Problem 5

(a) R code for generating simple randomization and statified block randomization is given below.

```
dat = data.frame(v1=c(1,1,0,1,1,0,1,1,0,0,0,1,1,0,1,0,1,0,1,0),

                  v2=c(0,1,0,0,0,0,0,1,1,1,1,0,1,0,1,1,0,0,0,1),
                  simple=NA,block=NA)

# Simple randomization
set.seed(33)
rnums = runif(20,0,1)
dat[,3] = ifelse(rnums < 0.5,1,0)
```

For block randomization, we apply a more technique. We introduce three more arrays: one to keep track of the current block size for each stratum, one to keep track of the current number of patients in a block for each stratum, and one to keep track of the number of patients assigned to group 1 in the current block for each stratum. We assign patients within each stratum to treatments on a rolling basis. Once a block is full, we randomly choose the next block size to be either 2 or 4.

```
blockSize = rep(0,4)
currCapl = rep(0,4)
currAssign = rep(0,4)


for(i in 1:20){
  # Interpret the 0's and 1's as TRUE and FALSE, respectively
  if(dat[i,1] & dat[i,2]){ #(V1, V2) = (1,1)
    j = 1
  }else if(dat[i,1] & !dat[i,2]){ #(V1, V2) = (1,0)
    j = 2
  }else if(!dat[i,1] & dat[i,2]){ #(V1, V2) = (0,1)
    j = 3
```

```
  }else{ #(V1, V2) = (0,0)
    j = 4
  }

  if(runif(1) < (blockSize[j]/2-currAssign[j])/(blockSize[j]-currCap[j])){
    dat[i,4] = 1
    currAssign[j] = currAssign[j]+1
  }else{
    dat[i,4] = 0
  }
  currCap[j] = currCap[j]+1

  if(currCap[j]==blockSize[j]){
    currCap[j] = currAssign[j] = 0
    blockSize[j] = ifelse(runif(1)<0.5,2,4)
  }
}
```

The results of both randomization schemes for this particular seed are given below. Different seeds may give different results.

```
   v1 v2 simple block
1   1  0      1     0
2   1  1      1     0
3   0  0      1     0
4   1  0      0     1
5   1  0      0     1
6   0  0      0     1
7   1  0      1     0
8   1  1      1     1
9   0  1      1     0
10  0  1      1     1
11  0  1      0     0
12  1  0      1     1
13  1  1      1     0
14  0  0      1     1
15  1  1      0     1
16  0  1      0     1
17  1  0      0     0
18  0  0      1     0
19  1  0      1     1
20  0  1      0     0
```

(b) For the above randomization, we check the balance in terms of absolute difference between patients assigned treatment A and patients assigned treatment B. R code is given below:

```
# (i) Overall balance
print(sprintf("Overall imbalance from simple rand: %d",
```

```
              abs(sum(dat[,3]==1)-sum(dat[,3]==0))))
print(sprintf("Overall imbalance from stratified block rand: %d",
              abs(sum(dat[,4]==1)-sum(dat[,4]==0))))

"Overall imbalance from simple rand: 4"
"Overall imbalance from stratified block rand: 0"

# (ii) Across strata
stratum = rep(0,20)
for(i in 1:20){
  if(dat[i,1] & dat[i,2]){ #(V1, V2) = (1,1)
    stratum[i] = 1
  }else if(dat[i,1] & !dat[i,2]){ #(V1, V2) = (1,0)
    stratum[i] = 2
  }else if(!dat[i,1] & dat[i,2]){ #(V1, V2) = (0,1)
    stratum[i] = 3
  }else{ #(V1, V2) = (0,0)
    stratum[i] = 4
  }
}

for(i in 1:4){
  print(sprintf("For stratum %d:",i))
  st = which(stratum==i)
  print(sprintf("Imbalance from simple randomization: %d",
                abs(sum(dat[st,3]==1)-sum(dat[st,3]==0))))
  print(sprintf("Imbalance from stratified block randomization: %d",
                abs(sum(dat[st,4]==1)-sum(dat[st,4]==0))))
}

"For stratum 1:"
"Imbalance from simple randomization: 2"
"Imbalance from stratified block randomization: 0"
"For stratum 2:"
"Imbalance from simple randomization: 1"
"Imbalance from stratified block randomization: 1"
"For stratum 3:"
"Imbalance from simple randomization: 1"
"Imbalance from stratified block randomization: 1"
"For stratum 4:"
"Imbalance from simple randomization: 2"
"Imbalance from stratified block randomization: 0"

# (iii) Marginally across prognostic factors
for(i in 0:1){
  print(sprintf("For V1 = %d:",i))
  st = which(dat[,1]==i)
  print(sprintf("Imbalance from simple rand: %d",
```

```
                abs(sum(dat[st,3]==1)-sum(dat[st,3]==0))))
  print(sprintf("Imbalance from stratified block randomization: %d",
                abs(sum(dat[st,4]==1)-sum(dat[st,4]==0))))
}

"For V1 = 0:"
"Imbalance from simple rand: 1"
"Imbalance from stratified block randomization: 1"
"For V1 = 1:"
"Imbalance from simple rand: 3"
"Imbalance from stratified block randomization: 1"

for(i in 0:1){
  print(sprintf("For V2 = %d:",i))
  st = which(dat[,2]==i)
  print(sprintf("Imbalance from simple rand: %d",
                abs(sum(dat[st,3]==1)-sum(dat[st,3]==0))))
  print(sprintf("Imbalance from stratified block randomization: %d",
                abs(sum(dat[st,4]==1)-sum(dat[st,4]==0))))
}

"For V2 = 0:"
"Imbalance from simple rand: 3"
"Imbalance from stratified block randomization: 1"
"For V2 = 1:"
"Imbalance from simple rand: 1"
"Imbalance from stratified block randomization: 1"
```

Thus, at least for this seed, it seems that the stratified block randomization produced better balance overall, across the strata $(V1, V2) = (1, 1)$ and $(V1, V2) = (0, 0)$, and marginally across $V1 = 1$ and $V2 = 0$. In general, one should see better balance in all three categories, though individual results will vary.