STOR 767 Spring 2019 Hw1: Computational Part

Due on 01/23/2019 in Class

Zhenghan Fang

Instruction.

- Homework 1 includes **Theoretical Part** (50%) and **Computational Part** (50%).
- For homework submission and grading, edit this document and create a PDF file to print and submit in class. Codes and key results should be displayed.

Exercise 1. *(5 pt)* **Hadamard matrix** is a useful construction for two-level orthogonal design. It's defined recursively by

$$\mathbf{H}_1 = \begin{pmatrix} 1 \end{pmatrix} \in \mathbb{R}^{1\times 1}, \quad \mathbf{H}_{2^k} = \begin{pmatrix} \mathbf{H}_{2^{k-1}} & \mathbf{H}_{2^{k-1}} \\ \mathbf{H}_{2^{k-1}} & -\mathbf{H}_{2^{k-1}} \end{pmatrix} \in \mathbb{R}^{2^k \times 2^k}. \quad (k \in \mathbb{N})$$

Create $\mathbf{H}_{2^4}$ in **R**.

```
H <- c(1)
for (i in 1:4){
  H <- cbind(rbind(H, H), rbind(H, -H))
}

# print result
H
```

```
##   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## H    1    1    1    1    1    1    1    1    1     1     1     1     1
## H    1   -1    1   -1    1   -1    1   -1    1    -1     1    -1     1
## H    1    1   -1   -1    1    1   -1   -1    1     1    -1    -1     1
## H    1   -1   -1    1    1   -1   -1    1    1    -1    -1     1     1
## H    1    1    1    1   -1   -1   -1   -1    1     1     1     1    -1
## H    1   -1    1   -1   -1    1   -1    1    1    -1     1    -1    -1
## H    1    1   -1   -1   -1   -1    1    1    1     1    -1    -1    -1
## H    1   -1   -1    1   -1    1    1   -1    1    -1    -1     1    -1
## H    1    1    1    1    1    1    1    1   -1    -1    -1    -1    -1
## H    1   -1    1   -1    1   -1    1   -1   -1     1    -1     1    -1
## H    1    1   -1   -1    1    1   -1   -1   -1    -1     1     1    -1
## H    1   -1   -1    1    1   -1   -1    1   -1     1     1    -1    -1
## H    1    1    1    1   -1   -1   -1   -1   -1    -1    -1    -1     1
## H    1   -1    1   -1   -1    1   -1    1   -1     1    -1     1     1
## H    1    1   -1   -1   -1   -1    1    1   -1    -1     1     1     1
## H    1   -1   -1    1   -1    1    1   -1   -1     1     1    -1     1
##   [,14] [,15] [,16]
## H     1     1     1
## H    -1     1    -1
## H     1    -1    -1
## H    -1    -1     1
## H    -1    -1    -1
## H     1    -1     1
## H    -1     1     1
## H     1     1    -1
## H    -1    -1    -1
## H     1    -1     1
```

```
## H    -1     1     1
## H     1     1    -1
## H     1     1     1
## H    -1     1    -1
## H     1    -1    -1
## H    -1    -1     1
```

Exercise 2. *(5 pt)* It has been shown that a LASSO estimate for the location of X is of the following thresholding form

$$\hat{\mu}_{\text{LASSO}} = \underset{\mu\in\mathbb{R}}{\operatorname{argmin}} \frac{1}{2}(X-\mu)^2 + \lambda|\mu| = \begin{cases} X+\lambda, & X \leqslant -\lambda \\ 0, & -\lambda < X \leqslant \lambda \\ X-\lambda, & X > \lambda \end{cases}$$

