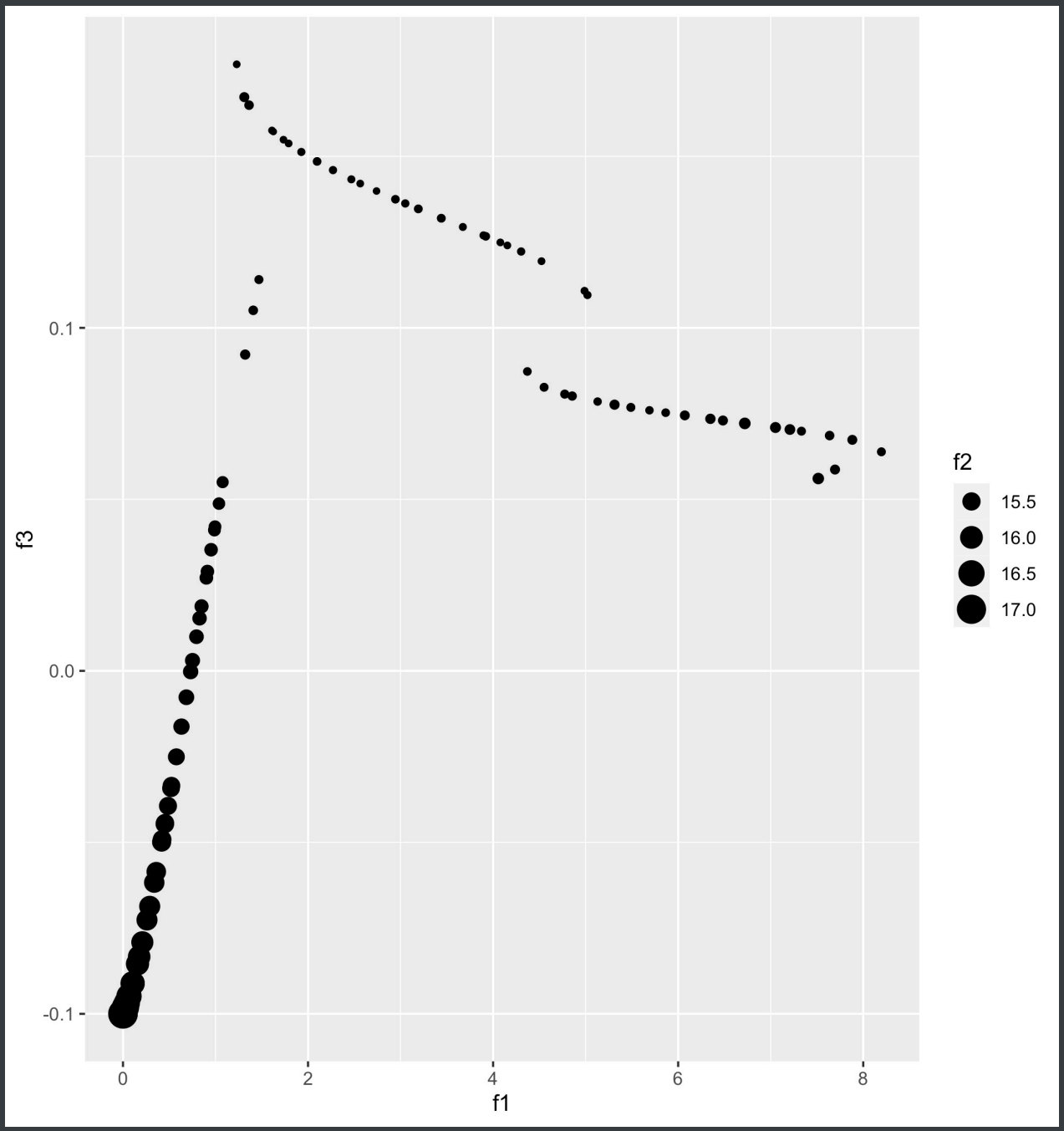
Yeo Ying Xuan 1003835

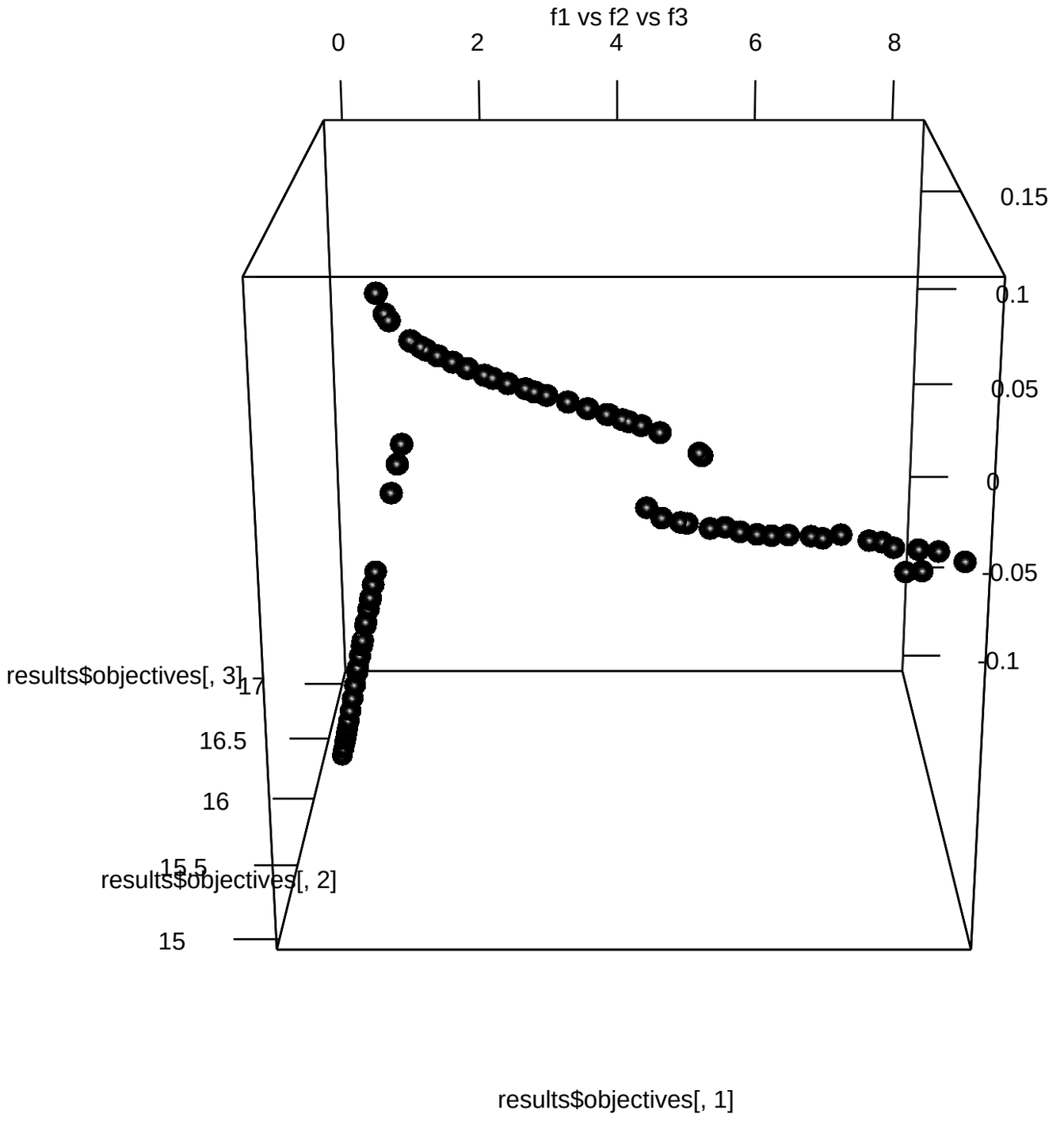
40.014 Engineering Systems Architecture

Spring 2020

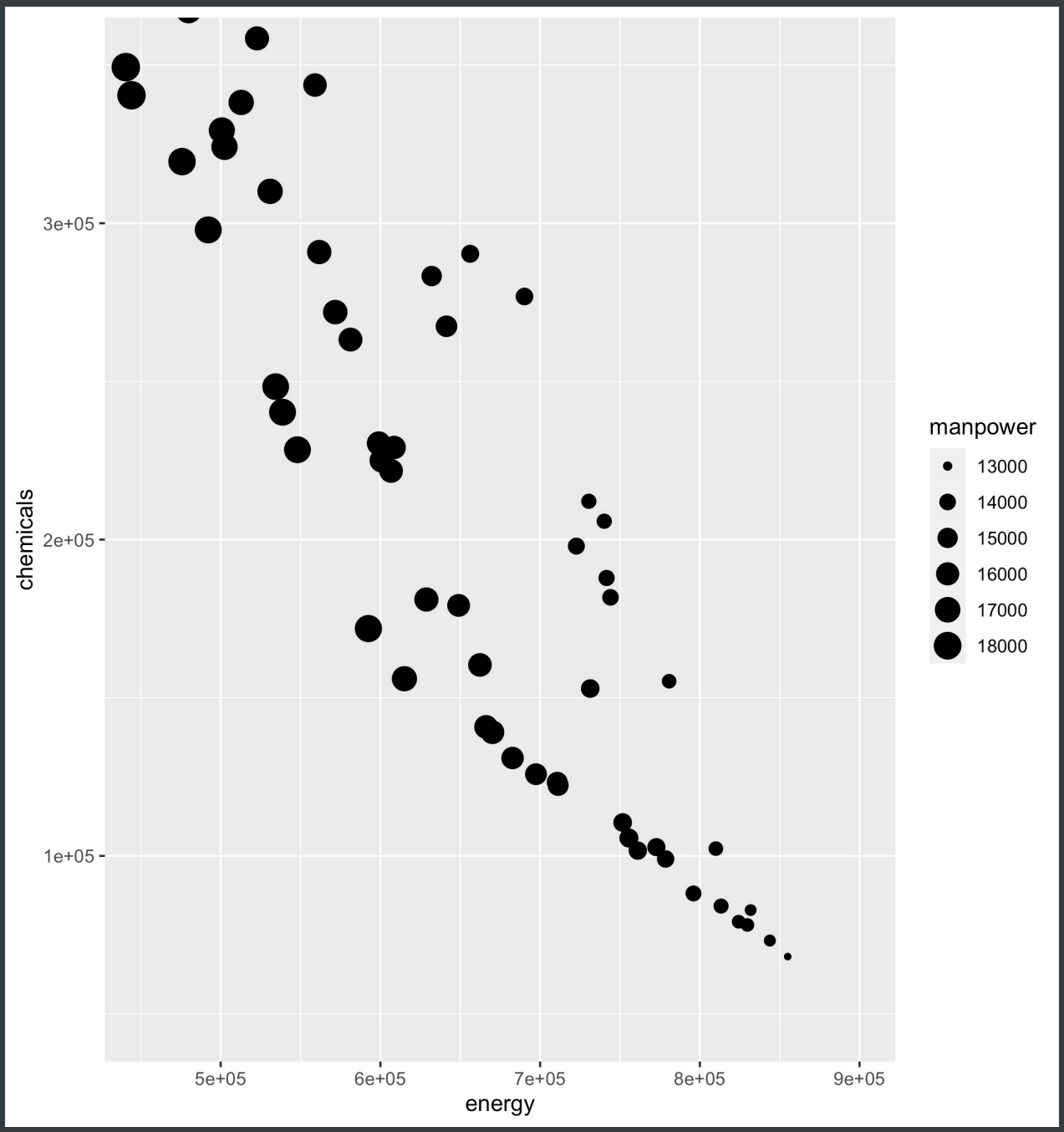
Lecture 2020 Activity

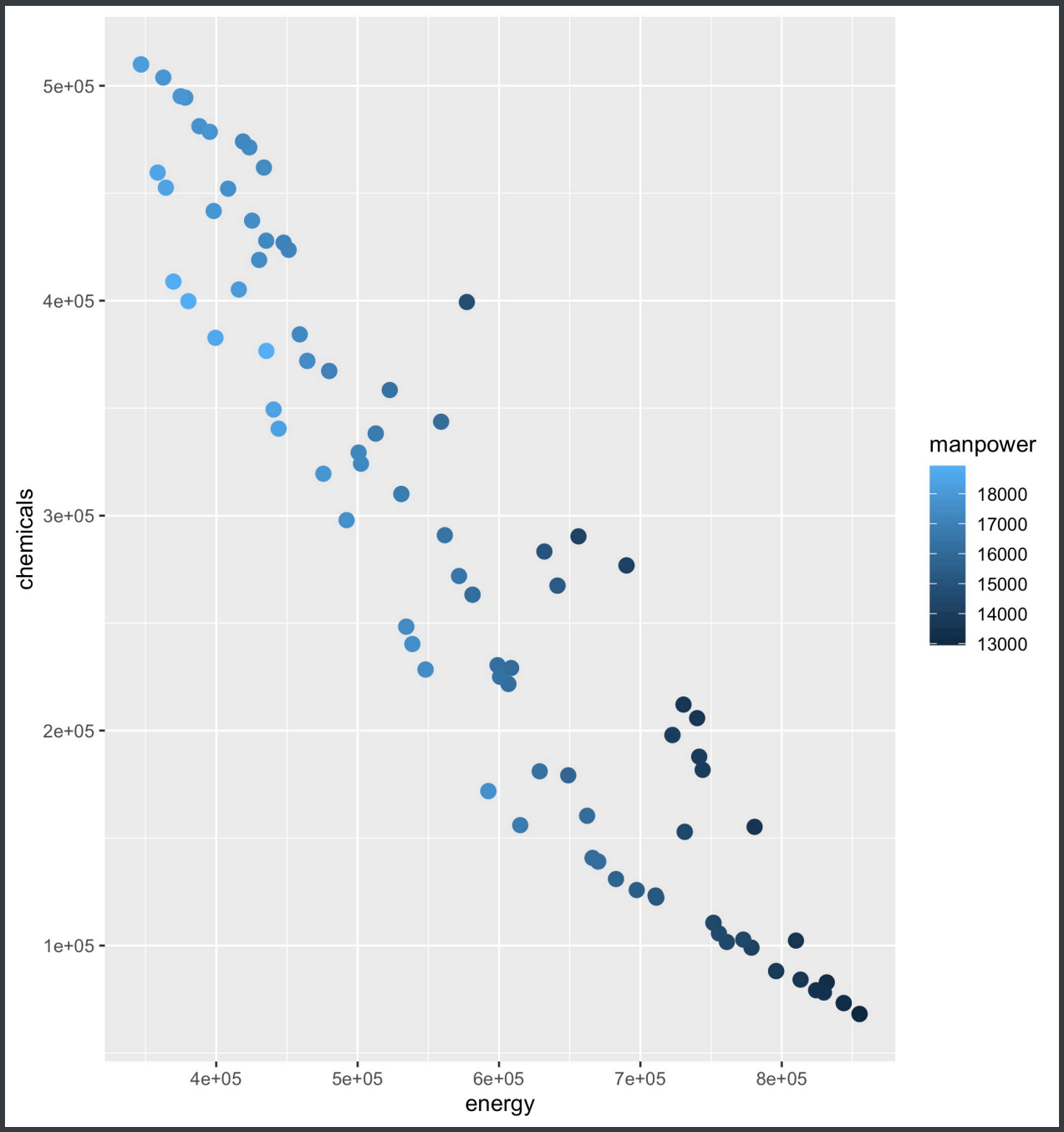
Activity 1a

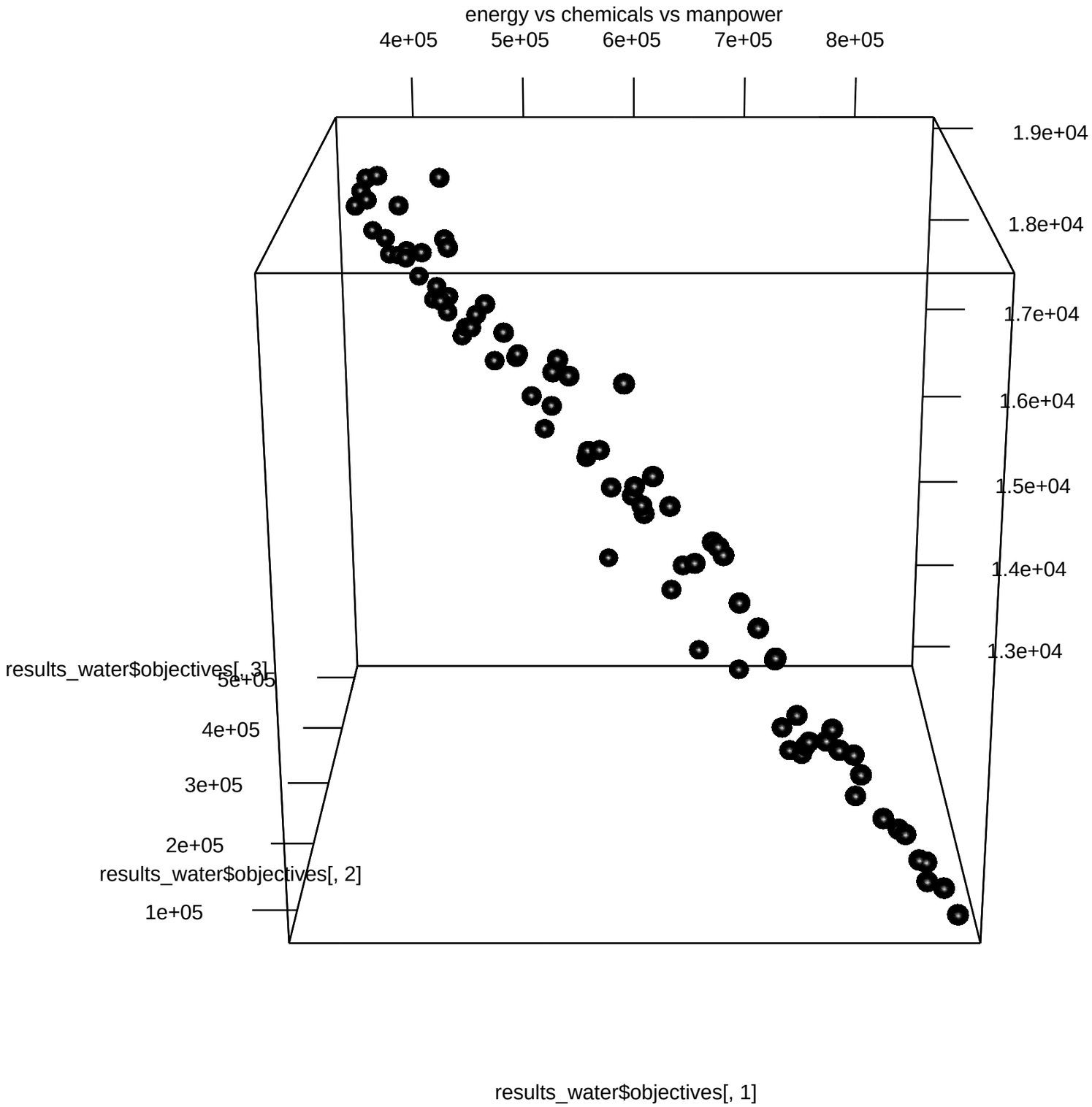



f1 vs f2 vs f3
0
2
4
6
8
0.15
0.1
0.05
0
−0.05
−0.1
results$objectives[, 3]
17
16.5
16
15.5
15
results$objectives[, 2]
results$objectives[, 1]

Activity 1b


chemicals
3e+05
2e+05
1e+05
5e+05
6e+05
7e+05
8e+05
9e+05
energy
manpower
13000
14000
15000
16000
17000
18000


chemicals
5e+05
4e+05
3e+05
2e+05
1e+05
energy
4e+05
5e+05
6e+05
7e+05
8e+05
manpower
18000
17000
16000
15000
14000
13000


energy vs chemicals vs manpower
4e+05
5e+05
6e+05
7e+05
8e+05
1.9e+04
1.8e+04
1.7e+04
1.6e+04
1.5e+04
1.4e+04
1.3e+04
results_water$objectives[, 3]
5e+05
4e+05
3e+05
2e+05
results_water$objectives[, 2]
1e+05
results_water$objectives[, 1]

R Script

```r
# Script for Activity 1a and 1b -- Lecture 20 (student's version)


######## Load data and packages

