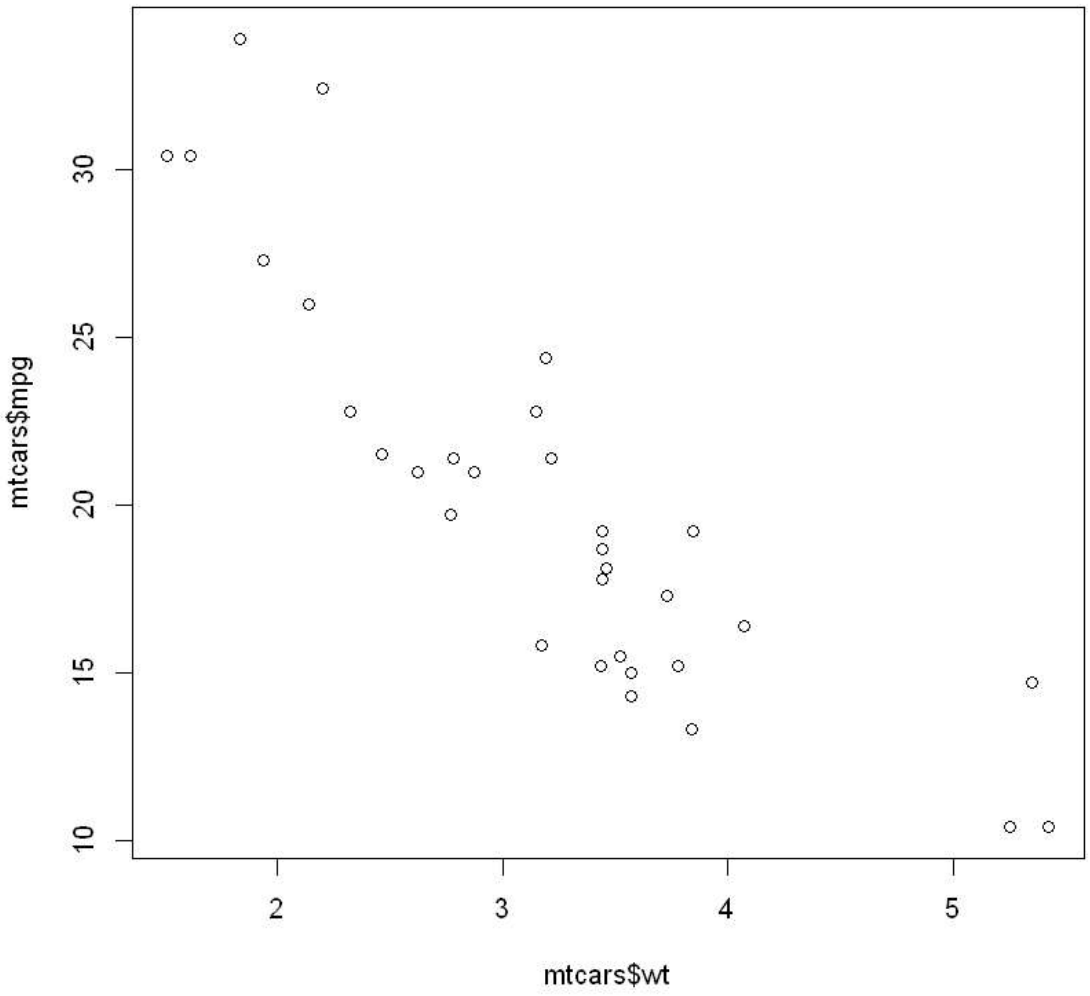
读书笔记

绘制散点图

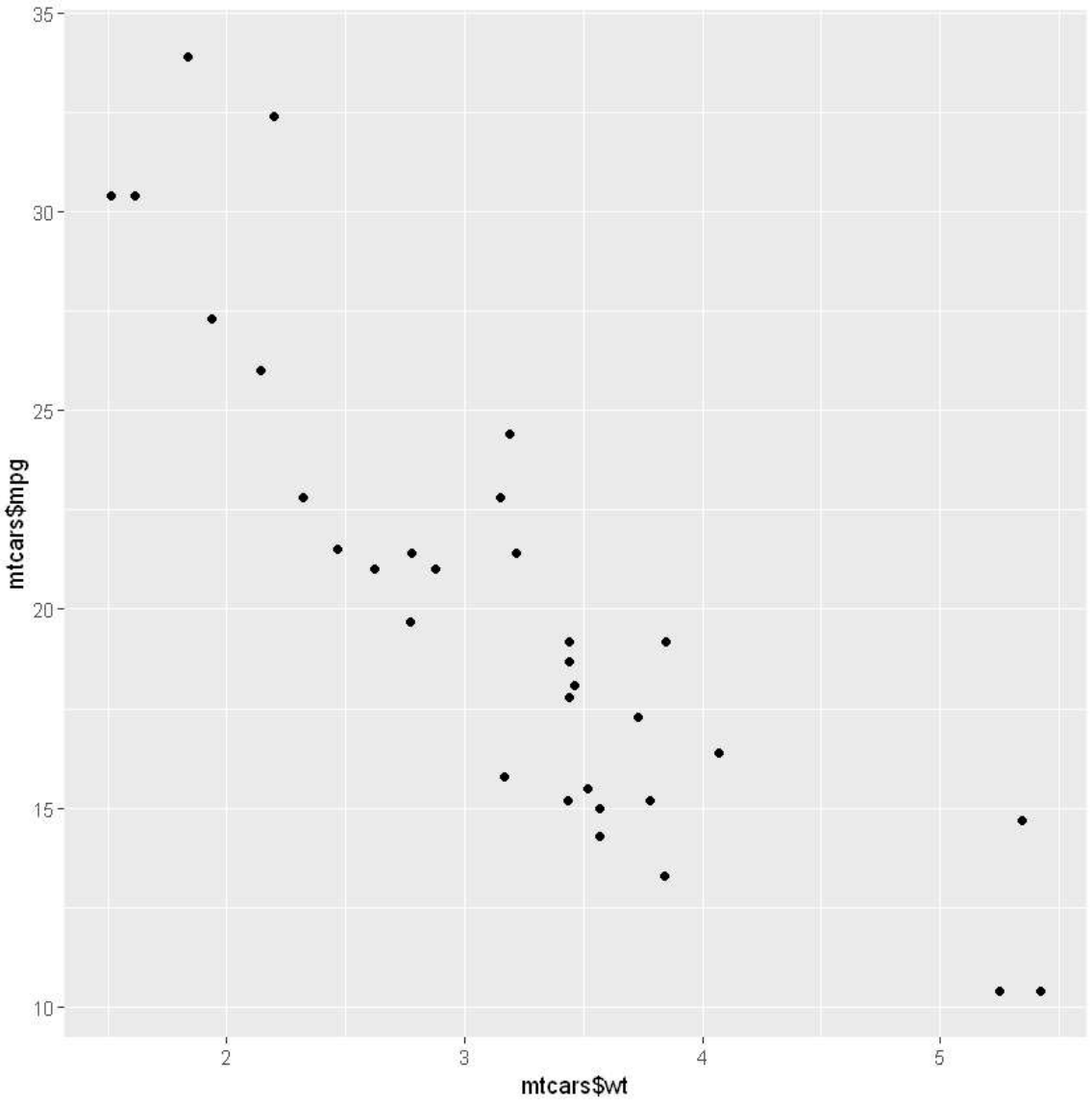
In [1]:

```
plot(mtcars$wt, mtcars$mpg)
```



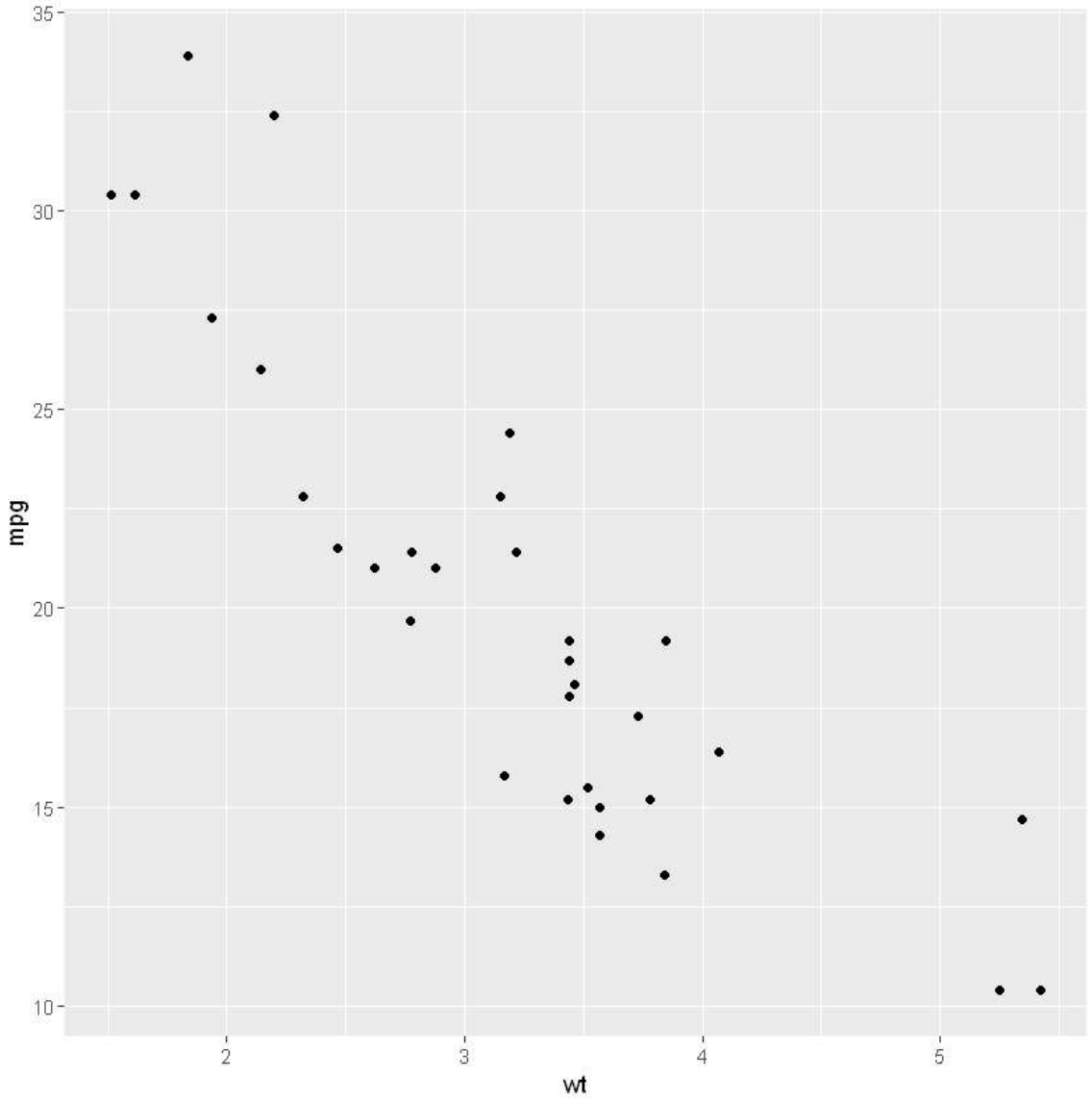
In [3]:

```
library(ggplot2)
qplot(mtcars$wt, mtcars$mpg) #用qplot
#qplot(wt, mpg, data = mtcars) 或者用这种方式
```



In [4]:

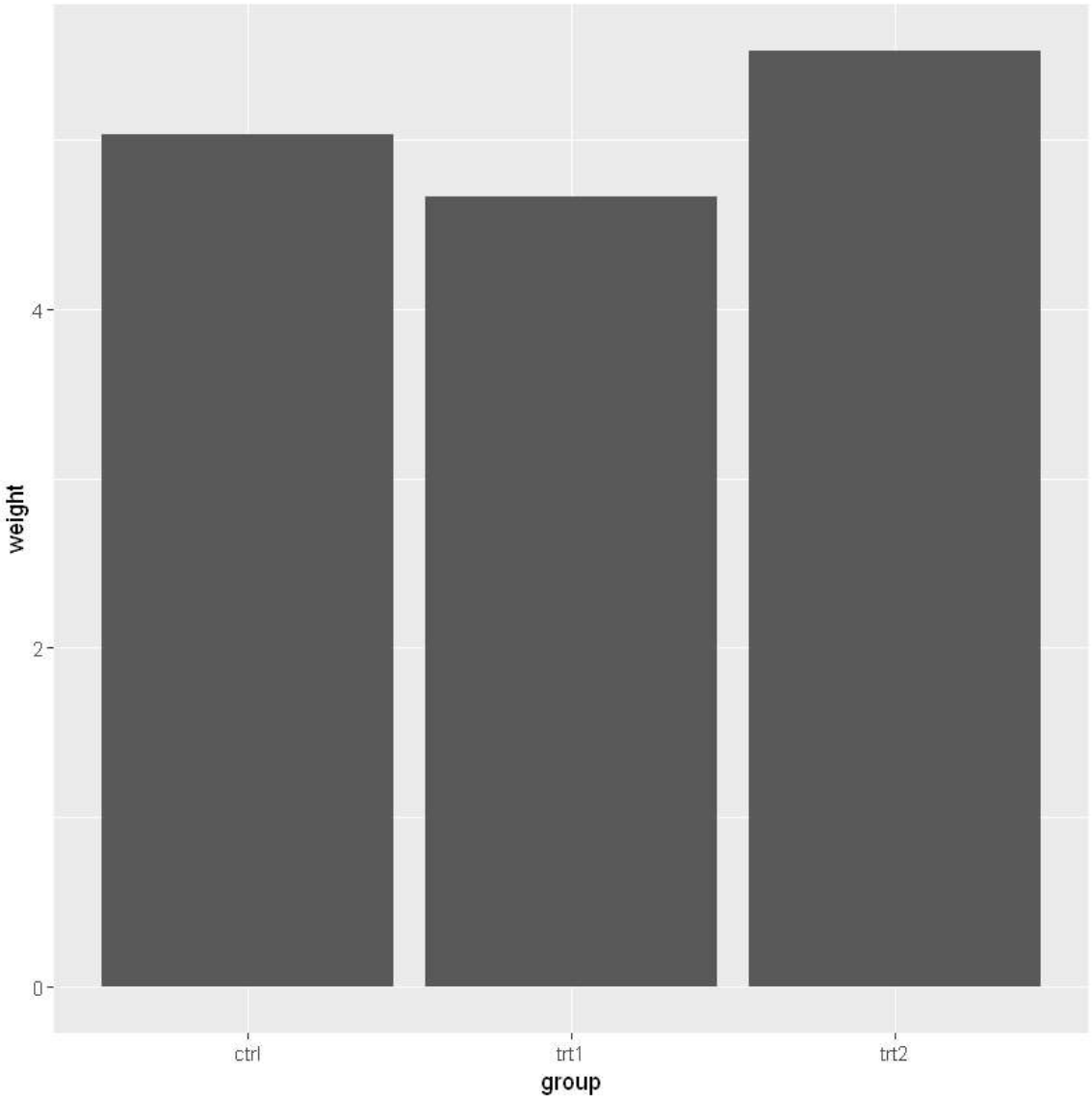
```
ggplot(mtcars, aes(x = wt, y = mpg))+
    geom_point() #用ggplot
```