Now let $\lambda = 1$ and consider 100 *i.i.d.* sample $\{X_i\}_{i=1}^n$ drawn from $\mathcal{N}(0,1)$. Return the vector of the LASSO estimates for their individual locations in **R**.

```r
x <- rnorm(100)
u <- x
u[x <= -1] <- x[x <= -1] + 1
u[x > -1 & x <= 1] <- 0
u[x >= 1] <- x[x >= 1] - 1

# print result
x
```

```
##   [1]  0.441400238 -0.184538295 -0.037266032 -1.383307339 -1.861093572
##   [6]  1.526559337 -0.445335803 -0.234773339  0.946675101 -0.126140588
##  [11] -0.133501000  0.349384431 -1.348071503 -0.026855595 -0.566121181
##  [16]  0.120191344 -1.620262672  0.239276266  0.019175204 -0.604487623
##  [21]  0.864372526 -0.046468560  0.588840478  0.226616826 -0.129507071
##  [26] -0.473702802 -0.214635913 -0.505718655  2.304043962 -1.908568728
##  [31] -0.876030318  1.537363353 -0.162516611  1.561470683 -3.329285763
##  [36]  0.466605490  0.806530832 -1.530731839  0.955304207 -0.741279119
##  [41] -0.850751477  0.619856989  1.204274545  1.268101412 -0.251528186
##  [46] -0.333201383 -0.652248189  0.300352032  0.285240686  0.567770574
##  [51] -1.410930080 -0.775213358  0.010945414 -0.112570597  0.359746014
##  [56] -0.033974323  2.392085462 -0.002870496  0.274765141 -0.063982512
##  [61]  0.249130485 -0.932648649  1.559610305 -1.952601692  0.236178273
##  [66]  0.708462485  1.176134105  0.170029842 -0.731188775 -0.347991248
##  [71]  0.278587309  0.077522585 -0.667247924 -0.202822288  1.755376397
##  [76]  1.509240362 -1.587806013  0.512617835 -1.434953478 -0.114557179
##  [81]  1.167232925 -0.110646458  1.741546764 -1.319812237  0.206128276
##  [86]  0.422669245  1.560582611  1.638082158  0.240715166 -0.001126631
##  [91]  0.953739763 -0.741429791 -0.837334032  0.984139452  0.117519892
##  [96]  0.747412915  1.349424028 -0.047335381 -0.451122209  0.076730888
```

```r
u
```

```
##   [1]  0.0000000  0.0000000  0.0000000 -0.3833073 -0.8610936  0.5265593
##   [7]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
##  [13] -0.3480715  0.0000000  0.0000000  0.0000000 -0.6202627  0.0000000
##  [19]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
##  [25]  0.0000000  0.0000000  0.0000000  0.0000000  1.3040440 -0.9085687
##  [31]  0.0000000  0.5373634  0.0000000  0.5614707 -2.3292858  0.0000000
##  [37]  0.0000000 -0.5307318  0.0000000  0.0000000  0.0000000  0.0000000
##  [43]  0.2042745  0.2681014  0.0000000  0.0000000  0.0000000  0.0000000
##  [49]  0.0000000  0.0000000 -0.4109301  0.0000000  0.0000000  0.0000000
```

```
## [55]  0.0000000  0.0000000  1.3920855  0.0000000  0.0000000  0.0000000
## [61]  0.0000000  0.0000000  0.5596103 -0.9526017  0.0000000  0.0000000
## [67]  0.1761341  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
## [73]  0.0000000  0.0000000  0.7553764  0.5092404 -0.5878060  0.0000000
## [79] -0.4349535  0.0000000  0.1672329  0.0000000  0.7415468 -0.3198122
## [85]  0.0000000  0.0000000  0.5605826  0.6380822  0.0000000  0.0000000
## [91]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
## [97]  0.3494240  0.0000000  0.0000000  0.0000000
```

Exercise 3. *(5 pt)* Table 1 presents a mixed 2-level and 3-level orthogonal design from (Wu and Hamada 2011). The first four rows in 2-level factors A, B and C, as a 2^{3-1} design, have been repeated for the next eight 4-row groups. Groups are embedded into a 3^{3-1} design in 3-level factors D, E and F. In particular, column C = column A $\times$ column B, column F = column D + column E (mod 3) by encoding $(-1, 0, 1)$ in $(1, 2, 0)$. Create such design matrix in **R** without reading from Table 1 directly.

```r
A <- rep(c(-1,1), 18)
B <- rep(c(-1,1), each = 2)
B <- rep(B, 9)
C <- A * B
D <- rep(c(-1, 0, 1), each = 4)
D <- rep(D, 3)
E <- rep(c(-1, 0, 1), each = 12)
F <- ((D + 2) %% 3 + (E + 2) %% 3) %% 3
F <- ((F + 1) %% 3) - 1

A <- factor(A)
B <- factor(B)
C <- factor(C)
D <- factor(D)
E <- factor(E)
F <- factor(F)
orth_arr <- data.frame(A, B, C, D, E, F)

# print result
orth_arr
```

```
##     A  B  C  D  E  F
## 1  -1 -1  1 -1 -1 -1
## 2   1 -1 -1 -1 -1 -1
## 3  -1  1 -1 -1 -1 -1
## 4   1  1  1 -1 -1 -1
## 5  -1 -1  1  0 -1  0
## 6   1 -1 -1  0 -1  0
## 7  -1  1 -1  0 -1  0
## 8   1  1  1  0 -1  0
## 9  -1 -1  1  1 -1  1
## 10  1 -1 -1  1 -1  1
## 11 -1  1 -1  1 -1  1
## 12  1  1  1  1 -1  1
## 13 -1 -1  1 -1  0  0
## 14  1 -1 -1 -1  0  0
## 15 -1  1 -1 -1  0  0
## 16  1  1  1 -1  0  0
## 17 -1 -1  1  0  0  1
## 18  1 -1 -1  0  0  1
```

Table 1 $2^{3-1} \times 3^{3-1}$ Orthogonal Array

	2-Level Factors			3-Level Factors		
	A	**B**	**C**	**D**	**E**	**F**
1	-1	-1	1	-1	-1	-1
2	1	-1	-1	-1	-1	-1
3	-1	1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1
5	-1	-1	1	0	-1	0
6	1	-1	-1	0	-1	0
7	-1	1	-1	0	-1	0
8	1	1	1	0	-1	0
9	-1	-1	1	1	-1	1
10	1	-1	-1	1	-1	1
11	-1	1	-1	1	-1	1
12	1	1	1	1	-1	1
13	-1	-1	1	-1	0	0
14	1	-1	-1	-1	0	0
15	-1	1	-1	-1	0	0
16	1	1	1	-1	0	0
17	-1	-1	1	0	0	1
18	1	-1	-1	0	0	1
19	-1	1	-1	0	0	1
20	1	1	1	0	0	1
21	-1	-1	1	1	0	-1
22	1	-1	-1	1	0	-1
23	-1	1	-1	1	0	-1
24	1	1	1	1	0	-1
25	-1	-1	1	-1	1	1
26	1	-1	-1	-1	1	1
27	-1	1	-1	-1	1	1
28	1	1	1	-1	1	1
29	-1	-1	1	0	1	-1
30	1	-1	-1	0	1	-1
31	-1	1	-1	0	1	-1
32	1	1	1	0	1	-1
33	-1	-1	1	1	1	0
34	1	-1	-1	1	1	0
35	-1	1	-1	1	1	0
36	1	1	1	1	1	0

```
## 19 -1  1 -1  0  0  1
## 20  1  1  1  0  0  1
## 21 -1 -1  1  1  0 -1
## 22  1 -1 -1  1  0 -1
## 23 -1  1 -1  1  0 -1
## 24  1  1  1  1  0 -1
## 25 -1 -1  1 -1  1  1
## 26  1 -1 -1 -1  1  1
## 27 -1  1 -1 -1  1  1
## 28  1  1  1 -1  1  1
## 29 -1 -1  1  0  1 -1
```

```
## 30  1 -1 -1  0  1 -1
## 31 -1  1 -1  0  1 -1
## 32  1  1  1  0  1 -1
## 33 -1 -1  1  1  1  0
## 34  1 -1 -1  1  1  0
## 35 -1  1 -1  1  1  0
## 36  1  1  1  1  1  0
```

