聚成三类对医生评价作图

1.数据预处理阶段

这里使用python进行处理，pandas库是主要的工具
先导入数据，查看数据排列结构，并且处理缺失值

In [1]:

```
import pandas as pd
df = pd.read_excel("grade.xlsx", sheet_name = "E Ye CG", index_col = "MED Ye CG")
df = df.fillna(0)
df.head()
```

Out[1]:

	A1_BL_YCG	A1_D1_YCG	A1_D2_YCG	A1_D4_YCG	A1_D5_YCG	A1_D6_YC
MED Ye CG						
1	0	0.0	0	0	0.0	0.0
3	0	0.0	0	0	0.0	0.0
4	0	0.0	0	0	0.0	0.0
6	0	0.0	0	0	0.0	0.0
7	0	0.0	0	0	0.0	0.0

5 rows × 162 columns

根据分类信息，也就是表格的最后一列，截取数据子集
pandas中使用groupby实现，R中使用subset函数实现

In [2]:

```
df1 = df.groupby('hcw').get_group(0)
df2 = df.groupby('hcw').get_group(1)
df3 = df.groupby('hcw').get_group(2)
```

1.1处理第一个分类

1.1.1以A5为例先处理一个孔的数据

将A5变为我们需要的数据格式，先刻画我们需要的数据形式

In [3]:

```
ks = pd.DataFrame(columns = ['kong', 'day', 'grade', 'freq'])
ks
```

Out[3]:

	kong	day	grade	freq

各列含义解释：
kong:孔位信息，就是A1～A6，A7是参照系不做考虑
day:时间节点信息，D1～W96，(每个孔下22个时间节点)
grade:医生评级，对应着7个水平的值，（每个时间节点下的医生有7个评级）
freq:每个医生评级对应的频数，（含义：每个孔下某一个时间节点，该时间节点有7个水平的医生评级，每个评级对应着一个频数）

1. 从df1中截取数据，截取A5孔从D1至W96的全部数据

In [4]:

```
kf = df1.loc[:,'A5_D1_YCG':'A5_W96_YCG']
kf.head()
```

Out[4]:

	A5_D1_YCG	A5_D2_YCG	A5_D4_YCG	A5_D5_YCG	A5_D6_YCG	A5_D7_YC
MED Ye CG						
1	2.0	2.0	1.0	1.0	1.0	1.0
8	2.0	3.0	2.0	2.0	2.0	2.0
10	1.5	1.0	1.0	1.0	1.0	1.0
11	1.0	2.0	2.0	2.0	2.0	2.0
12	2.0	2.0	2.0	2.0	1.0	2.0

5 rows × 22 columns

1. 统计每一列数据的频数：例如有多少个0，多少个1，多少个2...这一类信息
   使用pandas里series对象的value_counts()方法可以实现计数的功能
   但这里是一个dataframe对象，dataframe对象的每一列又是一个series对象
   所以可以对dataframe的每一列使用value_counts()方法，然后使用dataframe的apply()函数将这个方法用于全部列

In [5]:

```
kf = kf.apply(lambda x: x.value_counts())
kf
```

Out[5]:

	A5_D1_YCG	A5_D2_YCG	A5_D4_YCG	A5_D5_YCG	A5_D6_YCG	A5_D7_YCC
0.0	NaN	NaN	NaN	NaN	NaN	NaN
0.5	1.0	1.0	1.0	2.0	1.0	1.0
1.0	2.0	1.0	4.0	3.0	5.0	3.0
1.5	1.0	NaN	NaN	NaN	NaN	NaN
2.0	5.0	6.0	5.0	4.0	4.0	6.0
3.0	1.0	2.0	NaN	1.0	NaN	NaN

6 rows × 22 columns

1. 统计数据中虽然把出现过的数字0,0.5,1.0,1.5,2.0,3.0都统计到了，但是其他医生评级数据例如2.5就没有统计到
   虽然实际上2.5这个评级在该数据集中没出现过，但这个评级应该被统计为0
   下面解决这个问题,这里使用了datafarme对象的join()方法

