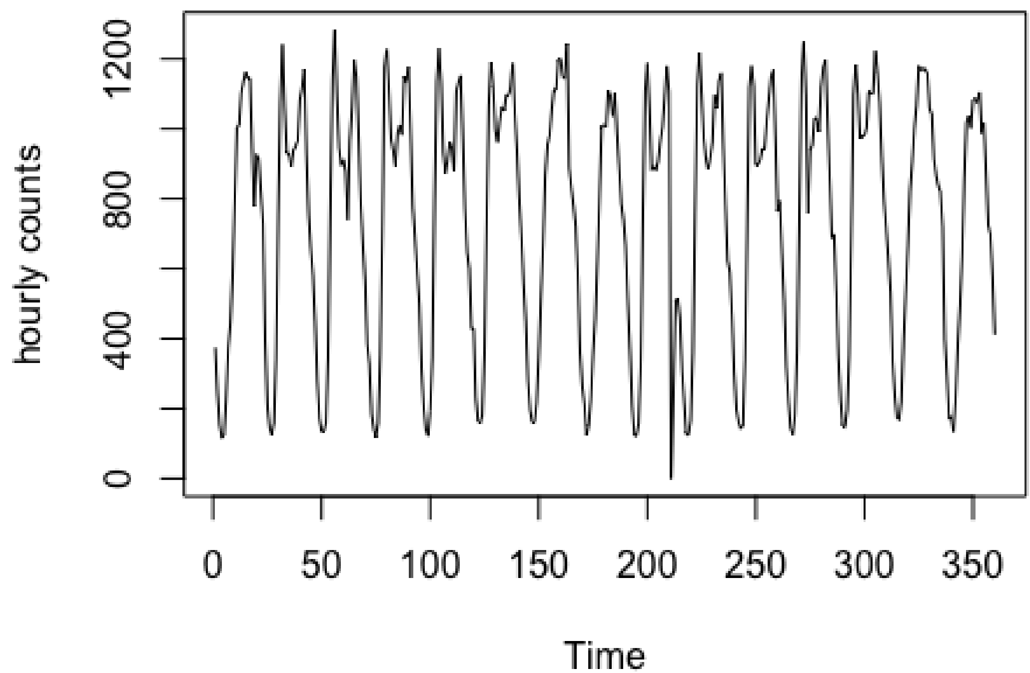
Assignment4 Yunzhi Wang 12149087

Part 1

```
#time series for Dayp
ddts <- ts(dd)
plot.ts(ddts)
```




```
fit_dd = auto.arima(ddts)
fit_dd

## Series: ddts
## ARIMA(2,0,3) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1      ma2      ma3      mean
##       1.8088  -0.8853  -0.5348  -0.2671  -0.1157  746.3187
## s.e.  0.0288   0.0287   0.0600   0.0596   0.0654    6.8585
##
## sigma^2 estimated as 13442:  log likelihood=-2220.77
## AIC=4455.53   AICc=4455.85   BIC=4482.74
```

```r
forecast(fit_dd, h=24)
```

```
##     Point Forecast    Lo 80     Hi 80      Lo 95     Hi 95
## 361       350.3265 201.7447  498.9084 123.090209  577.5629
## 362       361.4997 120.8482  602.1512  -6.544983  729.5444
## 363       412.0913 116.7639  707.4188 -39.573064  863.7557
## 364       482.4428 161.8027  803.0829  -7.933951  972.8195
## 365       564.9076 235.7696  894.0456  61.534427 1068.2807
## 366       651.7898 321.7877  981.7919 147.095161 1156.4844
## 367       735.9381 405.1617 1066.7144 230.059310 1241.8169
## 368       811.2304 474.6858 1147.7750 296.529816 1325.9309
## 369       872.9239 523.9782 1221.8696 339.257487 1406.5903
## 370       917.8599 551.4270 1284.2927 357.449213 1478.2705
## 371       944.5233 558.7272 1330.3194 354.499021 1534.5476
## 372       952.9704 549.1543 1356.7866 335.386895 1570.5540
## 373       944.6442 526.3968 1362.8917 304.989987 1584.2985
## 374       922.1050 493.9845 1350.2255 267.351218 1576.8588
## 375       888.7065 455.0737 1322.3393 225.522309 1551.8907
## 376       848.2483 412.4150 1284.0815 181.698857 1514.7976
## 377       804.6341 368.4252 1240.8430 137.510178 1471.7581
## 378       761.5616 325.3117 1197.8116  94.374911 1428.7483
## 379       722.2628 285.1875 1159.3381  53.813799 1390.7118
## 380       689.3104 250.0989 1128.5220  17.594346 1361.0265
## 381       664.4969 221.9179 1107.0760 -12.369325 1341.3632
## 382       648.7866 202.1235 1095.4498 -34.325689 1331.8990
## 383       642.3372 191.5597 1093.1147 -47.067437 1331.7418
## 384       644.5798 190.2678 1098.8918 -50.230505 1339.3901
```