Exercise 4. *(5 pt)* Thickness data from a paint experiment based on Table 1 design in (Wu and Hamada 2011) are collected as below. Compute the sum of squares for all factors (main effects) from scratch, *i.e.* without resorting to any ANOVA-type **R** functions. Compare them with outputs produced by `aov`.

```r
y <- scan(text = "0.755 0.550 0.550 0.600 0.900 0.875 1.000 1.000 1.400 1.225 1.225 1.475
0.600 0.600 0.625 0.500 0.925 1.025 0.875 0.850 1.200 1.250 1.150 1.150 0.500 0.550 0.575
0.600 0.900 1.025 0.850 0.975 1.100 1.200 1.150 1.300")

data.t = cbind(orth_arr, y)
mean.all <- mean(data.t$y)
n.all <- length(data.t$y)
Sum.Sq <- c()
for(i in c("A", "B", "C", "D", "E", "F")){
  mean.group <- tapply(data.t$y, data.t[[i]], mean)
  n.group <- tapply(rep(1, n.all), data.t[[i]], sum)
  Sum.Sq[i] <- sum(n.group * (mean.group - mean.all) ^ 2)
}

data.t.aov.A <- aov(y ~ A, data.t)
data.t.aov.B <- aov(y ~ B, data.t)
data.t.aov.C <- aov(y ~ C, data.t)
data.t.aov.D <- aov(y ~ D, data.t)
data.t.aov.E <- aov(y ~ E, data.t)
data.t.aov.F <- aov(y ~ F, data.t)

# print results
Sum.Sq
```

```
##            A            B            C            D            E
## 0.0061361111 0.0004694444 0.0051361111 2.5525291667 0.0371541667
##            F
## 0.0062375000
```

```r
summary(data.t.aov.A)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## A            1 0.0061 0.00614   0.075  0.786
## Residuals   34 2.7782 0.08171
```

```r
summary(data.t.aov.B)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## B            1 0.0005 0.00047   0.006   0.94
## Residuals   34 2.7839 0.08188
```

```r
summary(data.t.aov.C)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## C            1 0.0051 0.00514   0.063  0.804
```

```
## Residuals   34 2.7792 0.08174
```

```r
summary(data.t.aov.D)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)    
## D             2 2.5525   1.276   181.7 <2e-16 ***
## Residuals    33 0.2318   0.007                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(data.t.aov.E)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## E             2 0.0372 0.01858   0.223  0.801
## Residuals    33 2.7472 0.08325
```

```r
summary(data.t.aov.F)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## F             2 0.0062 0.00312   0.037  0.964
## Residuals    33 2.7781 0.08419
```

Exercise 5. *(10 pt)* Write a function `optim_gd(par, fn, gr, gr_lips, maxit = 10000, tol = 1e-5)` to find the minimizer of a smooth convex function using gradient descent.[1]

- `par`: initial values for the parameters to be optimized over.
- `fn`: objective function to be minimized f on domain $\mathcal{X}$.
- `gr`: gradient of objective function ∇f.
- `gr_lips`: Lipschitz gradient constant L_f, *i.e.*

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2 \leqslant L_f \|\boldsymbol{x} - \boldsymbol{y}\|_2. \quad (\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X})$$

- `maxit`: maximal number of iterations.
- `tol`: convergence tolerance parameter $\epsilon > 0$.

Iterations are performed by

$$\boldsymbol{x}^{k+1} := \boldsymbol{x}^k - \frac{1}{L_f} \nabla f(\boldsymbol{x}^k)$$

with stopping criterion

$$\frac{\|\nabla f(\boldsymbol{x}^k)\|_2}{\max\{1, \|\nabla f(\boldsymbol{x}^0)\|_2\}} \leqslant \epsilon.$$

Return a list with `par` = minimizer, `value` = optimal objective value, and `counts` = number of iterations performed. Apply it to the bivariate function[2]

$$f(x_1, x_2) = \log\left(1 + e^{-x_1 + x_2}\right) + \log\left(1 + e^{x_1}\right) + \log\left(1 + e^{-x_1 - x_2}\right)$$

with $L_f = \frac{5}{4}$ and initial value $(0, 0)$. Compare it with the built-in optimization function `optim` with `method = "BFGS"`.

```r
fn.biv <- function(x){
  x1 <- x[1]
  x2 <- x[2]
  log(1 + exp(- x1 + x2)) + log(1 + exp(x1)) + log(1 + exp(- x1 - x2))
}
```

[1] STOR 893 Fall 2018 lecture note http://quoctd.web.unc.edu/files/2018/10/lecture4-selected-cvx-methods.pdf.