In [6]:

```
grade = pd.DataFrame(index = [0, 0.5, 1, 1.5, 2, 2.5, 3])
kf = grade.join(kf)
kf
```

Out[6]:

	A5_D1_YCG	A5_D2_YCG	A5_D4_YCG	A5_D5_YCG	A5_D6_YCG	A5_D7_YCC
0.0	NaN	NaN	NaN	NaN	NaN	NaN
0.5	1.0	1.0	1.0	2.0	1.0	1.0
1.0	2.0	1.0	4.0	3.0	5.0	3.0
1.5	1.0	NaN	NaN	NaN	NaN	NaN
2.0	5.0	6.0	5.0	4.0	4.0	6.0
2.5	NaN	NaN	NaN	NaN	NaN	NaN
3.0	1.0	2.0	NaN	1.0	NaN	NaN

7 rows × 22 columns

1. 数据中有很多NaN，这是因为没有统计到这些数据，从而可以把他们填充为0

In [7]:

```
kf = kf.fillna(0)
```

1. 在该数据集后面加上一列grade，表示医生的评级信息，实际上该信息和我们的行索引是相同的

In [8]:

```
kf.loc[:,'grade'] = kf.index
kf
```

Out[8]:

	A5_D1_YCG	A5_D2_YCG	A5_D4_YCG	A5_D5_YCG	A5_D6_YCG	A5_D7_YCC
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.5	1.0	1.0	1.0	2.0	1.0	1.0
1.0	2.0	1.0	4.0	3.0	5.0	3.0
1.5	1.0	0.0	0.0	0.0	0.0	0.0
2.0	5.0	6.0	5.0	4.0	4.0	6.0
2.5	0.0	0.0	0.0	0.0	0.0	0.0
3.0	1.0	2.0	0.0	1.0	0.0	0.0

7 rows × 23 columns

1. 化长数据为宽数据，R中reshape包提供了melt()函数，同样pandas中也提供了melt()函数
   注意：pandas中melt()函数不是对象方法，调用方式是：（模块名.函数名）
   melt()函数的参数有：id_vars，var_name

In [9]:

```
pf = pd.melt(kf, id_vars = 'grade', var_name = 'day', value_name = 'freq')
pf.tail()
```

Out[9]:

	grade	day	freq
149	1.0	A5_W96_YCG	0.0
150	1.5	A5_W96_YCG	0.0
151	2.0	A5_W96_YCG	0.0
152	2.5	A5_W96_YCG	0.0
153	3.0	A5_W96_YCG	0.0

上述操作的含义是：每个grade有一个column值对应的属性，把这个属性命名为'day'，每个属性对应着一个值，把这个值命名为freq
具体讲就是：医生评级对应一个时点，每个时点对应这这个评级的频数。

1. 完成我们最终需要的数据集合，也就是ks所展现的结构

In [10]:

```
pf = ks.append(pf, sort=False)
pf.tail(7)
```

Out[10]:

	kong	day	grade	freq
147	NaN	A5_W96_YCG	0.0	10.0
148	NaN	A5_W96_YCG	0.5	0.0
149	NaN	A5_W96_YCG	1.0	0.0
150	NaN	A5_W96_YCG	1.5	0.0
151	NaN	A5_W96_YCG	2.0	0.0
152	NaN	A5_W96_YCG	2.5	0.0
153	NaN	A5_W96_YCG	3.0	0.0

1. 添加孔位信息，把孔位信息填充为‘A5’

In [11]:

```
pf.kong = 'A5'
```

1. 把day的信息进行调整，例如A5_W96_YCG改为W96
   利用字符串切片功能实现

In [12]:

```
pf.day = pf.day.apply(lambda x: x[3:-4])
pf.head()
```

Out[12]:

	kong	day	grade	freq
0	A5	D1	0.0	0.0
1	A5	D1	0.5	1.0
2	A5	D1	1.0	2.0
3	A5	D1	1.5	1.0
4	A5	D1	2.0	5.0

