Appendix 1 to Rozario D, Rozario N. Can machine learning optimize the efficiency of the operating room in the era of COVID19? *Can J Surg* 2020.

DOI: 10.1503/cjs.016520

Online appendices are unedited and posted as supplied by the authors.

The following information is a python code script which can be used to replicate the findings.

```
# Natasha Rozario
# This script reads data from an excel file about past procedures and the times they took, and uses google's or-tools
#   to calculate scheduling times for each procedure type that minimizes the undertime and overtime frequencies of
#   the operating room as a whole

from __future__ import print_function
from ortools.sat.python import cp_model
import pandas as pd
import sys
import math

#
=============================================================================
=================
# Parameters
targetOvertimeFrequency = 0.2
targetUndertimeFrequency = 0
undertimeCostWeight = 1
overtimeCostWeight = 1

arguments = sys.argv
excelInputFile = 'input.xlsx'
excelOutputFile = 'output.xlsx'

overtimeBlockLength = 15 # length of overtime blocks in minutes
noOfNursesPerBlock = 2.5 # number of nurses per overtime block
overtimeSalary = 75 # pay by block for overtime nurses

# Optional: get excel filenames from command line argument
if (len(arguments) > 1):
    excelInputFile = str(arguments[1])
    excelOutputFile = str(arguments[2])

# Read data from an excel file
data = pd.read_excel(r'' + excelInputFile, header=4)
# ------------------------------
# Input data from excel sheet columns

dfP = pd.DataFrame(data, columns=["Procedure"])
dfP = dfP.iloc[:, :].values.tolist()
dfP = [x for y in dfP for x in y]
```

Appendix 1 to Rozario D, Rozario N. Can machine learning optimize the efficiency of the operating room in the era of COVID19? *Can J Surg* 2020.

DOI: 10.1503/cjs.016520

Online appendices are unedited and posted as supplied by the authors.

```
dfD = pd.DataFrame(data, columns=["Date"])
dfD = dfD.get_values()
dfD = [x for y in dfD for x in y]

dfE = pd.DataFrame(data, columns=["Book Dur"])
dfE = dfE.iloc[:, :].values.tolist()
dfE = [x for y in dfE for x in y]

dfA = pd.DataFrame(data, columns=["Actual Dur"])
dfA = dfA.iloc[:, :].values.tolist()
dfA = [x for y in dfA for x in y]

rawProcedures = [] # a list of each procedure performed
procedureTypes = []  # a list of all the procedure codes
rawDays = []  # a list of dates for each procedure performed
days = []  # a list of the dates in the data
rawExpectedTimes = [] # a list of scheduled times for each procedure performed
rawActualTimes = [] # a list of actual times for each procedure performed

# Input data into arrays
for i in range(len(dfE)):
    if str(dfE[i]) != 'nan':
        if dfD[i] not in days:
            days.append(dfD[i])
        rawExpectedTimes.append(dfE[i])
        rawDays.append(dfD[i])
        rawActualTimes.append(dfA[i])
        rawProcedures.append(dfP[i])
        if dfP[i] not in procedureTypes:
            procedureTypes.append(dfP[i])

procedureTypes.sort() # sort procedure types alphabetically

proceduresPerDay = {w: [] for w in days}  # a map from a day to a list of procedures
performed on that day
for i in range(len(rawDays)):
    proceduresPerDay[rawDays[i]].append(rawProcedures[i])


expectedTimes = {w: [] for w in days}  # a map from a day to a list of booking times for that
day
for i in range(len(rawDays)):
    expectedTimes[rawDays[i]].append(rawExpectedTimes[i])

actualTimes = {w: [] for w in days}  # a map from a day to a list of actual procedure times for
that day
for i in range(len(rawDays)):
```

Appendix 1 to Rozario D, Rozario N. Can machine learning optimize the efficiency of the operating room in the era of COVID19? *Can J Surg* 2020.