[2] Negative log-likelihood of Logistic regression of the data $\{(-1, 1), (0, 0), (1, 1)\}$. Try performing `glm` to see whether the outputs coincide.

```
gr.biv <- function(x){
  x1 <- x[1]
  x2 <- x[2]
  gr.x1 <- - exp(- x1 + x2) / (1 + exp(- x1 + x2)) + exp(x1) / (1 + exp(x1)) - exp(- x1 - x2) / (1 + exp
  gr.x2 <- exp(- x1 + x2) / (1 + exp(- x1 + x2)) - exp(- x1 - x2) / (1 + exp(- x1 - x2))
  return(c(gr.x1, gr.x2))
}

optim_gd <- function(par, fn, gr, gr_lips, maxit = 10000, tol = 1e-5){
  gr_0 <- gr(par)
  for(i in 1:maxit){
    gr_i <- gr(par)
    if(sum(gr_i^2) / max(1, sum(gr_0^2)) <= tol) break
    par <- par - gr_i / gr_lips
  }
  list( par = par,
        value = fn(par),
        counts = i - 1)
}

optim_gd(c(0, 0), fn.biv, gr.biv, 5/4)
```

```
## $par
## [1] 0.6903931 0.0000000
##
## $value
## [1] 1.909545
##
## $counts
## [1] 7
```

```
optim(c(0, 0), fn.biv, method = "BFGS")
```

```
## $par
## [1]  6.931472e-01 -5.514553e-14
##
## $value
## [1] 1.909543
##
## $counts
## function gradient
##       18        5
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Exercise 6. *(10 pt)* Create the ANOVA table based on **Exercise 4** from scratch. Sum of squares, degrees of freedom, F-values, p-values and indicators of significance are to be reported. Comment on the relationship of all sum of squares and explain why.

```
data.t = cbind(orth_arr, y = y)
mean.all <- mean(data.t$y)
```

```r
n.all <- length(data.t$y)
SSb <- c()
dfb <- c()
MSb <- c()
SSw <- c()
dfw <- c()
MSw <- c()
F.aov <- c()
Pr <- c()
signif.indicator <- c()

for(i in c("A", "B", "C", "D", "E", "F")){
  mean.group <- tapply(data.t$y, data.t[[i]], mean)
  n.group <- tapply(rep(1, n.all), data.t[[i]], sum)
  SSb[i] <- sum(n.group * (mean.group - mean.all) ^ 2)

  dfb[i] <- length(levels(data.t[[i]])) - 1

  MSb[i] <- SSb[i] / dfb[i]

  mean.group.long <- data.t$y
  for(j in levels(data.t[[i]])){
    mean.group.long[data.t[[i]] == j] <- mean.group[j]
  }

  SSw[i] <- sum( (data.t$y - mean.group.long) ^ 2 )
  dfw[i] <- n.all - length(levels(data.t[[i]]))
  MSw[i] <- SSw[i] / dfw[i]

  F.aov[i] <- MSb[i] / MSw[i]

  Pr[i] <- 1 - pf( F.aov[i], dfb[i], dfw[i] )

  if(Pr[i] <= 0.001)
    sig <- "***"
  else if(Pr[i] <= 0.01)
    sig <- "**"
  else if(Pr[i] <= 0.05)
    sig <- "*"
  else if(Pr[i] <= 0.1)
    sig <- "."
  else
    sig <- " "

  signif.indicator[i] <- sig

}

my.aov.table <- data.frame(Sum_Sq = SSb,
                           Df = dfb,
                           F_value = F.aov,
                           p_value = Pr,
                           Signif = signif.indicator)
```

```r
# print results
my.aov.table
```

```
##         Sum_Sq Df      F_value   p_value Signif
## A 0.0061361111  1 7.509353e-02 0.7857181
## B 0.0004694444  1 5.733352e-03 0.9400865
## C 0.0051361111  1 6.283295e-02 0.8035817
## D 2.5525291667  2 1.816583e+02 0.0000000    ***
## E 0.0371541667  2 2.231505e-01 0.8011917
## F 0.0062375000  2 3.704595e-02 0.9636719
```

```r
summary(data.t.aov.A)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## A            1 0.0061 0.00614   0.075  0.786
## Residuals   34 2.7782 0.08171
```

```r
summary(data.t.aov.B)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## B            1 0.0005 0.00047   0.006   0.94
## Residuals   34 2.7839 0.08188
```

```r
summary(data.t.aov.C)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## C            1 0.0051 0.00514   0.063  0.804
## Residuals   34 2.7792 0.08174
```

```r
summary(data.t.aov.D)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## D            2 2.5525   1.276   181.7 <2e-16 ***
## Residuals   33 0.2318   0.007
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