# Remove all variables from the R environment to create a fresh start
rm(list=ls())

# Set the working folder
setwd("~/Documents/SUTD/Term 5/ESA/Week 12/Lecture 20")

# nsga2R -- containing the NSGA-II algorithm
if(!require(nsga2R)){
  install.packages("nsga2R")
  library(nsga2R)
}

# ggplot2
if(!require(ggplot2)) {
  install.packages("ggplot2")
  library(ggplot2)
}

if(!require(rgl)) {
  install.packages("rgl")
  library(rgl)
}


######## Activity 1a: function optimization in 2D (real-valued) -- FILL IN
THE LINE BELOW

# Define the objective function
Viennet  <- function(x){
  f1 <- 0.5*(x[1]^2+x[2]^2)+sin(x[1]^2+x[2]^2)
  f2 <- ((3*x[1]-2*x[2]+4)^2)/8 + ((x[1]-x[2]+1)^2)/27 + 15
```

```
  f3 <- (1/(x[1]^2+x[2]^2+1)) - 1.1*exp(-(x[1]^2+x[2]^2))
  return(c(f1, f2, f3))
}

# Set number of individuals and generations -- FILL IN THE LINES BELOW
numIndividuals <- 100
totalGen <- 100

# Note: NSGA2 minimizes all objective functions!

# Run NSGA2 -- FILL IN THE LINE BELOW
results <- nsga2R(fn = Viennet, varNo = 2, objDim = 3,
