Assignment 8 – DEA – Jaipreet Kaur

Jaipreet Kaur

Introduction

This R Markdown performs a full **Data Envelopment Analysis (DEA)** for six facilities using two inputs and two outputs under six technology assumptions: **CRS, VRS, IRS, DRS, FDH, and FRH**. It reports efficiency scores (input-oriented), peers, and lambdas, and ends with a comparative summary table.

- **Inputs**: Staff hours, Supplies (in $ thousands)
- **Outputs**: Reimbursed patient-days, Privately paid patient-days

  Note: FDH and FRH are implemented here with compact custom functions (input-oriented) so the file is self-contained and does not depend on extra packages beyond `Benchmarking`.

Packages

```
library(Benchmarking)
library(dplyr)
library(tidyr)
library(tibble)
library(purrr)
library(knitr)
library(kableExtra)
```

Data (Riverbend / Hope Valley)

```
riverbend <- tribble(
  ~DMU,         ~Staff_hours, ~Supplies, ~Reimbursed_patient,
~Privately_paid,
  "Facility 1", 187.5,        0.250,     17500,                4375,
  "Facility 2", 500.0,        0.875,     17500,                26250,
  "Facility 3", 400.0,        1.500,     52500,                13125,
  "Facility 4", 650.0,        2.500,     35000,                52500,
  "Facility 5", 437.5,        1.500,     23750,                31250,
  "Facility 6", 400.0,        0.875,     17500,                18750
)

X <- as.matrix(riverbend[, c("Staff_hours","Supplies")])
Y <- as.matrix(riverbend[, c("Reimbursed_patient","Privately_paid")])
rownames(X) <- rownames(Y) <- riverbend$DMU

knitr::kable(riverbend, caption = "Riverbend DEA Dataset (2 inputs, 2
outputs)") %>%
  kable_styling(full_width = FALSE)
```

Riverbend DEA Dataset (2 inputs, 2 outputs)

DMU

Staff_hours

Supplies

Reimbursed_patient

Privately_paid

Facility 1

187.5

0.250

17500

4375

Facility 2

500.0

0.875

17500

26250

Facility 3

400.0

1.500

52500

13125

Facility 4

650.0

2.500

35000

52500

Facility 5

437.5

1.500

23750

31250

Facility 6

400.0

0.875

17500

18750

Helper: Extract lambdas & peers from `Benchmarking`

```
get_lambdas <- function(model){
  # Handle Benchmarking versions without a 'lambdas()' function
  if (!"lambda" %in% names(model)) {
    message("Lambda values not directly stored in model; returning NA
matrix.")
    return(matrix(NA, nrow = nrow(X), ncol = nrow(X),
                  dimnames = list(rownames(X), rownames(X))))
  }
  L <- model$lambda
  if (is.null(L)) {
    message("No lambda values found in model; returning NA matrix.")
    L <- matrix(NA, nrow = nrow(X), ncol = nrow(X),
                dimnames = list(rownames(X), rownames(X)))
  }
  as.matrix(L)
}

get_peers <- function(model){
  pr <- peers(model)
  # handle cases where peers() returns varying dimensions
  nDMU <- nrow(X)
  out <- vector("list", length = nDMU)
  for (j in seq_len(nDMU)) {
    if (is.null(dim(pr))) next
    if (ncol(pr) < j) next
    idx <- pr[, j]
    idx <- idx[!is.na(idx) & idx > 0 & idx <= nDMU]
    out[[j]] <- if (length(idx) == 0) character(0) else rownames(X)[idx]
  }
  names(out) <- rownames(X)
  out
```

```
}

tabulate_peers <- function(peer_list){
  # Make a tall tibble: DMU | Peers (comma separated)
  tibble(
    DMU = names(peer_list),
    Peers = vapply(peer_list, function(v) if(length(v)==0) "" else paste(v,
collapse = ", "), character(1))
  )
}
```