绘制条形图

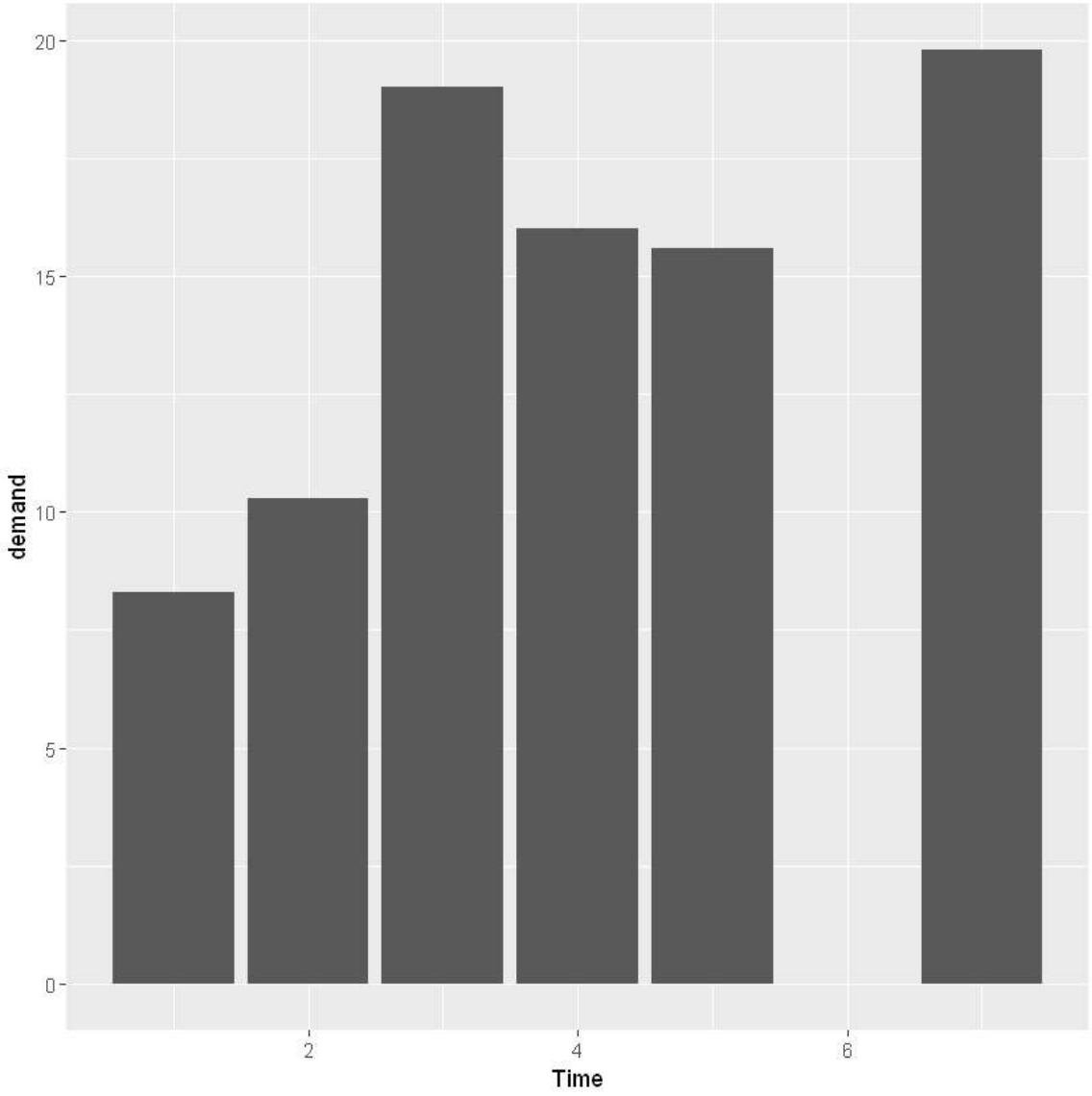
In [5]:

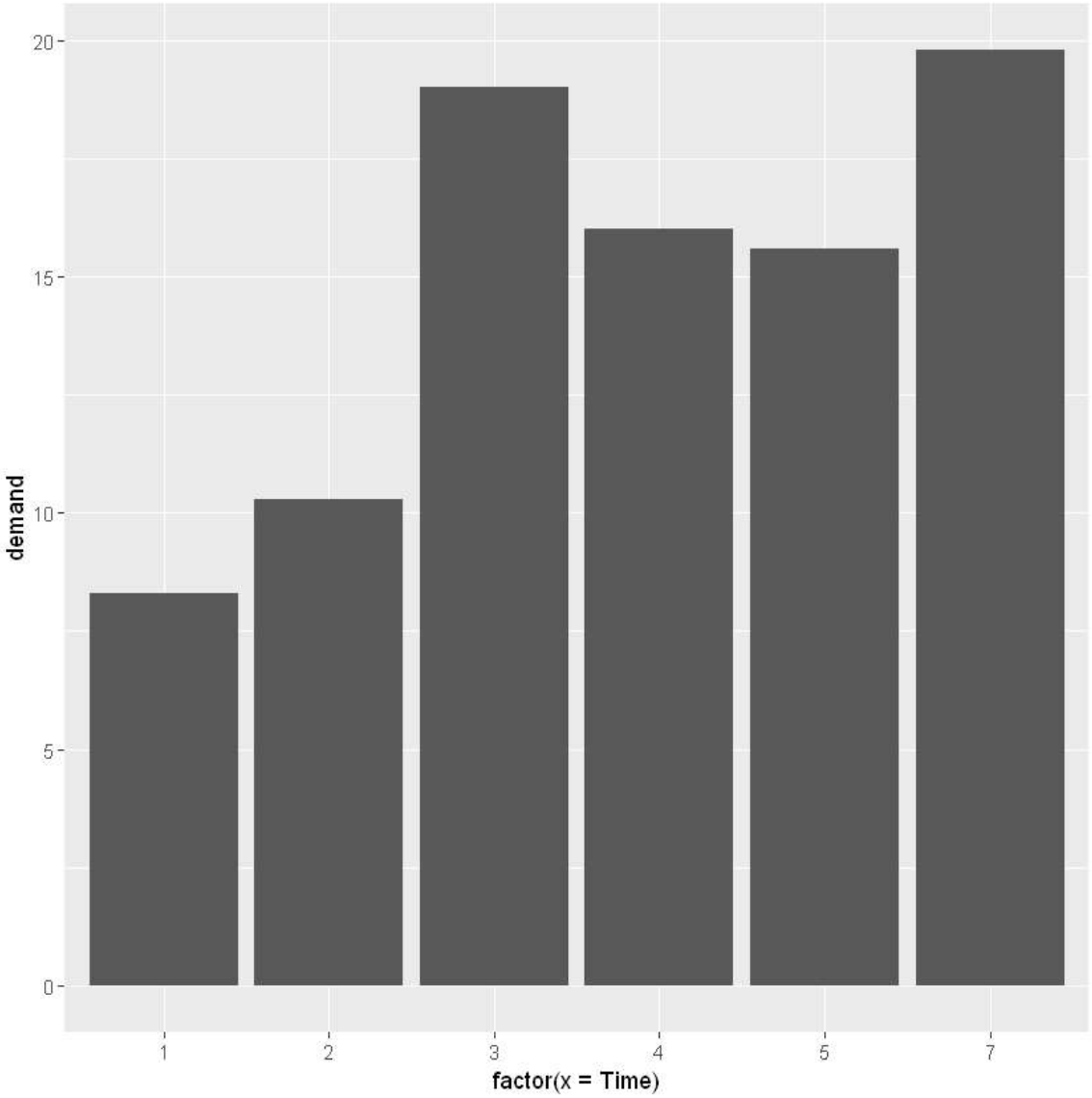
```
library(gcookbook)
ggplot(pg_mean, aes(x = group, y = weight)) +
    geom_bar(stat = "identity")
#其中stat="identity"表明取用样本点对应纵轴值
# 基函数：aes绑定条形图横轴纵轴
```



In [6]:

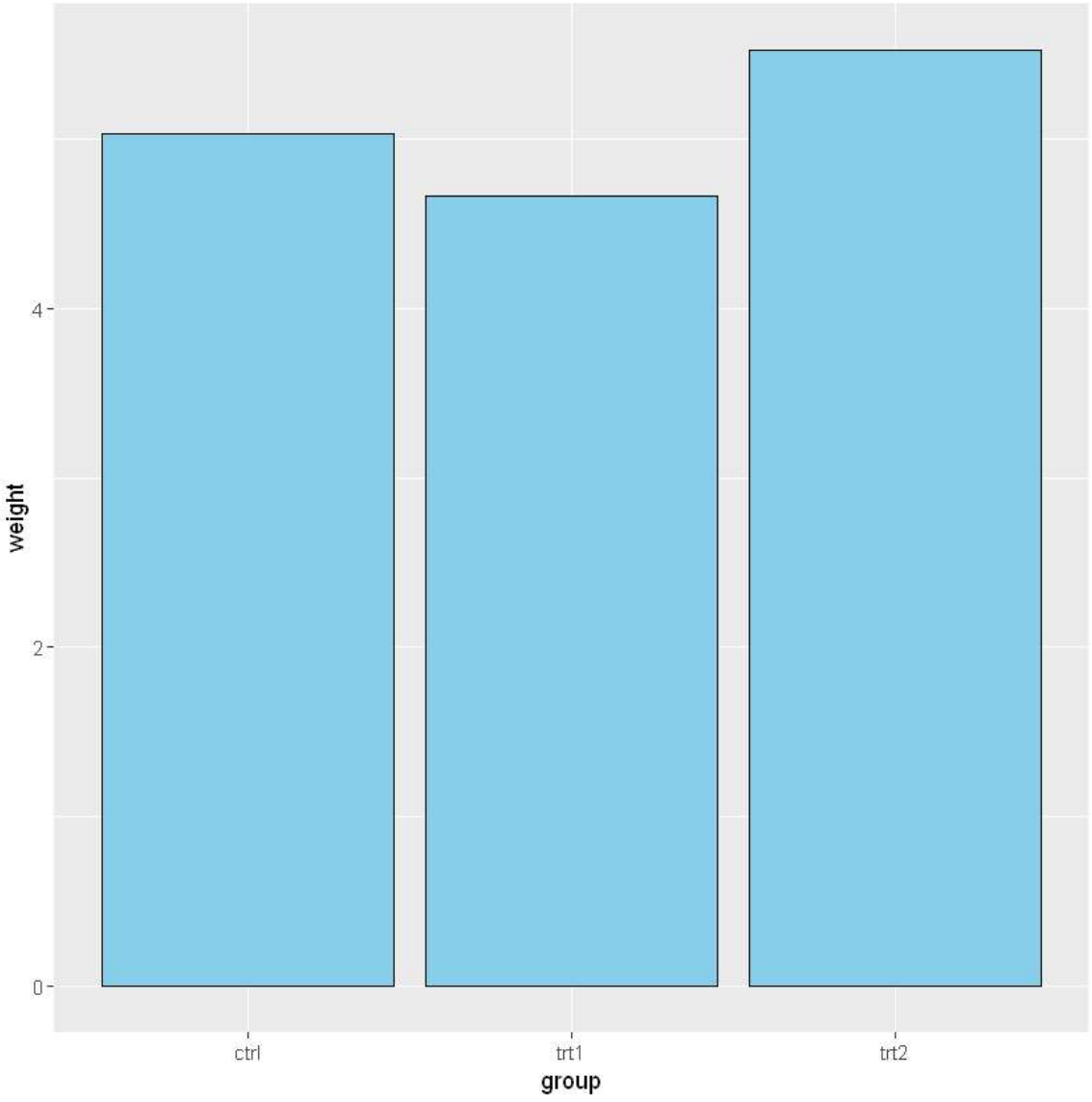
```
ggplot(BOD, aes(x = Time, y = demand)) +
    geom_bar(stat = "identity")

#Time转化为因子类型
ggplot(BOD, aes(factor(x = Time), y = demand)) +
    geom_bar(stat = "identity")
```


20
15
10
5
0
demand
2
4
6
Time


20
15
10
5
0
demand
1
2
3
4
5
7
factor(x = Time)

In [8]:

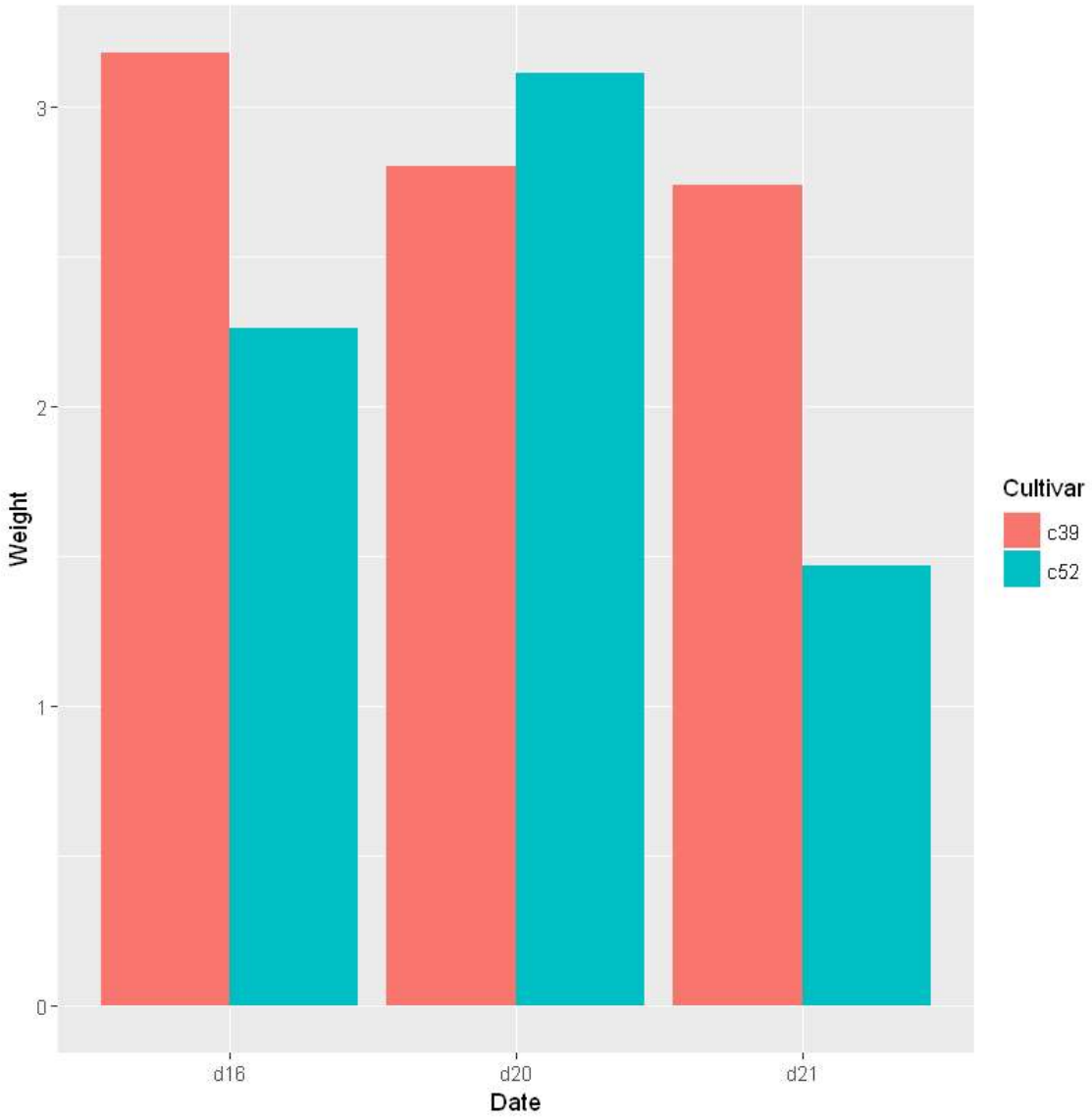
```
ggplot(pg_mean, aes(x = group, y = weight)) +
    geom_bar(stat = "identity", fill = "skyblue", color = "black")
#fill 填充色     color 边框色
```



