PATH 2020-06

Planning & Regional Development

5/29/2020

To do list: Copy and move input files - input variables and dates. DONE. Incorporate quarter-to-month interpolation. DONE. Bind dates, ridership, monthly economics, other. DONE. Confirm represents 2019 Q3 forecast outputs. DONE. Add and update diagnostics. DONE. Package outputs (inputs, forecast and diagnostics). Automate real fare calculation. Rebuild process with new component pieces. Clean code. Add narrative. Package (zip?) and share.

1. Setup setup setup.

```
setwd("~/Dropbox/Work and research/Port Authority/pathforecast")

cat("\014") # clear the console
```

```
rm(list=ls())
options(scipen=999)

library(broom)
library(knitr)
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

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

```
library(reshape2)
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 3.6.2
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(tseries)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(doBy)
library(mice)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'mice'
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
library(lmtest)
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:mice':
##
##     complete
```

```
## The following object is masked from 'package:reshape2':
##
##     smiths
```

```
library(zoo)
#install.packages("openxlsx")
library(openxlsx)
```

```
## Warning: package 'openxlsx' was built under R version 3.6.2
```

```
#install.packages("xlsxjars")
#library(xlsxjars)
#install.packages("XLConnect")
#library(XLConnect)
#install.packages("xlsx")
#library(xlsx)
#library(yardstick)

set.seed(101)

start = "2004-01-01"
end = "2019-08-01" #"2020-02-01" #"2019-12-01"
end_and_one = "2019-09-01" #"2020-03-01" #"2020-01-01"
extra = as.Date(end_and_one)-as.Date(end)
future = "2040-12-31"
```

```r
elapsed_months <- function(end_date, start_date) {
  ed <- as.POSIXlt(end_date)
  sd <- as.POSIXlt(start_date)
  12 * (ed$year - sd$year) + (ed$mon - sd$mon)
}
horizon = elapsed_months(future,start)+1
forec_horizon = elapsed_months(future,end)
forec_horizon
```

```
## [1] 256
```

2. Interpolation: convert quarterly data to monthly.

```r
jobs = read.csv("./input data/econ_vars_quar 2020_06.csv")
jobs1 = jobs#[complete.cases(jobs),]

# IF QUARTERS ARE 'AVERAGE' OF RELEVANT THREE MONTHS (define 'average' later)
jobs1$year=NULL
jobs1$quarter=NULL
jobs1$Indicator=NULL

jobs1$Month = as.Date(jobs1$Month, "%m/%d/%y")

jobs2 = jobs1
jobs2$quarter2 = NULL
jobs2 = read.zoo(jobs2) # Converts the data frame to a time series matrix

tt = as.yearmon(seq(start(jobs2), end(jobs2), "month")) # Makes months, different format (unsure why ne
jobs2$Indicator = NULL

zm = as.data.frame(na.spline(jobs2, as.yearmon, xout = tt))
zm$month_ = seq(as.Date("1996/1/1"), as.Date("2035/10/01"), by="month") # Add date

zm2 = subset(zm,zm$month_=="2035-10-01")
zm3 = subset(zm,zm$month_=="2035-10-01")
zm2$month_="2035-11-01"
zm3$month_="2035-12-01"
zm=rbind(zm,zm2)
zm=rbind(zm,zm3)
zm = zm[order(as.Date(zm$month_, format="%Y-%m-%d")),]

write.csv(zm,"./input data/econ_vars_months 2020_06.csv")
jobs_month = zm
names(jobs_month) = tolower(names(jobs_month))

rm(jobs,jobs1,jobs2,tt,zm,zm2,zm3)
```

3. Load data load data.

```
days = read.csv("./input data/dates_dummies.csv")
#; names(days) = c("month","weekdays","saturdays","sundays")
days$month = as.Date(days$month, "%m/%d/%y")

path = read.csv("./input data/PATH input 2020_06.csv")
path$month = as.Date(days$month, "%m/%d/%y")

other = read.csv("./input data/other.csv")
other$month = as.Date(other$month, "%m/%d/%y")

path = merge(days,path)#,by="month")
path = merge(path,other)#,by="month")
#path$month = as.Date(days$month,format="%m/%d/%y")
```

4.Prepare for model.

