Modelling and Predicting the Heat Consumption

Assignment 4 – 02417 Time Series Analysis – Anders Hørsted (s082382)

In this report models describing the heat consumption in the VEKS system are build. As data for the models, a year of hourly measurements of the heat consumption along with various climate variables are given. First a simple model, based only on the heat consumption measurements, is build. Then a more complex model that incorporates information from the climate data is build. For both models 1- and 6-hour predictions of the heat consumption are made and the performance is checked by comparing the predictions with the actual values.

Building the simple model

In the simple model only the data of past heat consumption is used. First the data is plotted and shown in figure 1.

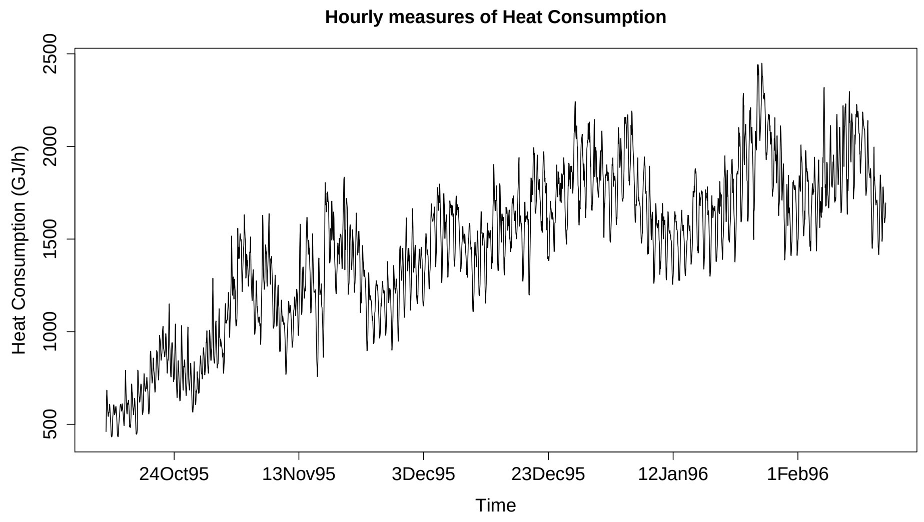


Figure 1: Heat Consumption time series

From the plot an upward trend is seen and high frequency oscillations are also noticed. Also it looks as if the variance increases along with the time. To look into this a range-mean plot is created and shown in figure 2.

Although a linear trend can be seen in the range-mean plot, most groups are gathered in what seems to be a random cluster. Only group 1 and 2 have lower means. The conclusion is that the original data is not transformed.

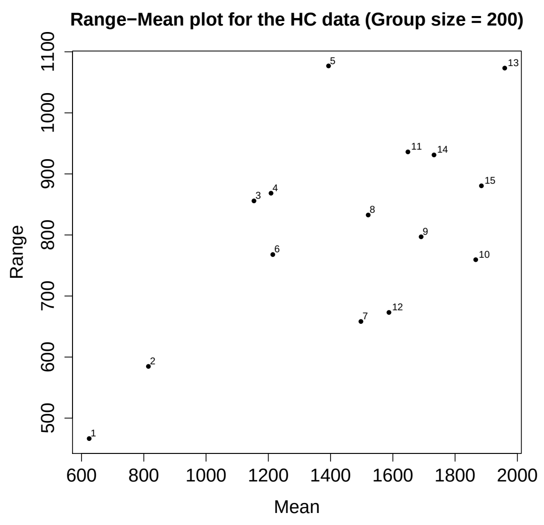


Figure 2: Range-Mean plot for the Heat Consumption series

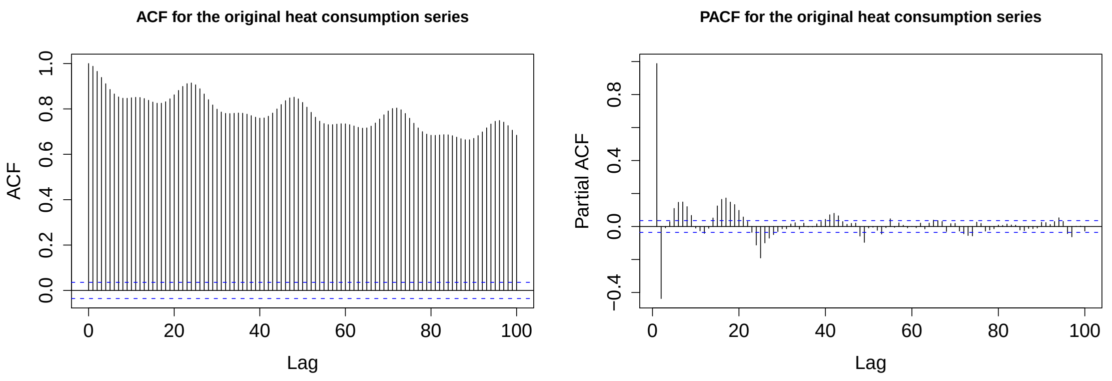


Figure 3: ACF and PACF for the Heat Consumption series

To further investigate the data the ACF and the PACF for the heat consumption series are plotted and shown in figure 3.

From the ACF it is seen that the series is non-stationary. Also a seasonallity with period 24 is seen. This isn't surprising since the series consists of hourly measurements. The PACF is numerically large for lag 1, and for higher lags it looks a bit like a damped sine function.

Removing the non-stationarity

The first problem to tackle is the non-stationarity. To remove the upward trend the original series Y_t is differenced once

$$W_t = \nabla Y_t$$

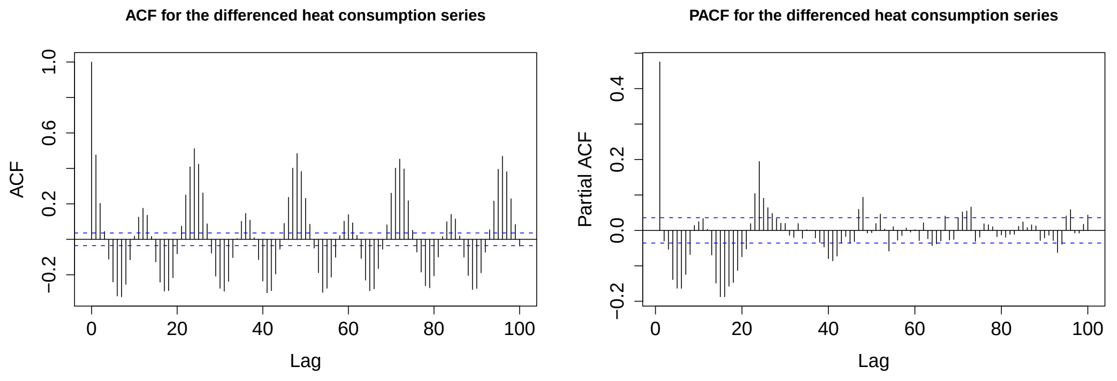


Figure 4: ACF and PACF for the differenced Heat Consumption series

This gives the new series W_t and the ACF and PACF for this series are shown in figure 4.

A lot of the correlation is gone for some of the lags, but there are still something to be removed. The ACF have some distinct patterns of spikes, that are separated by 24 lags. It therefore seems like a good idea to do a seasonal difference with period 24 on the W_t series. This gives

$$Z_t = \nabla_{24} W_t = \nabla \nabla_{24} Y_t$$

The ACF and PACF for this new series is calculated and shown in figure 5. A lot of the remaining correlation is gone. In the ACF the first 5 lags and the 24'th lag are different from zero. In the PACF the first 5 or 6 lags are different from zero, and then the lags that are a multiplum of 24 seems to slowly decay. Also two or three lags around the lags that are a multiplum of 24, are seen to be different from zero. These observations indicates that we probably need a non-seasonal ARMA(p, q) model and to get started we choose $p = q = 3$. For the seasonal correlations only the first lag (lag 24) is different from zero in the ACF and the seasonal lag slowly decays in the PACF. This indicates that a seasonal MA(1) should be appropriate. The first model we are trying to fit to the Y_t series is therefore a multiplicative $(3, 1, 3) \times (0, 1, 1)_{24}$ seasonal model.