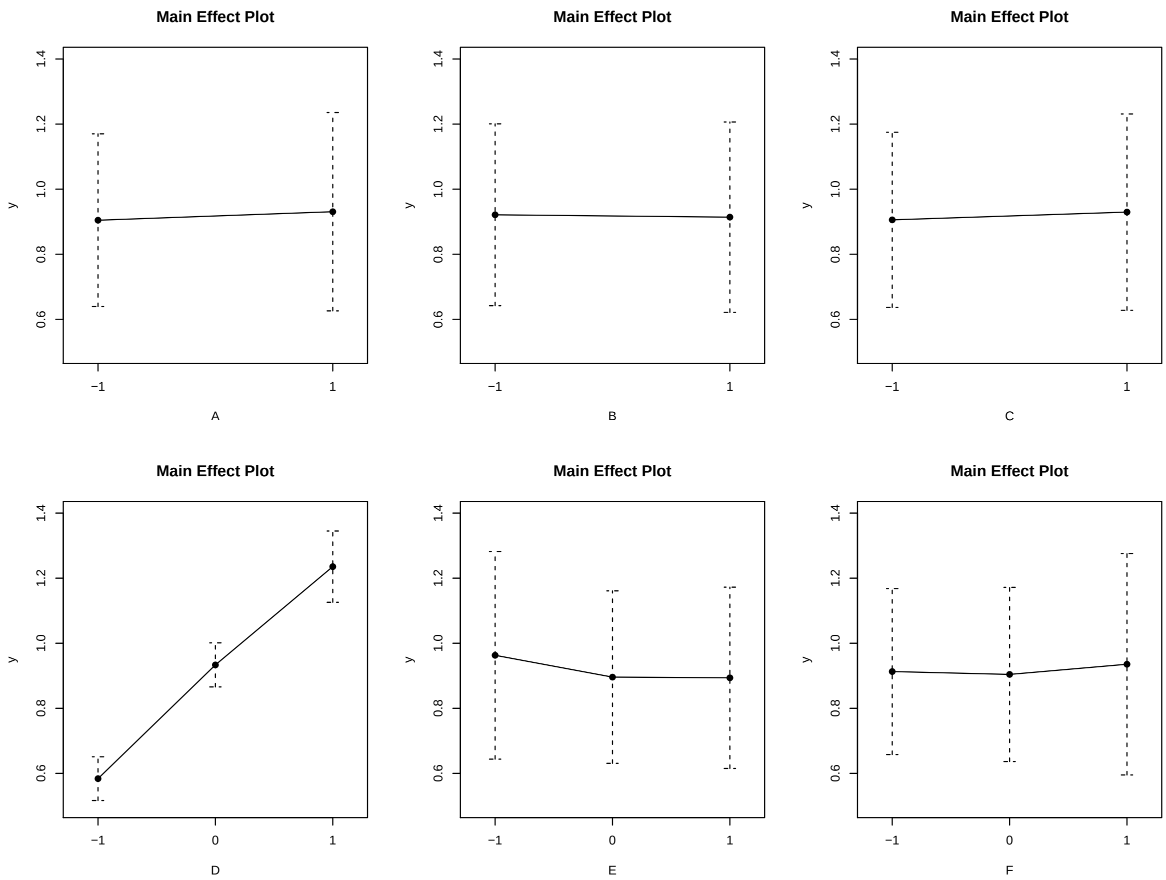
```r
summary(data.t.aov.E)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## E            2 0.0372 0.01858   0.223  0.801
## Residuals   33 2.7472 0.08325
```

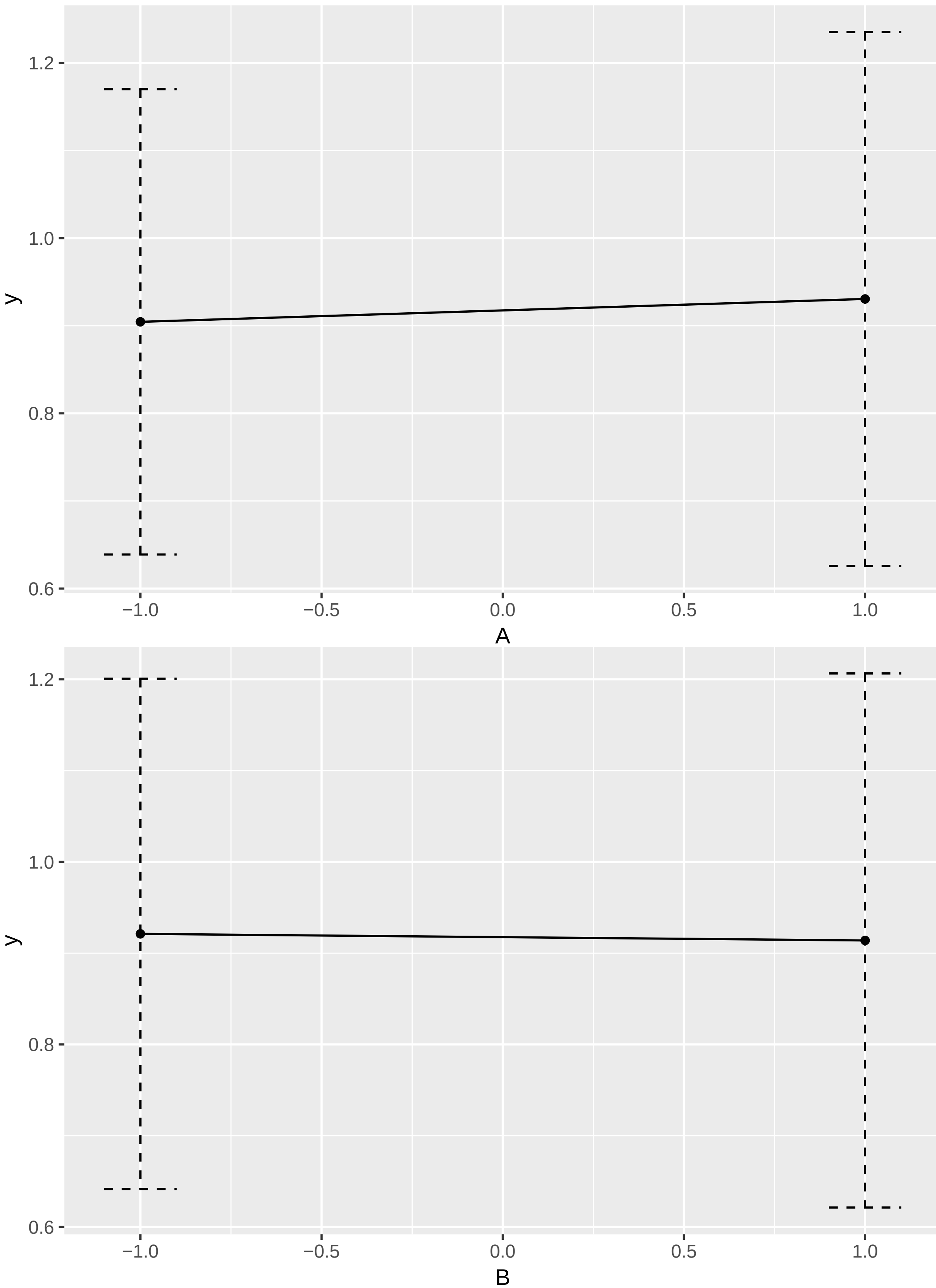
```r
summary(data.t.aov.F)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## F            2 0.0062 0.00312   0.037  0.964
## Residuals   33 2.7781 0.08419
```