簇状条形图

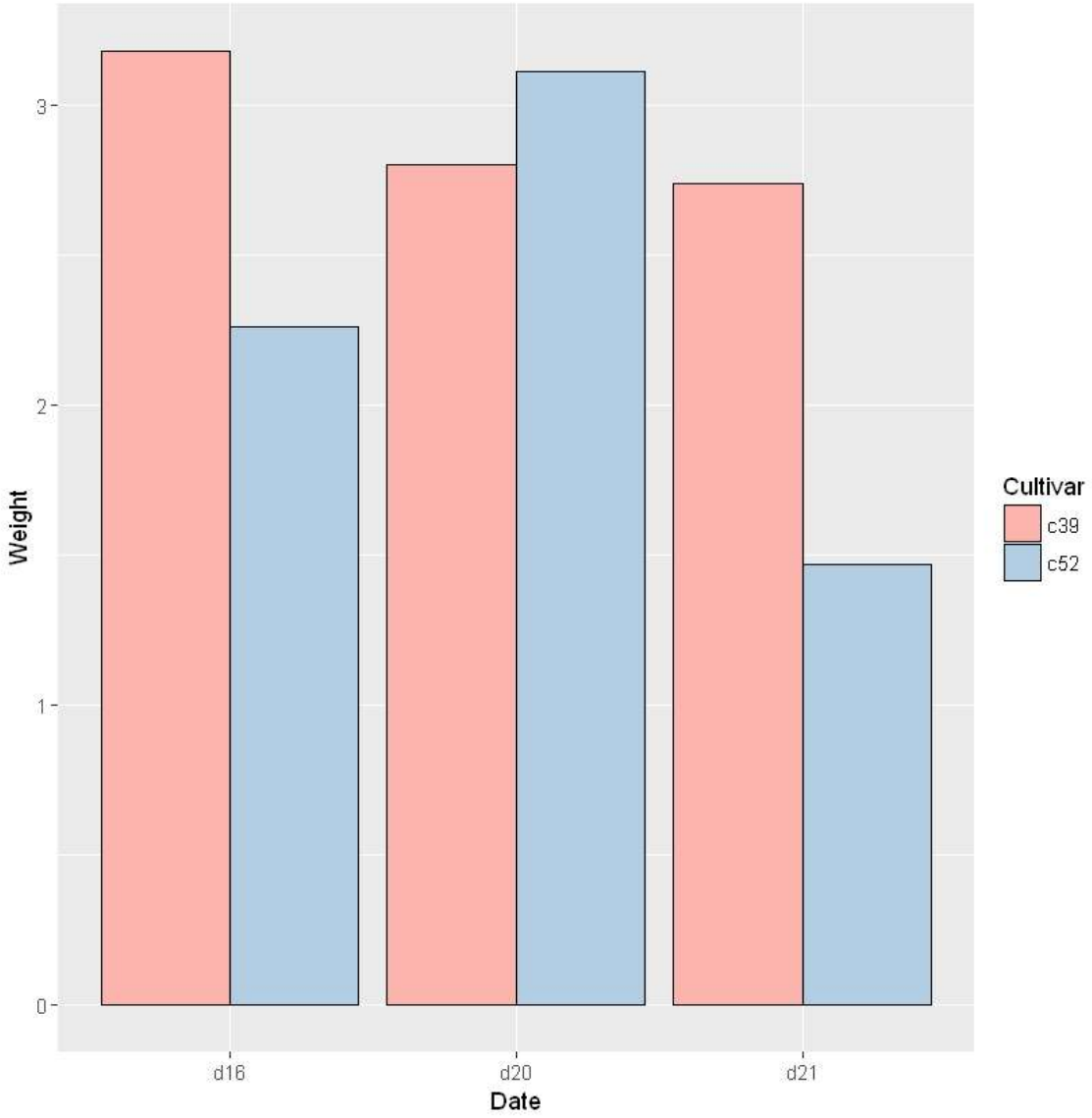
In [9]:

```
ggplot(cabbage_exp, aes(x = Date, y = Weight, fill = Cultivar)) +
    geom_bar(position = "dodge", stat = "identity")
```



In [10]:

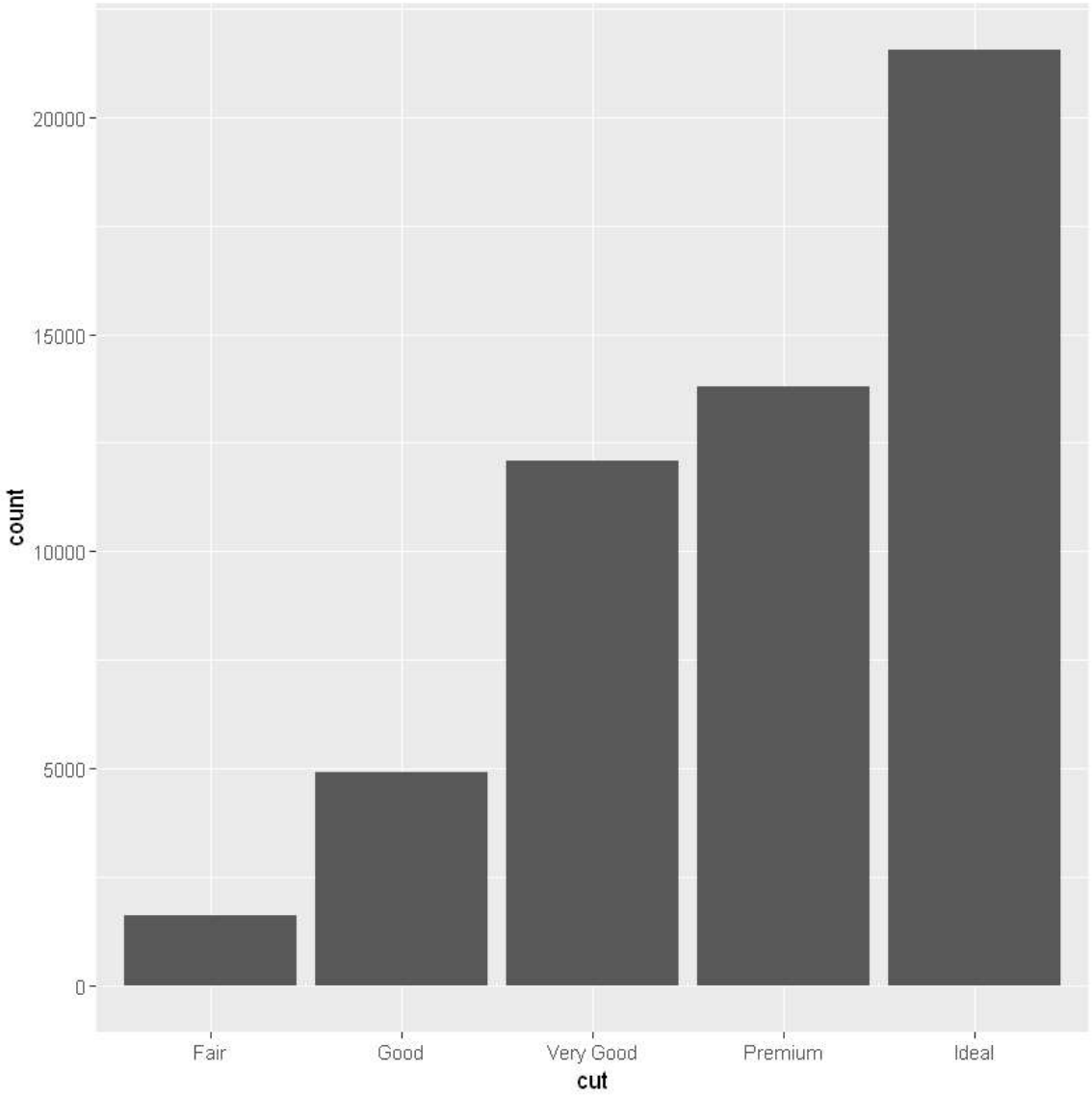
```
library(RColorBrewer) #自定义颜色
ggplot(cabbage_exp, aes(x = Date, y = Weight, fill = Cultivar)) +
    geom_bar(position = "dodge", stat = "identity", color = "black")+
    scale_fill_brewer(palette = "Pastel1")
```



