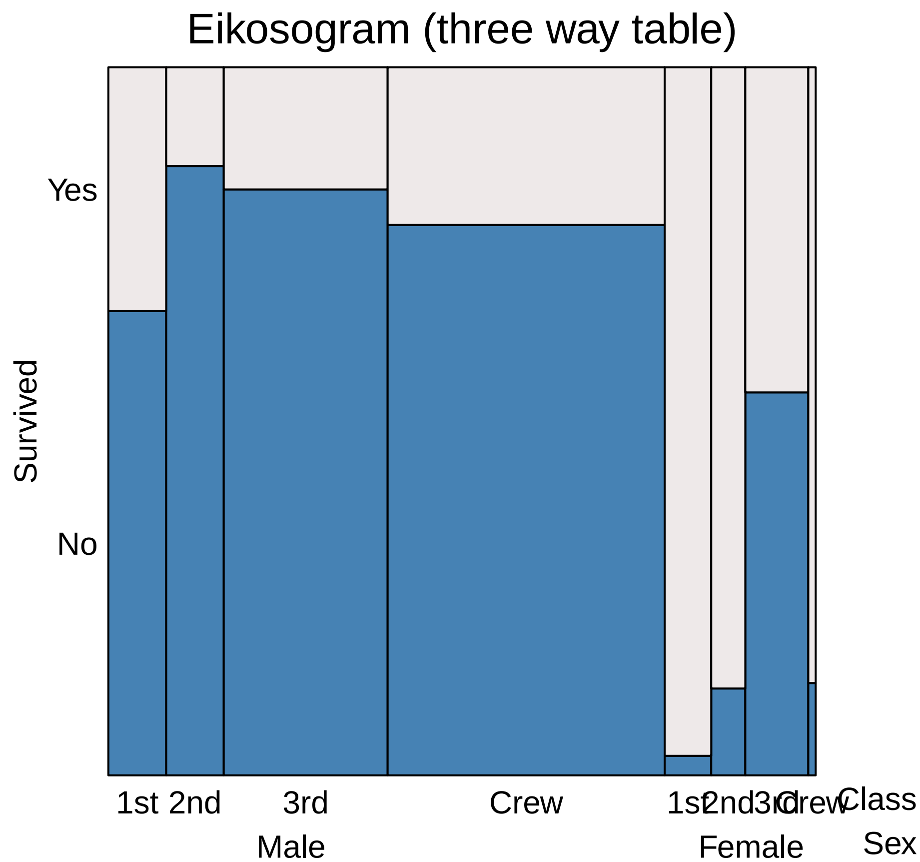



```
spineplot(TitanicSurvClass, main = "Spine plot (two way table)")
```



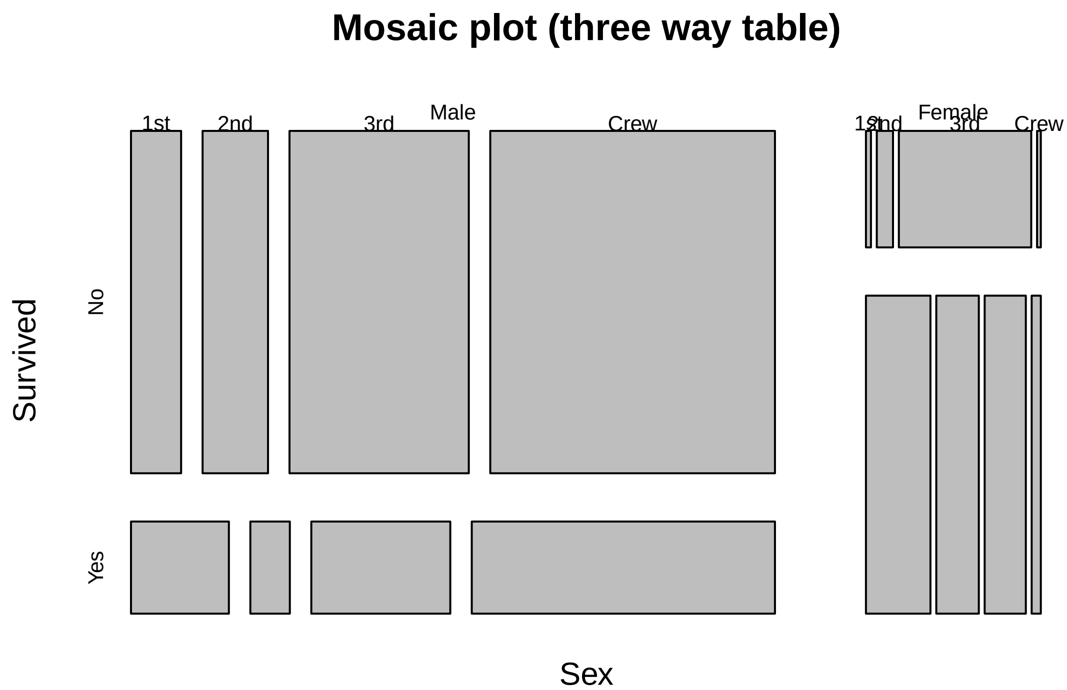
4.2.4.2 多分类 Eikosogram:

```
eikos(y = "Survived", # 纵坐标根据生还情况分类
      x = c("Class", "Sex"), # 横坐标根据舱位和性别分类
      data=Titanic, # 泰坦尼克数据集
      xaxs=FALSE, # 不显示概率
      yaxs=FALSE, # 不显示概率
      main="Eikosogram (three way table)") # 三分类
```