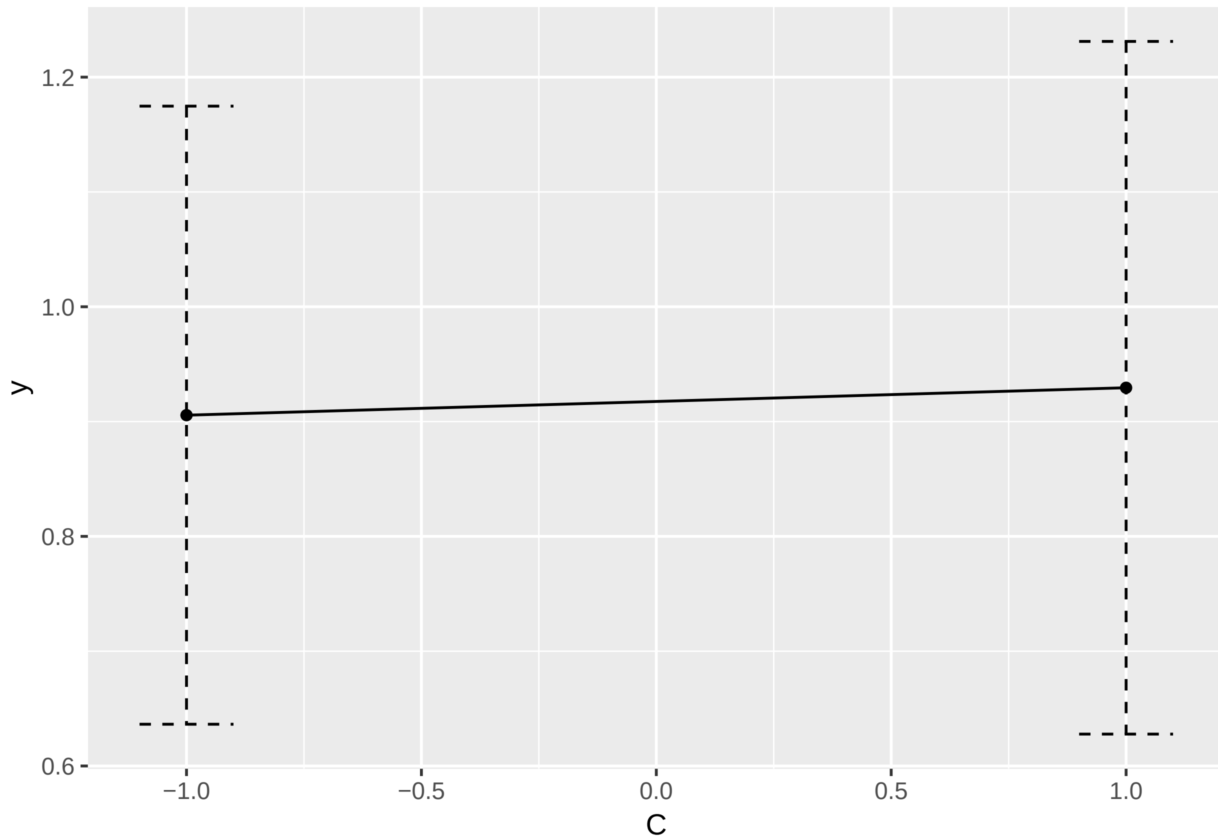
Exercise 7. *(10 pt)* Reproduce the code that generates the following plot based on **Exercise 4**.