We got Arima(2,0,3) Now we will change the p and a

```
## Series: ddts
## ARIMA(3,0,3) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ar3      ma1      ma2      ma3      mean
##       1.6567  -0.6057  -0.1392  -0.3870  -0.3533  -0.1644  746.3246
## s.e.  0.2818   0.5176   0.2584   0.2756   0.1751   0.1006    6.8677
##
## sigma^2 estimated as 13468:  log likelihood=-2220.62
## AIC=4457.24   AICc=4457.65   BIC=4488.33
```

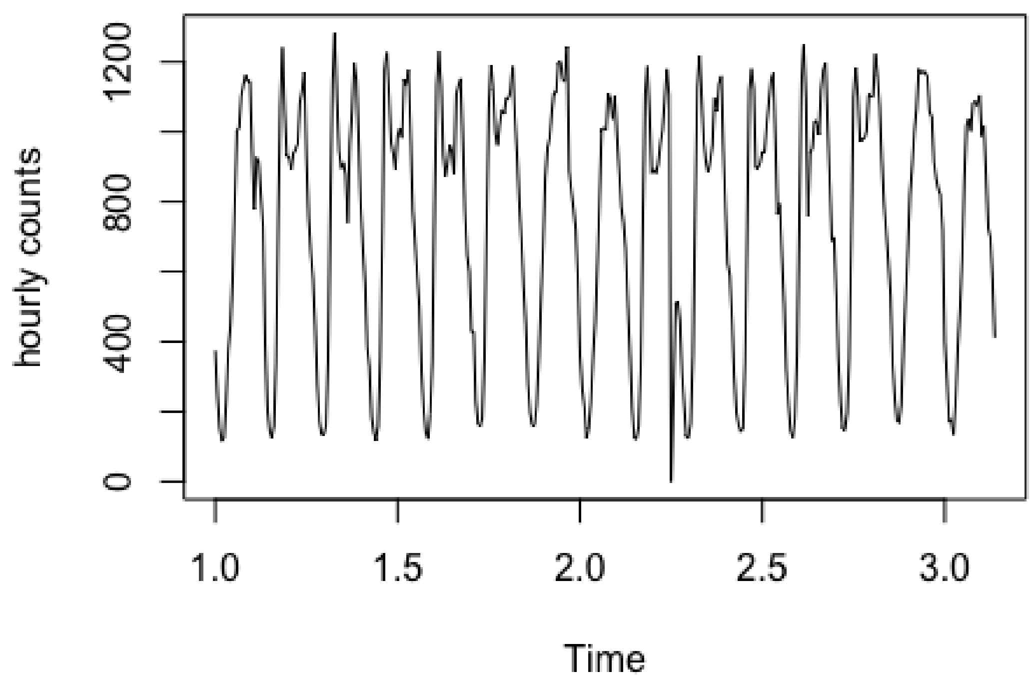
```
## Series: ddts
## ARIMA(2,0,4) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1      ma2      ma3     ma4      mean
##       1.8479  -0.9224  -0.6164  -0.2987  -0.1278  0.1245  746.3452
## s.e.  0.0402   0.0384   0.0954   0.0737   0.0662  0.1098    6.8874
##
## sigma^2 estimated as 13407:  log likelihood=-2219.86
## AIC=4455.73   AICc=4456.14   BIC=4486.81
```

```
## Series: ddts
## ARIMA(3,0,4) with non-zero mean
##
## Coefficients:
##          ar1     ar2      ar3     ma1      ma2      ma3      ma4      mean
##       0.8323  0.8768  -0.8601  0.4579  -0.7879  -0.3761  -0.1303  746.2890
## s.e.  0.0314  0.0269   0.0319  0.0618   0.0688   0.0607   0.0658    6.8532
##
## sigma^2 estimated as 13339:  log likelihood=-2219.24
## AIC=4456.47   AICc=4456.98   BIC=4491.45
```