1.2设计循环处理所有的孔位

上面我们成功地将数据变成了我们想要的结构
但是我们只处理了一个孔位的数据，现在我们需要处理6个孔的全部数据
df1是第一类人群的全部数据，我们以df1作为处理的起点

第一步：存储数据的容器——需要一个容器存储数据

In [13]:

```
ct = pd.DataFrame(columns=['kong','day','grade','freq'])
```

第二步：循环迭代，主要切片循环机制的设计

In [14]:

```
for i in range(1,7):
    cf = df1.loc[:,'A'+str(i)+'_D1_YCG':'A'+str(i)+'_W96_YCG']
    cf = cf.apply(lambda x: x.value_counts())
    grade = pd.DataFrame(index=[0,0.5,1,1.5,2,2.5,3])
    cf = grade.join(cf, sort=False)
    cf = cf.fillna(0)
    cf.loc[:,'grade'] = cf.index
    cf = pd.melt(cf, id_vars='grade', var_name='day', value_name='freq')
    col = pd.DataFrame(columns=['kong','day','grade','freq'])
    cf = col.append(cf, sort=False)
    cf.day = cf.day.apply(lambda x: x[3:-4])
    cf.kong = 'A'+str(i)
    ct = ct.append(cf, sort=False)
```

In [15]:

```
ct.tail()
```

Out[15]:

	kong	day	grade	freq
149	A6	W96	1.0	0.0
150	A6	W96	1.5	0.0
151	A6	W96	2.0	0.0
152	A6	W96	2.5	0.0
153	A6	W96	3.0	0.0

1.3完成其他类人群数据的读取

其实是上面代码重复
第二类人群的数据处理

In [16]:

```
at = pd.DataFrame(columns=['kong','day','grade','freq'])#要修改容器
for i in range(1,7):
    cf = df2.loc[:,'A'+str(i)+'_D1_YCG':'A'+str(i)+'_W96_YCG']#需要修改
    cf = cf.apply(lambda x: x.value_counts())
    grade = pd.DataFrame(index=[0,0.5,1,1.5,2,2.5,3])
    cf = grade.join(cf, sort=False)
    cf = cf.fillna(0)
    cf.loc[:,'grade'] = cf.index
    cf = pd.melt(cf, id_vars='grade', var_name='day', value_name='freq')
    col = pd.DataFrame(columns=['kong','day','grade','freq'])
    cf = col.append(cf, sort=False)
    cf.day = cf.day.apply(lambda x: x[3:-4])
    cf.kong = 'A'+str(i)
    at = at.append(cf, sort=False)#要修改容器
at.head()
```

Out[16]:

	kong	day	grade	freq
0	A1	D1	0.0	10.0
1	A1	D1	0.5	1.0
2	A1	D1	1.0	0.0
3	A1	D1	1.5	0.0
4	A1	D1	2.0	0.0

第三类人群的数据处理

In [17]:

```
bt = pd.DataFrame(columns=['kong','day','grade','freq'])#要修改容器
for i in range(1,7):
    cf = df3.loc[:,'A'+str(i)+'_D1_YCG':'A'+str(i)+'_W96_YCG']#需要修改
    cf = cf.apply(lambda x: x.value_counts())
    grade = pd.DataFrame(index=[0,0.5,1,1.5,2,2.5,3])
    cf = grade.join(cf, sort=False)
    cf = cf.fillna(0)
    cf.loc[:,'grade'] = cf.index
    cf = pd.melt(cf, id_vars='grade', var_name='day', value_name='freq')
    col = pd.DataFrame(columns=['kong','day','grade','freq'])
    cf = col.append(cf, sort=False)
    cf.day = cf.day.apply(lambda x: x[3:-4])
    cf.kong = 'A'+str(i)
    bt = bt.append(cf, sort=False)#修改容器
bt.tail()
```

Out[17]:

	kong	day	grade	freq
149	A6	W96	1.0	0.0
150	A6	W96	1.5	0.0
151	A6	W96	2.0	0.0
152	A6	W96	2.5	0.0
153	A6	W96	3.0	0.0

