附錄

附錄一 簡易插補法(不考慮葛特曼量表性質)

令 $N_{(i,j,k,l)}$表示回答(i,j,k,l)的個數，其中$(i,j,k,l) = (1,1,1,1)$，$(1,1,1,2)$，$(1,1,2,2)$，$(1,2,2,2)$及$(2,2,2,2)$，$N = N_{(1,1,1,1)}+N_{(1,1,1,2)}+N_{(1,1,2,2)} +N_{(1,2,2,2)} +N_{(2,2,2,2)}$。$n_{(a,b,c,d)}$表示拒答形式為$(a,b,c,d)$的個數，$a,b,c,d$=1,2,8 且 a,b,c,d 至少有一個為 8。又令 $T_{(i,j,k,l)}$表示可以插補成(i,j,k,l)的拒答形式之總數，例如 $T_{(1,1,1,1)}= \sum_{a,b,c,d=1,8} n_{(a,b,c,d)}$。拒答率為 r。

(1)將拒答以容許插補，即以 1 取代所有的 8，則插補的正確率為

$$r\{N_{(1,1,1,1)} + N_{(1,1,1,2)} * \frac{n_{(1,1,8,2)} + n_{(1,8,8,2)} + n_{(8,8,8,2)}}{T_{(1,1,1,2)}} + N_{(1,1,2,2)} * \frac{n_{(1,8,2,2)}}{T_{(1,1,2,2)}} + N_{(1,2,2,2)} * \frac{n_{(8,2,2,2)}}{T_{(1,2,2,2)}}\} / rN \tag{1.1}$$

，其中，$N_{(1,1,1,1)} = N_{(1,1,1,1)}*(T_{(1,1,1,1)}/T_{(1,1,1,1)})$，因為分子分母相同而可約分。

(2)將拒答以不容許插補，即以 2 取代所有的 8，則插補的正確率為

$$r\{N_{(1,1,2,2)}*\frac{n_{(1,1,8,8)} + n_{(1,1,8,2)}}{T_{(1,1,2,2)}} + N_{(1,2,2,2)}*\frac{n_{(1,8,8,8)} + n_{(1,8,2,2)} + n_{(1,8,8,2)} + n_{(1,2,8,8)}}{T_{(1,2,2,2)}} + N_{(2,2,2,2)}\}/rN \tag{2.1}$$

，其中，$N_{(2,2,2,2)} = N_{(2,2,2,2)}*(T_{(1,2,2,2)}/T_{(1,2,2,2)})$，因為分子分母相同而可約分。

(3)將拒答以多數回答插補，四題的多數回答依序為 1、2、2、2，分別插補拒答對應位置，插補的正確率為

$$r\{N_{(1,1,2,2)}*\frac{n_{(1,1,8,8)} + n_{(1,1,8,2)}}{T_{(1,1,2,2)}} + N_{(1,2,2,2)} + N_{(2,2,2,2)}*\frac{n_{(2,8,8,8)} + n_{(2,2,8,8)}}{T_{(2,2,2,2)}}\}/rN \tag{3.1}$$

，其中，$N_{(1,2,2,2)} = N_{(1,2,2,2)}*(T_{(1,2,2,2)}/T_{(1,2,2,2)})$，因為分子分母相同而可約分。

(4)將拒答以少數回答插補，四題的少數回答依序為 2、1、1、1，分別插補拒答對應位置，插補的正確率為

$$r\{N_{(1,1,1,1)}*\frac{n_{(1,1,8,8)}+n_{(1,8,8,8)}}{T_{(1,1,1,1)}}+N_{(1,1,1,2)}*\frac{n_{(1,1,8,2)}+n_{(1,8,8,2)}}{T_{(1,1,2,2)}}+N_{(1,1,2,2)}*\frac{n_{(1,8,2,2)}}{T_{(1,1,2,2)}}\}/rN \qquad \textbf{(4.1)}$$

實際帶入結果如下：

	容許	不容許	多數	少數
正確率	0.2316	***0.3107***	***0.2901***	0.0376

附錄表 1 正確率計算(不考慮葛特曼量表特性)