Mosaic plot：

```
TitanicSurvClassSex <- margin.table(Titanic,c(2,4,1)) # # 三层结构
mosaicplot(TitanicSurvClassSex,
           main="Mosaic plot (three way table)")
```


Mosaic plot (three way table)
Male
Female
1st
2nd
3rd
Crew
Survived
No
Yes
Sex

4.3 地图数据

4.3.1 一般地图

绘制世界地图：

```
# install.packages("ggplot2")
# install.packages("dplyr")
# install.packages("maps")
# install.packages("viridis")
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:graph':
##
##     union

## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(maps)
library(viridis)
```

```
## Loading required package: viridisLite
```

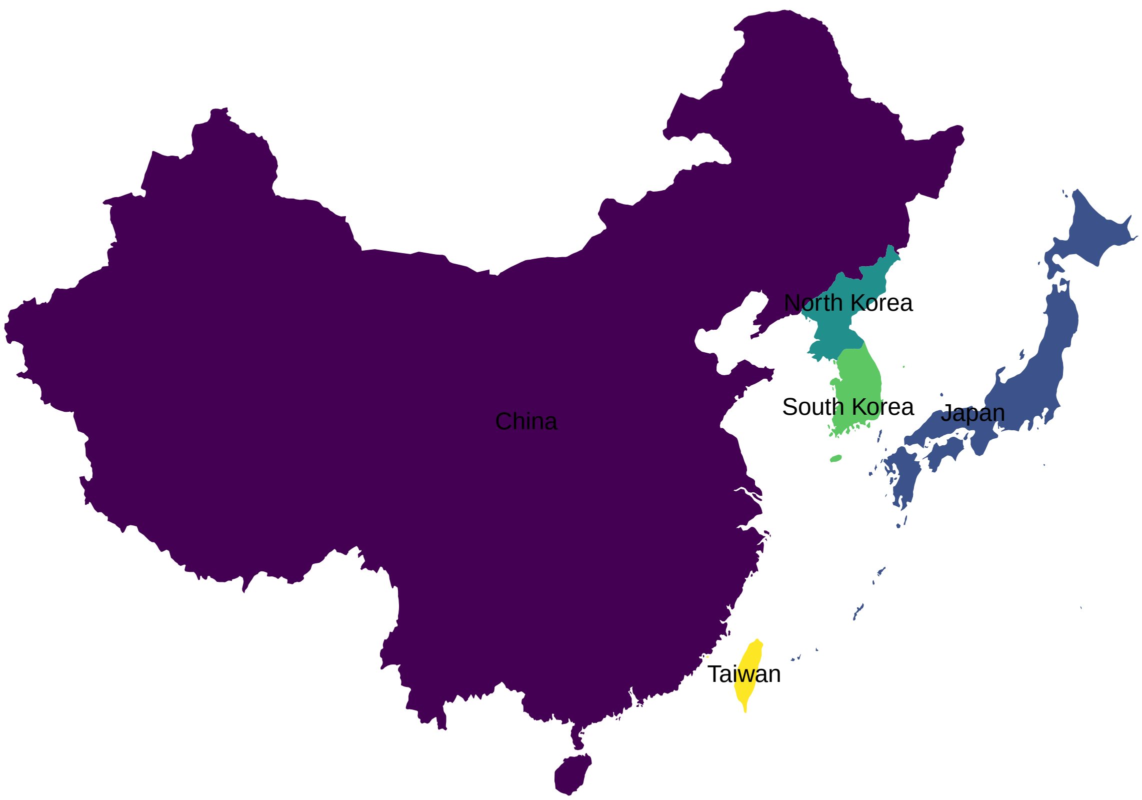
```
theme_set(
  theme_void()
  )
```

```
# 绘制世界地图
world_map <- map_data("world")
ggplot(world_map, aes(x = long, y = lat, group = group)) +
  geom_polygon(fill="lightgray", colour = "white")
```


绘制特定地图:

```
# 选取特定国家和区域
some.asia.countries <- c("China", "Japan", "North Korea", "South Korea", "Taiwan",
                         "Hongkong")
# 从地图集中取出所需地图信息
some.asia.maps <- map_data("world", region = some.asia.countries)
# 计算经纬度
region.lab.data <- some.asia.maps %>%
  group_by(region) %>%
  summarise(long = mean(long), lat = mean(lat))

# 绘图
ggplot(some.asia.maps, aes(x = long, y = lat)) +
```

```
  geom_polygon(aes( group = group, fill = region))+
  geom_text(aes(label = region), data = region.lab.data,  size = 3, hjust = 0.5)+
  scale_fill_viridis_d()+
  theme_void()+
  theme(legend.position = "none")
```