DEA under CRS, VRS, IRS, DRS (input-oriented)

```
dea_crs <- dea(X, Y, RTS = "crs", ORIENTATION = "in")
dea_vrs <- dea(X, Y, RTS = "vrs", ORIENTATION = "in")
dea_irs <- dea(X, Y, RTS = "irs", ORIENTATION = "in")
dea_drs <- dea(X, Y, RTS = "drs", ORIENTATION = "in")

core_models <- list(CRS = dea_crs, VRS = dea_vrs, IRS = dea_irs, DRS =
dea_drs)

core_eff <- map_dfr(names(core_models), function(nm){
  tibble(Model = nm, DMU = rownames(X), Efficiency =
efficiencies(core_models[[nm]]))
})

core_peers <- map(names(core_models), function(nm){
  tabulate_peers(get_peers(core_models[[nm]])) %>% mutate(Model = nm, .before
= 1)
}) %>% bind_rows()

# Lambdas as a long table
core_lambdas <- map_dfr(names(core_models), function(nm){
  L <- get_lambdas(core_models[[nm]])
  as_tibble(L, .name_repair = "minimal") %>%
    mutate(Peer = rownames(L)) %>%
    relocate(Peer) %>%
    pivot_longer(-Peer, names_to = "DMU", values_to = "Lambda") %>%
    mutate(Model = nm, .before = 1)
})
```

Results: Efficiency (CRS, VRS, IRS, DRS)

```
core_eff %>%
  mutate(Efficiency = round(Efficiency, 4)) %>%
  pivot_wider(names_from = Model, values_from = Efficiency) %>%
  arrange(DMU) %>%
  kable(caption = "Input-oriented Efficiency Scores under CRS, VRS, IRS,
DRS") %>%
  kable_styling(full_width = FALSE)
```

Input-oriented Efficiency Scores under CRS, VRS, IRS, DRS

DMU

CRS

VRS

IRS

DRS

Facility 1

1.0000

1.0000

1.0000

1.0000

Facility 2

1.0000

1.0000

1.0000

1.0000

Facility 3

1.0000

1.0000

1.0000

1.0000

Facility 4

1.0000

1.0000

1.0000

1.0000

Facility 5

0.9775

1.0000

1.0000

0.9775

Facility 6

0.8675

0.8963

0.8963

0.8675

Results: Peers and Lambdas (CRS, VRS, IRS, DRS)

```
# Peers
core_peers %>%
  arrange(Model, DMU) %>%
  kable(caption = "Peers by DMU under CRS, VRS, IRS, DRS") %>%
  kable_styling(full_width = FALSE)
```

Peers by DMU under CRS, VRS, IRS, DRS

Model

DMU

Peers

CRS

Facility 1

Facility 1, Facility 2, Facility 3, Facility 4, Facility 1, Facility 1

CRS

Facility 2

Facility 2, Facility 2

CRS

Facility 3

Facility 4, Facility 4

CRS

Facility 4

CRS

Facility 5

CRS

Facility 6

DRS

Facility 1

Facility 1, Facility 2, Facility 3, Facility 4, Facility 1, Facility 1

DRS

Facility 2

Facility 2, Facility 2

DRS

Facility 3

Facility 4, Facility 4

DRS

Facility 4

DRS

Facility 5

DRS

Facility 6

IRS

Facility 1

Facility 1, Facility 2, Facility 3, Facility 4, Facility 5, Facility 1

IRS

Facility 2

Facility 2

IRS

Facility 3

Facility 5

IRS

Facility 4

IRS

Facility 5

IRS

Facility 6

VRS

Facility 1

Facility 1, Facility 2, Facility 3, Facility 4, Facility 5, Facility 1

VRS

Facility 2

Facility 2

VRS

Facility 3

Facility 5

VRS

Facility 4

VRS

Facility 5

VRS

Facility 6

```
# Lambdas
core_lambdas %>%
  filter(Lambda > 1e-10) %>%
  mutate(Lambda = round(Lambda, 6)) %>%
  arrange(Model, DMU, desc(Lambda)) %>%
  kable(caption = "Positive Lambdas by Model/DMU/Peer (CRS, VRS, IRS, DRS)")
%>%
  kable_styling(full_width = FALSE)
```

Positive Lambdas by Model/DMU/Peer (CRS, VRS, IRS, DRS)