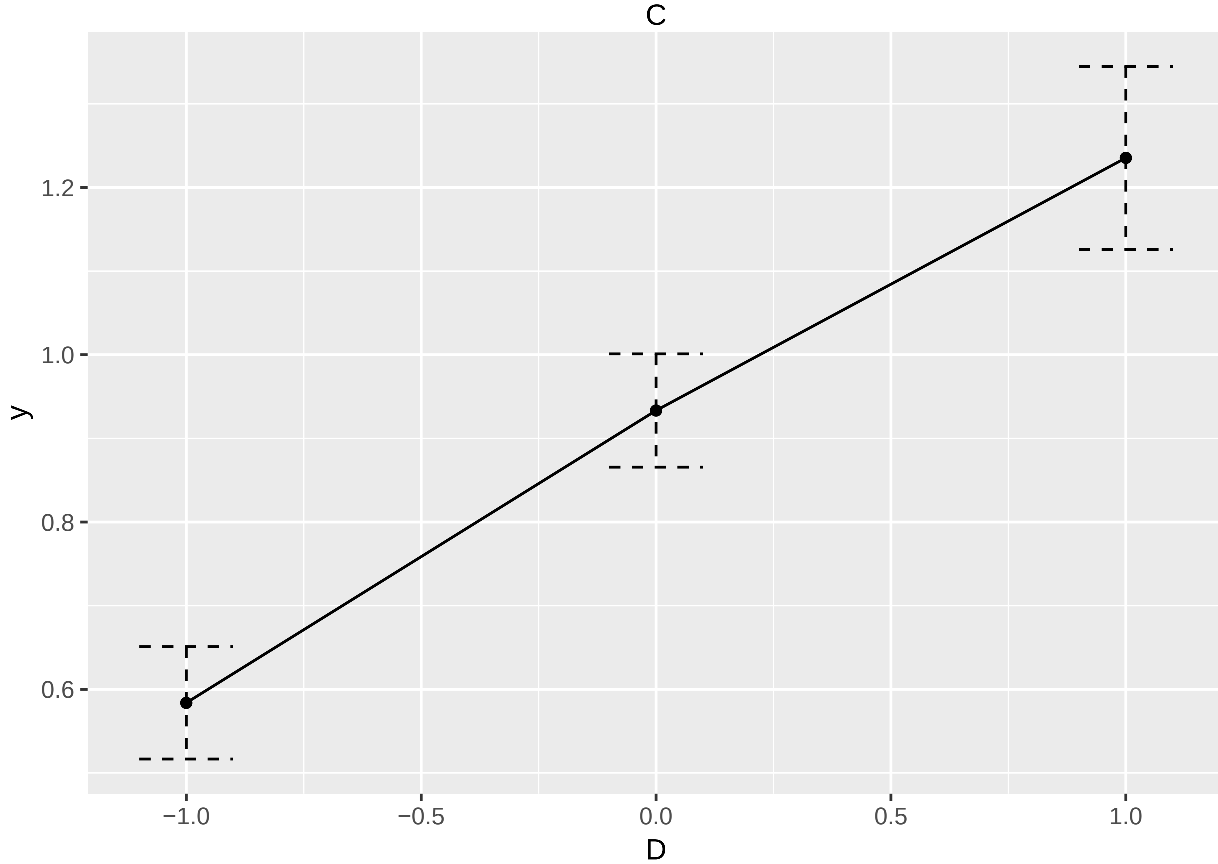


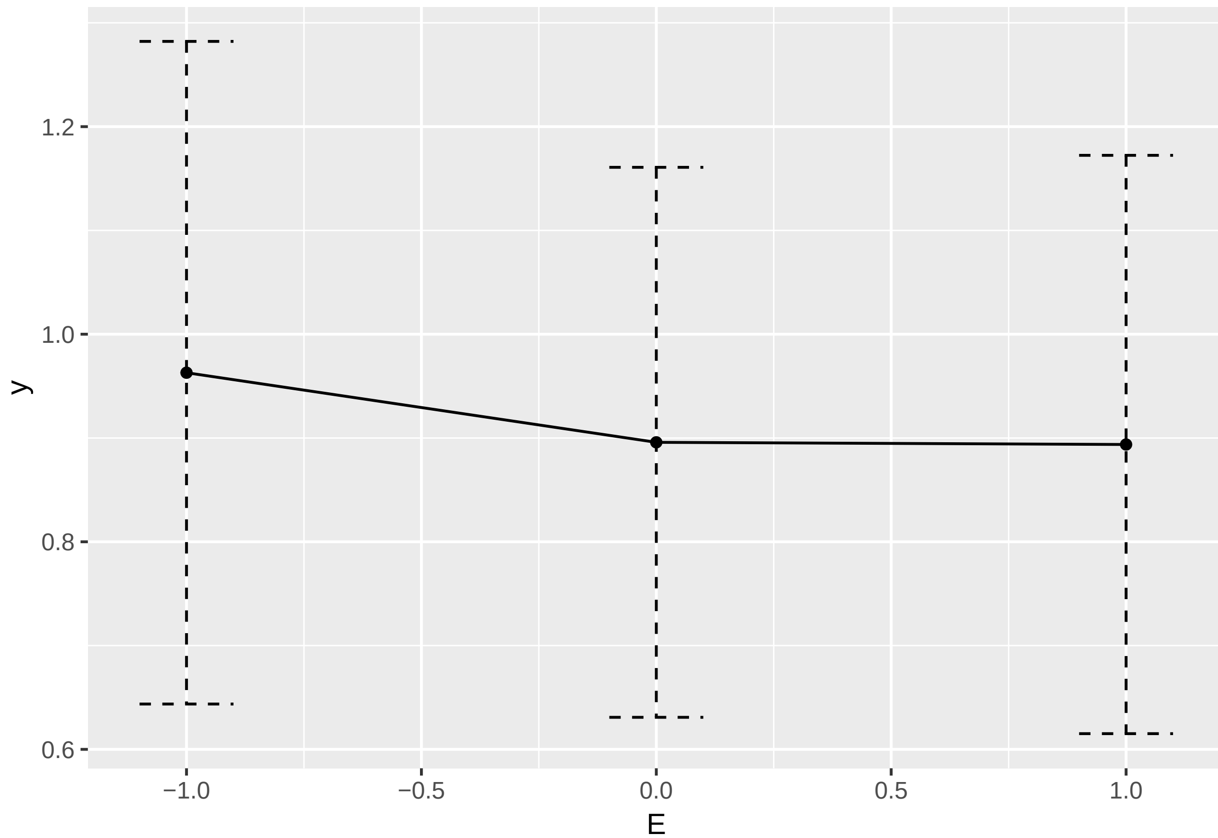

```
for(i in c("A", "B", "C", "D", "E", "F")){
  mean.group <- tapply(data.t$y, data.t[[i]], mean)
  var.group <- tapply(data.t$y, data.t[[i]], var)
  plot.data <- data.frame(fac = as.numeric(levels(data.t[[i]])),
             mean = mean.group,
             sd = sqrt(var.group) )
  fig <- ggplot(plot.data, aes(x=fac, y=mean)) +
    geom_line(linetype = 1) +
    geom_errorbar(aes(ymin=mean-sd, ymax=mean+sd), width=.2, linetype = 2) +
    geom_point() +
    xlab(i) +
    ylab("y")
  print(fig)
}
```