绘制频数图

In [11]:

```
ggplot(diamonds, aes(x = cut))+
    geom_bar(stat = "count")
```

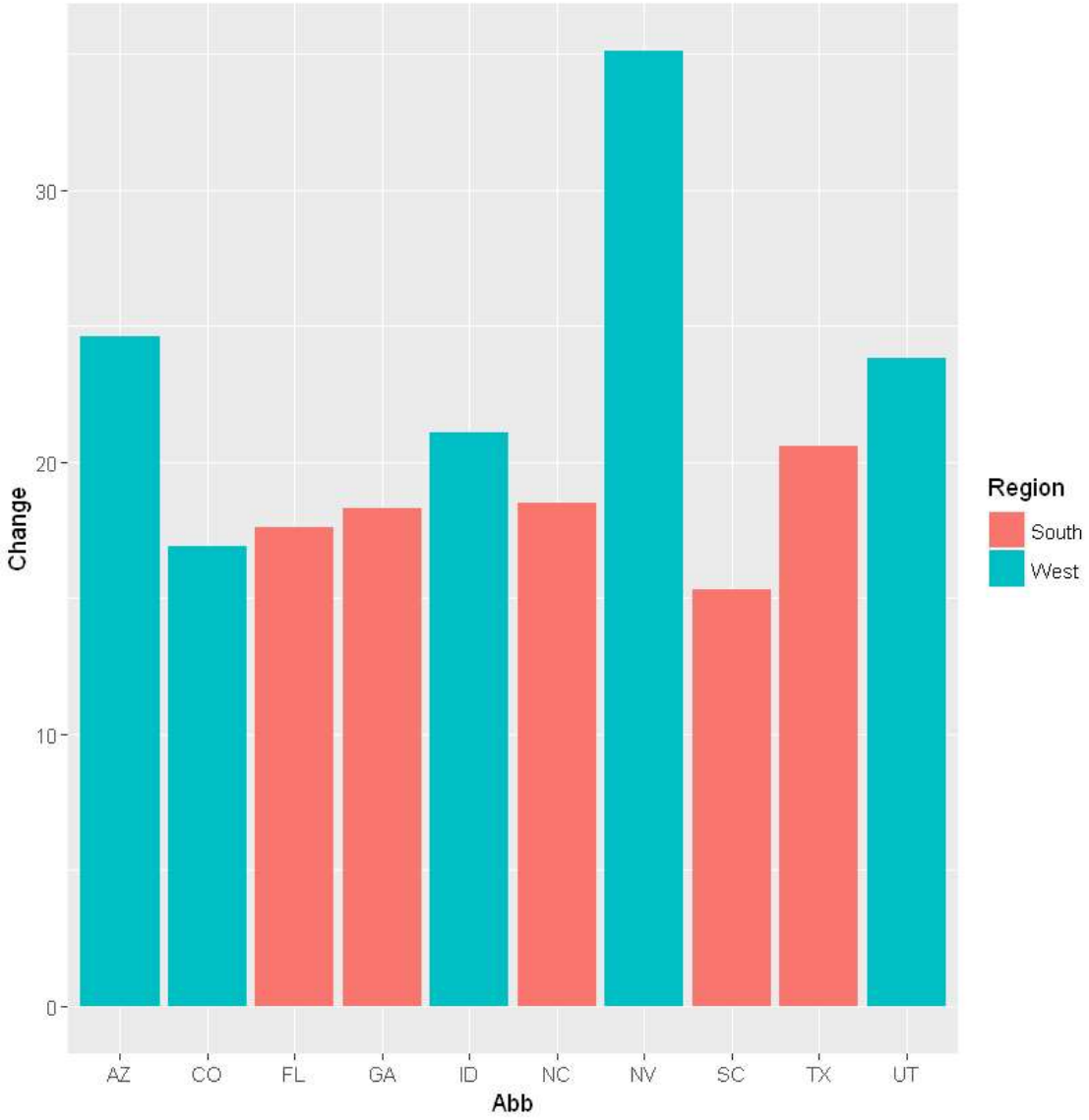


频数值和实际值要区分开来

着色

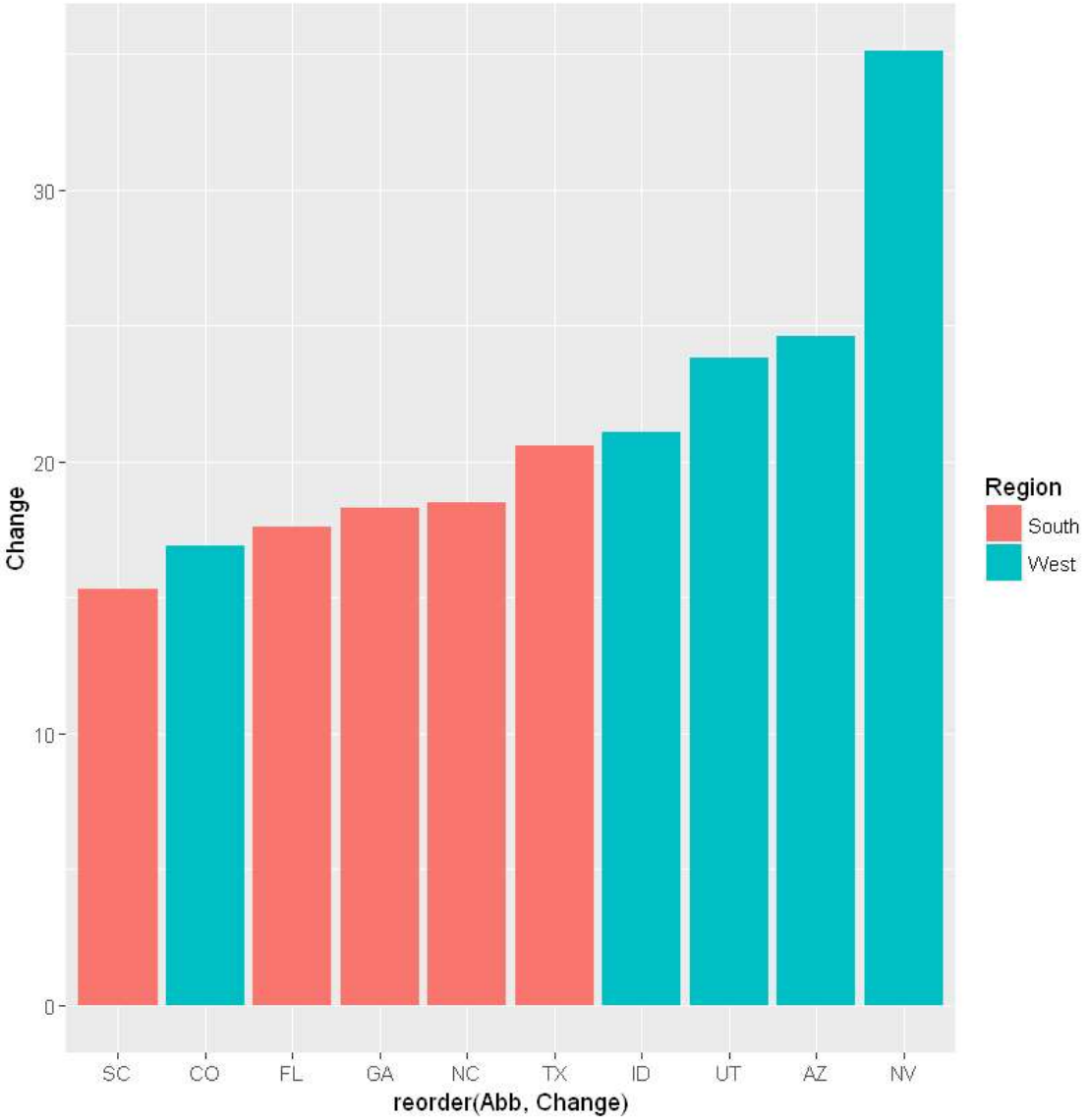
In [13]:

```
upc <- subset(uspopchange, rank(Change) > 40)
ggplot(upc, aes(x = Abb, y = Change, fill = Region))+
    geom_bar(stat = "identity")
```



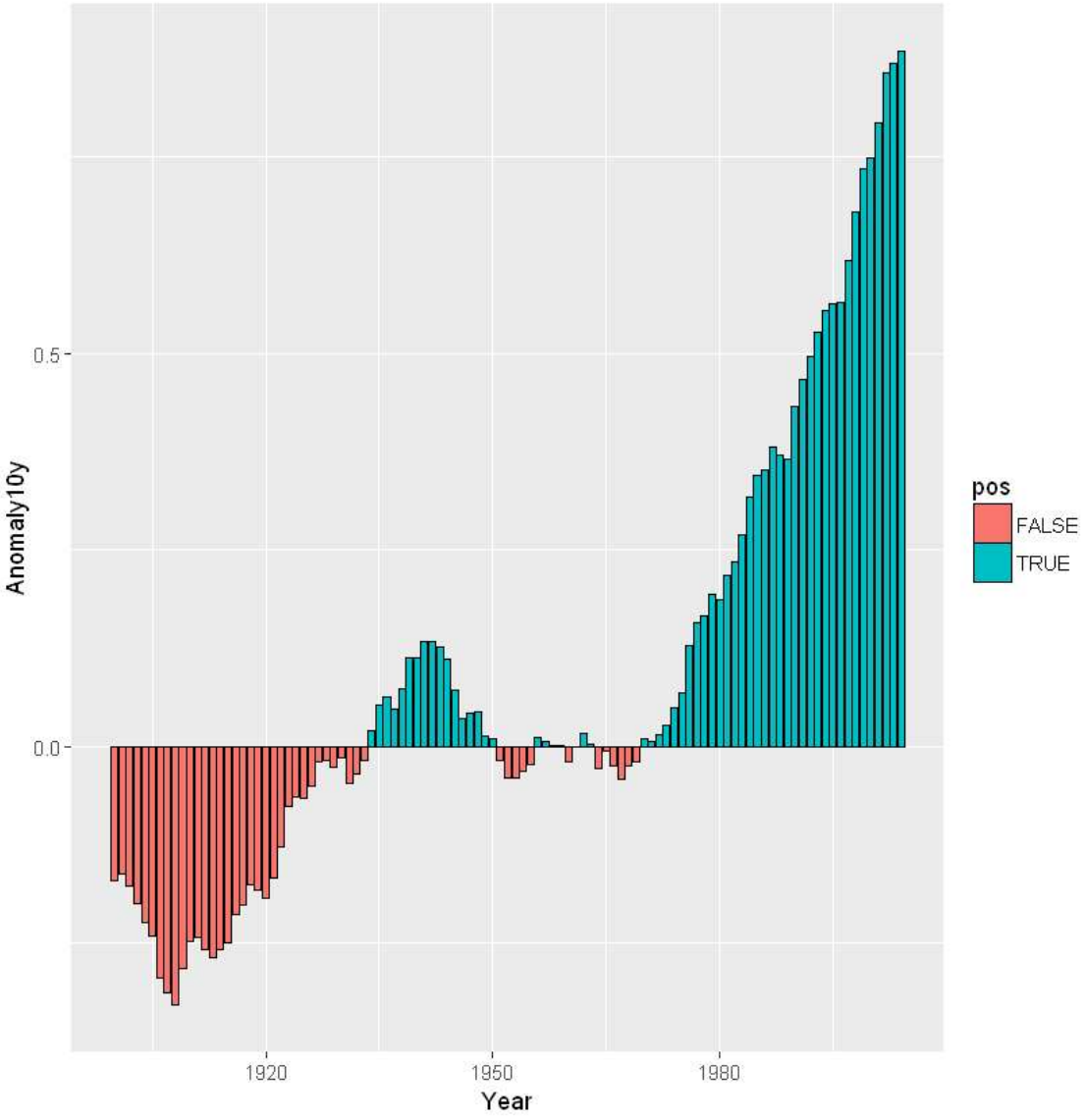
In [14]:

```
ggplot(upc, aes(x = reorder(Abb, Change), y = Change, fill = Region))+
    geom_bar(stat = "identity") #进行一个排序
```



In [15]:

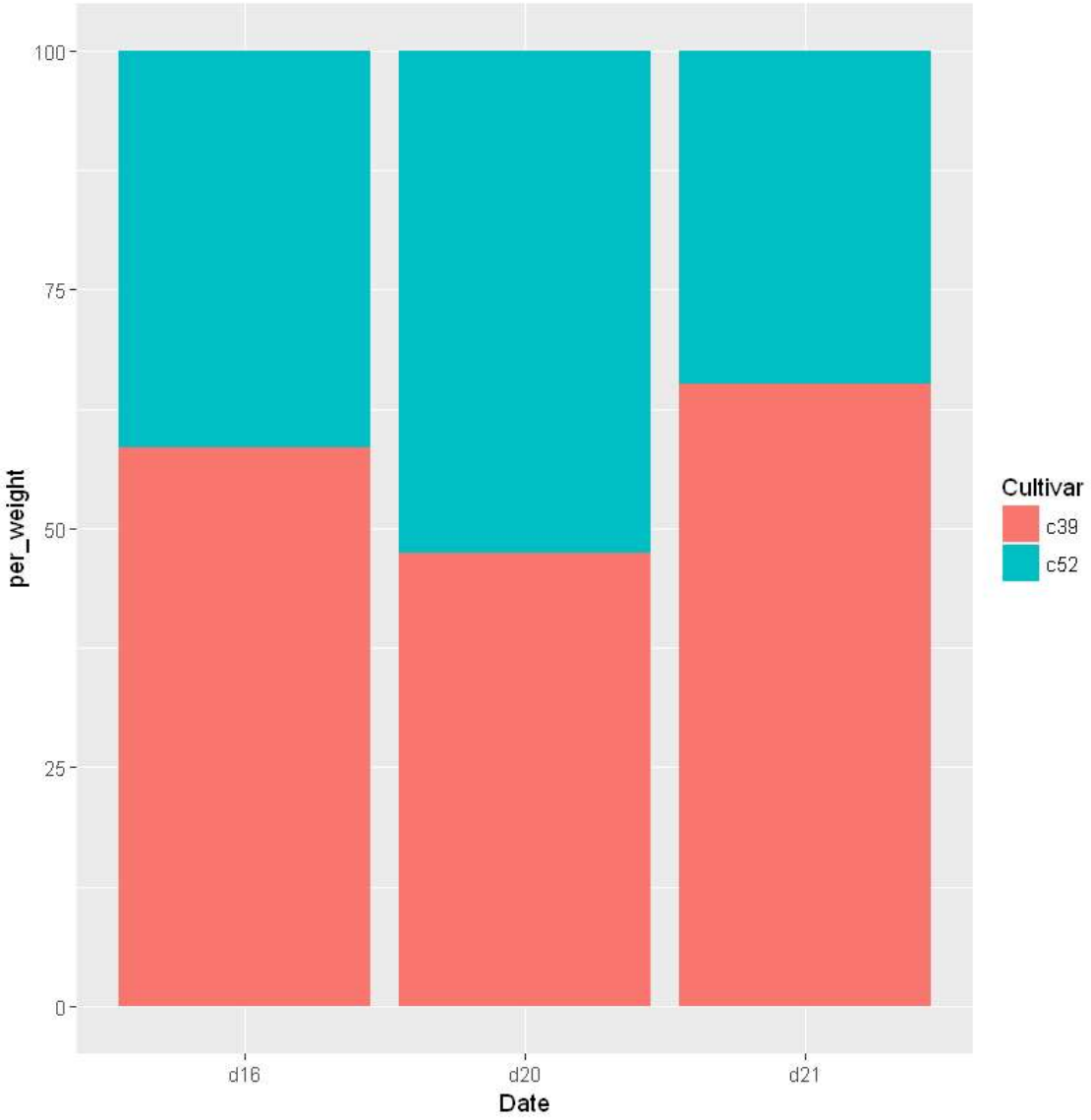
```
csub <- subset(climate, Source == "Berkeley" & Year >= 1900)
csub$pos <- csub$Anomaly10y >= 0 #增加一列数据框选出正值
ggplot(csub, aes(x = Year, y = Anomaly10y, fill = pos))+
    geom_bar(stat = "identity", position = "identity", color = "black", size = 0.1)
```



绘制百分比堆积图

In [17]:

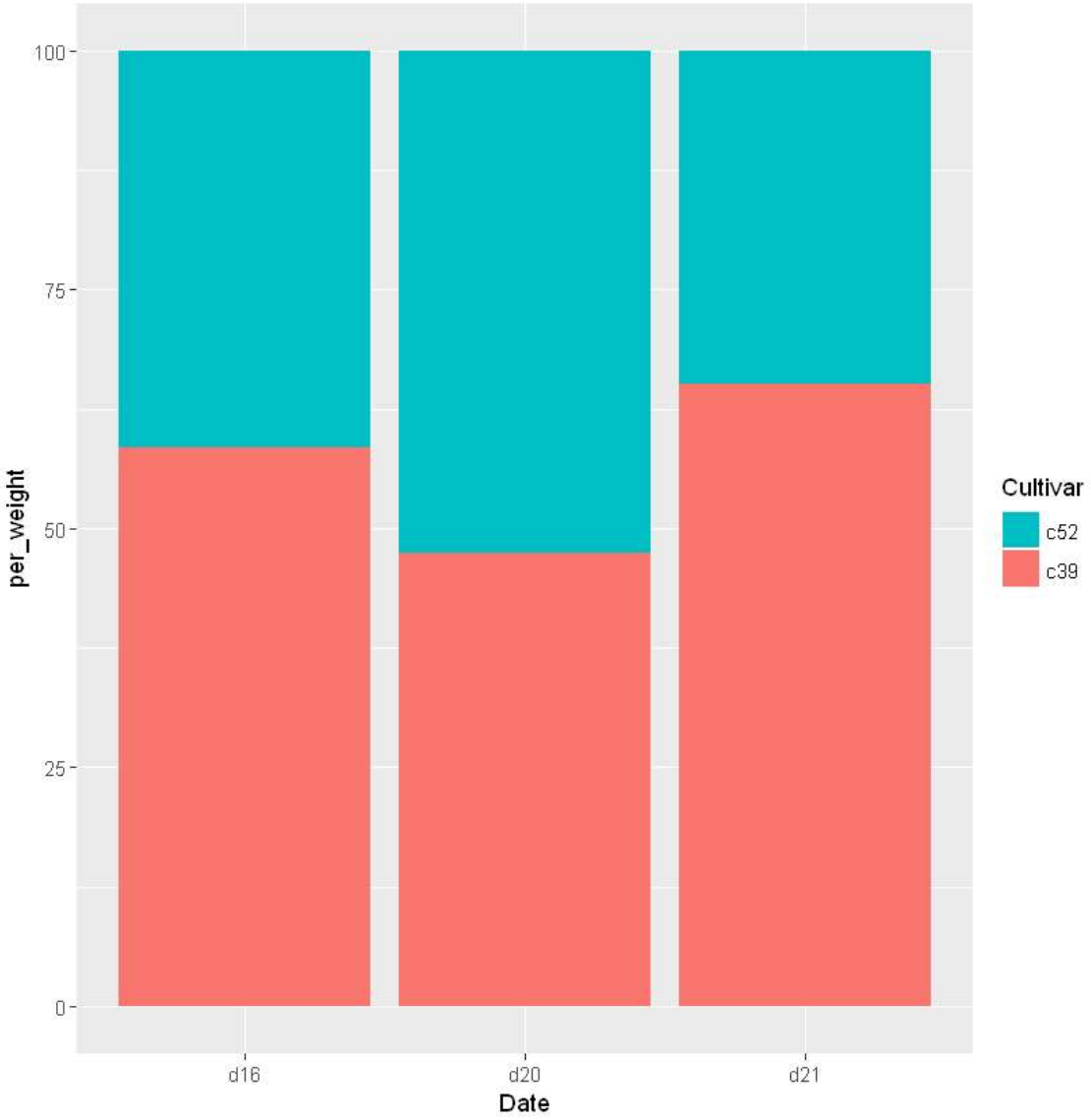
```
library(plyr)
ce <- ddply(cabbage_exp, "Date", transform, per_weight = Weight / sum(Weight) * 100)
ggplot(ce, aes(x = Date, y = per_weight, fill = Cultivar))+
    geom_bar(stat = "identity")
```



注意到图例标记与实际堆积图相反，可以设置

In [19]:

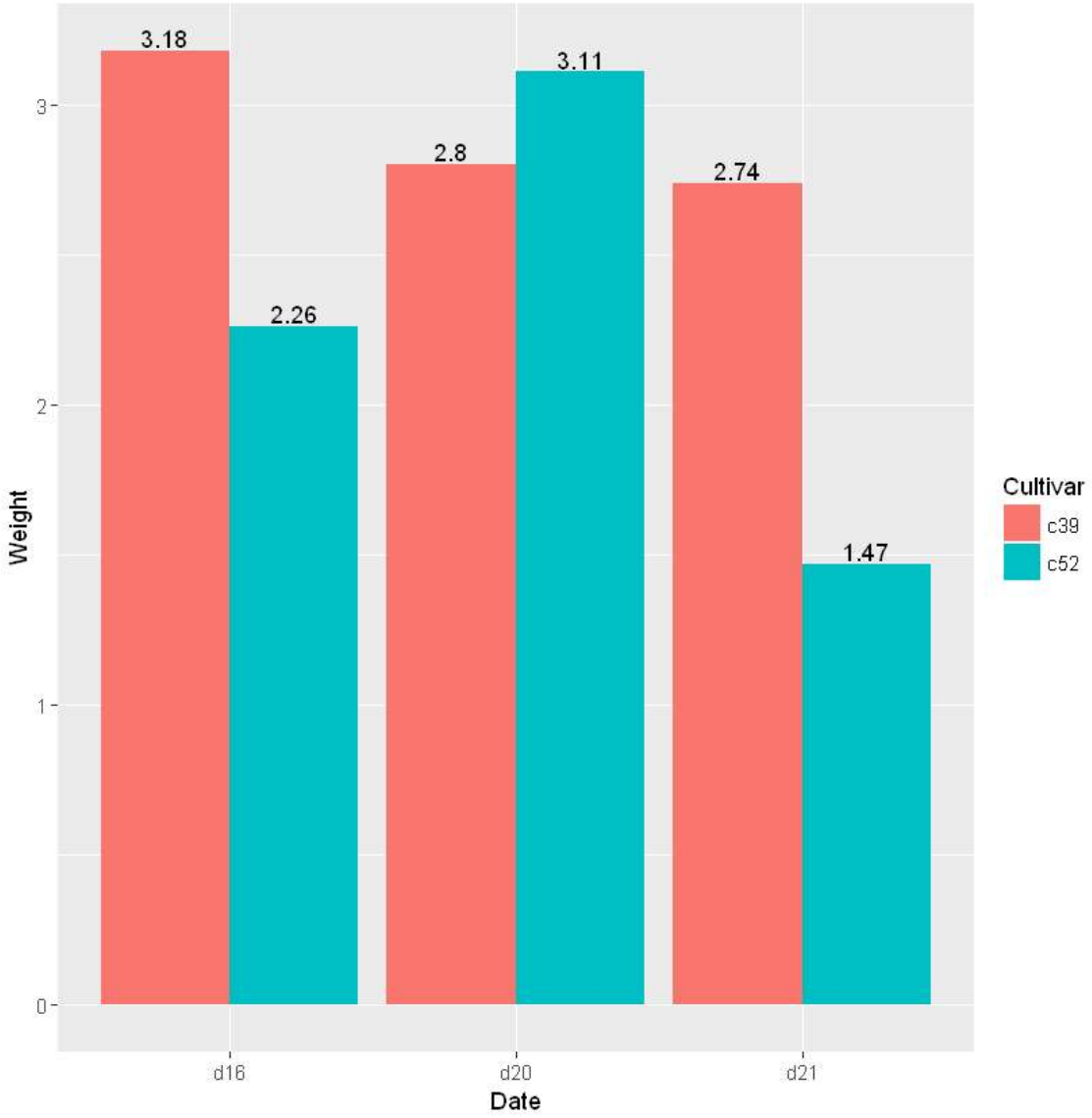
```
ggplot(ce, aes(x = Date, y = per_weight, fill = Cultivar))+
    geom_bar(stat = "identity")+
    guides(fill=guide_legend(reverse=TRUE))
```



添加数据标签

In [20]:

```
ggplot(cabbage_exp, aes(x = Date, y = Weight, fill = Cultivar))+
    geom_bar(position = "dodge", stat = "identity")+
    geom_text(aes(label = Weight), vjust = -0.2, position = position_dodge(0.9)) #vjust进行标签位置的微调，负数向上，正数向下
```



线性模型

lm()函数中常用参数

~	分隔符号，左边为响应变量，右边为解释变量。例如，要通过x、z和w预测y，代码为y ~ x + z + w
+	分隔预测变量
:	表示预测变量的交互项。例如，要通过x、z及x与z的交互项预测y，代码为y ~ x + z + x:z
*	表示所有可能交互项的简洁方式。代码y~ x z w可展开为y ~ x + z + w + x:z + x:w + z:w + x:z:w
^	表示交互项达到某个次数。代码y ~ (x + z + w)^2可展开为y ~ x + z + w + x:z + x:w + z:w
.	表示包含除因变量外的所有变量。例如，若一个数据框包含变量x、y、z和w，代码y ~ .可展开为y ~ x + z + w
-	减号，表示从等式中移除某个变量。例如，y ~ (x + z + w)^2 – x:w可展开为y ~ x + z + w + x:z + z:w
-1	删除截距项。例如，表达式y ~ x - 1拟合y在x上的回归，并强制直线通过原点
I()	从算术的角度来解释括号中的元素。例如，y ~ x + (z + w)^2将展开为y ~ x + z + w + z:w。
function	可以在表达式中用的数学函数。例如，log(y) ~ x + z + w表示通过x、z和w来预测log(y)

对模型进行操作的常用函数

summary()	展示拟合模型的详细结果
coefficients()	列出拟合模型的模型参数（截距项和斜率）
confint()	提供模型参数的置信区间（默认95%）
fitted()	列出拟合模型的预测值
residuals()	列出拟合模型的残差值
anova()	生成一个拟合模型的方差分析表，或者比较两个或更多拟合模型的方差分析表
vcov()	列出模型参数的协方差矩阵
AIC()	输出赤池信息统计量
plot()	生成评价拟合模型的诊断图
predict()	用拟合模型对新的数据集预测响应变量值

In [22]:

```
fit <- lm(weight~height, data = women)
summary(fit)
```

```
Call:
lm(formula = weight ~ height, data = women)

Residuals:
    Min      1Q  Median      3Q     Max
-1.7333 -1.1333 -0.3833  0.7417  3.1167

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -87.51667    5.93694  -14.74 1.71e-09 ***
height        3.45000    0.09114   37.85 1.09e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.525 on 13 degrees of freedom
Multiple R-squared:  0.991,     Adjusted R-squared:  0.9903
F-statistic:  1433 on 1 and 13 DF,  p-value: 1.091e-14
```

可见斜率为3.45，截距-87.52，p值小于0.001， R2系数为0.991

多项式回归

In [23]:

```
fit2 <- lm(weight~height + I(height^2), data = women)
summary(fit2)
```

```
Call:
lm(formula = weight ~ height + I(height^2), data = women)

Residuals:
     Min       1Q   Median       3Q      Max
-0.50941 -0.29611 -0.00941  0.28615  0.59706

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 261.87818   25.19677  10.393 2.36e-07 ***
height       -7.34832    0.77769  -9.449 6.58e-07 ***
I(height^2)   0.08306    0.00598  13.891 9.32e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3841 on 12 degrees of freedom
Multiple R-squared:  0.9995,    Adjusted R-squared:  0.9994
F-statistic: 1.139e+04 on 2 and 12 DF,  p-value: < 2.2e-16
```

多元线性回归

In [24]:

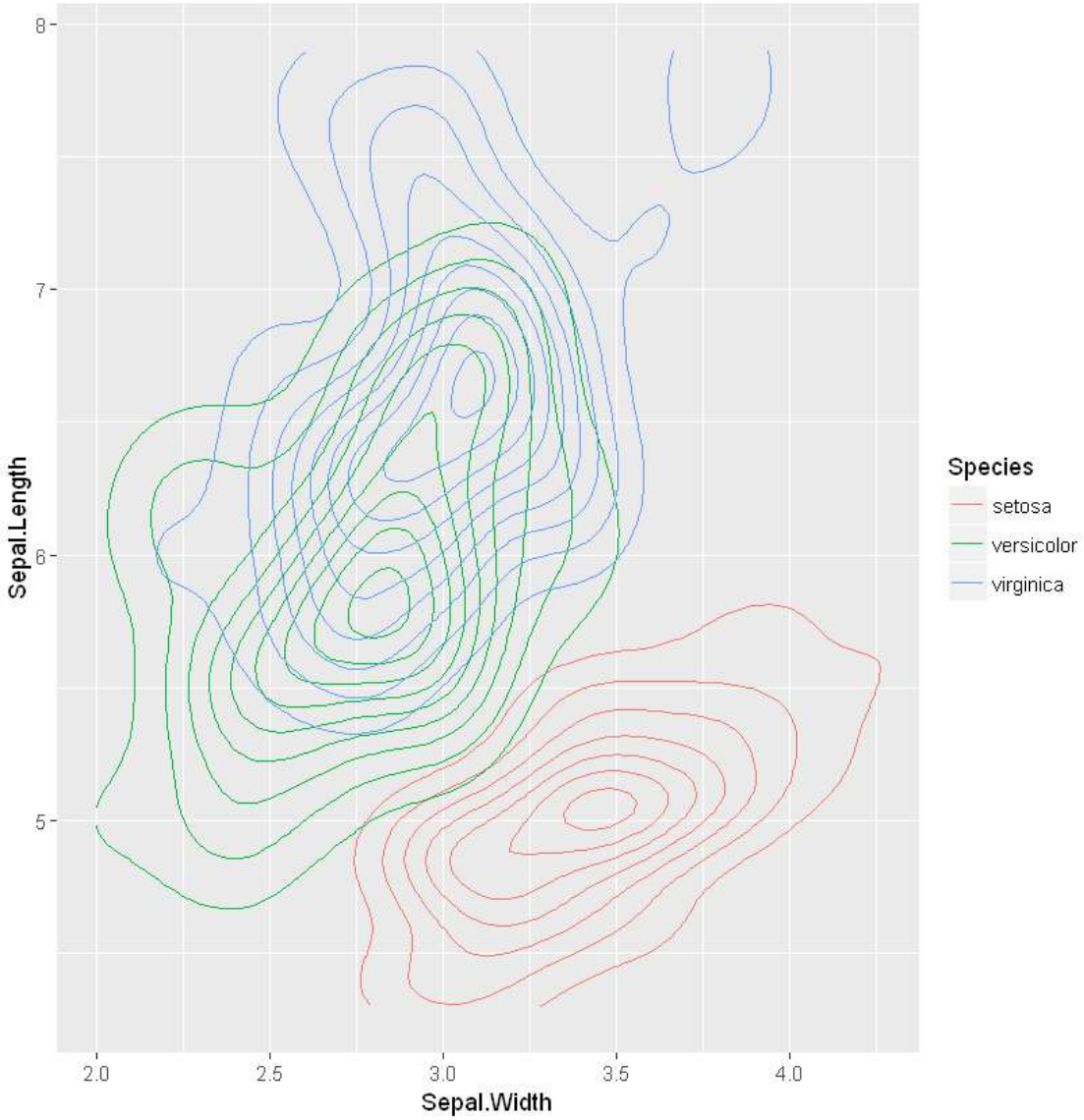
```
fit3 <- lm(mpg~ hp + wt + hp:wt, data = mtcars)
summary(fit3)
```

```
Call:
lm(formula = mpg ~ hp + wt + hp:wt, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.0632 -1.6491 -0.7362  1.4211  4.5513 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 49.80842    3.60516  13.816 5.01e-14 ***
hp          -0.12010    0.02470  -4.863 4.04e-05 ***
wt          -8.21662    1.26971  -6.471 5.20e-07 ***
hp:wt        0.02785    0.00742   3.753 0.000811 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.153 on 28 degrees of freedom
Multiple R-squared:  0.8848,    Adjusted R-squared:  0.8724 
F-statistic: 71.66 on 3 and 28 DF,  p-value: 2.981e-13
```

第二题 绘图

In [7]:

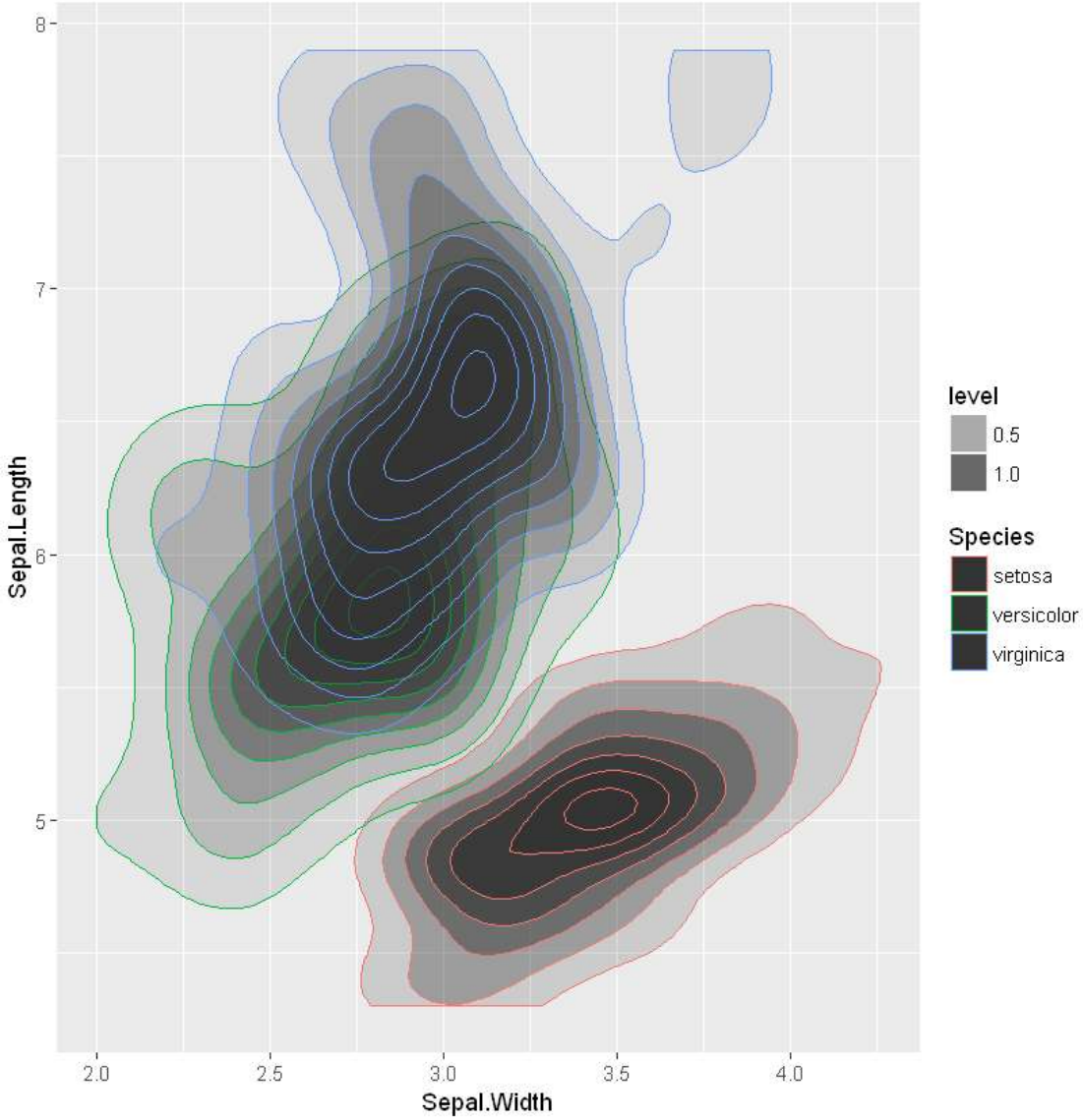
```
library(ggplot2)
p <- ggplot(iris, aes(x=Sepal.Width, y=Sepal.Length, color = Species))
p + stat_density2d()
```