DOI: 10.1503/cjs.016520

Online appendices are unedited and posted as supplied by the authors.

```
    actualTimes[rawDays[i]].append(rawActualTimes[i])

# ---------------------------------
# Pre-processing

originalBookTimes = {p: 0 for p in procedureTypes} # a map with average booking times for
each procedue
for p in procedureTypes:
    counter = 0
    for i in range(len(rawDays)):
        if p == rawProcedures[i]:
            counter += 1
            originalBookTimes[p] += rawExpectedTimes[i]
    originalBookTimes[p] = int(originalBookTimes[p] / counter)

expectedTotalTime = {}  # a map from a day to the sum of booking times for that day
for day in days:
    expectedTotalTime[day] = 0
    for time in expectedTimes[day]:
        expectedTotalTime[day] += time

actualTotalTime = {}  # a map from a day to the sum of actual procedure times for that day
for day in days:
    actualTotalTime[day] = 0
    for time in actualTimes[day]:
        actualTotalTime[day] += time

currentOvertimeCount = 0  # counts how many days the room went overtime
currentUndertimeCount = 0  # counts how many days the room went undertime
for day in days:
    if expectedTotalTime[day] + 0 < actualTotalTime[day]:
        currentOvertimeCount += 1
    elif actualTotalTime[day] + 15 < expectedTotalTime[day]:
        currentUndertimeCount += 1

minProcedureTime = {p: 1440 for p in procedureTypes}  # a map from a procedure to its
minimum case time
maxProcedureTime = {p: 0 for p in procedureTypes}  # a map from a procedure to its
maximum case time

for i in range(len(rawDays)):
    if rawActualTimes[i] < minProcedureTime[rawProcedures[i]]:
        minProcedureTime[rawProcedures[i]] = rawActualTimes[i]
    if rawActualTimes[i] > maxProcedureTime[rawProcedures[i]]:
        maxProcedureTime[rawProcedures[i]] = rawActualTimes[i]

count = len(days)  # number of days
```

Appendix 1 to Rozario D, Rozario N. Can machine learning optimize the efficiency of the operating room in the era of COVID19? *Can J Surg* 2020.

DOI: 10.1503/cjs.016520

Online appendices are unedited and posted as supplied by the authors.

```
#
================================================================
=================
# Model

print("Starting model optimisation", file=sys.stderr)

# Decision variables
model = cp_model.CpModel()  # Create the model

procedureSchedulingTimes = {}  # Create a decision variable for the scheduling time of each
procedure type
for p in procedureTypes:
    procedureSchedulingTimes[p] = model.NewIntVar(int(minProcedureTime[p]),
int(maxProcedureTime[p]),
                                                  "procedureSchedulingTime" + str(p))

totalScheduledTime = {}  # The total time scheduled for each day
overtimeTriggers = {}  # Set to 1 if overtime for each day
undertimeTriggers = {}  # Set to 1 if undertime for each day
for day in days:
    totalScheduledTime[day] = model.NewIntVar(0, 1000, "day" + str(day))
    overtimeTriggers[day] = model.NewIntVar(0, 1, "overtimeTrigger" + str(day))
    undertimeTriggers[day] = model.NewIntVar(0, 1, "undertimeTrigger" + str(day))

overtimeCount = model.NewIntVar(0, 1000, "overtimeCount")  # Number of days that went
overtime
undertimeCount = model.NewIntVar(0, 1000, "undertimeCount")  # Number of days that went
undertime
overtimeCost = model.NewIntVar(-1000, 1000, "overtimeCost")  # Overtime cost
undertimeCost = model.NewIntVar(-1000, 1000, "undertimeCost")  # Undertime cost
absOvertimeCost = model.NewIntVar(0, 100, "absOvertimeCost")  # |overtime cost|
absUndertimeCost = model.NewIntVar(0, 100, "absUndertimeCost")  # |undertime cost|
finalCost = model.NewIntVar(0, 100, "finalCost")  # final cost is sum of overtime and
undertime costs

# ------------------------------------------------------------------------
# Constraints
for day in days:
    # set totalScheduledTime to the sum of scheduled time in a day
    model.Add(totalScheduledTime[day] == (sum(procedureSchedulingTimes[i] for i in
proceduresPerDay[day])))

    # set overtime trigger to 1 if overtime
    model.Add(1000 * overtimeTriggers[day] >= int(actualTotalTime[day]) -
totalScheduledTime[day] - 0 - 1)
```

Appendix 1 to Rozario D, Rozario N. Can machine learning optimize the efficiency of the operating room in the era of COVID19? *Can J Surg* 2020.