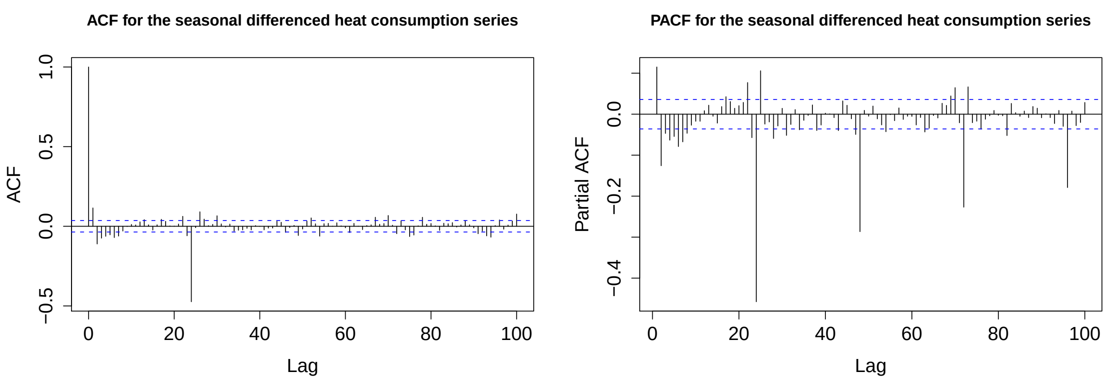
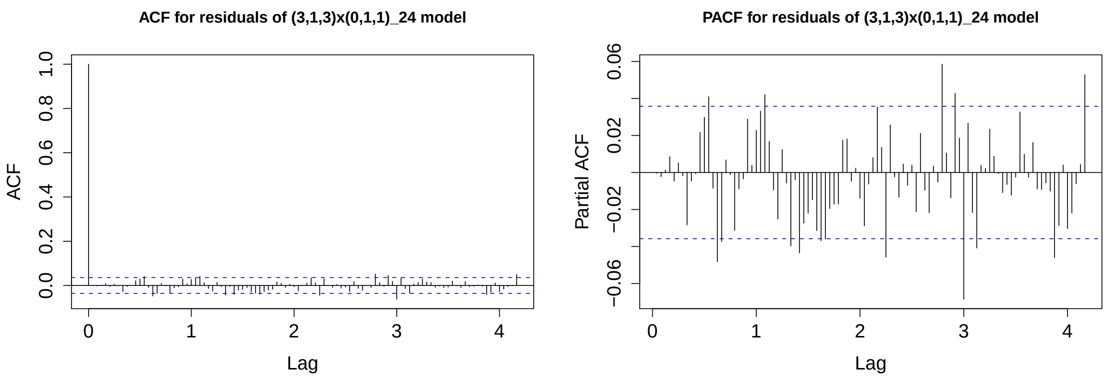


Figure 5: ACF and PACF for the combined differenced and seasonal diffirenced Heat Consumption series

Fitting the $(3,1,3) \times (0,1,1)_{24}$ model

A $(3,1,3) \times (0,1,1)_{24}$ model is fitted and the summary along with the ACF and PACF of the residuals are shown in figure 6.



```
Coefficients:
         ar1      ar2      ar3      ma1      ma2     ma3     sma1
      1.1925  -0.2091  -0.1392  -1.0620  -0.0348  0.1893  -0.9491
s.e.  0.2903   0.3321   0.1113   0.2891   0.2938  0.0782   0.0075

sigma^2 estimated as 1942:  log likelihood = -15517.02,  aic = 31050.03
```

Figure 6: The ACF and PACF for the residuals of the $(3,1,3) \times (0,1,1)_{24}$ model

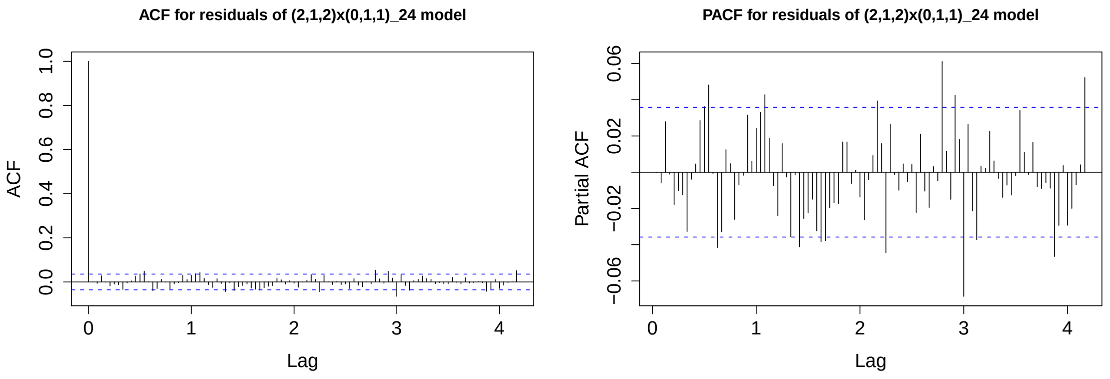
It is seen that the model fit isn't perfect. For a well fit model the residuals should be white noise and therefore the ACF and PACF should show no significant correlation for lags greater than zero. Especially lag 72 in both the ACF and PACF is significantly greater than zero. This could hint that we somehow need a seasonal AR part. But first we try to simplify the non-seasonal ARMA model. This will probably give good results since many of the coefficients in the $(3, 1, 3) \times (0, 1, 1)_{24}$ model is not significantly different from zero.

The $(3, 1, 2) \times (0, 1, 1)_{24}$ and $(2, 1, 3) \times (0, 1, 1)_{24}$ model

The model from the previous section is tried simplified by decreasing the nonseasonal AR- and MA-part separately. The results from fitting the $(3, 1, 2) \times (0, 1, 1)_{24}$ and $(2, 1, 3) \times (0, 1, 1)_{24}$ models are shown in appendix A.1. The ACF and PACF is almost identical to the $(3, 1, 3) \times (0, 1, 1)_{24}$model and the AIC score increases for both model. We therefore continue and tries the simpler $(2, 1, 2) \times (0, 1, 1)_{24}$ model.

The $(2, 1, 2) \times (0, 1, 1)_{24}$ model

Fitting the $(2, 1, 2) \times (0, 1, 1)_{24}$ model gives the summary and ACF/PACF shown in figure 7.



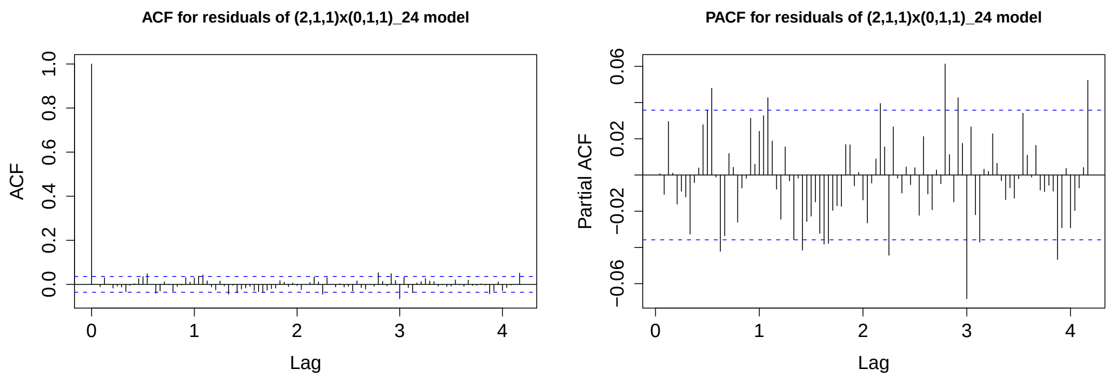
```
Coefficients:
         ar1      ar2      ma1      ma2     sma1
      1.0003  -0.1862  -0.8692  -0.0293  -0.9493
s.e.  0.1103   0.1059   0.1129   0.1120   0.0075

sigma^2 estimated as 1946:  log likelihood = -15519.81,  aic = 31051.62
```

Figure 7: The ACF and PACF for the residuals of the $(2, 1, 2) \times (0, 1, 1)_{24}$ model

The ACF and PACF still looks almost identical to the $(3, 1, 3) \times (0, 1, 1)_{24}$ model and the AIC score has only increase a little. Since this new model is simpler it seems more appropriate to use when we should try to make the seasonal part more complex. Looking at the R summary for this new model it is seen that the coefficient for the second order MA term is not significantly different from zero. Therefore we try to fit the even simpler $(2, 1, 1) \times (0, 1, 1)_{24}$ model.