AICc and BIC select the same best model for us.

Part 2

```
ddts_mon <- ts(dd, frequency = 168)
plot.ts(ddts_mon)
```




```
fit_ddMon <- auto.arima(ddts_mon, seasonal = TRUE)
fit_ddMon

## Series: ddts_mon
## ARIMA(0,1,2)(0,1,0)[168]
##
```

```
## Coefficients:
##           ma1      ma2
##       -0.4741  -0.4853
## s.e.   0.0593   0.0586
##
## sigma^2 estimated as 7080:  log likelihood=-1121.64
## AIC=2249.29   AICc=2249.42   BIC=2259.04
```

```
m <- forecast(fit_ddMon, h = 24)
m
```

```
##          Point Forecast      Lo 80     Hi 80      Lo 95     Hi 95
## 3.142857      231.35454  123.51849  339.1906   66.43353  396.2755
## 3.148810      140.97896   19.13917  262.8188  -45.35892  327.3168
## 3.154762      141.97896   20.06044  263.8975  -44.47933  328.4373
## 3.160714      176.97896   54.98176  298.9762   -9.59966  363.5576
## 3.166667      352.97896  230.90313  475.0548  166.28009  539.6778
## 3.172619      775.97896  653.82455  898.1334  589.15991  962.7980
## 3.178571     1125.97896 1003.74602 1248.2119  939.03981 1312.9181
## 3.184524     1205.97896 1083.66754 1328.2904 1018.91979 1393.0381
## 3.190476     1080.97896  958.58912 1203.3688  893.79984 1268.1581
## 3.196429      899.97896  777.51074 1022.4472  712.67998 1087.2779
## 3.202381      909.97896  787.43241 1032.5255  722.56018 1097.3977
## 3.208333      898.97896  776.35413 1021.6038  711.44047 1086.5175
## 3.214286      926.97896  804.27591 1049.6820  739.32083 1114.6371
## 3.220238      982.97896  860.19773 1105.7602  795.20127 1170.7567
## 3.226190     1022.97896  900.11960 1145.8383  835.08178 1210.8761
## 3.232143     1104.97896  982.04152 1227.9164  916.96237 1292.9956
## 3.238095     1196.97896 1073.96349 1319.9944 1008.84304 1385.1149
## 3.244048     1125.97896 1002.88551 1249.0724  937.72378 1314.2341
## 3.250000       18.07896 -105.09241  141.2503 -170.29540  206.4533
## 3.255952      270.97896  147.72970  394.2282   82.48549  459.4724
## 3.261905      525.97896  402.65187  649.3061  337.36646  714.5915
## 3.267857      534.97896  411.57409  658.3838  346.24750  723.7104
## 3.273810      476.97896  353.49636  600.4616  288.12861  665.8293
## 3.279762      326.97896  203.41867  450.5393  138.00981  515.9481
```

```
m$mean
```

```
## Time Series:
## Start = c(3, 25)
## End = c(3, 48)
## Frequency = 168
##  [1]  231.35454  140.97896  141.97896  176.97896  352.97896  775.97896
##  [7] 1125.97896 1205.97896 1080.97896  899.97896  909.97896  898.97896
## [13]  926.97896  982.97896 1022.97896 1104.97896 1196.97896 1125.97896
## [19]   18.07896  270.97896  525.97896  534.97896  476.97896  326.97896
```