```
before = subset(path,path$month<=end & path$month>="2004-01-01")
#before = head(path,218)
after = subset(path,path$month>end)
#after = tail(path,250)

### WEEKDAYS
oldreg=as.matrix(data.frame(before$man_hud_KEEP,                                before$dummy_2,before$dummy
                       before$dummy_6,before$dummy_7,before$dummy_8,before$dummy_9,before$dummy_10,before
                             before$supersandy, before$real_fare_q1_KEEP)) #real_fare_q4
newreg=as.matrix(data.frame(after$man_hud_KEEP,
                             after$dummy_2,after$dummy_3,after$dummy_4,after$dummy_5, after$dum_911_base
                             after$dummy_6, after$dummy_7, after$dummy_8, after$dummy_9, after$dummy_10,
                             after$supersandy, after$real_fare_q1_KEEP))

### SATURDAY & SUNDAY
oldregsat=as.matrix(data.frame(before$pop_hudson_KEEP,before$dummy_2, before$dummy_3, before$dummy_4,be
                             before$dummy_5, before$dummy_6, before$dummy_7, before$dummy_8, before$dummy
                             before$dummy_12,before$supersandy, before$end_close,
                             before$real_fare_q1_KEEP))
newregsat=as.matrix(data.frame(after$pop_hudson_KEEP,after$dummy_2, after$dummy_3, after$dummy_4,after$
                             after$dummy_5, after$dummy_6, after$dummy_7, after$dummy_8, after$dummy_9,
                             after$dummy_12,after$supersandy, after$end_close,
                             after$real_fare_q1_KEEP))

#t.test(before$sun,before$real_fare)
```

5. Models and forecasts.

5a. Fit models (estimate equations).

Weekday

```
summary(before)
```

```
##      month            num_wkdayholminor num_satholmajor    num_sun
##  Min.   :2004-01-01   Min.   :18.00     Min.   :4.000   Min.   :4.000
##  1st Qu.:2007-11-23   1st Qu.:20.00     1st Qu.:4.000   1st Qu.:4.000
##  Median :2011-10-16   Median :21.00     Median :5.000   Median :4.000
##  Mean   :2011-10-16   Mean   :20.99     Mean   :5.096   Mean   :4.346
##  3rd Qu.:2015-09-08   3rd Qu.:22.00     3rd Qu.:6.000   3rd Qu.:5.000
##  Max.   :2019-08-01   Max.   :23.00     Max.   :7.000   Max.   :5.000
##
##     weekdays       saturdays        sundays       dum_911_base
##  Min.   :18.00   Min.   :4.000   Min.   :4.000   Min.   :0.000000
##  1st Qu.:21.00   1st Qu.:4.000   1st Qu.:4.000   1st Qu.:0.000000
##  Median :21.00   Median :5.000   Median :4.000   Median :0.000000
##  Mean   :21.36   Mean   :4.734   Mean   :4.346   Mean   :0.005319
##  3rd Qu.:22.00   3rd Qu.:5.000   3rd Qu.:5.000   3rd Qu.:0.000000
##  Max.   :23.00   Max.   :7.000   Max.   :5.000   Max.   :1.000000
##
##    supersandy      summer_of_hell      end_close           mon
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   : 1.000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.: 3.000
##  Median :0.00000   Median :0.00000   Median :0.0000   Median : 6.000
##  Mean   :0.01064   Mean   :0.01064   Mean   :0.2287   Mean   : 6.415
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.: 9.000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :12.000
##
##     dummy_1           dummy_2           dummy_3           dummy_4
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.08511   Mean   :0.08511   Mean   :0.08511   Mean   :0.08511
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##
##     dummy_5           dummy_6           dummy_7           dummy_8
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.08511   Mean   :0.08511   Mean   :0.08511   Mean   :0.08511
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##
##     dummy_9           dummy_10          dummy_11          dummy_12
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.07979   Mean   :0.07979   Mean   :0.07979   Mean   :0.07979
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##
##       week             sat              sun            sat_alt
##  Min.   :121710   Min.   : 59748   Min.   : 37912   Min.   : 70082
##  1st Qu.:233081   1st Qu.: 99235   1st Qu.: 73949   1st Qu.: 98245
##  Median :250484   Median :110079   Median : 83081   Median :111618
##  Mean   :246583   Mean   :108200   Mean   : 81976   Mean   :109729
##  3rd Qu.:264202   3rd Qu.:118217   3rd Qu.: 91394   3rd Qu.:119979
```