The $(2, 1, 1) \times (0, 1, 1)_{24}$ model

The $(2, 1, 1) \times (0, 1, 1)_{24}$ model is fitted and gives the results shown in figure 8.



```
Coefficients:
         ar1      ar2      ma1     sma1
      1.0278  -0.2133  -0.8979  -0.9493
s.e.  0.0268   0.0184   0.0215   0.0075

sigma^2 estimated as 1946:  log likelihood = -15519.84,  aic = 31049.68
```

Figure 8: The ACF and PACF for the residuals of the $(2,1,1) \times (0,1,1)_{24}$ model

It is seen that this model gives almost identical ACF and PACF and the AIC-score has decreased. As the model is simpler than the previous models, this model[*] is used as starting point for trying other seasonal ARMA-models.

The $(2,1,1) \times (0,1,2)_{24}$ model

The problem with the previous models is the high correlation at lag 72. Maybe it can be removed by increasing the order of the seasonal MA-part. The results from fitting the $(2,1,1) \times (0,1,2)_{24}$ model is shown in figure 9.

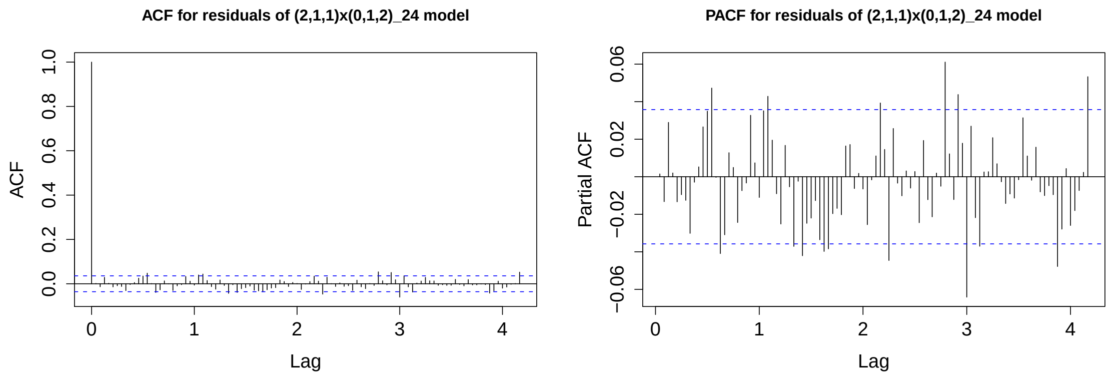
No progress is detected except for a slight decrease of the AIC score. The primary problem with high correlation at lag 72 remains. We try to increase the seasonal AR-part instead.

The $(2,1,1) \times (1,1,1)_{24}$ model

A model with an seasonal AR-part of order 1 is now fitted and the results are shown in figure 10.

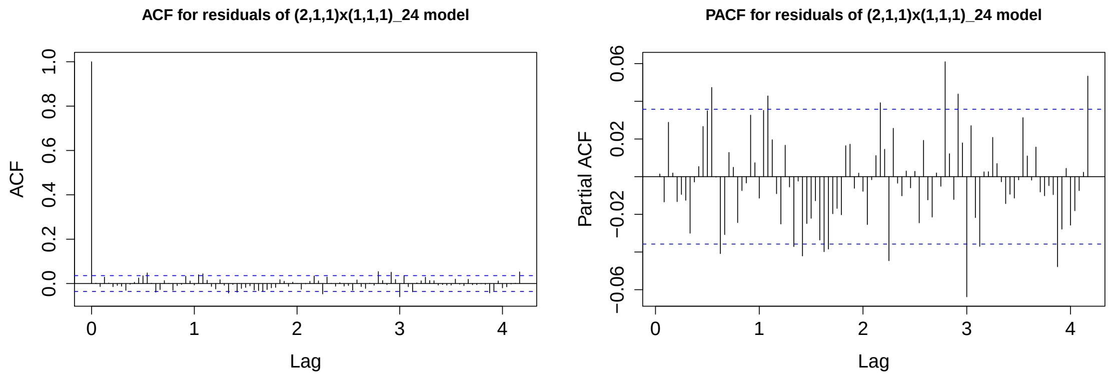
As usual no progress is detected, so another combination of the seasonal ARMA part should be tried. This is indeed what was done (see code in appendix A.4 for a complete list) but no model was found that was significantly better than the models already tried. Somehow there is some structure in the heat consumption series that is difficult to capture with a multiplicative seasonal model. Even though no model have

[*]The even simpler $(1,1,1) \times (0,1,1)_{24}$ model was also fitted (see appendix A.2) but correlation begin to show for the first 5 lags.



```
Coefficients:
         ar1      ar2      ma1     sma1     sma2
      1.0289  -0.2118  -0.9005  -0.9138  -0.0386
s.e.  0.0274   0.0185   0.0224   0.0183   0.0183

sigma^2 estimated as 1942:  log likelihood = -15517.64,  aic = 31047.28
```

Figure 9: The ACF and PACF for the residuals of the $(2, 1, 1) \times (0, 1, 2)_{24}$ model



```
Coefficients:
         ar1      ar2      ma1    sar1     sma1
      1.0294  -0.2118  -0.9008  0.0413  -0.9546
s.e.  0.0274   0.0185   0.0224  0.0195   0.0075

sigma^2 estimated as 1942:  log likelihood = -15517.61,  aic = 31047.22
```

Figure 10: The ACF and PACF for the residuals of the $(2, 1, 1) \times (1, 1, 1)_{24}$ model

been found that captures the structure perfectly, a model should be chosen to make predictions for one and six hours ahead. Apart from the ACF and the PACF we should also make sign tests, Portmanteau test and QQ-plots for the different models. This is done in the code in appendix A.4 and the results for the different models are found to be close to each other[†].

Choosing $(2,1,1) \times (0,1,2)_{24}$ as the final model

Since many models performs almost identically, the choice of a final model is somewhat difficult. The $(2,1,1) \times (0,1,2)_{24}$ model was chosen as the final model though, since it has one of the lowest AIC-score and all coefficients are significantly different from zero. To get a better picture of the model fit the output from the `tsdiag` function is shown in figure 11.