2.画图——数据准备阶段

上面的数据预处理已经解决了数据的格式问题。现在，我们要在R中使用ggplot2进行作图，涉及到从pandas的dataframe到R的dataframe，也涉及从python到R的转换。
这里采用的策略是把我们之前处理好的数据写入磁盘，然后用R程序再次从文件中读取。

In [18]:

```
ct.to_csv('ou1.csv')
at.to_csv('ou2.csv')
bt.to_csv('ou3.csv')
```

切换服务jupyter的程序为R，用R提供的函数读取三个.csv文件

In [7]:

```
df1 <- read.table('ou1.csv', sep = ',', header = TRUE)
df2 <- read.table('ou2.csv', sep = ',', header = TRUE)
df3 <- read.table('ou3.csv', sep = ',', header = TRUE)
```

In [8]:

```
head(df1)
```

X	kong	day	grade	freq
0	A1	D1	0.0	9
1	A1	D1	0.5	1
2	A1	D1	1.0	0
3	A1	D1	1.5	0
4	A1	D1	2.0	0
5	A1	D1	2.5	0

其中X是多余的列，产生于pandas数据框保存为csv文件过程中，需要去除

In [10]:

```
df1 <- subset(df1, select = -X)
df2 <- subset(df2, select = -X)
df3 <- subset(df3, select = -X)
```

In [11]:

```
tail(df1)
```

	kong	day	grade	freq
919	A6	W96	0.5	0
920	A6	W96	1.0	0
921	A6	W96	1.5	0
922	A6	W96	2.0	0
923	A6	W96	2.5	0
924	A6	W96	3.0	0

设定day的因子水平

In [14]:

```
lev <- c('D1','D2','D4','D5','D6','D7','W2','W3','W4','W6','W8','W10','W12','W16','W20','W24','W
30','W36','W42','W48','W78','W96')
df1$day <- factor(df1$day, levels = lev)
df2$day <- factor(df2$day, levels = lev)
df3$day <- factor(df3$day, levels = lev)
```

设定grade的因子水平

In [15]:

```
grade <- c(3,2.5,2,1.5,1,0.5,0)
df1$grade <- factor(df1$grade, levels = grade)
df2$grade <- factor(df2$grade, levels = grade)
df3$grade <- factor(df3$grade, levels = grade)
```

3.画图

3.1画其中第一幅图

1. 导入ggplot2包

In [16]:

```
library(ggplot2)
```

1. 创建ggplot对象

In [18]:

```
p1 <- ggplot(df1, aes(x = day, y = freq, fill = grade))
```

1. 添加几何对象为bar，方法为identity

In [27]:

```
p1 <- p1 + geom_bar(stat = 'identity', width = 1, colour = "black")
```