```
##  Max.   :295574   Max.   :137725   Max.   :105328   Max.   :137725
##  NA's   :3        NA's   :3        NA's   :3        NA's   :43
##     sun_alt       avg_wkdayholminor_tstile avg_satholmajor_tstile
##  Min.   : 49634   Min.   :121710           Min.   : 59748
##  1st Qu.: 75649   1st Qu.:233191           1st Qu.: 98255
##  Median : 85282   Median :251548           Median :108878
##  Mean   : 85076   Mean   :247219           Mean   :107652
##  3rd Qu.: 92981   3rd Qu.:264720           3rd Qu.:118234
##  Max.   :123865   Max.   :295574           Max.   :137725
##  NA's   :43
##  avg_sun_tstile     real_fare      man_hud_KEEP   man_hud_opt    man_hud_pess
##  Min.   : 37912   Min.   :1.735   Min.   :2316   Min.   :2316   Min.   :2316
##  1st Qu.: 74477   1st Qu.:1.953   1st Qu.:2404   1st Qu.:2404   1st Qu.:2404
##  Median : 83124   Median :2.136   Median :2492   Median :2492   Median :2492
##  Mean   : 82013   Mean   :2.270   Mean   :2552   Mean   :2553   Mean   :2552
##  3rd Qu.: 91317   3rd Qu.:2.750   3rd Qu.:2708   3rd Qu.:2709   3rd Qu.:2708
##  Max.   :105328   Max.   :2.869   Max.   :2895   Max.   :2898   Max.   :2892
##
##  population_hud_opt population_hud_pess real_fare_q1_KEEP pop_hudson_KEEP
##  Min.   :614.1      Min.   :614.1       Min.   :1.804     Min.   :614.1
##  1st Qu.:616.6      1st Qu.:616.6       1st Qu.:2.033     1st Qu.:616.6
##  Median :648.0      Median :648.0       Median :2.233     Median :648.0
##  Mean   :644.5      Mean   :644.5       Mean   :2.362     Mean   :644.5
##  3rd Qu.:666.0      3rd Qu.:666.0       3rd Qu.:2.810     3rd Qu.:666.0
##  Max.   :681.1      Max.   :681.1       Max.   :2.986     Max.   :681.2
##
```

```
fit = arima(ts(before$avg_wkdayholminor_tstile),xreg = oldreg, order=c(0,0,1), include.mean=T)#method="
#fit = arima(ts(before$avg_wkdayholminor_tstile),xreg = oldreg, order=c(0,0,1), include.mean=T)# as of
  #fit = auto.arima(ts(before$week),xreg=oldreg, ic="aic", trace=TRUE, allowdrift=FALSE)#,lambda=0, sea
```

Saturday

```
fitsat = arima(ts(before$avg_satholmajor_tstile),xreg=oldregsat,order=c(1,1,0))# as of 2018-09 (1,1,0)
#fitsat = arima(ts(before$avg_satholmajor_tstile),xreg=oldregsat,order=c(1,1,0))# as of 2018-09 (1,1,0)
  #fitsat = auto.arima(ts(before$sat_mice),xreg=oldregsat, ic="aic", trace=TRUE, allowdrift=FALSE ,lamb
```

Sunday

```
fitsun = arima(ts(before$avg_sun_tstile),xreg=oldregsat,order=c(1,1,1))# as of 2018-09 (1,1,1) before t
#fitsun = arima(ts(before$avg_sun_tstile),xreg=oldregsat,order=c(1,1,1))# as of 2018-09 (1,1,1) before
  #fitsun = auto.arima(ts(before$sun_mice),xreg=oldregsat, ic="aic", trace=TRUE, allowdrift=FALSE,lambd
```

5b. Predict (forecast).