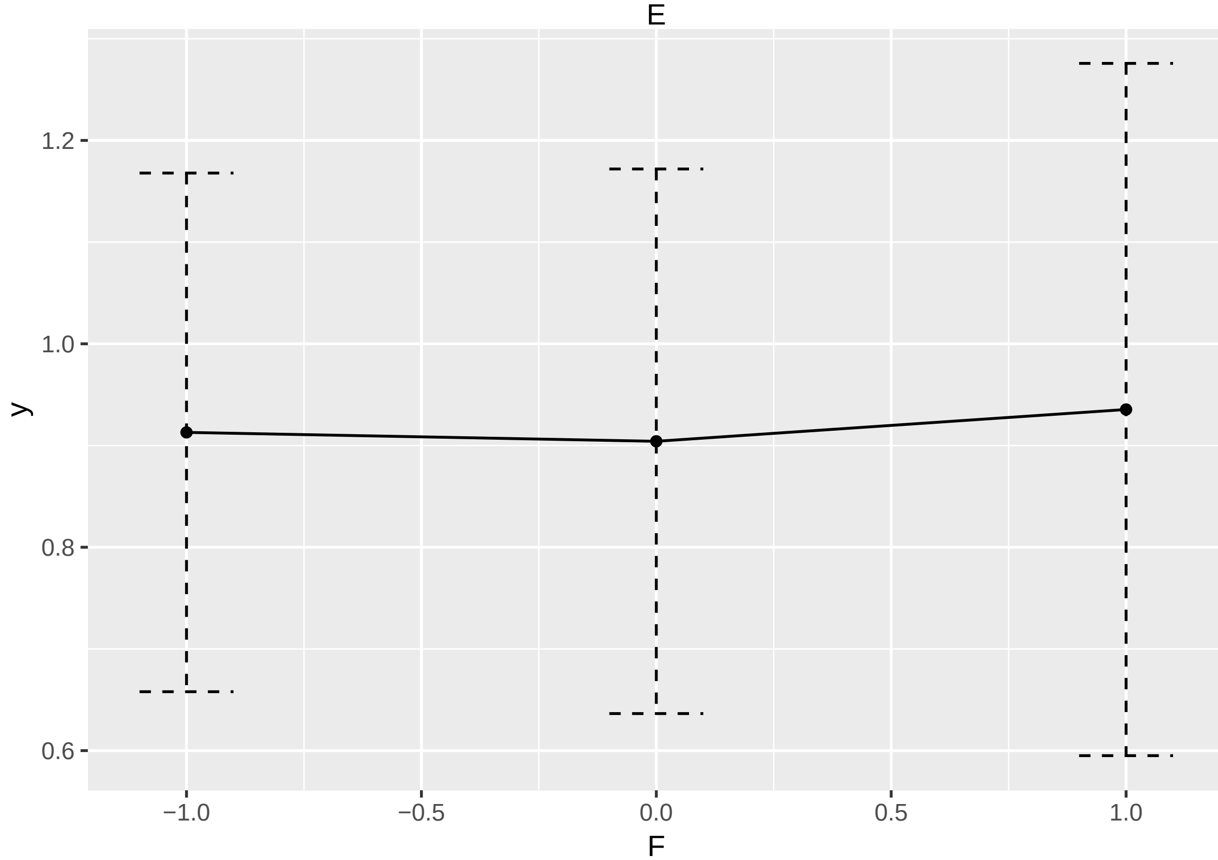


1.2
1.0
0.8
0.6
y
−1.0
−0.5
0.0
0.5
1.0
A
1.2
1.0
0.8
0.6
y
−1.0
−0.5
0.0
0.5
1.0
B


1.2
1.0
0.8
0.6
y
−1.0
−0.5
0.0
0.5
1.0
C


1.2
1.0
0.8
0.6
y
−1.0
−0.5
0.0
0.5
1.0
D


y
1.2
1.0
0.8
0.6
−1.0
−0.5
0.0
0.5
1.0
E


y
1.2
1.0
0.8
0.6
−1.0
−0.5
0.0
0.5
1.0
F

Biblography

Wu, C.F. Jeff, and Michael S. Hamada. 2011. *Experiments: Planning, Analysis, and Optimization*. Vol. 552. John Wiley & Sons.