DOI: 10.1503/cjs.016520

Online appendices are unedited and posted as supplied by the authors.

```
    # set overime trigger to 0 if not overtime
    model.Add(1500 * (1 - overtimeTriggers[day]) >= totalScheduledTime[day] -
int(actualTotalTime[day]) + 0)

    # set undertime trigger to 1 if undertime
    model.Add(1000 * undertimeTriggers[day] >= totalScheduledTime[day] - 15 -
int(actualTotalTime[day]) - 1)
    # set undertime trigger to 0 if not undertime
    model.Add(1000 * (1 - undertimeTriggers[day]) >= int(actualTotalTime[day]) -
totalScheduledTime[day] + 15)

model.Add(overtimeCount == (sum(overtimeTriggers[day] for day in days)))  # count how
many days went overtime
model.Add(undertimeCount == (sum(undertimeTriggers[day] for day in days)))  # count how
many days went undertime

model.Add((overtimeCost == overtimeCount - int(count * targetOvertimeFrequency)))  #
calculate overtime cost
model.Add((undertimeCost == undertimeCount - int(count * targetUndertimeFrequency)))  #
calculate undertime cost

# calculate absolute value of overtime cost
model.AddAbsEquality(absOvertimeCost, int(overtimeCostWeight) * overtimeCost)
# calculate absolute value of undertime cost
model.AddAbsEquality(absUndertimeCost, int(undertimeCostWeight) * undertimeCost)
model.Add((finalCost == overtimeCost + undertimeCost))  # calculate final cost

model.Minimize(finalCost)  # Objective is to minimize final cost

solver = cp_model.CpSolver()
print("Starting to solve", file=sys.stderr)
status = solver.Solve(model)

#
================================================================
=================
# Print output to console and excel sheet
if status == cp_model.OPTIMAL or status == cp_model.FEASIBLE:
    print("Success!")
    rows = ["Number of days", "Number of cases", "Original overtime frequency", "Original
undertime frequency", "Model overtime frequency",
            "Model undertime frequency", "Model cases achieved", "Original Overtime Minutes
Used", "Model Overtime Minutes Used",
            "Original Overtime Cost", "Model Overtime Cost", "Original OR minutes used", "Model
OR minutes used", "", "Procedure Type"] + procedureTypes

    dashboard = pd.DataFrame(index=rows,
```

Appendix 1 to Rozario D, Rozario N. Can machine learning optimize the efficiency of the operating room in the era of COVID19? *Can J Surg* 2020.