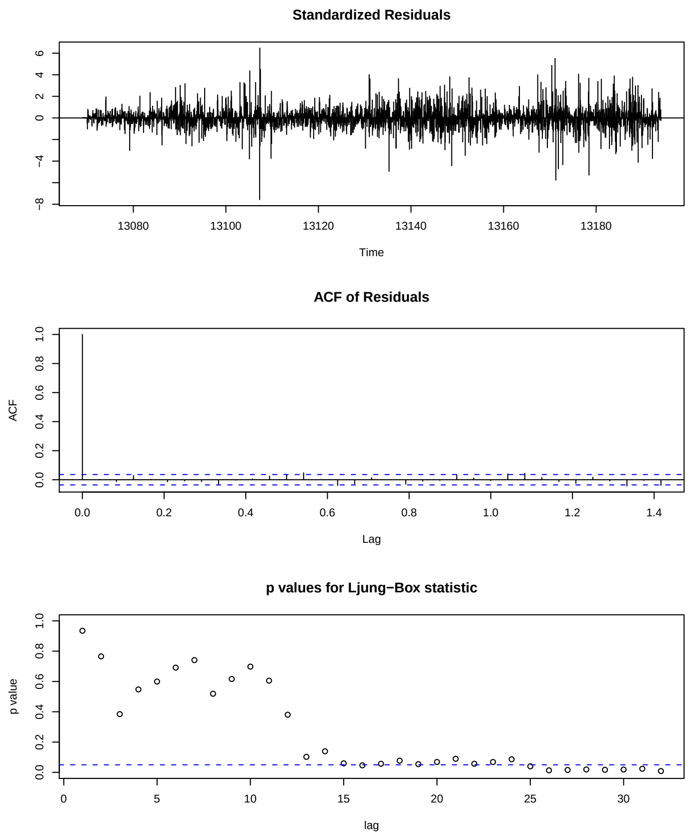


Figure 11: Output from the `tsdiag` function for the $(2,1,1) \times (0,1,2)_{24}$ model

From the `tsdiag` it is seen that the Portmanteau test (Ljung-Box is just an adjusted Portmanteau test) shows significantly deviations from a chi squared distribution when lags larger than 15 are included in the sum of squares of residual autocorrelation values. It reinforce that there is some seasonal structure that isn't captured by the chosen model. One possibility is that there is a second seasonallity with another period than 24. This won't be examined further in this report, but it could be worth investigating.

[†]Since almost 30 models was tested all results isn't included in the appendices. See instead `https://github.com/alphabits/dtu-fall-2011/tree/master/02417/assignment-4` for all plots, tables etc

The model also do not pass the sign test. The p-value is $3.8 \cdot 10^{-8}$.

Apart from the `tsdiag` plot and the sign test, a QQ-plot of the residuals is also created and shown in figure 12. It is seen that the residual distribution is more heavy tailed than a normal distribution. This is expected when a model don't fit the data well since the residuals will tend to be larger than "expected". This is worth remembering when we now turn to prediction and confidence intervals for the predictions.

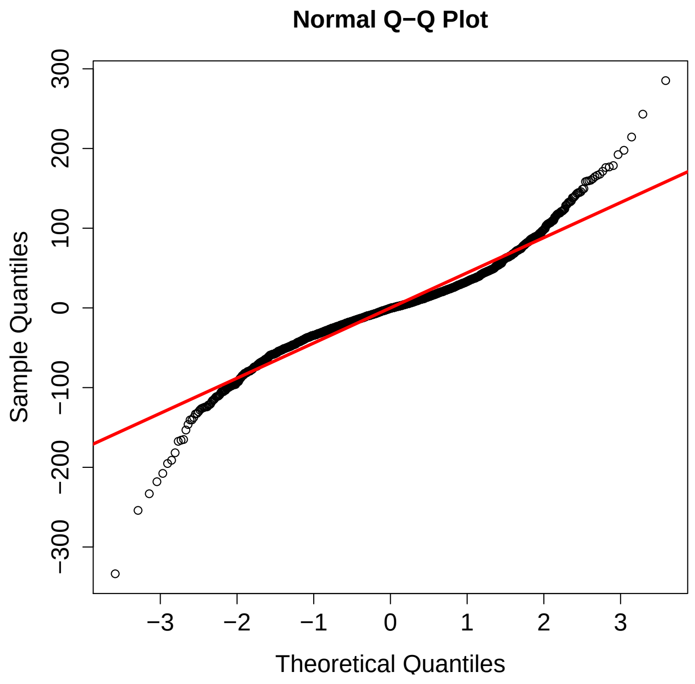


Figure 12: QQ-plot for the $(2,1,1) \times (0,1,2)_{24}$ model

Predicting 6 hours ahead with the $(2,1,1) \times (0,1,2)_{24}$ model

The heat consumption 1 and 6 hours ahead should now be predicted. The relevant theory is described in section 5.7.1 in [1] but the predictions, and the corresponding confidence intervals, are calculated using the `predict` function in R. The predictions 6 hours ahead are shown in figure 13 and the values are given in table 1.

Both from the plot and the table of the predicted values it is seen how the uncertainty in the predictions increases as the time gets futher into the future. Already for 6 hours into the future the uncertainty so large that the prediction may be almost useless. It is also worth noticing that the calculated 95% prediction interval do not include the uncertainty in the estimated model parameters. The actual confidence interval will therefore be larger than what is shown in the plot and table.

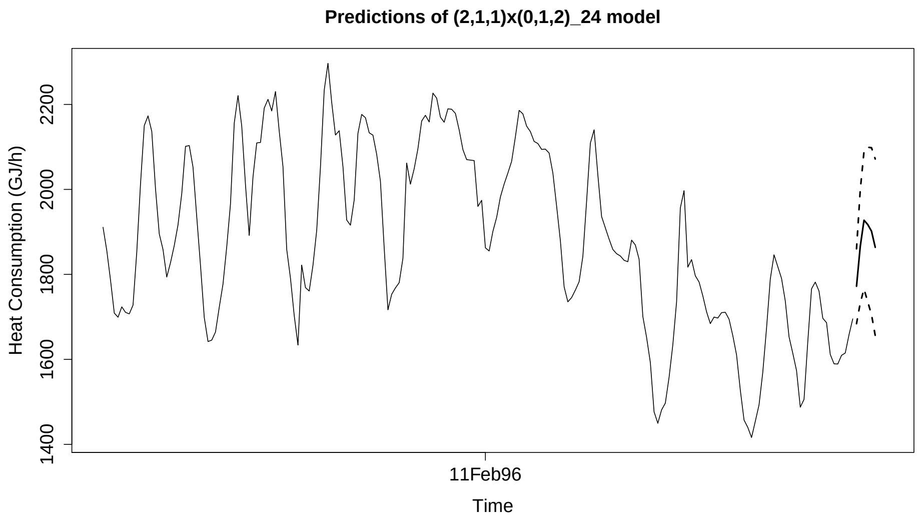


Figure 13: Predictions of $(2, 1, 1) \times (0, 1, 2)_{24}$ model

Time ahead	Prediction	Error	Interval
1	1772.05	86.38	[1685.67; 1858.43]
2	1865.74	130.24	[1735.50; 1995.99]
3	1927.45	158.65	[1768.79; 2086.10]
4	1917.34	178.21	[1739.13; 2095.55]
5	1901.66	192.56	[1709.10; 2094.21]
6	1863.67	203.70	[1659.97; 2067.37]

Table 1: Predictions of $(2, 1, 1) \times (0, 1, 2)_{24}$ model

Building the complex model

In the previous section we had trouble finding a suitable model describing the heat consumption data. In this section we will try to incorporate additional climate data in the model. By including this extra information we hope to build a better model. Specifically we have 3 extra time series with hourly measurements of Ambient Air Temperature, the Wind Speed and the Global Radiation for the same time period as the heat consumption measurements. To build a good model it should be determined which of the climate time series that are correlated with the heat consumption time series.

Estimating cross correlation functions

The estimated cross correlation function (CCF) between two time series is given by equation 6.17 in [1]. Unfortunately the estimated cross correlation function may show significant correlations even for two uncorrelated series. The time series therefore needs to be pre-whitened before calculating the CCF. The pre-whitening and estimation of the CCF is done by the `prewhiten` function from the package `TSA`. Using this function

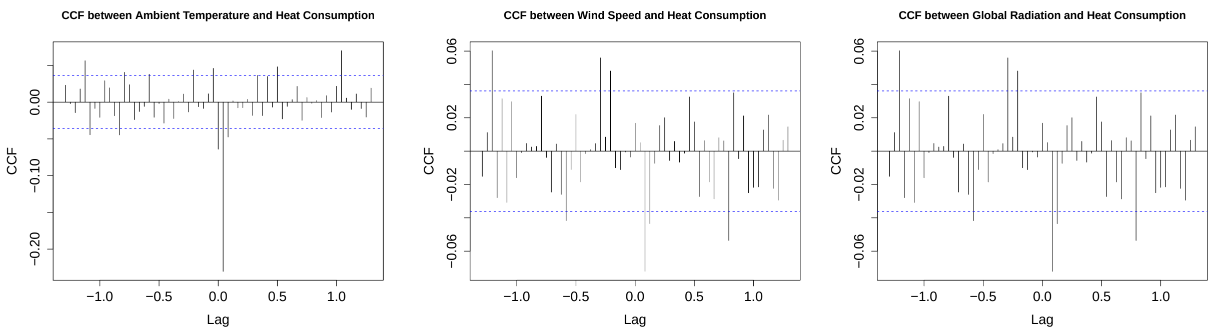


Figure 14: CCFs for the Heat Consumption data and the three different climate time series

the CCF is calculated for the combination of the heat consumption series and the three climate series. The result is shown in figure 14

From the figure it is seen that the CCF with the largest correlation values is the CCF between the ambient temperature and the heat consumption. From the plot it is seen that approximately $\rho_{\epsilon\alpha}(1) \approx -0.22$ where ϵ is the white noise from the temperature series and α is the "white noise" of the heat consumption series. The interpretation is that increasing temperature results in less heat consumption the next hour, which seems quite sensible.