4.3.2 数据地图

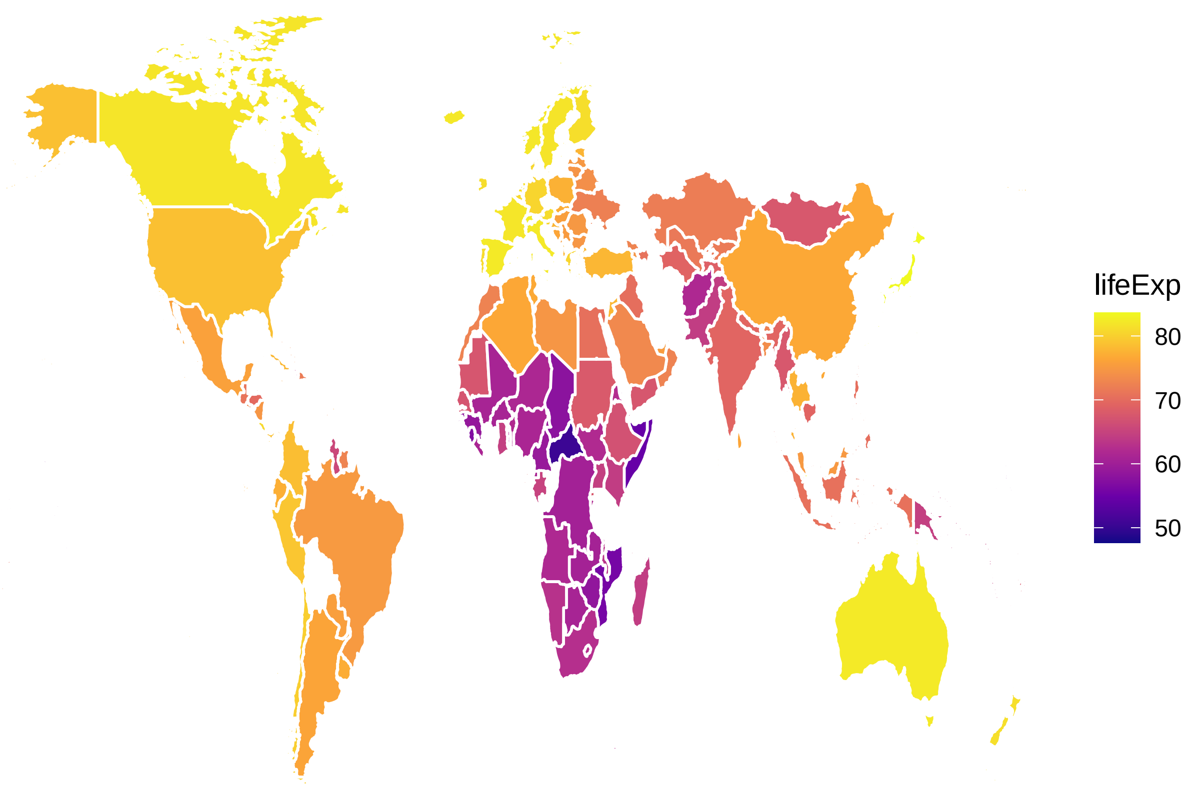
绘制全球寿命地图：

```
# install.packages("WHO")
# 或者到 https://cran.r-project.org/src/contrib/Archive/WHO/手动下载安装
library("WHO")
life.exp <- get_data("WHOSIS_000001")
```

```
## Warning: `as_data_frame()` was deprecated in tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
```

```
life.exp <- life.exp %>%
  filter(year == 2015 & sex == "Both sexes") %>%  # 选择 2015 数据和两个性别
  select(country, value) %>%                      # 选择感兴趣的列
  rename(region = country, lifeExp = value) %>%   # 重命名
  mutate(
    region = ifelse(region == "United States of America", "USA", region)
    )
```

```
world_map <- map_data("world")
life.exp.map <- left_join(life.exp, world_map, by = "region")

ggplot(life.exp.map, aes(map_id = region, fill = lifeExp))+
  geom_map(map = life.exp.map,  color = "white")+
  expand_limits(x = life.exp.map$long, y = life.exp.map$lat)+
  scale_fill_viridis_c(option = "C")
```



绘制美国各州人均 gdp 地图：

```
# install.packages("raster")
gdp <- read.csv("gdp-by-state.csv", header = TRUE)
gdp$region <- tolower(gdp$region)
head(gdp)
```

```
##        region   gdp
## 1     alabama 37508
## 2      alaska 63610
## 3     arizona 39583
## 4    arkansas 36714
## 5 california 60359
## 6    colorado 54026
```

```
# provinces <- getData(country="Canada", level=1)
states_map <- map_data("state")
wealth_map <- left_join(states_map, gdp, by = "region")
```

```
# ggplot(wealth_map, aes(long, lat, group = group) +
#   geom_polygon(aes(fill = gdp), color = "white") +
#   scale_fill_viridis_c(option = "C")
```

5 三维数据

6 更高维度数据

7 Java 封装及调用

8 Python 封装、调用及类似实现方法