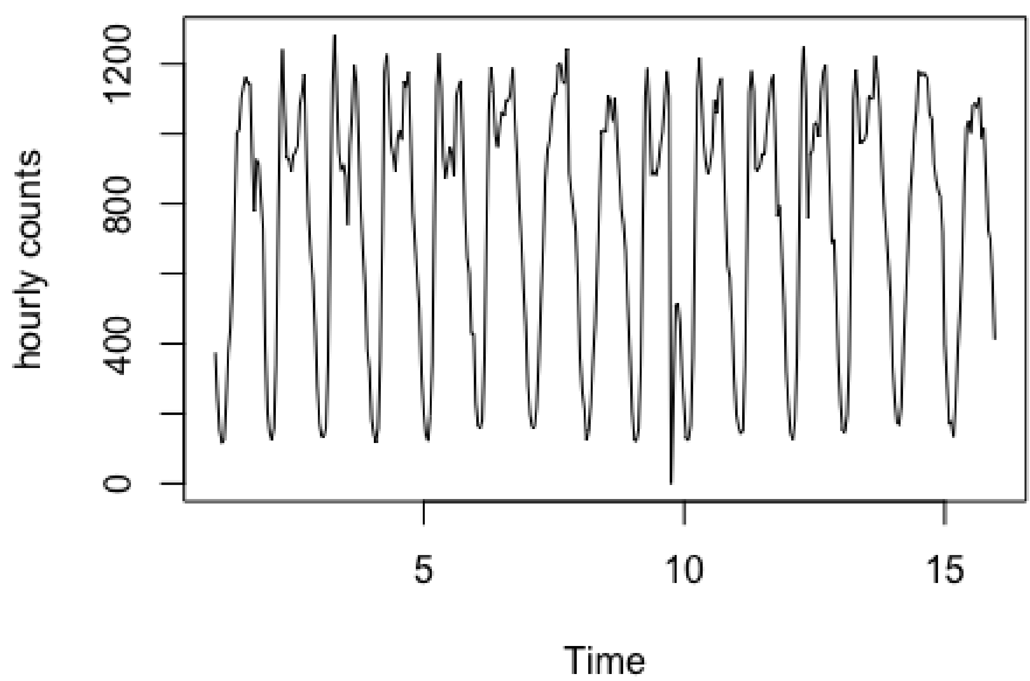
What is shown above is the July 1st data, which we observed from a day of the week perspective. So the hourly counts at 8:00, 9:00, 17:00 and 18:00 is 1218.4032, 1067.4875, 1154.9122， 1143.3740.

```r
E1 <- 1105 - 1205.97855
E2 <- 1233 - 1080.97855
E3 <- 1059 - 1196.97855
E4 <- 1142 - 1125.97855
mm <- list(E1, E2, E3, E4)
mmm <- unlist(mm)
sum(mmm^2)

## [1] 52601.96
```