與表 4.3 比較可發現以不容許及多數回答插補的正確率相同，因為不論是否考慮葛特曼量表的性質，在這筆資料下這兩種插補法會有相同的結果，例如(1,2,8,8)考慮葛特曼量表性質應直接補為(1,2,2,2)，但若不考慮，以不容許及多數回答插補也同樣會補成(1,2,2,2)。不過另外兩種方法的插補結果就顯示，有考慮葛特曼量表性質的插補會有比較高的正確率。

附錄二 R 製造拒答的程式碼(拒答率 5%)

```
data1=read.table("D:/(1111).txt",header=T,sep="")
attach(data1)
M1=sample(number,19,replace=FALSE)#抽出要將(1,1,1,1)製造成拒答的號碼
M1for8888=M1[c(1:15)]#製造成 8888 的號碼
M1for1888=M1[c(16:19)]#製造成 1888 的號碼
data1[M1for8888,c(7:10)]=8
data1[M1for1888,c(8:10)]=8

data2=read.table("D:/(1112).txt",header=T,sep="")
attach(data2)
M2=sample(number,7,replace=FALSE)#抽出要將(1,1,1,2)製造成拒答的號碼
M2for8888=M2[c(1:5)]#製造成 8888 的號碼
M2for1888=M2[6] #製造成 1888 的號碼
M2for1182=M2[7] #製造成 1182 的號碼
data2[M2for8888,c(7:10)]=8
data2[M2for1888,c(8:10)]=8
data2[M2for1182,9]=8

data3=read.table("D:/(1122).txt",header=T,sep="")
attach(data3)
M3=sample(number,16,replace=FALSE)#抽出要將(1,1,2,2)製造成拒答的號碼
M3for8888=M3[c(1:11)]#製造成 8888 的號碼
M3for1888=M3[c(12:14)]#製造成 1888 的號碼
M3for1182=M3[c(15:16)]#製造成 1182 的號碼
data3[M3for8888,c(7:10)]=8
data3[M3for1888,c(8:10)]=8
data3[M3for1182,9]=8

data4=read.table("D:/(1222).txt",header=T,sep="")
attach(data4)
M4=sample(number,23,replace=FALSE)#抽出要將(1,2,2,2)製造成拒答的號碼
M4for8888=M4[c(1:18)]#製造成 8888 的號碼
M4for1888=M4[c(19:22)]#製造成 1888 的號碼
M4for8882=M4[23]#製造成 8882 的號碼
data4[M4for8888,c(7:10)]=8
data4[M4for1888,c(8:10)]=8
```

```
data4[M4for8882,c(7:9)]=8

data5=read.table("D:/(2222).txt",header=T,sep="")
attach(data5)
M5=sample(number,24,replace=FALSE)#抽出要將(2,2,2,2)製造成拒答的號碼
M5for8888=M5[c(1:18)]#製造成 8888 的號碼
M5for2888=M5[c(19:21)]#製造成 2888 的號碼
M5for8882=M5[22]#製造成 8882 的號碼
M5for2288=M5[c(23:24)]#製造成 2288 的號碼
data5[M5for8888,c(7:10)]=8
data5[M5for2888,c(8:10)]=8
data5[M5for8882,c(7:9)]=8
data5[M5for2288,c(9:10)]=8
data=rbind(data1,data2,data3,data4,data5)#合併成一組資料
```

附錄三 R 執行多重插補法的程式碼(模式一，拒答率 5%)

```
library(mice)
H=mice(data[,2:7],m=3,im="logreg")
H=complete(H,"long")
H=cbind(H[,1:6],data[,8])
H[(H[,6]==0),7]=0
#利用邏輯斯迴歸的模型來插補 H，插補三次，若 H 插補成 0(不容許)，則 K 必為 0。

K=mice(H,m=3,im="logreg")
K=complete(K,"long")
K=cbind(K[,1:7],data[,9])
K[(K[,7]==0),8]=0
#利用邏輯斯迴歸的模型來插補 K，插補三次，若 K 插補成 0(不容許)，則 C 必為 0。

C=mice(K,m=3,im="logreg")
C=complete(C,"long")
C=cbind(C[,1:8],data[,10])
C[(C[,8]==0),9]=0
#利用邏輯斯迴歸的模型來插補 C，插補三次，若 C 插補成 0(不容許)，則 S 必為 0。

S=mice(C,m=3,im="logreg")
S=complete(S,"long")
ALL=cbind(S[,1:9],data[,11])
#將插補後所有資料合併。

pattern=1000*ALL[,6]+100*ALL[,7]+10*ALL[,8]+ALL[,9]
#將四個問題合併為一個，分別標記為 1111、1110、1100、1000，及 0000。

T=table(pattern,ALL[,10])
#產生插補前後的列連表。

p5=1-sum(c(T[1,1:4],T[2,1:3],T[2,5],T[3,1:2],T[3,4:5],T[4,1],T[4,3:5],T[5,2:5]))/(81*89)
#用 1 扣除回答錯誤的比例即為正確率。
```