可以见到三种类别的大致分布范围。接着用polygon对各自范围内部进行“填充”

In [8]:

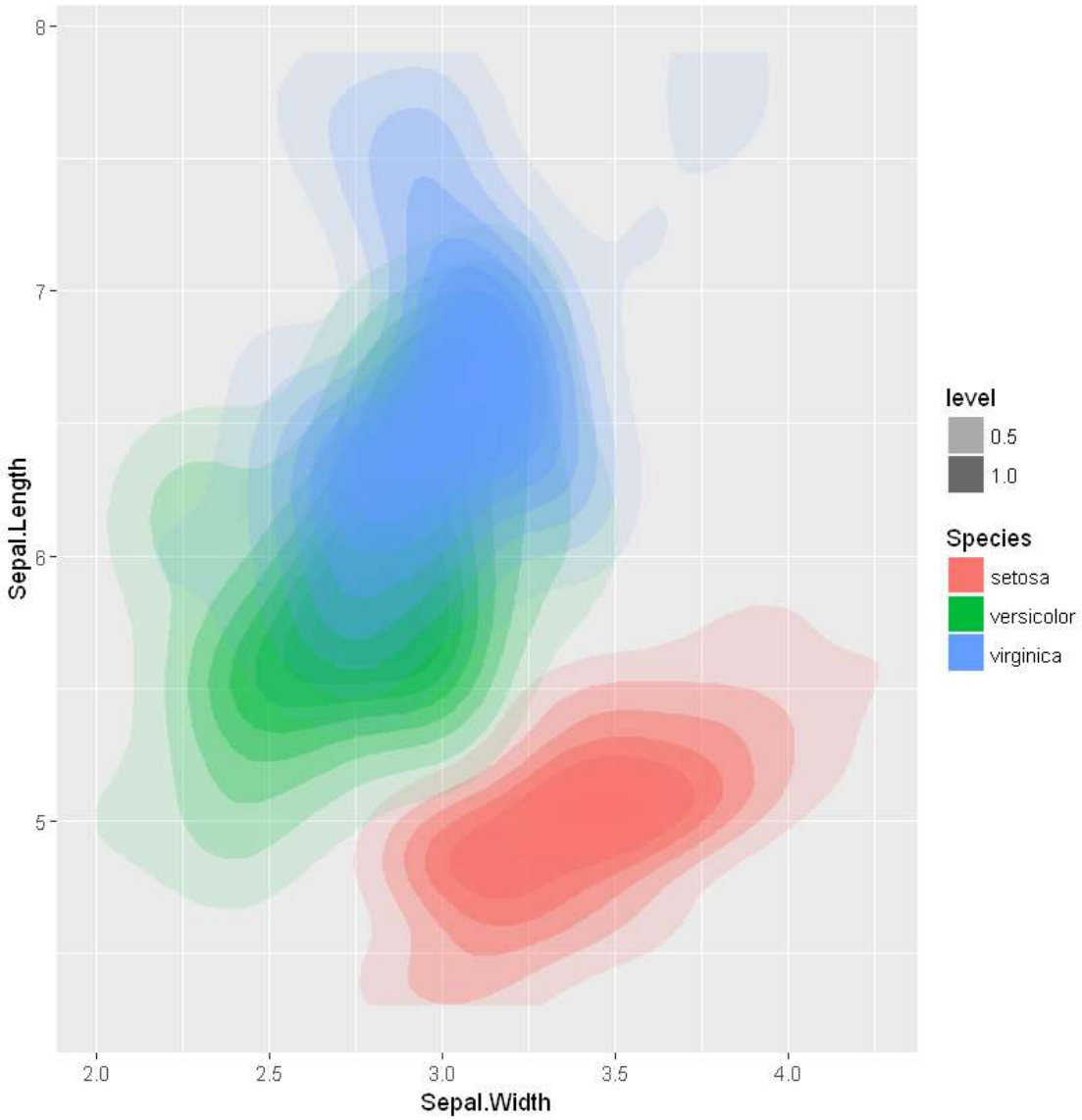
```
p + stat_density2d(aes(alpha = ..level..), geom ="polygon")
```



发现没有区分度，三种类别的边框颜色不一样，但是填充色一致，无法区分。修改color为fill参数

In [11]:

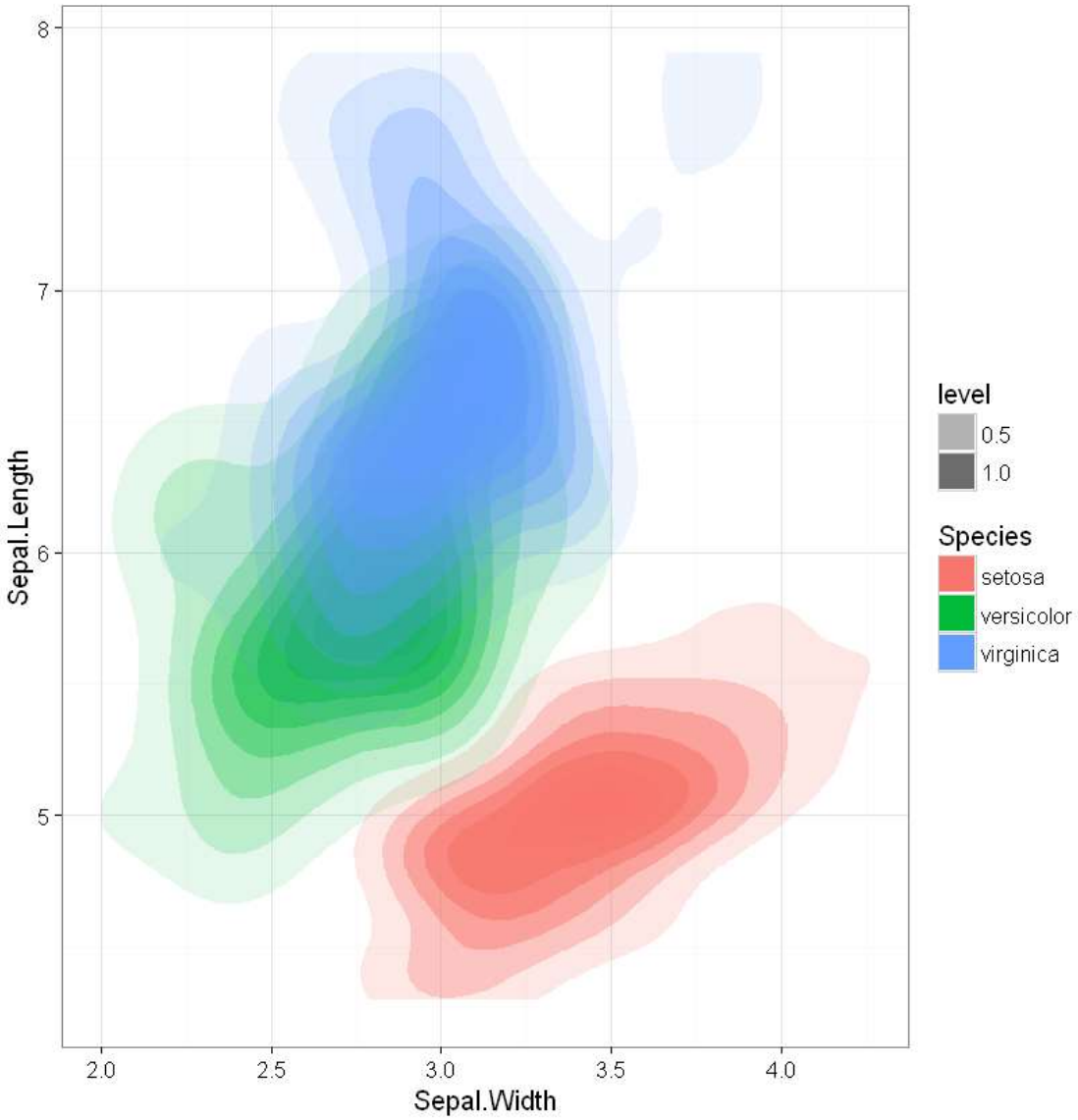
```
p <- ggplot(iris, aes(x=Sepal.Width, y=Sepal.Length, fill = Species))
p + stat_density2d(aes(alpha = ..level..), geom ="polygon")
```



此时大致正常，得到初步图像。接下来进行细节上的处理，先修改背景颜色和网格线分布

In [6]:

```
p + stat_density2d(aes(alpha = ..level..), geom ="polygon") +
    theme(panel.grid.minor = element_blank())+  #去掉网格线
    theme_bw() #修改背景颜色
```

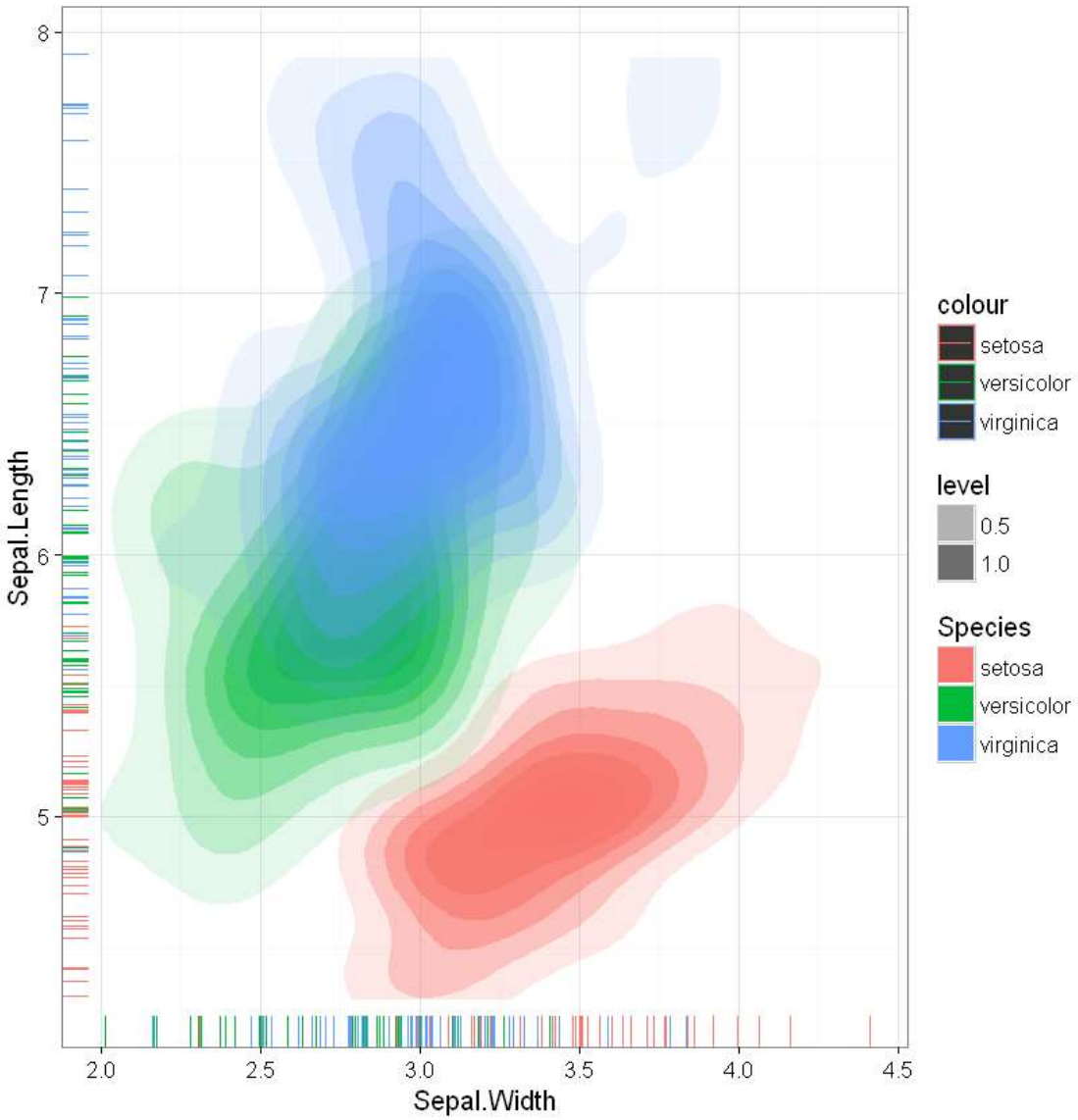


接着用geom_rug添加边际毛毯。这里要注意的地方有两点

1. 有很多重叠值，需要用“jitter”
2. 三种种类的边际毛毯线也需要区分

In [13]:

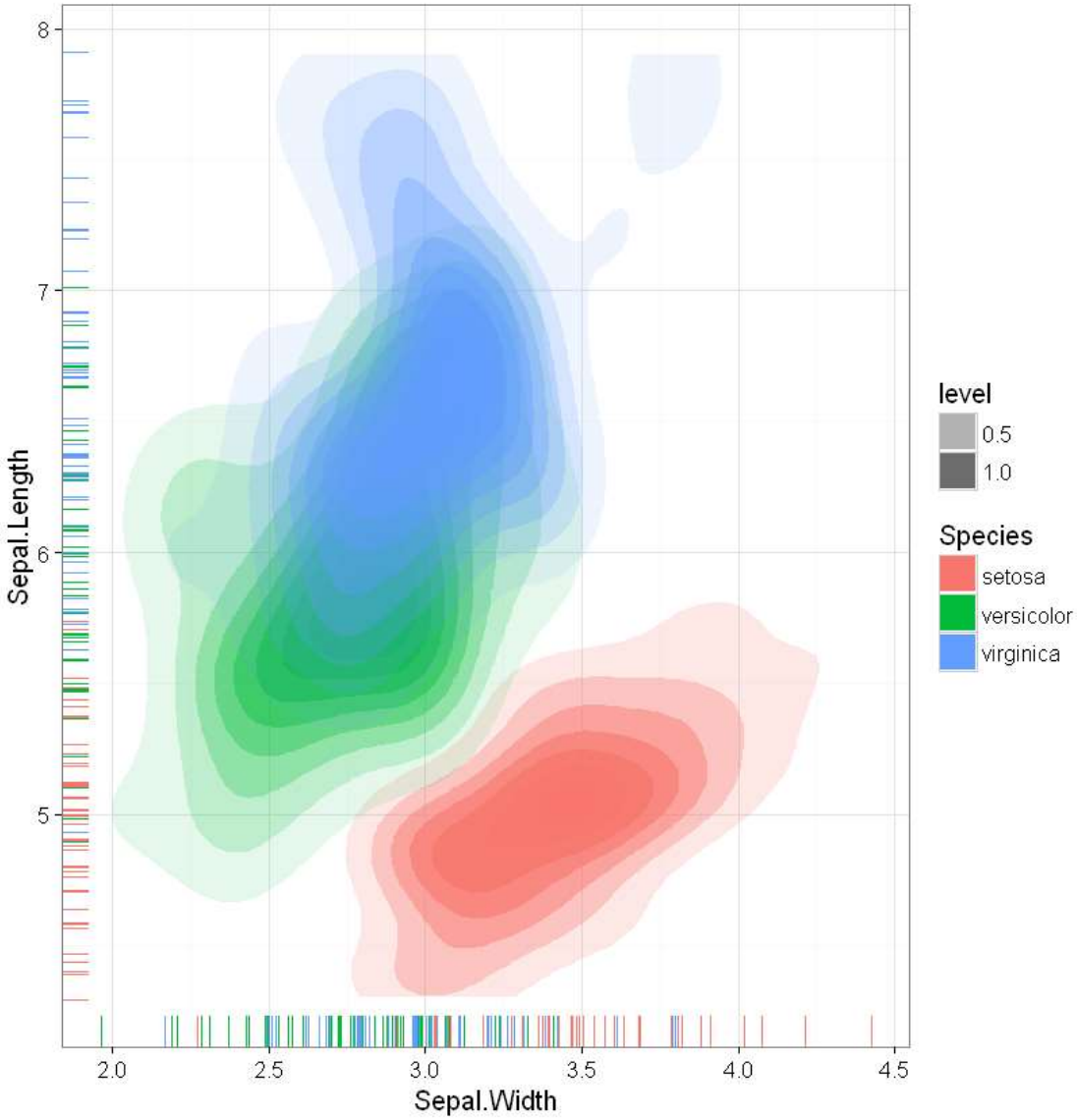
```
p + stat_density2d(aes(alpha = ..level..), geom ="polygon") +
    theme(panel.grid.minor = element_blank())+  #去掉网格线
    theme_bw()+ #修改背景颜色
    geom_rug(aes(color = Species), position = "jitter", size = 0.2)#增加边际毛毯
```



发现右边多出来colour的图例，需要处理掉

In [14]:

```
p + stat_density2d(aes(alpha = ..level..), geom ="polygon") + #添加透明度水平
    theme(panel.grid.minor = element_blank())+ #去掉次网格线
    theme_bw()+ #修改背景颜色
    geom_rug(aes(color = Species), position = "jitter", size = 0.2)+ #增加边际毛毯
    guides(color = FALSE) #删除图例
```



三、Kaggle比赛之house price

读入数据

In [1]:

```
setwd("G:/graduate-student/研究生/课件/R/kaggle")
train <- read.csv("train.csv", stringsAsFactors = FALSE)
test <- read.csv("test.csv", stringsAsFactors = FALSE)
head(train)
head(test)
```

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandCo
1	60	RL	65	8450	Pave	NA	Reg	Lvl
2	20	RL	80	9600	Pave	NA	Reg	Lvl
3	60	RL	68	11250	Pave	NA	IR1	Lvl
4	70	RL	60	9550	Pave	NA	IR1	Lvl
5	60	RL	84	14260	Pave	NA	IR1	Lvl
6	50	RL	85	14115	Pave	NA	IR1	Lvl

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	Land
1461	20	RH	80	11622	Pave	NA	Reg	Lvl
1462	20	RL	81	14267	Pave	NA	IR1	Lvl
1463	60	RL	74	13830	Pave	NA	IR1	Lvl
1464	60	RL	78	9978	Pave	NA	IR1	Lvl
1465	120	RL	43	5005	Pave	NA	IR1	HLS
1466	60	RL	75	10000	Pave	NA	IR1	Lvl

In [3]:

```
dim(train)
dim(test)
```

1460 81

1459 80

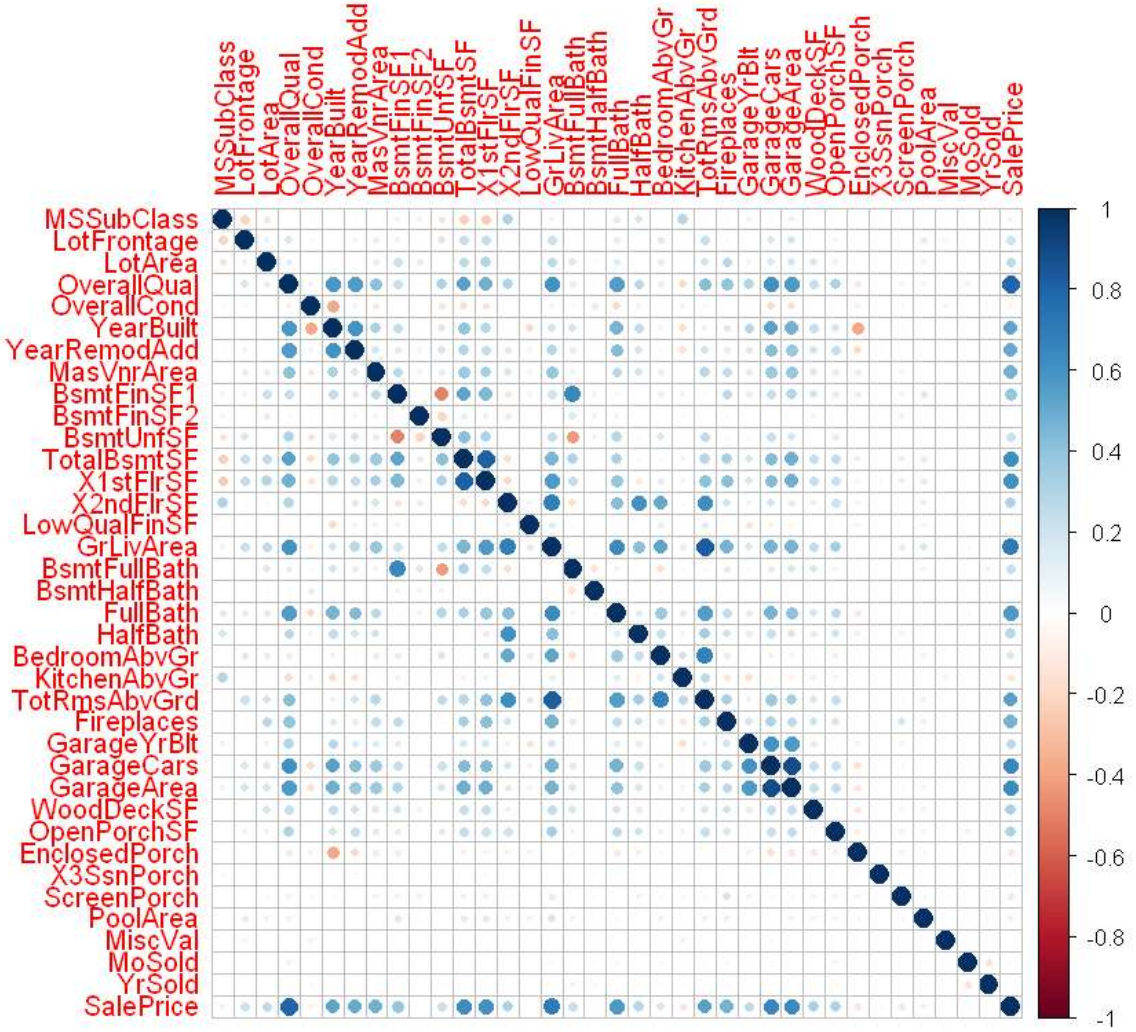
train数据集一共81行1460列，除去Id一共79个特征值，这些特征与SalePrice值相关。这其中36个数值型特征，43个分类（枚举）型特征，对于数值型特征，可以用相关系数的值去表征相关性，但是对于枚举型特征，我们需要寻找另外的参数来判断该特征对于目标标签的相关性，决定系数R2的大小决定了相关的密切程度。现在把这两部分特征分开进行处理，找出其中和SalePrice相关性最强的特征。

In [3]:

```
train_num <- train[,2, drop = FALSE]
for (i in c(3:79)) {
    if(class(train[2, i]) == "integer"){
        train_num <- cbind(train_num, train[, i, drop = FALSE])
    }
}

train_cat <- train[,3, drop = FALSE]
for (i in c(3:79)) {
    if(class(train[2, i]) == "character"){
        train_cat <- cbind(train_cat, train[, i, drop = FALSE])
    }
}
train_num[is.na(train_num)] <- -1
train_num <- cbind(train_num, train[, 81, drop = FALSE])

train_cat[is.na(train_cat)] <- "*MISSING*"
train_cat <- cbind(train_cat, train[, 81, drop = FALSE])
```

In [4]:

```
library(corrplot)
mcor <- cor(train_num)
corrplot(mcor)
```



从上述37个数值变量中选出相关性最强的特征，认为相关性大于0.4则表示考虑该特征

In [5]:

```
mcor <- as.data.frame(mcor)
subset(mcor, mcor$SalePrice > 0.4 & mcor$SalePrice < 1 | mcor$SalePrice < -0.4, select = c(SaleP
rice))
```

	SalePrice
OverallQual	0.7909816
YearBuilt	0.5228973
YearRemodAdd	0.5071010
MasVnrArea	0.4725851
TotalBsmtSF	0.6135806
X1stFlrSF	0.6058522
GrLivArea	0.7086245
FullBath	0.5606638
TotRmsAbvGrd	0.5337232
Fireplaces	0.4669288
GarageCars	0.6404092
GarageArea	0.6234314

将这12个特征提取出来组成一个数据框。先获得这些特征在train_num里面的索引

In [7]:

```
index1 <- c()
for (i in c(1:36)) {
    if(mcor$SalePrice[i] > 0.4){
        index1 <- c(index1, i)
    }
}
index1
```

4 6 7 8 12 13 16 19 23 24 26 27

根据索引生成只有这些数字特征的数据框

In [8]:

```
last1 <- train_num[,index1[1], drop = FALSE]
for (i in c(2:12)) {
    last1 <- cbind(last1, train_num[,index1[i], drop = FALSE])
}
head(last1)
```

OverallQual	YearBuilt	YearRemodAdd	MasVnrArea	TotalBsmtSF	X1stFlrSF	GrLiv
7	2003	2003	196	856	856	1710
6	1976	1976	0	1262	1262	1262
7	2001	2002	162	920	920	1786
7	1915	1970	0	756	961	1717
8	2000	2000	350	1145	1145	2198
5	1993	1995	0	796	796	1362

选出了这12个数值特征，再从42个枚举特征中选出与房价最密切的特征，通过R2系数来决定,选择R2系数大于0.15的特征值。首先仍然是先获得这些特征的下标

In [10]:

```
r2 <- c()
index2 <- c()
for (i in c(1:42)) {
    model = lm(data = train_cat, SalePrice ~train_cat[,i])
    a <- summary(model)
    r2 <- c(r2, a$r.squared)
    if(a$r.squared > 0.15){
        index2 <- c(index2, i)
    }
}
index2
```

10 17 18 19 20 22 23 26 29 32 34 35 36

In [11]:

```
length(index2)
```

13

一共有13个枚举变量，把这13个枚举特征也挑选出来生成一个数据框

In [12]:

```
last2 <- train_cat[,index2[1], drop = FALSE]
for (i in c(2:13)) {
    last2 <- cbind(last2, train_cat[,index2[i], drop = FALSE])
}
head(last2)
```

Neighborhood	Exterior1st	Exterior2nd	MasVnrType	ExterQual	Foundation	BsmtC
CollgCr	VinylSd	VinylSd	BrkFace	Gd	PConc	Gd
Veenker	MetalSd	MetalSd	None	TA	CBlock	Gd
CollgCr	VinylSd	VinylSd	BrkFace	Gd	PConc	Gd
Crawfor	Wd Sdng	Wd Shng	None	TA	BrkTil	TA
NoRidge	VinylSd	VinylSd	BrkFace	Gd	PConc	Gd
Mitchel	VinylSd	VinylSd	None	TA	Wood	Gd

至此，数据特征和枚举特征都已经挑出来了，可以进行建模预测房价。

In [14]:

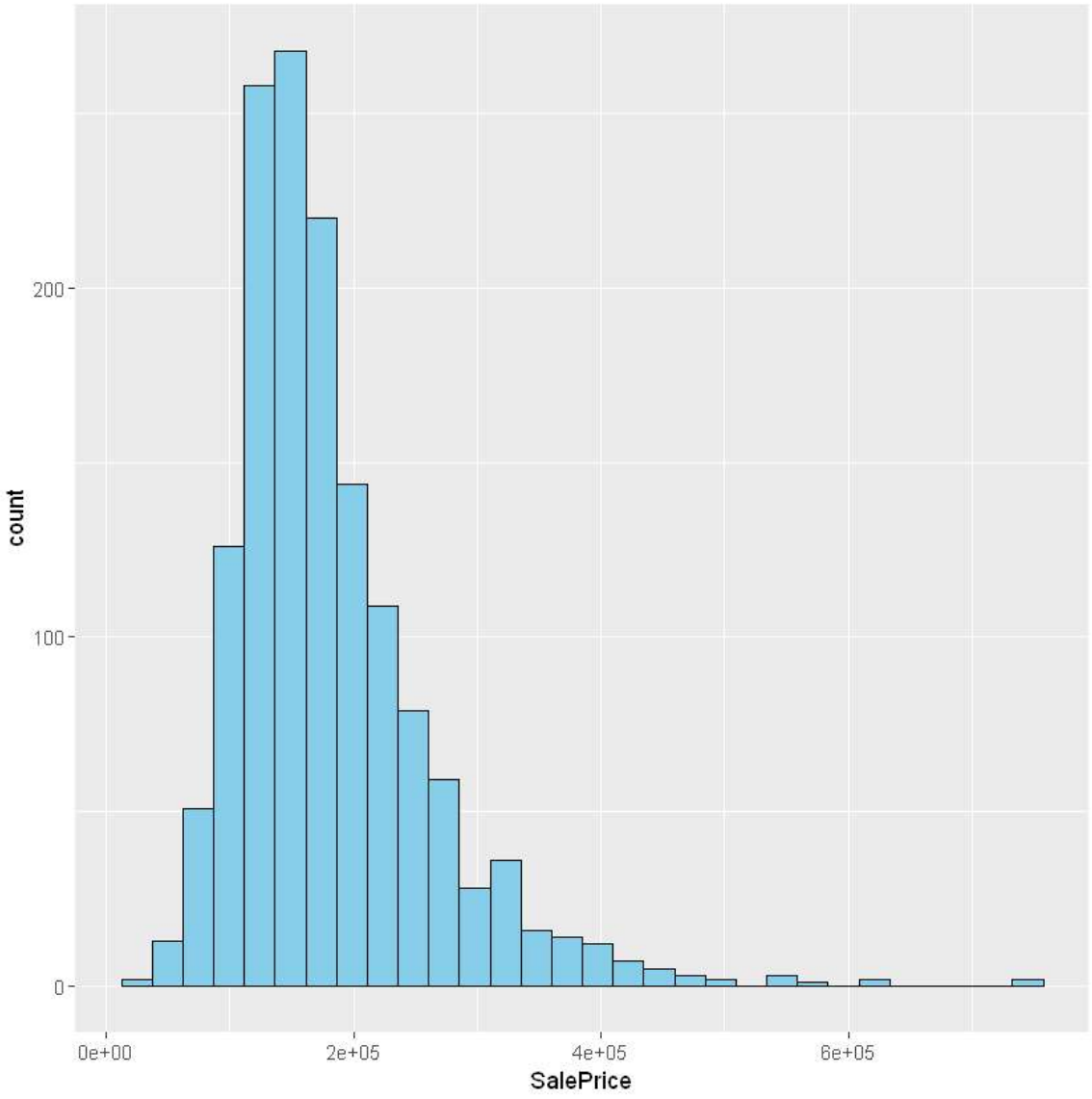
```
last <- cbind(last1, last2)
last <- cbind(last, train[, 81, drop = FALSE])
library(rpart)
model <- rpart(data = last, SalePrice~.)
pred = predict(model, test)
pred <- as.data.frame(pred)
ans <- cbind(test$Id, pred)
names(ans) <- c("Id", "SalePrice")
write.csv(ans, "sample_submission.csv")
```