DOI: 10.1503/cjs.016520

Online appendices are unedited and posted as supplied by the authors.

```
                    columns=["A", "B", "C", ])

    for r in rows:
        dashboard.at[r, "A"] = r

    # ------------------------------------------------------------------------
    # Output basic stats
    dashboard.at["", ""] = ""
    dashboard.at["Number of days", "B"] = count
    print("number of days: ", count)
    dashboard.at["Number of cases", "B"] = len(rawExpectedTimes)

    print("number of cases: ", len(rawExpectedTimes))
    dashboard.at["Original overtime frequency", "B"] = round(currentOvertimeCount / count, 2)
    print("current overtime frequency: %.0f%%" % (100 * currentOvertimeCount / count))
    dashboard.at["Original undertime frequency", "B"] = round(currentUndertimeCount / count,
2)
    print("current undertime frequency: %.0f%%" % (100 * currentUndertimeCount / count))
    dashboard.at["Model overtime frequency", "B"] = round(solver.Value(overtimeCount) /
count, 2)
    print("model overtime frequency: %.0f%%" % (100 * solver.Value(overtimeCount) / count))
    dashboard.at["Model undertime frequency", "B"] = round(solver.Value(undertimeCount) /
count, 2)
    print("model undertime frequency: %.0f%%" % (100 * solver.Value(undertimeCount) /
count))

    # ------------------------------------------------------------------------
    # Output overtime minutes and overtime cost
    originalOverMinutes = 0
    modelOverMinutes = 0

    originalCost = 0
    modelCost = 0

    for day in days:
        if solver.Value(totalScheduledTime[day]) < actualTotalTime[day]:
            modelOverMinutes += actualTotalTime[day] - solver.Value(totalScheduledTime[day])
            modelCost += math.ceil((actualTotalTime[day] -
solver.Value(totalScheduledTime[day])) / overtimeBlockLength) * noOfNursesPerBlock *
overtimeSalary
        if expectedTotalTime[day] < actualTotalTime[day]:
            originalOverMinutes += actualTotalTime[day] - expectedTotalTime[day]
            originalCost += math.ceil((actualTotalTime[day] - expectedTotalTime[day]) /
overtimeBlockLength) * noOfNursesPerBlock * overtimeSalary


    print("Original overtime minutes used: ", originalOverMinutes)
```

Appendix 1 to Rozario D, Rozario N. Can machine learning optimize the efficiency of the operating room in the era of COVID19? *Can J Surg* 2020.

DOI: 10.1503/cjs.016520

Online appendices are unedited and posted as supplied by the authors.

```
    dashboard.at["Original Overtime Minutes Used", "B"] = originalOverMinutes
    print("Model overtime minutes used: ", modelOverMinutes)
    dashboard.at["Model Overtime Minutes Used", "B"] = modelOverMinutes

    print("Original overtime cost: ", originalCost)
    dashboard.at["Original Overtime Cost", "B"] = originalCost
    print("Model overtime cost: ", modelCost)
    dashboard.at["Model Overtime Cost", "B"] = modelCost

    print("\n")

    # ------------------------------------------------------------------------
    # Output original and machine learning scheduling times for each procedure
    dashboard.at["Procedure Type", "B"] = "Original Time"
    dashboard.at["Procedure Type", "C"] = "Machine Learning Time"
    for p in procedureTypes:
        print(p + ": " + "%d" % solver.Value(procedureSchedulingTimes[p]))
        dashboard.at[p, "B"] = originalBookTimes[p]
        dashboard.at[p, "C"] = solver.Value(procedureSchedulingTimes[p])

    # ------------------------------------------------------------------------
    # Output cases completed with model and OR minutes used by original and machine
learning methods
    totalNoCases = 0
    originalCases = 0
    modelCases = 0
    originalMinutes = 0
    modelMinutes = 0
    totalMinutes = 0

    print("\n")
    for day in days:
        noOfCases = len(actualTimes[day])
        totalNoCases += noOfCases

        originalMinutes += actualTotalTime[day]
        modelMinutes += solver.Value(totalScheduledTime[day])

        ORTime = 450 - (noOfCases - 1) * 15
        totalMinutes += ORTime

        originalTime = (ORTime - actualTotalTime[day])
        modelTime = (ORTime - solver.Value(totalScheduledTime[day]))
        originalCases += noOfCases
        if originalTime < 0:
            for i in reversed(range(len(actualTimes[day]))):
                originalTime += actualTimes[day][i]
```

Appendix 1 to Rozario D, Rozario N. Can machine learning optimize the efficiency of the operating room in the era of COVID19? *Can J Surg* 2020.