                  lowerBounds = rep(-3, 2), upperBounds = rep(3, 2),
                  popSize = numIndividuals, generations = totalGen)

# Analyze the results
results$parameters        # Value of the decision variables
results$objectives        # Objective function values
results$paretoFrontRank   # Index denoting whether one solution is
efficient (1 -> Pareto efficient; 0 -> otherwise)

# Visualize the results with ggplot -- FILL IN THE LINES BELOW
mydata <- data.frame(f1=results$objectives[,1], f2=results$objectives[,2],
f3=results$objectives[,3])
a_plot1 <- ggplot(data = mydata,
       mapping = aes(x = f1, y = f3, size = f2)) +
  geom_point()
a_plot1
# recall, goal is to minimise
# we can minimise f1 and f3, but the problem is that when we do this, f2 is
maximised
# strong conflict between: f1 and f2, f3 and f2
# as we minimise f1 and f3, a strong tension is created with f2 as it is
maximised

# can also use colour to look at the plot

ggsave("1a Plot1.jpg", a_plot1)



# Alternative 3D Plot
a_plot2 <- plot3d(results$objectives[,1], results$objectives[,2],
results$objectives[,3],
       type = "s", size = 1, lit = TRUE, main = "f1 vs f2 vs f3")
```

```r
a_plot2
rgl.postscript("1a Plot2.pdf" , fmt = "pdf")

######## Activity 1b: water supply problem

# Define the objective function
Water_supply  <- function(x){

  # calculate the penalty term (for not supplying 250,000 m^3 of water) --
FILL IN THE LINES BELOW
  pen <- sqrt(.Machine$double.xmax)  # penalty term
  # supply must be larger than 250000 m3 of water
  # if supply is larger than 250 000, supply value will be positive, hence
the constraint portion is to get minimum number between 0 and supply
(supply did not hit 250000)
  # 1 unit because 1 unit of source for all four sources
  supply <- 1.00 * x[1] + 1.00 * x[2] + 1.00 * x[3] + 1.00 * x[4] - 250000
  constr <- min(supply, 0)* pen

  # Calculate the obj. functions, to which we add the penalty -- FILL IN
THE LINES BELOW
  # penalty is added as a positive value, although it is calculated as a
negative value (it's just equation manipulation)