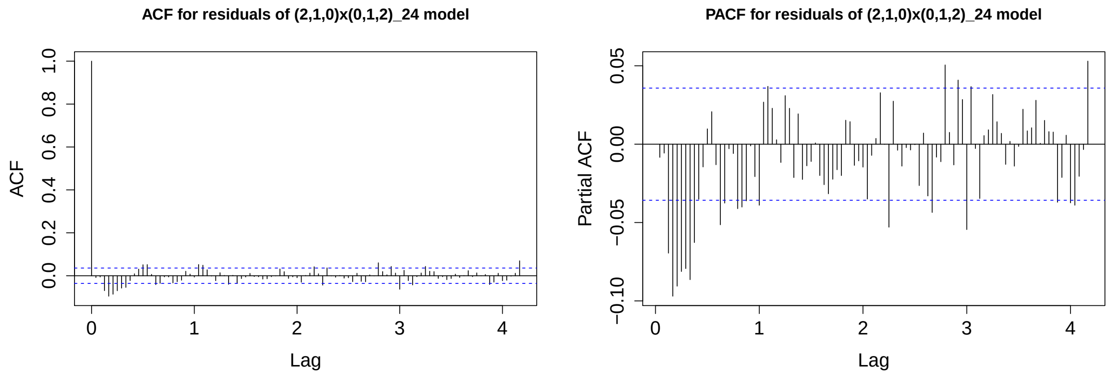
Using the Ambient Temperature as exogenous input

A model that includes the ambient temperature as an exogenous variable is now created. The model order is still given as $(2, 1, 1) \times (0, 1, 2)_{24}$. Using the `arima` function the model parameters are estimated and gives the model summary and residual ACF/PACF shown in figure 15

Although the AIC score has decreased a higher correlation is seen in the residuals for the first 6 lags. All parameters are seen to be significant and maybe a better model could be obtained by raising the non-seasonal ARIMA order. Therefore a $(2, 1, 3) \times (0, 1, 2)_{24}$ is fitted to the data, still with the ambient temperature as exogenous input. The results are shown in figure 16.

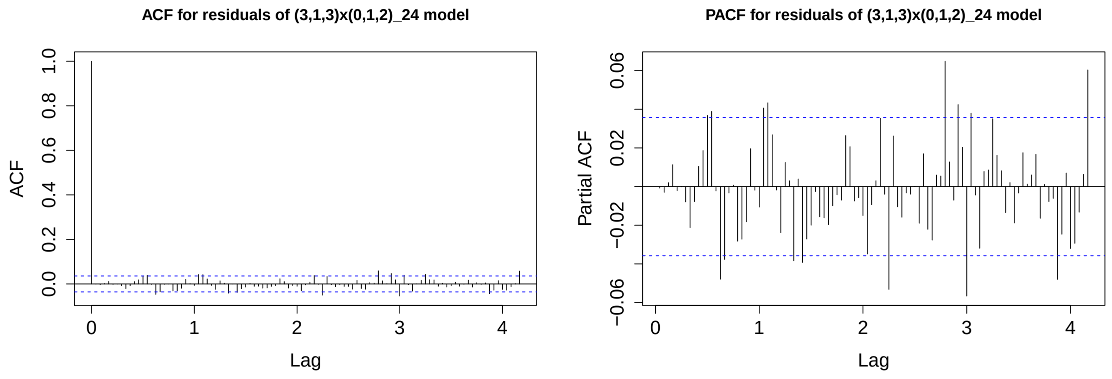
The AIC score has decreased further and the high correlation for lags below 6 has disappeared. The PACF still shows the same structure as for the model without the temperature series as input. Allthough the model should be further refined we accept it for now, and look at some of the other model checks. In figure 17

Generally it is the same structure as for the model without extra input. Although for lags between 15 and 25 the model with extra input has slightly higher p-values for the Portmanteau test. A sign test for the residuals gives a p-value of 3.3% which is much better than for the simple model. The QQ-plot for the residuals is shown in figure 18 and it looks almost completely identical with the QQ plot for the simple model.



```
Coefficients:
         ar1      ar2     sma1     sma2      xreg
      0.0940  -0.1241  -0.9125  -0.0413  -19.2452
s.e.  0.0187   0.0184   0.0185   0.0186    1.6365

sigma^2 estimated as 1909:  log likelihood = -15492.28,  aic = 30994.56
```

Figure 15: Summary for the $(2,1,1) \times (0,1,2)_{24}$ transfer model



```
Coefficients:
         ar1      ar2      ar3      ma1      ma2     ma3     sma1     sma2
      1.0886  -0.1235  -0.1588  -1.0603  -0.0789  0.2196  -0.9232  -0.0298
s.e.  0.2494   0.2807   0.1048   0.2478   0.2718  0.0848   0.0186   0.0187
          xreg
      -23.1620
s.e.    1.4765

sigma^2 estimated as 1817:  log likelihood = -15418.3,  aic = 30854.6
```

Figure 16: Summary for the $(2,1,3) \times (0,1,2)_{24}$ transfer model

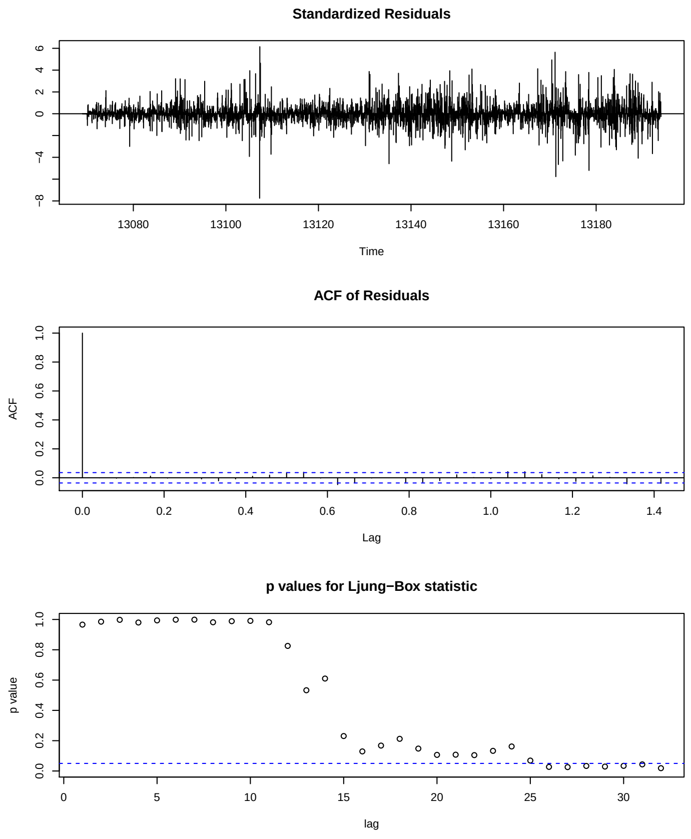


Figure 17: Output from the `tsdiag` function for the $(2,1,3) \times (0,1,2)_{24}$ transfer model