DOI: 10.1503/cjs.016520

Online appendices are unedited and posted as supplied by the authors.

```
            originalCases -= 1
            if originalTime > 0:
                break
        modelCases += noOfCases
        if modelTime < 0:
            for i in reversed(range(len(actualTimes[day]))):
                modelTime += actualTimes[day][i]
                modelCases -= 1
                if modelTime > 0:
                    break

    print("Cases achieved with machine learning model: %.0f%%" % (100 * modelCases /
totalNoCases))
    dashboard.at["Model cases achieved", "B"] =  round(modelCases / totalNoCases, 2)
    print("OR minutes used with original model: %.0f%%" % (100 * originalMinutes /
totalMinutes))
    dashboard.at["Original OR minutes used", "B"] =  round(originalMinutes / totalMinutes, 2)
    print("OR minutes used with machine learning model: %.0f%%" % (100 * modelMinutes /
totalMinutes))
    dashboard.at["Model OR minutes used", "B"] = round(modelMinutes / totalMinutes, 2)

    dashboard.to_excel(r'' + excelOutputFile, index=False)

else:

    print("No feasible solution found.")
```

Appendix 1 to Rozario D, Rozario N. Can machine learning optimize the efficiency of the operating room in the era of COVID19? *Can J Surg* 2020.

DOI: 10.1503/cjs.016520

Online appendices are unedited and posted as supplied by the authors.

Surgery Division 1 2017 - 2019

	Surgeon 1		Surgeon 2		Surgeon 3		Surgeon 4		Surgeon 5		Surgeon 6		Surgeon 7	
Number of Days	155		135		176		154		194		183		140	
Number of Cases	847		444		733		703		753		655		567	
Machine Learning Cases Achieved	99.00%		95.67%		94.67%		91.67%		91.33%		93.00%		96.33%	
	Original	**ML Model**	**Original**	**ML Model**	**Original**	**ML Model**	**Original**	**ML Model**	**Original**	**ML Model**	**Original**	**Machine Learning**	**Original**	**ML Model**
Overtime Frequency	13.33%	21.33%	48.67%	20.67%	32.67%	21.67%	55.33%	28.00%	54.67%	22.33%	58.33%	25.00%	47.33%	27.67%
Undertime Frequency	74.00%	17.00%	31.00%	13.00%	56.67%	14.33%	35.33%	10.00%	30.00%	24.67%	27.33%	15.33%	40.33%	10.67%
Overtime Minutes Used	623	1586	2933	2127	2200	2825	3926	2525	4997	3303	5041	2525	3066	2757
Overtime Cost	$9937.50	$26250.00	$42750.00	$32437.50	$33375.00	$42000.00	$56812.50	$39187.50	$72000.00	$48562.50	$73312.50	$39750.00	$44625.00	$40312.50
Percentage of OR Minutes Used	75.00%	76.33%	71.67%	71.67%	73.67%	73.00%	88.33%	86.67%	81.67%	83.00%	76.67%	77.00%	76.67%	74.67%
Scheduling Times (mins)														
Procedure 1	55	56												
Procedure 2	60	61	60	71	52	59	88	114	74	86	68	63	74	66
Procedure 3			53	64										
Procedure 4									94	112			106	103
Procedure 5			86	64										
Procedure 6			102	96	78	78								
Procedure 7											157	143		
Procedure 8	139	125							205	262				
Procedure 9			170	176	173	143			174	160			169	154
Procedure 10	54	54	69	73	62	66	70	63	73	68	86	81	70	76
Procedure 11	50	37					55	56	65	70	65	57	46	37
Procedure 12									94	111				
Procedure 13	80	74												
Procedure 14	50	40	61	64	60	57	70	71	62	70	78	81	62	65
Procedure 15							35	37						
Procedure 16							58	72						
Procedure 17					31	24								
Procedure 18					31	24								
Procedure 19							48	58						
Procedure 20	58	60					92	91	96	100	100	109	78	80
Procedure 21			112	135	111	70								
Procedure 22			72	82			72	62						
Procedure 23	45	34									57	58	44	42
Procedure 24	45	36	65	47	44	51	61	59	51	59	53	54	48	44
Procedure 25											25	39		

Appendix 1 to Rozario D, Rozario N. Can machine learning optimize the efficiency of the operating room in the era of COVID19? *Can J Surg* 2020.