```
pathpredict = predict(fit, n.ahead=forec_horizon, newxreg=newreg, level=95)#interval = "prediction", co
pathpredictsat = predict(fitsat, n.ahead=forec_horizon, newxreg=newregsat) # predict
pathpredictsun = predict(fitsun, n.ahead=forec_horizon, newxreg=newregsat) # predict
```

6. Clean and consolidate results for export.

```
pathpredict_by_month = as.data.frame(cbind(pathpredict$pred,pathpredictsat$pred,pathpredictsun$pred)); r
head(pathpredict_by_month,3)
```

```
##   week_avg  sat_avg  sun_avg
## 1 297069.1 107832.8 85560.62
## 2 298801.3 114105.4 83080.65
## 3 292963.2 107481.8 79396.11
```

```
end
```

```
## [1] "2019-08-01"
```

```
future
```

```
## [1] "2040-12-31"
```

```
pathpredict_by_month$month = seq(as.Date(end)+extra,as.Date(future),by="mon")

## Add old stuff (January 2017, for example) back to the pile.
before_mini = data.frame((before$week*before$weekdays),(before$sat*before$saturdays),(before$sun*before$
  names(before_mini) = c("week","sat","sun","month")

## Now multiply by number of days per month ...
pathpredict_by_month = merge(pathpredict_by_month,days)
pathpredict_by_month$week = pathpredict_by_month$week_avg*pathpredict_by_month$weekdays
pathpredict_by_month$sat = pathpredict_by_month$sat_avg*pathpredict_by_month$saturdays
pathpredict_by_month$sun = pathpredict_by_month$sun_avg*pathpredict_by_month$sundays


pathpredict_mini = data.frame(pathpredict_by_month$month,pathpredict_by_month$week,pathpredict_by_month$
  names(pathpredict_mini) = c("month","week","sat","sun")
```

7. Model diagnostics.

```
out1 = tidy(fit)
  #out2 = tidy(glance(fit)) ## why is this crashing my program?
out2 = glance(fit)
out2.5 = accuracy(fit)
out3 = tidy(fitsat)
out4 = glance(fitsat)
out4.5 = accuracy(fitsat)
out5 = tidy(fitsun)
out6 = glance(fitsun)
out6.5 = accuracy(fitsun)

#accuracy(fit)[,'MAPE']

pathpredict_month_backup = pathpredict_by_month
pathpredict_month = rbind(before_mini,pathpredict_mini) #meh. figure this out later


pathpredict_month$year = year(pathpredict_month$month)
```

```
pathpredict_year = summaryBy(week + sat + sun ~ year, data = pathpredict_month, FUN = sum); names(pathpr
pathpredict_year$total = pathpredict_year$week + pathpredict_year$sat + pathpredict_year$sun
pathpredict_month$year = NULL
years = pathpredict_year
pathpredict_year[14,5] = 76565451
pathpredict_year[15,5] = 78517120


resids = as.data.frame(cbind(as.vector(resid(fit)),as.vector(resid(fitsat)),as.vector(resid(fitsun))));
```