提交得分0.24348，接下来进行改良。首先观察一下房价分布

In [17]:

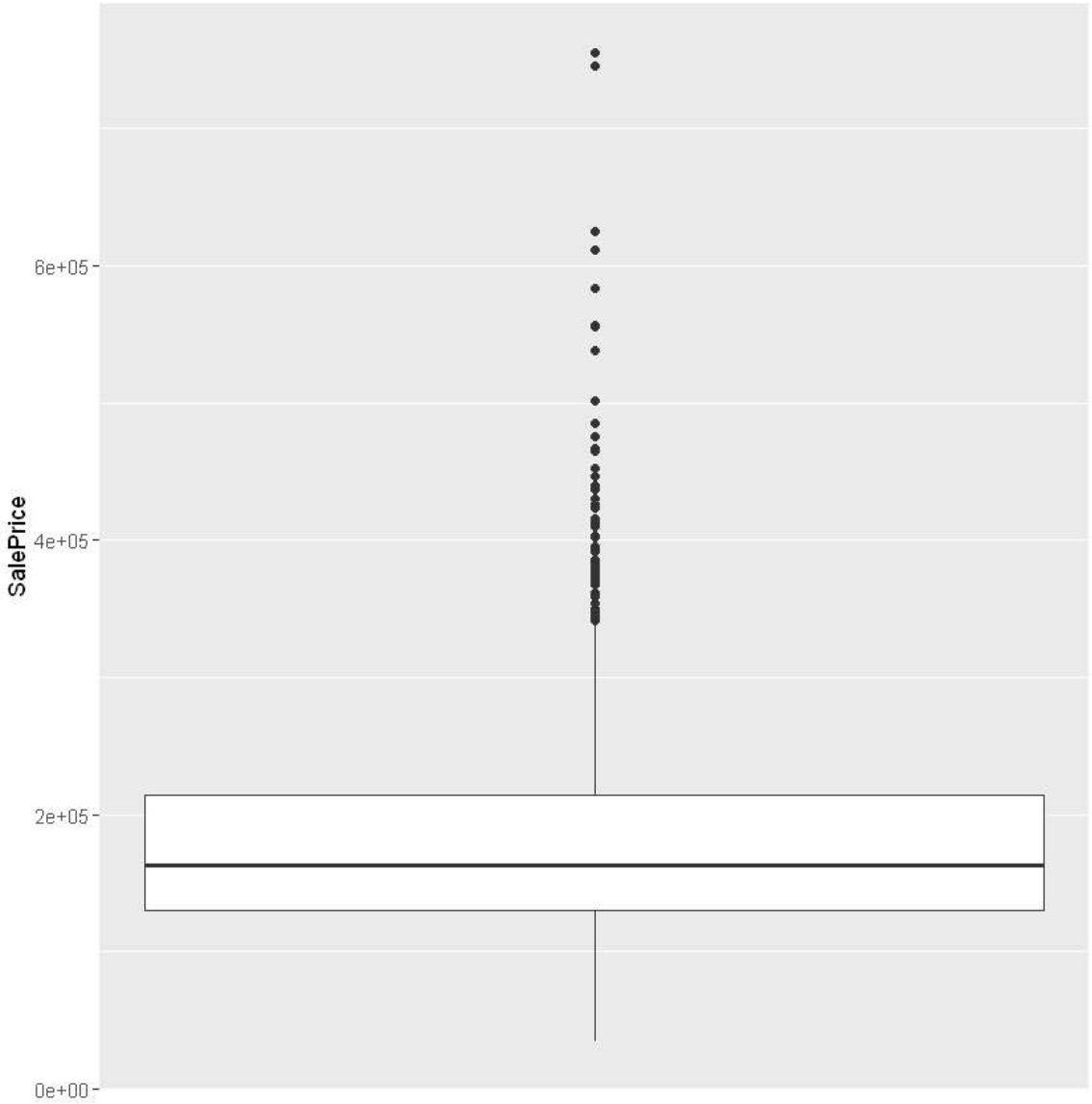
```
saleprice <- train[, 81, drop = FALSE]
library(ggplot2)
ggplot(saleprice, aes(x=SalePrice)) +
    geom_histogram(fill = "skyblue", color = "black")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



In [18]:

```
ggplot(saleprice, aes(x = 1, y = SalePrice))+
    geom_boxplot()+
    scale_x_continuous(breaks=NULL)+
    theme(axis.title.x = element_blank())
```

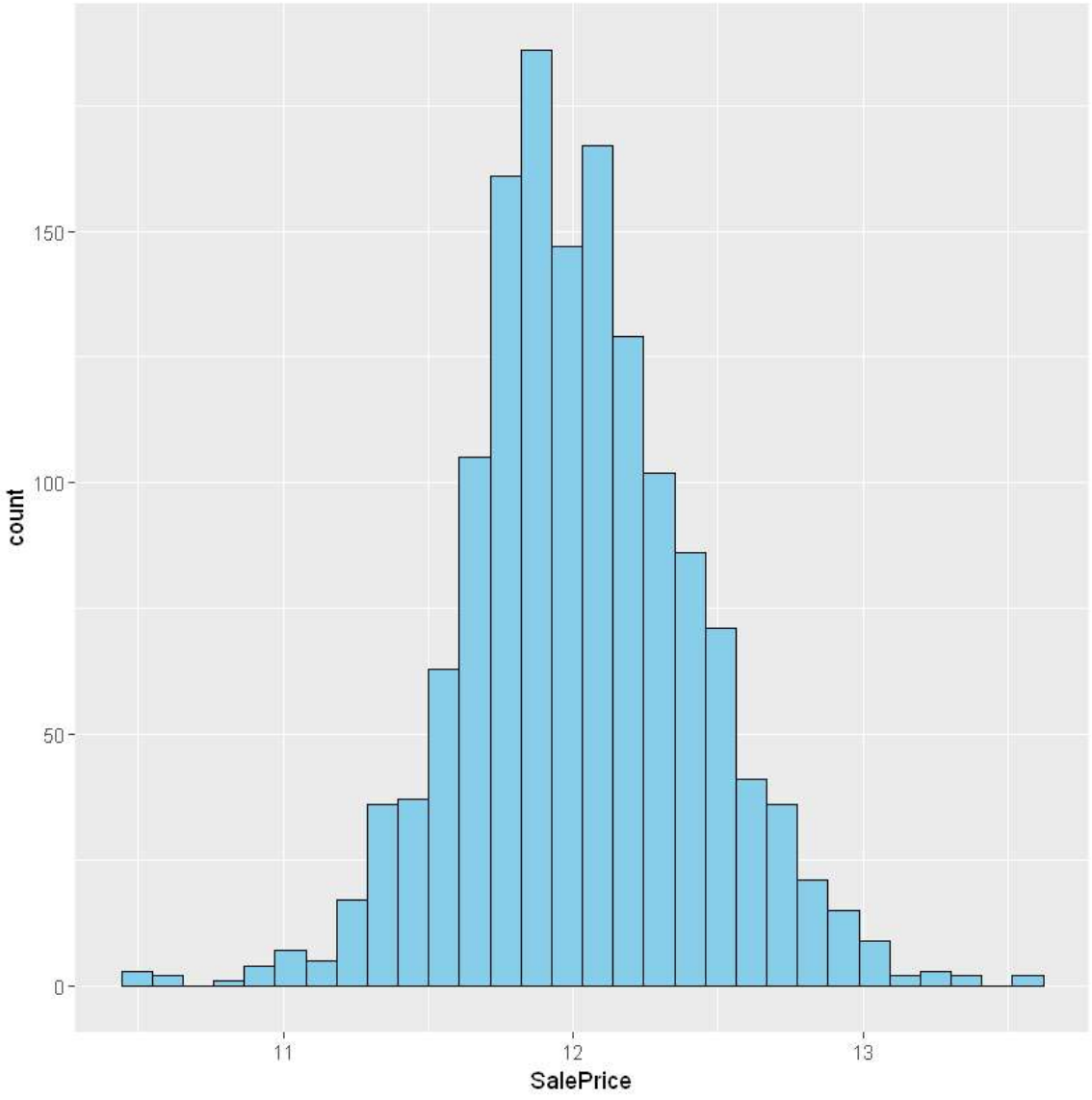


可见房价的极大值和极小值分布及不对阵。把房价取对数观察一下

In [19]:

```
saleprice_log <- log(saleprice)
ggplot(saleprice_log, aes(x=SalePrice)) +
    geom_histogram(fill = "skyblue", color = "black")
```

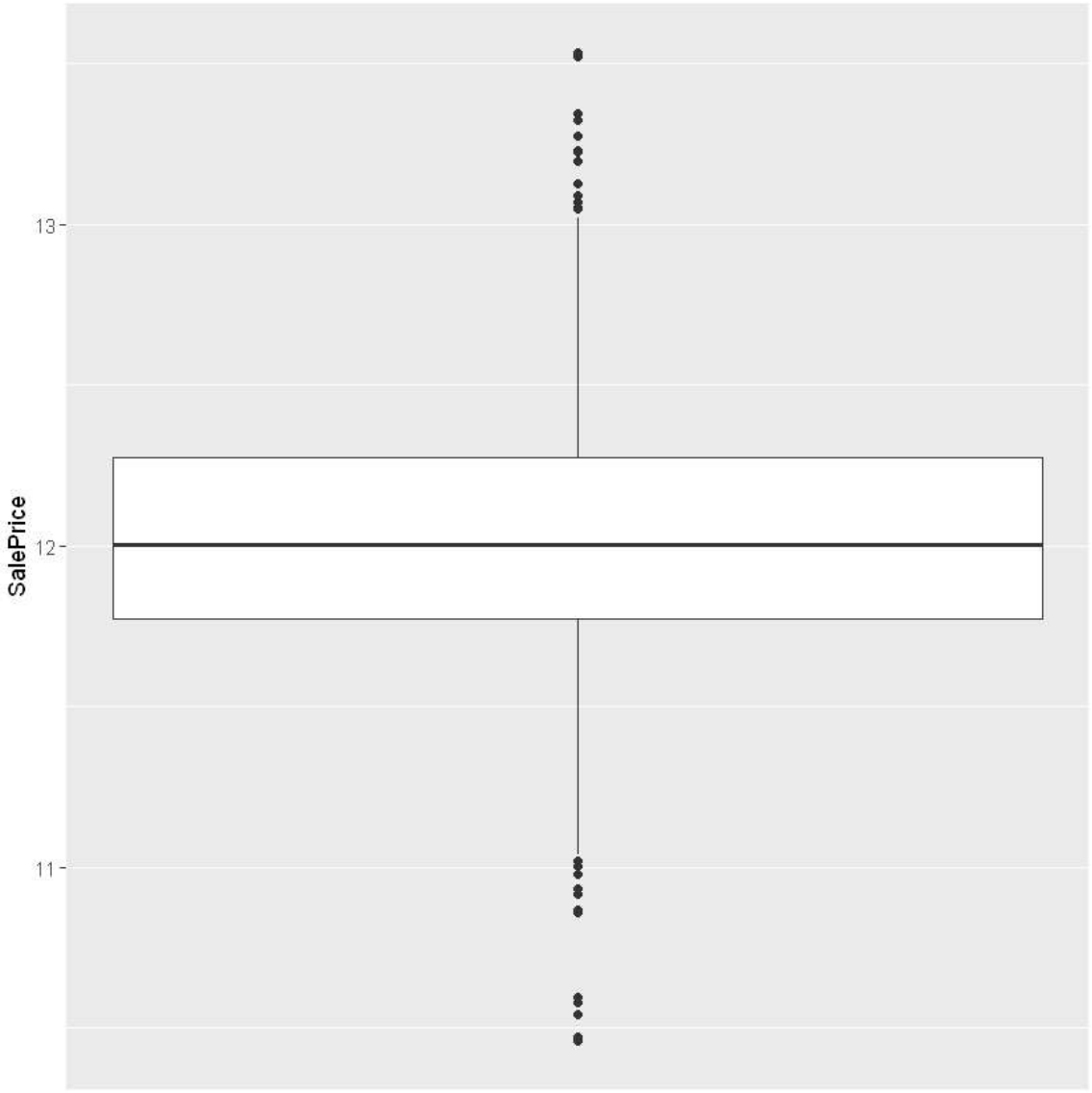
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



基本上呈正态分布，再观察一下箱线图

In [20]:

```
ggplot(saleprice_log, aes(x = 1, y = SalePrice))+
    geom_boxplot()+
    scale_x_continuous(breaks=NULL)+
    theme(axis.title.x = element_blank())
```



可见极大值和极小值的影响现在就是对称的，那么如果我们预测房价的值预测过大或者过小的影响就是对称的了。这样对房价的影响更小。然后用取对数正则化的房价数据进行建模预测。

In [21]:

```
last <- cbind(last1, last2)
last <- cbind(last, saleprice_log[,1, drop = FALSE])
library(rpart)
pred = predict(model, test)
pred <- as.data.frame(pred)
pred <- exp(pred)
ans <- cbind(test$Id, pred)
names(ans) <- c("Id", "SalePrice")
write.csv(ans, "sample_submission.csv")
```

用正则化过后的数据提交，得分0.2444，进步32名