DOI: 10.1503/cjs.016520

Online appendices are unedited and posted as supplied by the authors.

Procedure 26					37	35					46	46	37	33

Appendix 1 to Rozario D, Rozario N. Can machine learning optimize the efficiency of the operating room in the era of COVID19? *Can J Surg* 2020.

DOI: 10.1503/cjs.016520

Online appendices are unedited and posted as supplied by the authors.

Surgery Division 2 2017 - 2019

	Surgeon 1		Surgeon 2		Surgeon 3		Surgeon 4		Surgeon 5		Surgeon 6		Surgeon 7		Surgeon 8	
Number of Days	201		230		232		308		253		70		304		240	
Number of Cases	946		758		552		1161		879		173		821		561	
Machine Learning Cases Achieved	99.00%		99.33%		99.33%		100.00%		98.67%		97.00%		100.00%		95.00%	
	Original	**ML Model**	**Original**	**ML Model**	**Original**	**ML Model**	**Original**	**ML Model**	**Original**	**ML Model**	**Original**	**ML Model**	**Original**	**ML Model**	**Original**	**ML Model**
Overtime Frequency	45.67%	29.00%	46.00%	27.67%	36.33%	25.00%	57.33%	38.00%	55.67%	31.33%	53.00%	29.00%	49.33%	34.67%	62.67%	19.67%
Undertime Frequency	33.67%	20.33%	32.67%	18.00%	45.00%	19.00%	19.33%	19.67%	31.67%	19.00%	33.00%	19.00%	30.00%	21.67%	22.33%	23.33%
Overtime Minutes Used	2121	1850	2481	2027	2379	2093	3449	2852	4738	3311	1626	986	3079	2965	7980	4558
Overtime Cost	$35062.50	$30562.50	$41250.00	$35812.50	$38625.00	$34312.50	$60187.50	$48562.50	$72562.50	$52312.50	$23437.50	$15562.50	$53250.00	$48750.00	$113250.00	$68250.00
Percentage of OR Minutes Used	80.67%	81.00%	81.33%	82.00%	58.33%	58.67%	62.00%	62.00%	77.33%	77.33%	61.00%	60.00%	59.33%	60.00%	59.67%	61.67%
Scheduling Times (mins)																
Procedure 1			115	117	124	108	76	77	87	80	110	118	91	102		
Procedure 2	151	121														
Procedure 3											132	55			129	119
Procedure 4	87	78														
Procedure 5	115	104														
Procedure 6					55	71	50	55			50	72	55	49		
Procedure 7															59	56
Procedure 8	94	93														
Procedure 9									33	40						
Procedure 10					190	163	142	120	131	130			142	133		
Procedure 11															133	186
Procedure 12			143	128												
Procedure 13	116	111													120	171
Procedure 14			113	114												
Procedure 15					137	137	94	94	123	130			107	100		
Procedure 16													118	121		
Procedure 17			145	138												
Procedure 18	91	98	133	125												
Procedure 19									113	115						
Procedure 20															118	118
Procedure 21			105	104												
Procedure 22	132	109														
Procedure 23									37	40			40	58		
Procedure 24			114	114							128	127				
Procedure 25					88	125										
Procedure 26					82	81					92	75			99	108
Procedure 27							103	80								
Procedure 28											90	122			92	150
Procedure 29			106	122												
Procedure 30	106	102	117	113	138	122	111	101			111	96				
Procedure 31											85	115				
Procedure 32															119	120
Procedure 33									152	162						
Procedure 34	36	35	43	48	50	49	32	36	54	60	44	92	38	40	72	95
Procedure 35					93	98	95	82	90	139	75	127	105	75	108	126

Appendix 1 to Rozario D, Rozario N. Can machine learning optimize the efficiency of the operating room in the era of COVID19? *Can J Surg* 2020.

DOI: 10.1503/cjs.016520

Online appendices are unedited and posted as supplied by the authors.

Procedure 36							130	97	139	125			137	144		
Procedure 37	54	52			60	50	85	93					47	51		