1. 按照kong进行分面

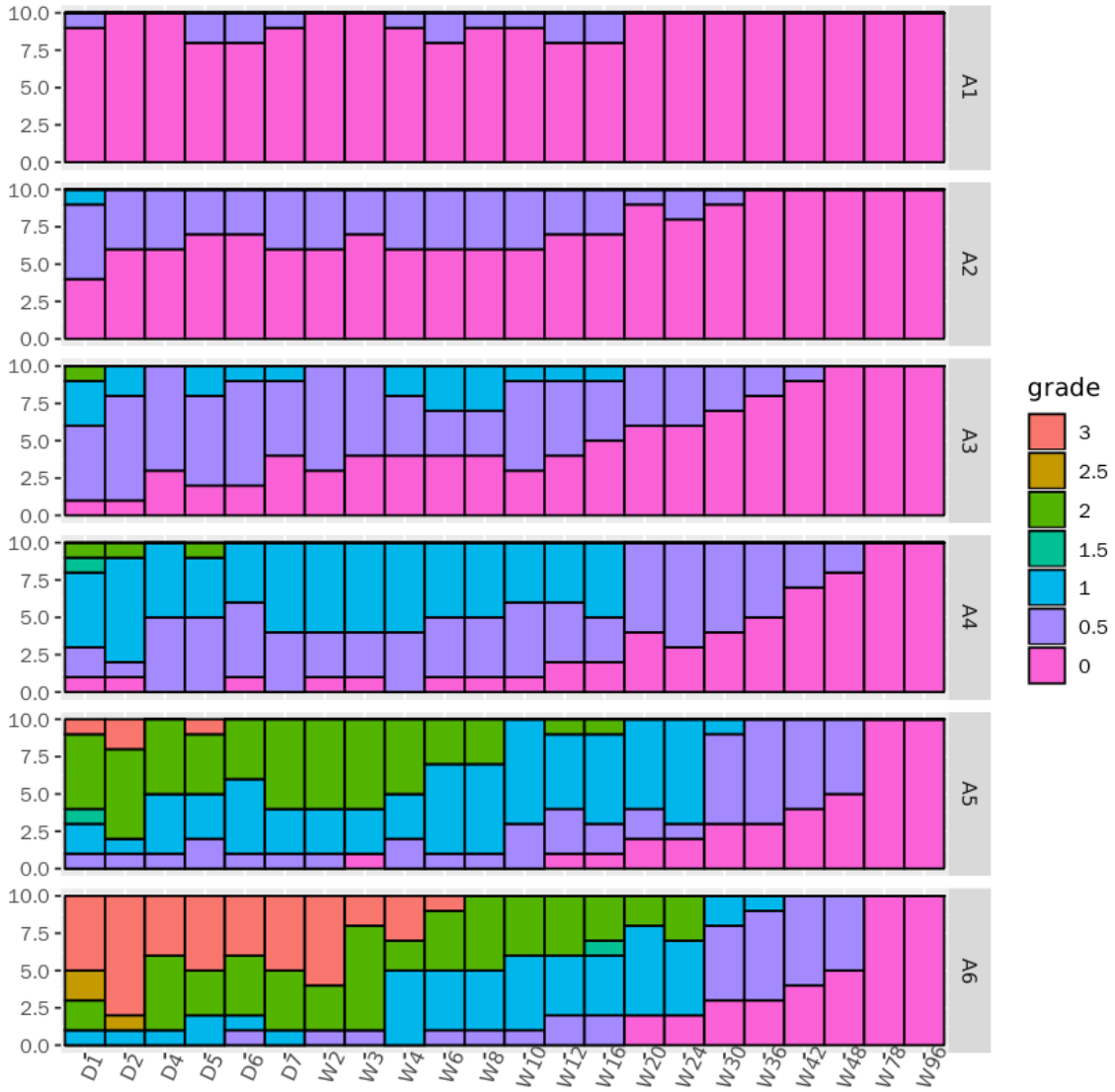
In [28]:

```
p1 <- p1 + facet_grid(kong ~ .)
```

1. 美化：去除x轴和y轴的label，同时修改刻度文字的排列

In [29]:

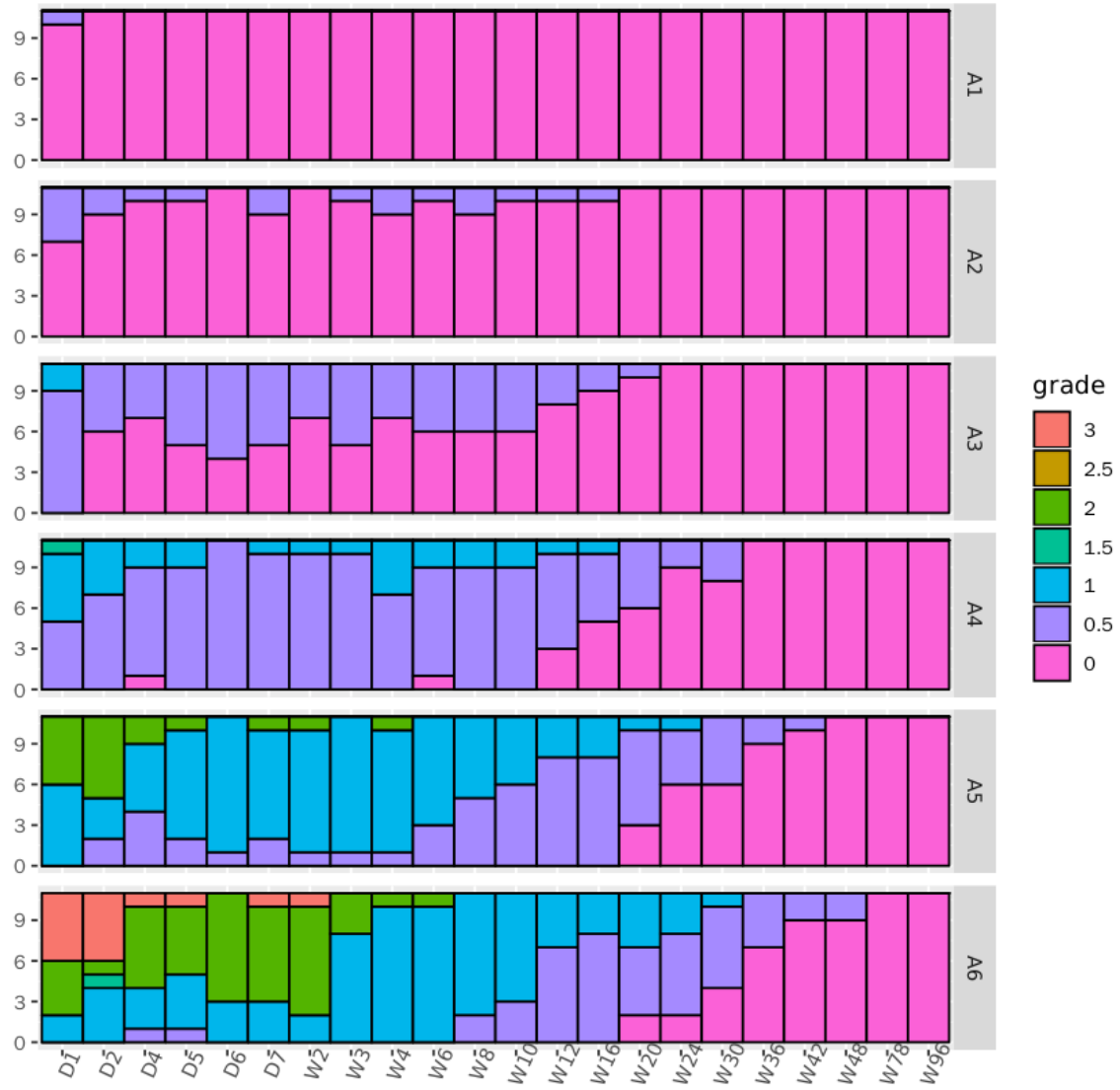
```
p1 + theme(axis.text.x = element_text(angle = 65)) + theme(axis.title = element_blank())
```