Part 3

```r
ddhour <- ts(dd, frequency = 24)
plot.ts(ddhour)
```




```r
fit_ddhour <- auto.arima(ddhour, seasonal = TRUE)
fit_ddhour

## Series: ddhour
## ARIMA(2,0,1)(2,0,0)[24] with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1    sar1    sar2      mean
##       1.7922  -0.8685  -0.9146  0.4866  0.1010  743.7285
```

```
## s.e.  0.0299   0.0291   0.0257  0.0555  0.0557   13.6798
##
## sigma^2 estimated as 10736:  log likelihood=-2184.1
## AIC=4382.2   AICc=4382.52   BIC=4409.4
```

```
n <- forecast(fit_ddhour, h=24 )
n
```

```
##          Point Forecast      Lo 80     Hi 80       Lo 95     Hi 95
## 16.00000       288.2882 155.49802  421.0784  85.203136  491.3733
## 16.04167       292.1628 115.48895  468.8366  21.963480  562.3621
## 16.08333       295.9552  96.05371  495.8567  -9.767754  601.6782
## 16.12500       369.1913 158.54877  579.8338  47.041351  691.3412
## 16.16667       416.8688 202.86415  630.8735  89.576924  744.1607
## 16.20833       533.8043 319.56394  748.0446 206.151962  861.4566
## 16.25000       661.2118 446.46016  875.9635 332.777493  989.6462
## 16.29167       756.5510 538.92261  974.1794 423.717100 1089.3849
## 16.33333       854.1021 630.72667 1077.4774 512.478886 1195.7252
## 16.37500       970.2430 739.05473 1201.4312 616.671058 1323.8149
## 16.41667      1044.1506 804.51100 1283.7902 677.653446 1410.6478
## 16.45833      1037.2573 789.90297 1284.6117 658.961477 1415.5532
## 16.50000      1004.1431 750.73333 1257.5528 616.586313 1391.6998
## 16.54167      1009.4817 752.02743 1266.9360 615.739371 1403.2240
## 16.58333       981.8841 722.24134 1241.5268 584.794786 1378.9734
## 16.62500       945.2634 684.78798 1205.7389 546.900599 1343.6263
## 16.66667       933.5054 672.90457 1194.1062 534.950825 1332.0600
## 16.70833       846.3404 585.71482 1106.9659 447.747989 1244.9327
## 16.75000       845.5000 584.52750 1106.4724 446.377028 1244.6229
## 16.79167       757.6471 495.83187 1019.4624 357.235255 1158.0590
## 16.83333       677.0131 413.91637  940.1098 274.641368 1079.3848
## 16.87500       665.1818 400.57256  929.7911 260.496884 1069.8667
## 16.91667       614.4318 348.33397  880.5297 207.470284 1021.3934
## 16.95833       534.6246 267.27539  801.9738 125.749272  943.4999
```

The hourly counts for July.1st at 8:00, 9:00, 17：00 and 18:00 is: 756.5516, 854.0998, 933.5026, 846.3402

```
m1 <- 1105 - 756.5516
m2 <- 1233 - 854.0998
m3 <- 1059 - 933.5026
m4 <- 1142 - 846.3402
mn <- list(m1, m2, m3, m4)
mnm <- unlist(mn)
sum(mnm^2)
```