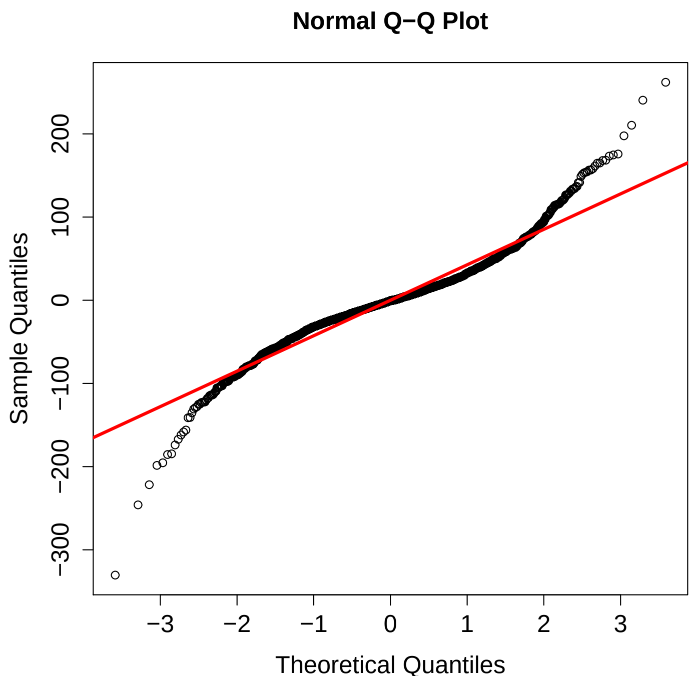


Figure 18: QQ plot for the $(2, 1, 3) \times (0, 1, 2)_{24}$ transfer model

Prediting with the $(2,1,3)\times(0,1,2)_{24}$ model with exogenous input

We are now ready to make predictions with the found model. The 6 step ahead predictions are shown in figure 19

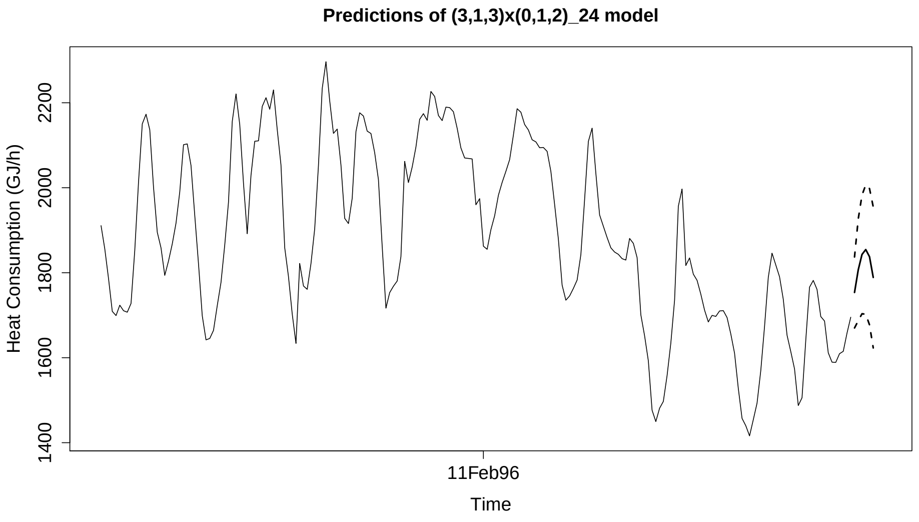
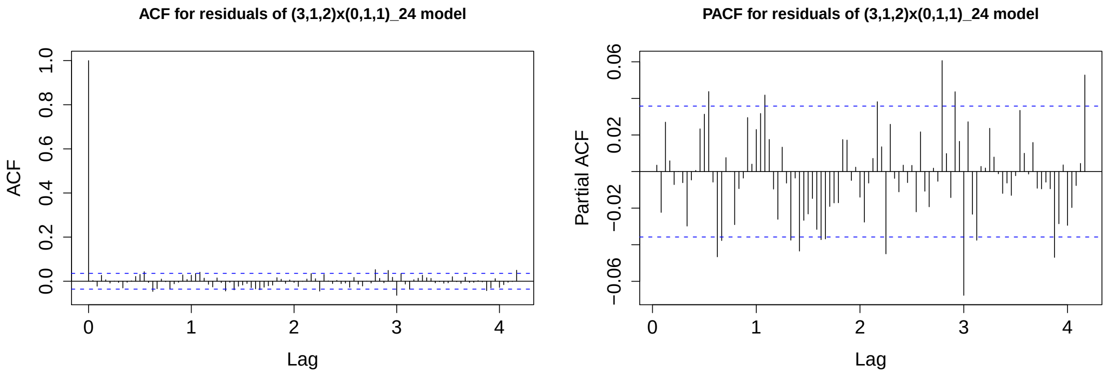


Figure 19: Predictions from the $(2,1,3)\times(0,1,2)_{24}$ transfer model

A Appendices

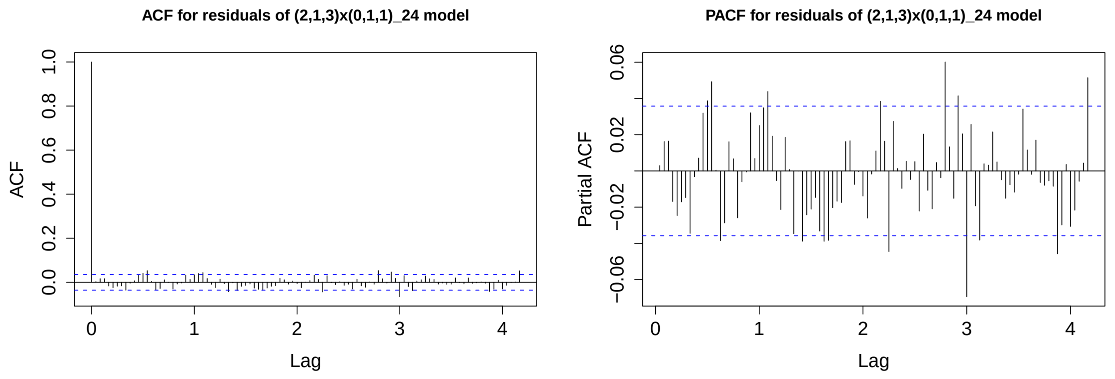
All R code used for this assignment is included here. All source code incl. latex code for this report can be found at `https://github.com/alphabits/dtu-fall-2011/tree/master/02417/assignment-4`

A.1 Results for the $(3,1,2)\times(0,1,1)_{24}$and $(2,1,3)\times(0,1,1)_{24}$model



```
Coefficients:
         ar1      ar2     ar3      ma1     ma2     sma1
      1.6659  -0.8669  0.1189  -1.5396  0.5877  -0.9493
s.e.  0.2648   0.2747  0.0612   0.2639  0.2371   0.0075

sigma^2 estimated as 1945:  log likelihood = -15519.09,  aic = 31052.18
```

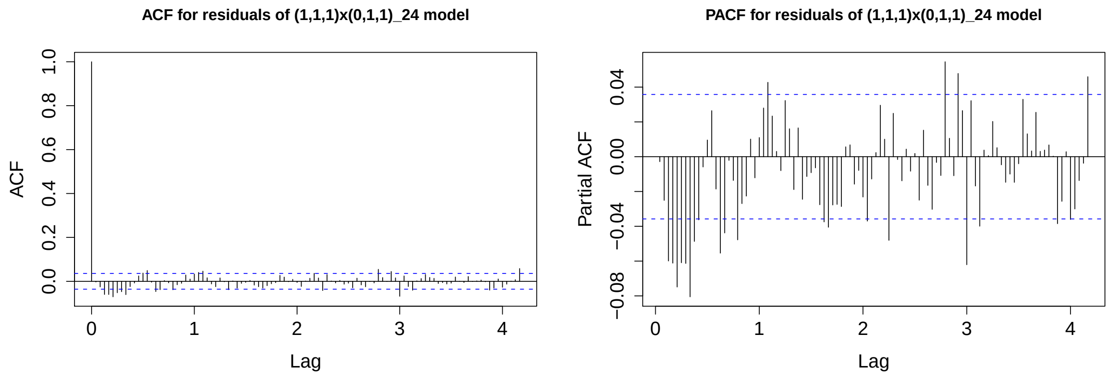
Figure 20: Summary for the $(3,1,2)\times(0,1,1)_{24}$ model



```
Coefficients:
         ar1     ar2      ma1      ma2      ma3     sma1
       0.286  0.4233  -0.1568  -0.5664  -0.1243  -0.9489
s.e.     NaN     NaN      NaN      NaN      NaN   0.0075

sigma^2 estimated as 1947:  log likelihood = -15521.16,  aic = 31056.33
```