  f1 <- 0.52*x[1] + 1.40*x[2] + 2.10*x[3] + 4.51*x[4] - constr
  f2 <- 0.20*x[1] + 4.00*x[2] + 1.20*x[3] + 0.30*x[4] - constr
  f3 <- 0.03*x[1] + 0.08*x[2] + 0.10*x[3] + 0.06*x[4] - constr

  # Return the value of the fitness functions
  return(c(f1,f2,f3))
}

# Set number of individuals and generations -- FILL IN THE LINE BELOW
numIndividuals <- 100
totalGen <- 500

# Run NSGA2
results_water <- nsga2R(fn = Water_supply, varNo = 4, objDim = 3,
                        lowerBounds = c(0,0,0,0), upperBounds =
c(70*10^3,100*10^3,120*10^3,10^9),
                        popSize = numIndividuals, generations = totalGen)

# Visualize the results with ggplot
mydata <- data.frame(energy=results_water$objectives[,1],
chemicals=results_water$objectives[,2],
manpower=results_water$objectives[,3])
```

```r
b_plot1 <- ggplot(data = mydata,
       mapping = aes(x = energy, y = chemicals, size = manpower)) +
  geom_point() # +
#  coord_cartesian(xlim=c(450000,900000), ylim=c(50000,350000))
#
b_plot1