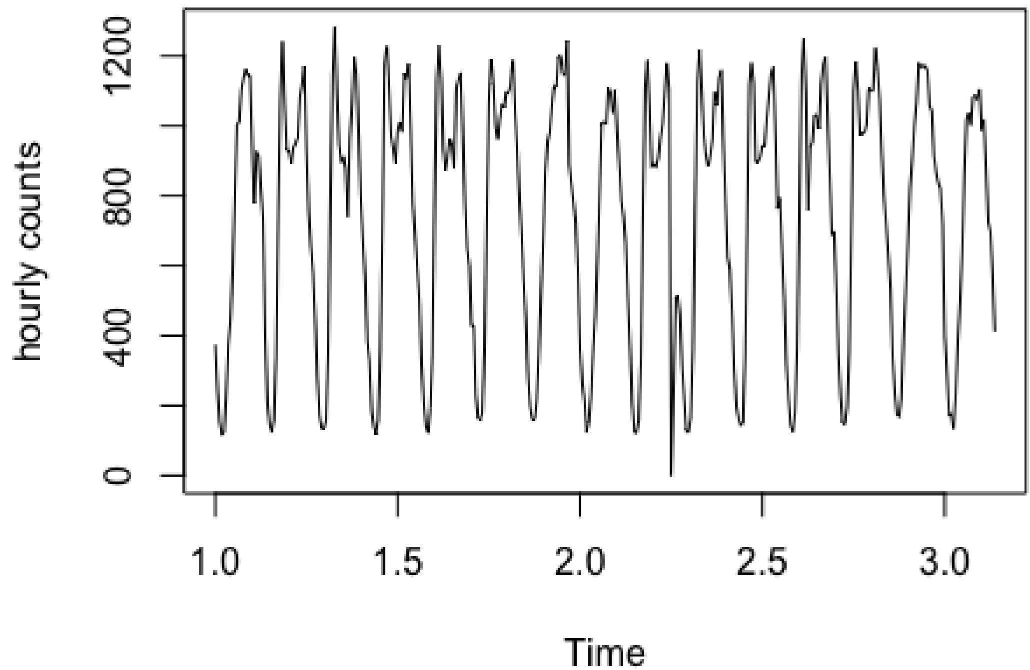
```
## [1] 368146
```

Part4

From the above two results in part 2 and 3, we can see that the sum of squared errors in part 3 is much greater than part2. Therefore we assume the model in Part2 can help us in better prediction.

Part5 Repeat Part 2 with Holt-Winters additive and multiplicative seasonality models

```
library(forecast)
library(stats)
ddts_mon <- ts(dd, frequency = 168)
plot.ts(ddts_mon)
```




```
#additive
ddts_monm <- HoltWinters(ddts_mon, l.start = 375, seasonal = "additive")
hwa <- forecast(ddts_monm, h = 24)
hwa$mean

## Time Series:
## Start = c(3, 25)
## End = c(3, 48)
## Frequency = 168
##  [1]  256.48019  169.97708  170.33706  204.91133  380.69691  803.39618
##  [7] 1153.06271 1232.59829 1107.17256  925.72302  935.06812  923.93406
## [13]  952.14225 1008.15402 1048.18663 1130.13589 1222.09112 1151.14753
## [19]   43.17001  295.95083  550.97450  560.05472  502.03375  352.06040
```

```
#multiplicative
# ddts_monmm <- HoltWinters(ddts_mon, seasonal = "multiplicative")
# hwm <- forecast(ddts_monmm, h = 24)
# hwm$mean
#both
ddts_monmm <- HoltWinters(ddts_mon, l.start = 375, seasonal =
"multiplicative")
hwm <- forecast(ddts_monmm, h = 24)
hwm$mean

## Time Series:
## Start = c(3, 25)
## End = c(3, 48)
## Frequency = 168
##  [1]  229.2586268  134.8311249  135.8100578  174.0439512  366.5916682
##  [6]  829.1483132 1211.4592586 1298.1194250 1160.8697282  962.6246091
## [11]  972.6824175  960.5180718  991.3286405 1052.4222692 1096.0994160
## [16] 1185.4567373 1285.7180516 1208.3892021    0.1090495  275.8520852
## [21]  553.9051313  563.7821603  500.5220412  336.9664647
```

The hourly counts for 8:00, 9:00, 17:00 and 18:00 ： 1232.59825, 1107.17252, 1222.09104, 1151.14745

```
library(hydroGOF)

## Warning: package 'hydroGOF' was built under R version 3.3.2

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 3.3.2

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

g1 <- 1105 - 1232.59825
g2 <- 1233 - 1107.17252
g3 <- 1059 - 1222.09104
g4 <- 1142 - 1151.14745
gg <- list(g1, g2, g3, g4)
ggu <- unlist(gg)
sum(ggu^2)

## [1] 58796.23
```

Part6: Compared with the result in Part2, we can see the mse in Part 2 is bigger than that of Part5 additive. So part 5 additive model is better than part2.

```r
library(hydroGOF)
#part 2 and the observed values comparison
mse(m$mean, p)

## [1] 49202.88

#part 5 and the observed values comparison
mse(hwa$mean,p)

## [1] 46585.51

mse(hwm$mean,p)

## [1] 53099.81
```

The mean of squared error we got from part 2 is 49202.88, in Part5 additive it's 46585.51, Part5 multiplicative it's 53099.81.