Model

Peer

DMU

Lambda

CRS

Facility 1

L_Facility 1

1.000000

CRS

Facility 6

L_Facility 1

0.342857

CRS

Facility 5

L_Facility 1

0.200000

CRS

Facility 2

L_Facility 2

1.000000

CRS

Facility 6

L_Facility 2

0.394993

CRS

Facility 5

L_Facility 2

0.080481

CRS

Facility 3

L_Facility 3

1.000000

CRS

Facility 4

L_Facility 4

1.000000

CRS

Facility 5

L_Facility 4

0.538331

CRS

Facility 6

L_Facility 4

0.131075

DRS

Facility 1

L_Facility 1

1.000000

DRS

Facility 6

L_Facility 1

0.342857

DRS

Facility 5

L_Facility 1

0.200000

DRS

Facility 2

L_Facility 2

1.000000

DRS

Facility 6

L_Facility 2

0.394993

DRS

Facility 5

L_Facility 2

0.080481

DRS

Facility 3

L_Facility 3

1.000000

DRS

Facility 4

L_Facility 4

1.000000

DRS

Facility 5

L_Facility 4

0.538331

DRS

Facility 6

L_Facility 4

0.131075

IRS

Facility 1

L_Facility 1

1.000000

IRS

Facility 6

L_Facility 1

0.401440

IRS

Facility 2

L_Facility 2

1.000000

IRS

Facility 6

L_Facility 2

0.342261

IRS

Facility 3

L_Facility 3

1.000000

IRS

Facility 4

L_Facility 4

1.000000

IRS

Facility 5

L_Facility 5

1.000000

IRS

Facility 6

L_Facility 5

0.256299

VRS

Facility 1

L_Facility 1

1.000000

VRS

Facility 6

L_Facility 1

0.401440

VRS

Facility 2

L_Facility 2

1.000000

VRS

Facility 6

L_Facility 2

0.342261

VRS

Facility 3

L_Facility 3

1.000000

VRS

Facility 4

L_Facility 4

1.000000

VRS

Facility 5

L_Facility 5

1.000000

VRS

Facility 6

L_Facility 5

0.256299

FDH and FRH (input-oriented) – self-contained implementations

We implement compact, vectorized functions for **FDH** and **FRH**.
Definitions (input-oriented):

- **FDH**: A DMU is compared to **observed** DMUs that **weakly dominate** it on outputs (≥) and use **no more** inputs after radial contraction by θ. For each candidate peer *j* with $Y_j \geq Y_0$, the feasible contraction is $\theta_{0j} = \max_i\{x_{ij}/x_{i0}\}$. The FDH efficiency is $\min_j \theta_{0j}$. Peers are all *j* that attain this minimum (within a small tolerance). Lambdas in FDH are 0/1 at the chosen peers (ties possible).

- **FRH** (Free Replication Hull): Like FDH but allows **integer replication** of an observed DMU *j*. Minimal replication factor for outputs is $k_{0j} = \max_r \lceil y_{0r}/y_{jr} \rceil$ (with division-by-zero guarded). Then the needed contraction is $\theta_{0j} = \max_i\{\left(k_{0j} \cdot x_{ij}\right)/x_{i0}\}$. Take the minimum across *j*. Peers are the *j* that attain the minimum; the associated replication factors are the minimizing k_{0j}.

```
fdh_input_oriented <- function(X, Y, tol = 1e-9){
  n <- nrow(X)
  p <- ncol(X); q <- ncol(Y)
  eff  <- rep(NA_real_, n)
  peer <- vector("list", n)
```

```r
  lambda <- matrix(0, nrow = n, ncol = n, dimnames = list(rownames(X),
rownames(X)))

  for (o in 1:n){
    # Candidates that weakly dominate on outputs
    dominates <- apply(Y, 1, function(yj) all(yj >= Y[o, ] - tol))
    if (!any(dominates)){
      # No one dominates; DMU is FDH-efficient (theta = 1) with itself as
peer
      eff[o] <- 1
      peer[[o]] <- rownames(X)[o]
      lambda[rownames(X)[o], rownames(X)[o]] <- 1
      next
    }
    # For each dominating j, compute theta_j = max_i x_ij / x_i0
    ratios <- apply(X[dominates, , drop = FALSE], 1, function(xj){
      max(xj / X[o, ])
    })
    theta_min <- min(ratios, na.rm = TRUE)
    eff[o] <- theta_min
    best_js <- names(ratios)[abs(ratios - theta_min) <= max(1e-9, tol *
max(1, theta_min))]
    peer[[o]] <- best_js
    # Assign equal lambdas across ties (0/1 style; here we split ties equally
for readability)
    lambda[best_js, rownames(X)[o]] <- 1/length(best_js)
  }
  list(eff = eff, peers = peer, lambdas = lambda)
}

frh_input_oriented <- function(X, Y, tol = 1e-9){
  n <- nrow(X)
  eff  <- rep(NA_real_, n)
  peer <- vector("list", n)
  lambda <- matrix(0, nrow = n, ncol = n, dimnames = list(rownames(X),
rownames(X)))
  k_rep <- rep(NA_real_, n)  # store the minimizing replication factor for
reporting

  for (o in 1:n){
    theta_j <- c()
    k_j <- c()
    names_vec <- c()
    for (j in 1:n){
      # Compute minimal integer replication to cover outputs of o using j
      yj <- Y[j, ]; y0 <- Y[o, ]
      # If any yj == 0 and y0 > 0 for that output, replication impossible =>
skip
      feasible <- TRUE
```

```
      need_k <- c()
      for (r in seq_along(yj)){
        if (yj[r] <= tol && y0[r] > tol){ feasible <- FALSE; break }
        if (yj[r] <= tol && y0[r] <= tol){ need_k <- c(need_k, 0) } else {
          need_k <- c(need_k, ceiling(y0[r] / yj[r]))
        }
      }
      if (!feasible) next
      k <- max(1, max(need_k, na.rm = TRUE))
      theta <- max((k * X[j, ]) / X[o, ])
      theta_j <- c(theta_j, theta)
      k_j <- c(k_j, k)
      names_vec <- c(names_vec, rownames(X)[j])
    }
    if (length(theta_j) == 0){
      # No feasible replication found (should be rare if outputs are
nonzero); mark as efficient
      eff[o] <- 1
      peer[[o]] <- rownames(X)[o]
      k_rep[o] <- 1
      lambda[rownames(X)[o], rownames(X)[o]] <- 1
      next
    }
    theta_min <- min(theta_j, na.rm = TRUE)
    eff[o] <- theta_min
    best_idx <- which(abs(theta_j - theta_min) <= max(1e-9, tol * max(1,
theta_min)))
    best_js <- names_vec[best_idx]
    peer[[o]] <- best_js
    # Equal-share lambdas across tied best_js (for readability)
    lambda[best_js, rownames(X)[o]] <- 1/length(best_js)
    # Store the corresponding k for the first best (if multiple ties, the
first is stored)
    k_rep[o] <- k_j[best_idx[1]]
  }
  list(eff = eff, peers = peer, lambdas = lambda, k = k_rep)
}
```