Figure 21: Summary for the $(2, 1, 3) \times (0, 1, 1)_{24}$ model

A.2 Results for the $(1, 1, 1) \times (0, 1, 1)_{24}$ model



```
Coefficients:
           ar1     ma1     sma1
       -0.2106  0.3725  -0.9484
s.e.    0.0905  0.0854   0.0076

sigma^2 estimated as 2002:  log likelihood = -15562.02,  aic = 31132.05
```

Figure 22: Summary for the $(1, 1, 1) \times (0, 1, 1)_{24}$ model

A.3 Loading data

```
dat = read.table('../data/veks.csv', sep=',', header=TRUE)
dat[,"day"] = 12965 + dat[,"rdays"]


hc = dat[["HC.f"]]
ta = dat[["Ta.f"]]
w = dat[["W.f"]]
gr = dat[["GR.f"]]

#st = 1995.5 + dat[2500,"rdays"]/365
ind = 2500:5500
pred.ind = 2500:5508
pred.only.ind = 5501:5506
st = dat[ind[1],"jdate"]

day = dat[["day"]][ind]
day.pred = dat[["day"]][pred.only.ind]
day.inc.pred = dat[["day"]][pred.ind]
hc.ts = ts(hc[ind], start=st, frequency=24)
ta.ts = ts(ta[ind], start=st, frequency=24)
w.ts = ts(w[ind], start=st, frequency=24)
gr.ts = ts(w[ind], start=st, frequency=24)
```

A.4 R code for task 1 and task 2

```
source('loaddata.R')
source('functions.R')


SAVEPLOTS = TRUE


plot.and.save('hc-ts.pdf', 12, 7, my.ts.plot, day, hc.ts,
              main="Hourly measures of Heat Consumption",
              ylab="Heat Consumption (GJ/h)", xlab="Time")


org.acf.main.tmpl = "%s for the original heat consumption series"
plot.and.save('acf-hc.pdf', 7, 5, my.acf, as.vector(hc.ts),
              main=sprintf(org.acf.main.tmpl, "ACF"))
plot.and.save('pacf-hc.pdf', 7, 5, my.pacf, as.vector(hc.ts),
              main=sprintf(org.acf.main.tmpl, "PACF"))

plot.and.save('range-mean-plot.pdf', 7, 7, range.mean.plot, hc.ts, 200,
              main="Range-Mean plot for the HC data (Group size = 200)")

hc.d = diff(hc.ts, 1)

hc.diff.acf.main.tmpl = "%s for the differenced heat consumption series"
plot.and.save('acf-hc-diff.pdf', 7, 5, my.acf, as.vector(hc.d),
              main=sprintf(hc.diff.acf.main.tmpl, "ACF"))
plot.and.save('pacf-hc-diff.pdf', 7, 5, my.pacf, as.vector(hc.d),
              main=sprintf(hc.diff.acf.main.tmpl, "PACF"))


hc.d.season = diff(hc.d, 24)
hc.diff.season.acf.main.tmpl = "%s for the seasonal differenced heat consumption series"
plot.and.save('acf-hc-diff-season.pdf', 7, 5, my.acf, as.vector(hc.d.season),
              main=sprintf(hc.diff.season.acf.main.tmpl, "ACF"))
plot.and.save('pacf-hc-diff-season.pdf', 7, 5, my.pacf, as.vector(hc.d.season),
              main=sprintf(hc.diff.season.acf.main.tmpl, "PACF"))


# Testing lots of different models

models = list(
    list(name='m1', nonseas=c(0,1,2), seas=c(0,1,2)),
    list(name='m2', nonseas=c(0,1,4), seas=c(0,1,1)),
    list(name='m3', nonseas=c(0,1,4), seas=c(0,1,2)),
    list(name='m4', nonseas=c(4,1,4), seas=c(0,1,1)),
    list(name='m5', nonseas=c(4,1,0), seas=c(0,1,1)),
    list(name='m6', nonseas=c(1,1,0), seas=c(2,0,0)),
    list(name='m7', nonseas=c(3,1,3), seas=c(0,1,1)),
    list(name='m8', nonseas=c(2,1,2), seas=c(0,1,1)),
    list(name='m9', nonseas=c(1,1,1), seas=c(0,1,1)),
    list(name='m10', nonseas=c(3,1,2), seas=c(0,1,1)),
    list(name='m11', nonseas=c(2,1,3), seas=c(0,1,1)),
    list(name='m12', nonseas=c(3,1,0), seas=c(1,0,0)),
    list(name='m13', nonseas=c(1,1,2), seas=c(0,1,1)),
    list(name='m14', nonseas=c(2,1,3), seas=c(0,1,2)),
    list(name='m15', nonseas=c(2,1,3), seas=c(1,1,1)),
    list(name='m16', nonseas=c(2,1,2), seas=c(0,1,1), transform=log),
    list(name='m17', nonseas=c(2,1,2), seas=c(0,1,2)),
    list(name='m18', nonseas=c(2,1,1), seas=c(0,1,2)),
    list(name='m19', nonseas=c(2,1,1), seas=c(1,1,2)),
    list(name='m20', nonseas=c(2,1,1), seas=c(0,1,2), transform=log),
    list(name='m21', nonseas=c(2,1,1), seas=c(0,1,1)),
    list(name='m22', nonseas=c(2,1,1), seas=c(0,1,2)),
    list(name='m23', nonseas=c(2,1,1), seas=c(0,1,3)),
    list(name='m24', nonseas=c(1,1,2), seas=c(2,1,1)),
    list(name='m25', nonseas=c(2,1,1), seas=c(0,2,1)),
    list(name='m26', nonseas=c(2,0,1), seas=c(0,1,2)),
    list(name='m27', nonseas=c(2,1,1), seas=c(1,0,2)),
    list(name='m28', nonseas=c(2,1,1), seas=c(1,1,1)),
    list(name='m29', nonseas=c(2,1,1), seas=c(1,0,4))
)

# Iterate models and assign in current scope
for (m in models) {
    model.name = m[["name"]]
    model.data = if ("transform" %in% names(m)) m[["transform"]](hc.ts) else hc.ts
    assign(model.name, get.model(model.name, arima, model.data, order=m[["nonseas"]],
                           seasonal=list(order=m[["seas"]], period=24)))
}

plot.and.save('m22-predictions.pdf', 12, 7,
              plot.forecast, m22,
              main=sprintf("Predictions of %s model", get.model.string(models[[22]])))
save.prediction.table('m22-predictions.tex', m22)

# Save all info about 'interesting' models for the report

for (model.setup in models) {
    model.name = model.setup[["name"]]
    model = get(model.name)
    description = get.model.string(model.setup)
    # QQ plot
    plot.and.save(sprintf('%s-qq.pdf', model.name), 7, 7, qq, model)
    # Model summary
    save.model.summary(model, sprintf('%s-summary.txt', model.name))
    # Sign test
    save.sign.test(residuals(model), sprintf('%s-sign-test.txt', model.name))
    # tsdiag
    plot.and.save(sprintf('%s-tsdiag.pdf', model.name), 7, 8.5, tsdiag, model, gof.lag=32)
    # ACF and PACF of residuals
    plot.and.save(sprintf('%s-residual-acf.pdf', model.name), 7, 5, my.acf, residuals(model),
                  main=sprintf("ACF for residuals of %s model", description))
    plot.and.save(sprintf('%s-residual-pacf.pdf', model.name), 7, 5, my.pacf, residuals(model),
                  main=sprintf("PACF for residuals of %s model", description))
}
```