# pareto front obtained
# want to minimise energy consumption, we can do this
# but as we minimise energy consumption, chemicals go up -> shows strong
tradeoff
# as energy is minimised, manpower goes up
# conclusion: show tradeoff between energy and manpower, energy and
chemicals

ggsave("1b Plot1.jpg", b_plot1)

# Alternative visualization
b_plot2 <- ggplot(data = mydata,
       mapping = aes(x = energy, y = chemicals)) +
  geom_point(aes(color=manpower),size=3) # +
# coord_cartesian(xlim=c(450000,900000), ylim=c(50000,350000))
#
b_plot2
ggsave("1b Plot2.jpg", b_plot2)
# Alternative (3D) visualization
b_plot3 <- plot3d(results_water$objectives[,1],
results_water$objectives[,2], results_water$objectives[,3],
       type="s", size=1, lit=TRUE, main = "energy vs chemicals vs
manpower")
b_plot3
rgl.postscript("1b Plot3.pdf" , fmt = "pdf")


# OPTIONAL: compare decision and objective space
mydata_sol <- data.frame(groundwater = results_water$parameters[,1],
surfacewater = results_water$parameters[,2],
                         imported = results_water$parameters[,3],
desalinated = results_water$parameters[,4],
                         solution = c(seq(1, numIndividuals)))

mydata <- data.frame(energy = results_water$objectives[,1], chemicals =
results_water$objectives[,2],
                     manpower = results_water$objectives[,3], solution =
c(seq(1, numIndividuals)))
```

```r
if(!require(gridExtra)) {
  install.packages("gridExtra")
  library(gridExtra)
}

p1 <- ggplot(data = mydata_sol,
             mapping = aes(x = groundwater, y = surfacewater, fill =
imported, color = desalinated, size = solution)) +
  geom_point(shape = 21) +
  scale_colour_gradient(low = "red", high = "green") +
  scale_size_continuous(range = c(1, 12)) +
  labs(title = "decision space")

p2 <- ggplot(data = mydata,
             mapping = aes(x = energy, y = chemicals, fill = manpower, size
= solution)) +
  geom_point(shape = 21) +
  scale_colour_gradient(low = "red", high = "green") +
  scale_size_continuous(range = c(1, 12)) +
  labs(title = "objective space")

grid.arrange(p1, p2, nrow = 1)
```