Compute FDH and FRH

```
fdh_res <- fdh_input_oriented(X, Y)
frh_res <- frh_input_oriented(X, Y)

fdh_eff_tbl <- tibble(Model = "FDH", DMU = rownames(X), Efficiency =
fdh_res$eff)
frh_eff_tbl <- tibble(Model = "FRH", DMU = rownames(X), Efficiency =
frh_res$eff, k_replication = frh_res$k)

fdh_peers_tbl <- tibble(Model = "FDH", DMU = rownames(X),
                        Peers = vapply(fdh_res$peers, function(v)
```

```
if(length(v)==0) "" else paste(v, collapse = ", "), character(1)))

frh_peers_tbl <- tibble(Model = "FRH", DMU = rownames(X),
                        Peers = vapply(frh_res$peers, function(v)
if(length(v)==0) "" else paste(v, collapse = ", "), character(1)))
```

FDH/FRH Results

```
bind_rows(fdh_eff_tbl %>% select(-Model),
          frh_eff_tbl %>% select(-k_replication, -Model)) %>%
  mutate(Efficiency = round(Efficiency, 4)) %>%
  pivot_wider(names_from = DMU, values_from = Efficiency) %>%
  kable(caption = "Input-oriented Efficiency: FDH and FRH") %>%
  kable_styling(full_width = FALSE)
```

Input-oriented Efficiency: FDH and FRH

Facility 1

Facility 2

Facility 3

Facility 4

Facility 5

Facility 6

1, 1

1, 1

1, 1

1, 1

1, 1

1, 1

```
fdh_peers_tbl %>%
  kable(caption = "FDH Peers by DMU") %>%
  kable_styling(full_width = FALSE)
```

FDH Peers by DMU

Model

DMU

Peers

FDH

Facility 1

Facility 1

FDH

Facility 2

Facility 2

FDH

Facility 3

Facility 3

FDH

Facility 4

Facility 4

FDH

Facility 5

Facility 5

FDH

Facility 6

Facility 6

```
frh_peers_tbl %>%
  mutate(Note = paste0("Minimal replication factor k*: ", frh_res$k)) %>%
  kable(caption = "FRH Peers by DMU (with minimal replication factor)") %>%
  kable_styling(full_width = FALSE)
```

FRH Peers by DMU (with minimal replication factor)