A.5 R code for task 3 and task 4

```
source('loaddata.R')
source('functions.R')
library('TSA')


SAVEPLOTS = TRUE


ta.d = diff(diff(ta.ts, 1), 24)
w.d = diff(diff(w.ts, 1), 24)
gr.d = diff(diff(gr.ts, 1), 24)
hc.d = diff(diff(hc.ts, 1), 24)

# Modelling the ambient temperature

plot.and.save('ta-ccf.pdf', 7, 6,
              prewhiten, ta.d, hc.d,
              main="CCF between Ambient Temperature and Heat Consumption")
plot.and.save('w-ccf.pdf', 7, 6,
              prewhiten, w.d, hc.d,
              main="CCF between Wind Speed and Heat Consumption")
plot.and.save('gr-ccf.pdf', 7, 6,
              prewhiten, gr.d, hc.d,
              main="CCF between Global Radiation and Heat Consumption")

m22 = get.model('m22', function(){})


models = list(
    list(name='tfm1', nonseas=c(2,1,0), seas=c(0,1,2), exo=ta.ts),
    list(name='tfm2', nonseas=c(3,1,3), seas=c(0,1,2), exo=ta.ts),
    list(name='tfm3', nonseas=c(2,1,3), seas=c(0,1,2), exo=ta.ts),
    list(name='tfm4', nonseas=c(3,1,3), seas=c(0,1,2), exo=ta.ts[2:3001])
)

# Iterate models and assign in current scope
for (m in models) {
    model.name = m[["name"]]
    model.data = if ("transform" %in% names(m)) m[["transform"]](hc.ts) else hc.ts
    if (model.name == 'tfm4') {
        model.data = model.data[1:3000]
    }
    assign(model.name, get.model(model.name, arima, model.data, order=m[["nonseas"]],
                           seasonal=list(order=m[["seas"]], period=24), xreg=m[["exo"]]))
}


for (model.setup in models) {
    model.name = model.setup[["name"]]
    model = get(model.name)
    description = get.model.string(model.setup)

    # QQ plot
    plot.and.save(sprintf('%s-qq.pdf', model.name), 7, 7, qq, model)
    # Model summary
    save.model.summary(model, sprintf('%s-summary.txt', model.name))
    # Sign test
    save.sign.test(residuals(model), sprintf('%s-sign-test.txt', model.name))
    # tsdiag
    plot.and.save(sprintf('%s-tsdiag.pdf', model.name), 7, 8.5, tsdiag, model, gof.lag=32)
    # ACF and PACF of residuals
    plot.and.save(sprintf('%s-residual-acf.pdf', model.name), 7, 5, my.acf, residuals(model),
                  main=sprintf("ACF for residuals of %s transfer model", description))
    plot.and.save(sprintf('%s-residual-pacf.pdf', model.name), 7, 5, my.pacf, residuals(model),
                  main=sprintf("PACF for residuals of %s transfer model", description))
}
```

A.6 Helper functions

```
# Loads the textxy function
library('calibrate')
library('date')


my.acf = function(x, ...) {acf(x, lag.max=100, ...)}
my.pacf = function(x, ...) {pacf(x, lag.max=100, ...)}
my.acfs = function(x, ...) {
    par(mfrow=c(1,2))
    my.acf(x, ...)
    my.pacf(x, ...)
}

plot.and.save = function(filename, width, height, plotfunction, ...) {
    save.the.plot = exists('SAVEPLOTS') && SAVEPLOTS
    if (save.the.plot) {pdf(sprintf('../plots/%s', filename), width, height)}
    plotfunction(cex.axis=1.4, cex.lab=1.4, cex.main=1.4, ...)
    if (save.the.plot) {dev.off()}
}

get.model = function(model.id, model.fun, ...) {
    filename = sprintf('../data/%s.rda', model.id)

    if (!file.exists(filename)) {
        model = model.fun(...)
        save(model, file=filename)
    }
    load(filename)

    return(model)
}
```

```
get.string.from.struct = function(struct) {
    return(sprintf('(%d,%d,%d)', struct[1], struct[2], struct[3]))
}

get.model.string = function(model.setup) {
    return(sprintf('%sx%s_24', get.string.from.struct(model.setup[["nonseas"]]),
                   get.string.from.struct(model.setup[["seas"]])))
}

range.mean.calculate = function(x, bin.size) {
    n = length(x)
    means = c()
    ranges = c()
    cur.group = c()

    for (i in 1:n) {
        cur.group = c(cur.group, x[i])
        if (i %% bin.size == 0) {
            means = c(means, mean(cur.group))
            ranges = c(ranges, max(cur.group)-min(cur.group))
            cur.group = c()
        }
    }

    return(list(means=means, ranges=ranges))
}

range.mean.plot = function(x, bin.size, ...) {
    res = range.mean.calculate(x, bin.size)
    means = res$means
    ranges = res$ranges

    plot(means, ranges, pch=20, xlab='Mean', ylab='Range', ...)
    textxy(means, ranges, 1:length(means), cx=0.75)
}

my.ts.plot.base = function(x, y, days.ticks, ...) {
    days = days.ticks
    plot(x, y, xaxt='n', cex.axis=1.4, cex.lab=1.4, cex.main=1.4, type="l", ...)
    axis(1, days, as.character(as.date(days)), cex.axis=1.4, cex.lab=1.4)
}

my.ts.plot = function(x, y, ...) {
    days = seq(13080, 13180, 20)
    return(my.ts.plot.base(x, y, days, ...))
}

plot.forecast = function(model, ...) {
    predictions = predict(model, n.ahead=length(pred.only.ind))
    pred = predictions$pred
    se = predictions$se
    n = length(day)
    ind = (n-200):n
    first.day = day[ind[1]]
    last.day = day.inc.pred[length(day.inc.pred)]
    my.ts.plot.base(day[ind], as.vector(hc.ts)[ind], c(13185, 13190), xlim=c(first.day, last.day),
                    xlab="Time", ylab="Heat Consumption (GJ/h)", ...)
    lines(day.pred, pred, lwd=2)
    lines(day.pred, pred + 1.96*se, lty=2, lwd=2)
    lines(day.pred, pred - 1.96*se, lty=2, lwd=2)
}
```

```
save.prediction.table = function(filename, model, steps.ahead=6) {
    predictions = predict(model, n.ahead=length(pred.only.ind))
    l = c()
    pred = predictions$pred
    err = 1.96*predictions$se
    l = c(l, 'Time ahead & Prediction & Error & Interval \\\\\\hline')
    for (i in 1:steps.ahead) {
        app = if (i!=steps.ahead) "\\\\" else ""
        l = c(l, sprintf('%d & %.02f & %.02f & [%.02f; %.02f]%s', i, pred[i],
                      err[i], pred[i]-err[i], pred[i]+err[i], app))
    }
    f = file(sprintf('../tables/%s', filename))
    writeLines(l, f)
    close(f)
}

sign.test = function(res) {
    n = length(res)
    binom.test(sum(1*(res[2:n]*res[1:(n-1)] < 0)), n-1)
}

save.sign.test = function(res, filename) {
    sink(sprintf('../tables/%s', filename))
    print(sign.test(res))
    sink()
}

qq = function(model, ...) {
    plot.qq.res(residuals(model), model$sigma2, ...)
}

plot.qq.res = function(res, sigma, ...) {
    qqnorm(res, ...)
    lines((-4):4,((-4):4)*sqrt(sigma), type="l", lwd=3, col='red')
}

save.model.summary = function(model, filename) {
    sink(sprintf('../tables/%s', filename))
    print(model)
    sink()
}
```

References

[1] Henrik Madsen, *Time Series Analysis*. Chapman & Hall/CRC, 1st Edition, 2008.