附錄四 R 執行最鄰近插補法的程式碼(拒答率 5%)

```
data1=data[((data[,7]<=2)+(data[,8]<=2)+(data[,9]<=2)+(data[,10]<=2))>=4,2:6]#不含拒答部份。

data=data[((data[,7]>2)+(data[,8]>2)+(data[,9]>2)+(data[,10]>2))!=0,]#拒答部份。

oo1=order(data[1:19,7],data[1:19,8],data[1:19,9],data[1:19,10],decreasing=T)
z1=cbind(data[1:19,2],data[1:19,3],data[1:19,4],data[1:19,5],data[1:19,6])#排序(1,1,1,1)部分。

oo2=order(data[20:26,7],data[20:26,8],data[20:26,9],data[20:26,10],decreasing=T)
z2=cbind(data[20:26,2],data[20:26,3],data[20:26,4],data[20:26,5],data[20:26,6])#排序(1,1,1,2)部份。

oo3=order(data[27:42,7],data[27:42,8],data[27:42,9],data[27:42,10],decreasing=T)
z3=cbind(data[27:42,2],data[27:42,3],data[27:42,4],data[27:42,5],data[27:42,6])#排序(1,1,2,2)部分。

oo4=order(data[43:65,7],data[43:65,8],data[43:65,9],data[43:65,10],decreasing=T)
z4=cbind(data[43:65,2],data[43:65,3],data[43:65,4],data[43:65,5],data[43:65,6])#排序(1,2,2,2)部分。

oo5=order(data[66:89,7],data[66:89,8],data[66:89,9],data[66:89,10],decreasing=T)
z5=cbind(data[66:89,2],data[66:89,3],data[66:89,4],data[66:89,5],data[66:89,6])#排序(2,2,2,2)部分。

a=rbind(z1[oo1,],z2[oo2,],z3[oo3,],z4[oo4,],z5[oo5,])#所有包含拒答資料。

s=NULL
for (i in 1:1696)
for (j in 1:89)
{ b=(ifelse(a[j,1]==data1[i,1],0,1)+abs(a[j,2]-data1[i,2])/(max(data1[,2])-min(data1[,2]))+ifelse(a[j,3]==data1[i,3],0,1)+abs(a[j,4]-data1[i,4])/(max(data1[,4])-min(data1[,4]))+abs(a[j,5]-data1[i,5])/(max(data1[,5])-min(data1[,5])))/5
s=c(s,b)}
out=matrix(s,nrow=1696,byrow=T)
#依據混合距離公式計算 1696*89 筆資料距離。
```

```
s1.8888=NULL
for (i in 1:15)
{b=ifelse(min(out[,i])==min(out[1:380,i]),1,0)
s1.8888=c(s1.8888,b)}
#找出原本回答(1,1,1,1)被製造成(8,8,8,8)的資料的最鄰近資料點，若其回答為(1,1,1,1)則插補正確，記為 1。

s1.1888=NULL
for (i in 1:4)
{b=ifelse(min(out[1:1254,15+i])==min(out[1:380,15+i]),1,0)
s1.1888=c(s1.1888,b)}