3.2对所有数据作图

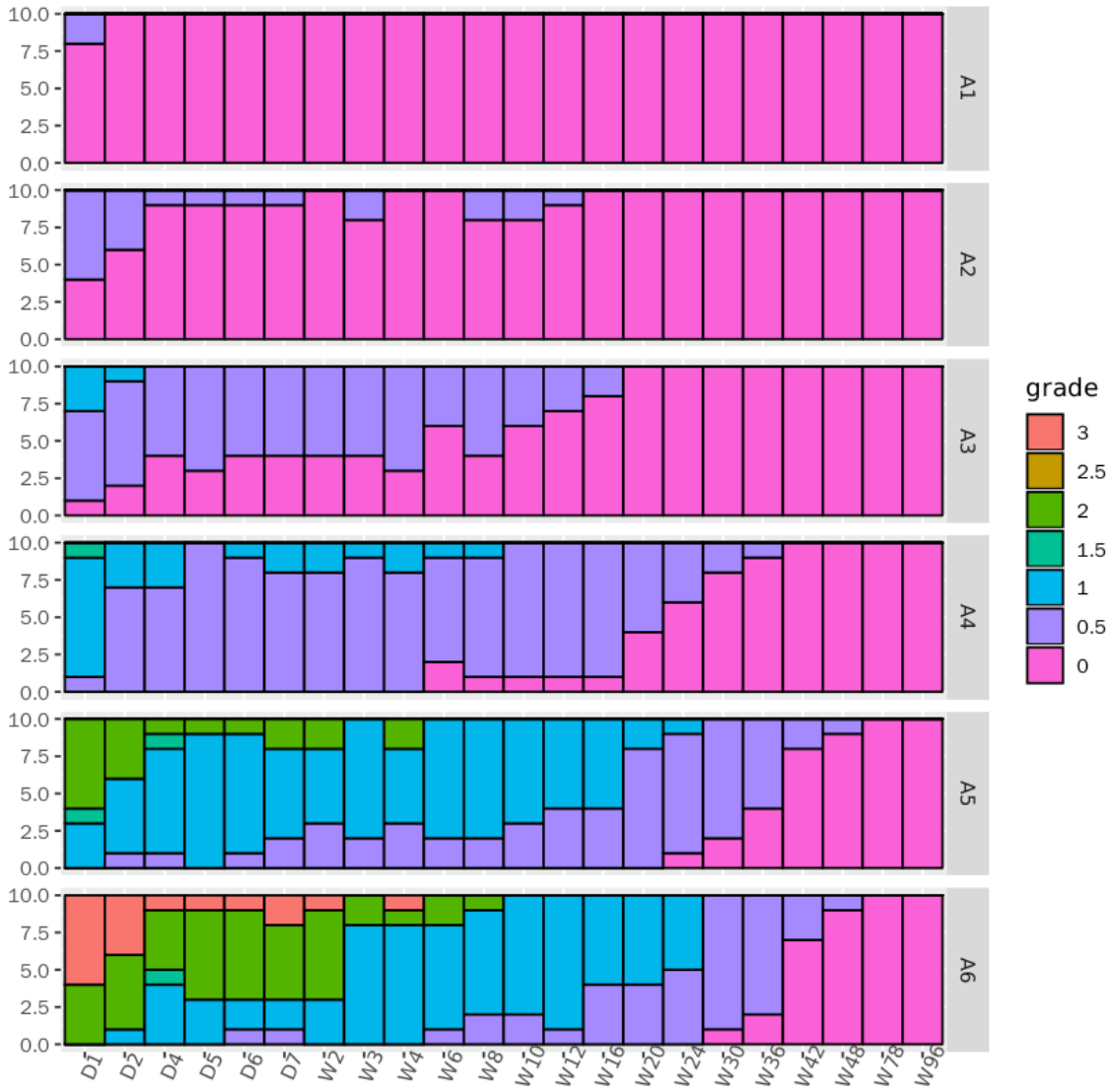
In [30]:

```
ggplot(df2, aes(x = day, y = freq, fill = grade)) +
geom_bar(stat = 'identity',width = 1, colour = "black") + facet_grid(kong ~ .) +
theme(axis.text.x = element_text(angle = 65)) +
theme(axis.title = element_blank())
```



In [31]:

```
ggplot(df3, aes(x = day, y = freq, fill = grade)) +
geom_bar(stat = 'identity', width = 1, colour = "black") + facet_grid(kong ~ .) +
theme(axis.text.x = element_text(angle = 65)) +
theme(axis.title = element_blank())
```



4.总结

关键:理解ggplot2的作图才是本文的关键，我们最终需要的图：横坐标是每一个时间点，纵轴表示的是频数，颜色映射的是医生评级。从而这就决定了，我们所需数据表的格式，表示时间点的day因子（顺序有实际意义），表示医生某一个评级的频数freq，频数对应的医生评级是多少。
注意:上述的关键点才是，我思考这个问题的方式，就是先考虑怎么用ggplot2画图，然后再考虑需要什么样子的dataframe格式。最后就是，用相关工具去实现这样的dataframe。先考虑**ggplot2**画图怎么实现的，然后就是构造符合要求的**data**。