```
out1
```

```
## # A tibble: 17 x 3
##    term                        estimate  std.error
##    <fct>                          <dbl>      <dbl>
##  1 ma1                            0.583     0.0477
##  2 intercept                -102680.    27073.
##  3 before.man_hud_KEEP          151.       15.9
##  4 before.dummy_2              5142.     3444.
##  5 before.dummy_3              7748.     4541.
##  6 before.dummy_4             13160.     4542.
##  7 before.dummy_5             15899.     4541.
##  8 before.dum_911_base       -21942.    11415.
##  9 before.dummy_6             19189.     4541.
## 10 before.dummy_7             15141.     4542.
## 11 before.dummy_8              9080.     4544.
## 12 before.dummy_9             19451.     4597.
## 13 before.dummy_10            17378.     4617.
## 14 before.dummy_11            11258.     4667.
## 15 before.dummy_12             1442.     3534.
## 16 before.supersandy         -76224.    10174.
## 17 before.real_fare_q1_KEEP  -19880.     6985.
```

```
out2
```

```
## # A tibble: 1 x 4
##    sigma logLik   AIC   BIC
##    <dbl>  <dbl> <dbl> <dbl>
## 1 10957. -2016. 4067. 4126.
```

```
out2.5
```

```
##                     ME     RMSE      MAE        MPE     MAPE     MASE      ACF1
## Training set 3.141493 10957.35 8345.181 -0.2913649 3.638641 1.128409 0.3021086
```

```
out3
```

```
## # A tibble: 17 x 3
##    term                         estimate std.error
##    <fct>                           <dbl>     <dbl>
##  1 ar1                           -0.554    0.0616
##  2 before.pop_hudson_KEEP       580.      668.
##  3 before.dummy_2              6696.     1847.
##  4 before.dummy_3             20234.     1677.
##  5 before.dummy_4             22236.     2027.
##  6 before.dum_911_base          887.     6299.
##  7 before.dummy_5             14092.     2018.
```

```
##  8 before.dummy_6            21080.    2141.
##  9 before.dummy_7            17123.    2120.
## 10 before.dummy_8            15770.    2162.
## 11 before.dummy_9            16841.    2058.
## 12 before.dummy_10           20420.    2075.
## 13 before.dummy_11           14783.    1727.
## 14 before.dummy_12           14785.    1918.
## 15 before.supersandy        -44033.    4191.
## 16 before.end_close         -12888.    2132.
## 17 before.real_fare_q1_KEEP -25863.    9896.
```

```
out4
```

```
## # A tibble: 1 x 4
##   sigma logLik   AIC   BIC
##   <dbl>  <dbl> <dbl> <dbl>
## 1 6083. -1895. 3826. 3884.
```

```
out4.5
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 72.52746 6066.528 4737.999 -0.1610564 4.464436 0.5634586
##                     ACF1
## Training set -0.1693733
```

```
out5
```

```
## # A tibble: 18 x 3
##    term                        estimate std.error
##    <fct>                          <dbl>     <dbl>
##  1 ar1                          -0.0448    0.0896
##  2 ma1                          -0.779     0.0523
##  3 before.pop_hudson_KEEP      340.      193.
##  4 before.dummy_2             5340.     1607.
##  5 before.dummy_3             8073.     1589.
##  6 before.dummy_4            14119.     1603.
##  7 before.dum_911_base       -2784.     5028.
##  8 before.dummy_5            18641.     1609.
##  9 before.dummy_6            24063.     1614.
## 10 before.dummy_7            17621.     1617.
## 11 before.dummy_8            16075.     1622.
## 12 before.dummy_9            20764.     1641.
## 13 before.dummy_10           17986.     1635.
## 14 before.dummy_11           14125.     1636.
## 15 before.dummy_12           16170.     1660.
## 16 before.supersandy        -37133.     3682.
## 17 before.end_close         -10414.     1410.
## 18 before.real_fare_q1_KEEP -12656.     5684.
```

```
out6
```

```
## # A tibble: 1 x 4
##   sigma logLik   AIC   BIC
##   <dbl>  <dbl> <dbl> <dbl>
## 1 4870. -1854. 3745. 3807.
```

```
out6.5
```

```
##                     ME     RMSE      MAE         MPE     MAPE      MASE
```

```
## Training set 205.2657 4857.256 3525.345 -0.06577637 4.516437 0.5203284
##                     ACF1
## Training set -0.00942956
```

8. Save (export).