#找出原本回答(1,1,1,1)被製造成(1,8,8,8)的資料的最鄰近資料點，若其回答為(1,1,1,1)則插補正確，記為 1。

s2.8888=NULL
for (i in 1:5)
{b=ifelse(min(out[,19+i])==min(out[381:513,19+i]),1,0)
s2.8888=c(s2.8888,b)}
#找出原本回答(1,1,1,2)被製造成(8,8,8,8)的資料的最鄰近資料點，若其回答為(1,1,1,2)則插補正確，記為 1。

s2.1888=ifelse(min(out[1:1254,25])==min(out[381:513,25]),1,0)
#找出原本回答(1,1,1,2)被製造成(1,8,8,8)的資料的最鄰近資料點，若其回答為(1,1,1,2)則插補正確，記為 1。

s2.1182=ifelse(min(out[382:794,26])==min(out[381:513,26]),1,0)
#找出原本回答(1,1,1,2)被製造成(1,1,8,2)的資料的最鄰近資料點，若其回答為(1,1,1,2)則插補正確，記為 1。

s3.8888=NULL
for (i in 1:11)
{b=ifelse(min(out[,26+i])==min(out[514:793,26+i]),1,0)
s3.8888=c(s3.8888,b)}
#找出原本回答(1,1,2,2)被製造成(8,8,8,8)的資料的最鄰近資料點，若其回答為(1,1,2,2)則插補正確，記為 1。
```

```
s3.1888=NULL
for (i in 1:3)
{b=ifelse(min(out[1:1254,37+i])==min(out[514:793,37+i]),1,0)
s3.1888=c(s3.1888,b)}
#找出原本回答(1,1,2,2)被製造成(1,8,8,8)的資料的最鄰近資料點，若其回答為(1,1,2,2)則插補正確，記為 1。

s3.1182=NULL
for (i in 1:2)
{b=ifelse(min(out[381:793,40+i])==min(out[514:793,40+i]),1,0)
s3.1182=c(s3.1182,b)}
#找出原本回答(1,1,2,2)被製造成(1,1,8,2)的資料的最鄰近資料點，若其回答為(1,1,2,2)則插補正確，記為 1。

s4.8888=NULL
for (i in 1:19)
{b=ifelse(min(out[,42+i])==min(out[794:1254,42+i]),1,0)
s4.8888=c(s4.8888,b)}
#找出原本回答(1,2,2,2)被製造成(8,8,8,8)的資料的最鄰近資料點，若其回答為(1,2,2,2)則插補正確，記為 1。

s4.1888=NULL
for (i in 1:4)
{b=ifelse(min(out[1:1254,61+i])==min(out[794:1254,61+i]),1,0)
s4.1888=c(s4.1888,b)}
#找出原本回答(1,2,2,2)被製造成(1,8,8,8)的資料的最鄰近資料點，若其回答為(1,2,2,2)則插補正確，記為 1。

s5.8888=NULL
for (i in 1:19)
{b=ifelse(min(out[,65+i])==min(out[1255:1696,65+i]),1,0)
s5.8888=c(s5.8888,b)}
#找出原本回答(2,2,2,2)被製造成(8,8,8,8)的資料的最鄰近資料點，若其回答為(2,2,2,2)則插補正確，記為 1。

s5.2888=c(1,1,1,1,1)
#強制(2,8,8,8)要插補正確。
```

```
corrate=sum(c(s1.8888,s1.1888,s2.8888,s2.1888,s2.1182,s3.8888,s3.1888,s3.1182,s4.8888,s4.1888,s5.8888,s5.2888))/89
#加總後除以拒答個數(在此為 89 個)。
```