Model

DMU

Peers

Note

FRH

Facility 1

Facility 1

Minimal replication factor k*: 1

FRH

Facility 2

Facility 2

Minimal replication factor k*: 1

FRH

Facility 3

Facility 3

Minimal replication factor k*: 1

FRH

Facility 4

Facility 4

Minimal replication factor k*: 1

FRH

Facility 5

Facility 5

Minimal replication factor k*: 1

FRH

Facility 6

Facility 6

Minimal replication factor k*: 1

```
# Lambdas (show positive entries)
fdh_lambda_long <- as_tibble(fdh_res$lambdas, rownames = "Peer") %>%
  pivot_longer(-Peer, names_to = "DMU", values_to = "Lambda") %>%
  filter(Lambda > 1e-10) %>% mutate(Model = "FDH", .before = 1)

frh_lambda_long <- as_tibble(frh_res$lambdas, rownames = "Peer") %>%
  pivot_longer(-Peer, names_to = "DMU", values_to = "Lambda") %>%
  filter(Lambda > 1e-10) %>% mutate(Model = "FRH", .before = 1)
```

```r
bind_rows(fdh_lambda_long, frh_lambda_long) %>%
  mutate(Lambda = round(Lambda, 6)) %>%
  arrange(Model, DMU, desc(Lambda)) %>%
  kable(caption = "Positive Lambdas (FDH & FRH)") %>%
  kable_styling(full_width = FALSE)
```

Positive Lambdas (FDH & FRH)

Model

Peer

DMU

Lambda

FDH

Facility 1

Facility 1

1

FDH

Facility 2

Facility 2

1

FDH

Facility 3

Facility 3

1

FDH

Facility 4

Facility 4

1

FDH

Facility 5

Facility 5

1

FDH

Facility 6

Facility 6

1

FRH

Facility 1

Facility 1

1

FRH

Facility 2

Facility 2

1

FRH

Facility 3

Facility 3

1

FRH

Facility 4

Facility 4

1

FRH

Facility 5

Facility 5

1

FRH

Facility 6

Facility 6

1

Comparative Summary Across All Six Assumptions

```
all_eff <- bind_rows(
  core_eff,
  fdh_eff_tbl,
  frh_eff_tbl %>% select(Model, DMU, Efficiency)
) %>% mutate(Efficiency = round(Efficiency, 4))

all_eff %>%
  pivot_wider(names_from = Model, values_from = Efficiency) %>%
  arrange(DMU) %>%
  kable(caption = "Comparative Efficiency (Input-oriented): CRS, VRS, IRS,
DRS, FDH, FRH") %>%
  kable_styling(full_width = FALSE)
```

Comparative Efficiency (Input-oriented): CRS, VRS, IRS, DRS, FDH, FRH

DMU

CRS

VRS

IRS

DRS

FDH

FRH

Facility 1

1.0000

1.0000

1.0000

1.0000

1

1

Facility 2

1.0000

1.0000

1.0000

1.0000

1

1

Facility 3

1.0000

1.0000

1.0000

1.0000

1

1

Facility 4

1.0000

1.0000

1.0000

1.0000

1

1

Facility 5

0.9775

1.0000

1.0000

0.9775

1

1

Facility 6

0.8675

0.8963

0.8963

0.8675

1

1

Discussion: Compare & Contrast

- **CRS vs VRS**: VRS typically yields efficiency scores **≥ CRS** because it allows variable scale; any gap suggests **scale inefficiency**.
- **IRS/DRS** break down scale returns. If a unit is efficient under **IRS** but not **DRS**, it suggests operating under **increasing returns** (benefits from scaling up), and vice versa.
- **FDH** forms a **non-convex** technology from observed points (no convex combinations). It generally **weakly envelops** VRS (may classify more units as efficient).
- **FRH** extends FDH by allowing **integer replication** of observed DMUs (a replication hull). Where outputs differ drastically by proportion, FRH can improve benchmarking by scaling observed practices discretely.
- **Peers & Lambdas**: Under convex models (CRS/VRS/IRS/DRS), lambdas are continuous and identify **virtual benchmarks**; under FDH/FRH here we report discrete/tied peers with equal-share lambdas for readability.

Managerial interpretation: Identify **inefficient** facilities ($\theta > 1$) and use the peer sets to set **input reduction targets** at current output mix. Check IRS/DRS to decide whether **scaling up or down** could move the facility to the efficient frontier.