```
tail(pathpredict_year)
```

```
##    year     week     sat     sun     total
## 32 2035 81510650 8176599 6976975  96664224
## 33 2036 82428256 8213474 7003227  97644956
## 34 2037 83129361 8250410 7029522  98409292
## 35 2038 83833420 8287408 7055860  99176688
## 36 2039 84541604 8324463 7082240  99948307
## 37 2040 85238575 8357797 7105810 100702182
```

```
#write.csv(pathpredict_month_backup, "./PATH output test 20200529.csv")

#write.csv(pathpredict_by_month,"./PATH forecast products/PATH forecast output/PATH q2/PATH month_ 2019
#  write.csv(fitted(fit), "./PA PATH output & viz/PATH fitted _week 2020q1.csv")
#  write.csv(fitted(fitsat), "./PA PATH output & viz/PATH fitted _sat 2020q1.csv")
#  write.csv(fitted(fitsun), "./PA PATH output & viz/PATH fitted _sun 2020q1.csv")
#  write.csv(resids, "./PATH forecast products/PATH forecast output/PATH q2/PATH residuals _nodummy 201

#write.csv(pathpredict_year,"./PA PATH output & viz/PATH ANNUAL_Q4 FARE TRNSTL RIDERS PESS.csv")
write.csv(pathpredict_by_month, "./output data/monthly 2020-06.csv")
```

XXX. Try saving diagnostics.

```
#xlcFreeMemory() # (this is related to the free memory command above (when loading XLConnect))

#wb = loadWorkbook(("./Diagnostics V1.xlsx"), create = TRUE)
#createSheet(wb, name = "Annual")
#writeWorksheet(wb, path_ann, sheet = "Annual", startRow = 3, startCol = 3, header = TRUE)
#writeWorksheet(wb, path_month, sheet = "Monthly", startRow = 3, startCol = 15, header = TRUE)
#writeWorksheet(wb, as.numeric(fitted(fit)), sheet = "Monthly", startRow = 3, startCol = 25, header=TRU
#writeWorksheet(wb, as.numeric(fitted(fitsat)), sheet = "Monthly", startRow = 3, startCol = 26, header=
#writeWorksheet(wb, as.numeric(fitted(fitsun)), sheet = "Monthly", startRow = 3, startCol = 27, header=
#createSheet(wb, name = "Diagnostics")
#writeWorksheet(wb, out1, sheet = "Diagnostics", startRow = 4, startCol = 2, header = TRUE)
#writeWorksheet(wb, out2, sheet = "Diagnostics", startRow = 24, startCol = 2, header = TRUE)
#writeWorksheet(wb, out3, sheet = "Diagnostics", startRow = 38, startCol = 2, header = TRUE)
#writeWorksheet(wb, out4, sheet = "Diagnostics", startRow = 60, startCol = 2, header = TRUE)
#writeWorksheet(wb, out5, sheet = "Diagnostics", startRow = 72, startCol = 2, header = TRUE)
#writeWorksheet(wb, out6, sheet = "Diagnostics", startRow = 92, startCol = 2, header = TRUE)
#writeWorksheet(wb, durbinWatsonTest(as.vector(resid(fit))), sheet = "Diagnostics", startRow=102, start
#writeWorksheet(wb, durbinWatsonTest(as.vector(resid(fitsat))), sheet = "Diagnostics", startRow=104, st
#writeWorksheet(wb, durbinWatsonTest(as.vector(resid(fitsun))), sheet = "Diagnostics", startRow=106, st
#setColumnWidth(wb, sheet = "Diagnostics", column = 3:4, width = -1)  #automatically adjust column widt
#writeWorksheet(wb, resids, sheet = "Diagnostics", startRow = 4, startCol = 24, header = TRUE)
#saveWorkbook(wb)
```