附錄五 其他模式的多重插補法

(一) 模式二

模式一中進行的多重插補法，是利用人口變數插補變數 H，再利用人口變數以及插補後的 H 的資料來插補變數 K，並且延續這樣的過程直到補完變數 S 為止。現在考慮另一種方法，同樣先利用人口變數插補變數 H，但是接下來只依據插補後的變數 H，來插補變數 K，接著以插補過後的變數 K 插補變數 C，再以插補過後的變數 C 插補變數 S。模擬同樣也是進行 500 次。這麼做的理由是因為考慮葛特曼量表的特性，此外，若是較少的變數也可以插補出不錯的結果，則當然就不需要放入過多的變數。下表將列出這種模式的多重插補法之下，一些常用統計量的結果比較。

拒答比例	5%	10%	15%	20%	25%	30%
最小值	0.2649	0.2852	0.2784	0.2863	0.2886	0.2990
第一四分位數	0.2904	0.3014	0.2959	0.2934	0.2969	0.3098
中位數	0.3003	0.3077	0.3042	0.2982	0.3023	0.3135
第三四分位數	0.3159	0.3162	0.3094	0.3036	0.3069	0.3178
最大值	0.3504	0.3415	0.3304	0.3253	0.3218	0.3323
平均數	**0.3020**	**0.3086**	**0.3031**	**0.2988**	**0.3024**	**0.3134**
標準差	**0.0177**	**0.0123**	**0.0101**	**0.0081**	**0.0074**	**0.0073**

附錄表 2 模式二多重插補法之下的常用統計量比較

由附錄表 2 的結果，只給定一個變數來進行多重插補法，其正確率不論拒答率多少大約都是 30%左右，跟表 4.6 的結果比起來相對比較低了約 2 個百分點。此外若與簡易插補法中的不容許插補比起來，也是比較低的。表示如此的作法可能因為給定的變數較少而損失很多訊息，造成插補正確率較低。不過因為給定的變數比較少，花費的時間也相對較少，研究者若是覺得這樣的減少還可以接受的話，用這樣的多重插補法也不失為一個選擇。

(二) 模式三

跟模式二類似，現在考慮以折衷的方式，也就是說，先以人口變項插補變數 H，再藉由變數 H 來插補 K。接下來不一樣的是，合併變數 H 及 K 的訊息來插補 C，再來藉由變數 H、K 及 C 的訊息來插補 S。如此進行的原因同樣是想比較使用變數的多寡是否真正會使得插補結果不同，甚至變好或變壞。附錄表 3 將列出這種累進變數(人口變項除外)模式的多重插補法之下，一些常用統計量的結果比較。

拒答比例	5%	10%	15%	20%	25%	30%
最小值	0.2340	0.2784	0.2770	0.2763	0.2791	0.2819
第一四分位數	0.2815	0.2954	0.2955	0.2944	0.2969	0.2972
中位數	0.2907	0.3021	0.3035	0.2991	0.2999	0.3026
第三四分位數	0.3067	0.3118	0.3077	0.3043	0.3039	0.3099
最大值	0.3525	0.3313	0.3225	0.3108	0.3125	0.3263
平均數	**0.2934**	**0.3042**	**0.3015**	**0.2983**	**0.2998**	**0.3029**
標準差	**0.0208**	**0.0114**	**0.0102**	**0.0078**	**0.0071**	**0.0080**

附錄表 3 模式三多重插補法之下的常用統計量比較

在這樣的模式下進行多重插補法的結果，大致與第二種模式的結果差不多，其正確率大約也會是 30%左右。跟第一種模式比起來，這種模式在插補後面變數的時候並沒有把人口變項加進去當解釋變數，附錄表 3 的結果表示加上人口變項會有比較好的正確率，因此考慮第四種模式。

(三) 模式四

最後考慮只利用人口變項的訊息分別就 H、K、C 及 S 四個變數來進行多重插補法。即用人口變項插補變數 H，再用人口變項插補變數 K，一直進行到人口變項插補變數 S。利用這樣的插補模式來確認人口變項對於欲插補的變數是否有一定影響。結果將列於附錄表 4。

拒答比例	5%	10%	15%	20%	25%	30%
最小值	0.2608	0.2917	0.3027	0.2940	0.2978	0.3052
第一四分位數	0.3067	0.3112	0.3165	0.3101	0.3093	0.3155
中位數	0.3178	0.3186	0.3240	0.3166	0.3150	0.3212
第三四分位數	0.3322	0.3303	0.3324	0.3239	0.3207	0.3250
最大值	0.3611	0.3506	0.3480	0.3352	0.3336	0.3386
平均數	**0.3183**	**0.3173**	**0.3139**	**0.3166**	**0.3153**	**0.3203**
標準差	**0.0208**	**0.0125**	**0.0107**	**0.0097**	**0.0085**	**0.0086**

附錄表 4 模式四多重插補法之下的常用統計量比較

模式四的插補正確率相較於模式二與三又高了一些，但是比模式一低一些，大約在 31%~32%左右。此結果是可以想見的，跟模式二與三比起來，模式四給定的解釋變數是比較多的，因此可能得到比較高的正確率，但是花費的時間其實跟模式一相差不多，故一般來說不太會建議用這樣的模式來插補。