Time Series HW 3

Kenny Flagg and Paul Harmon

September 16, 2016

HW 3

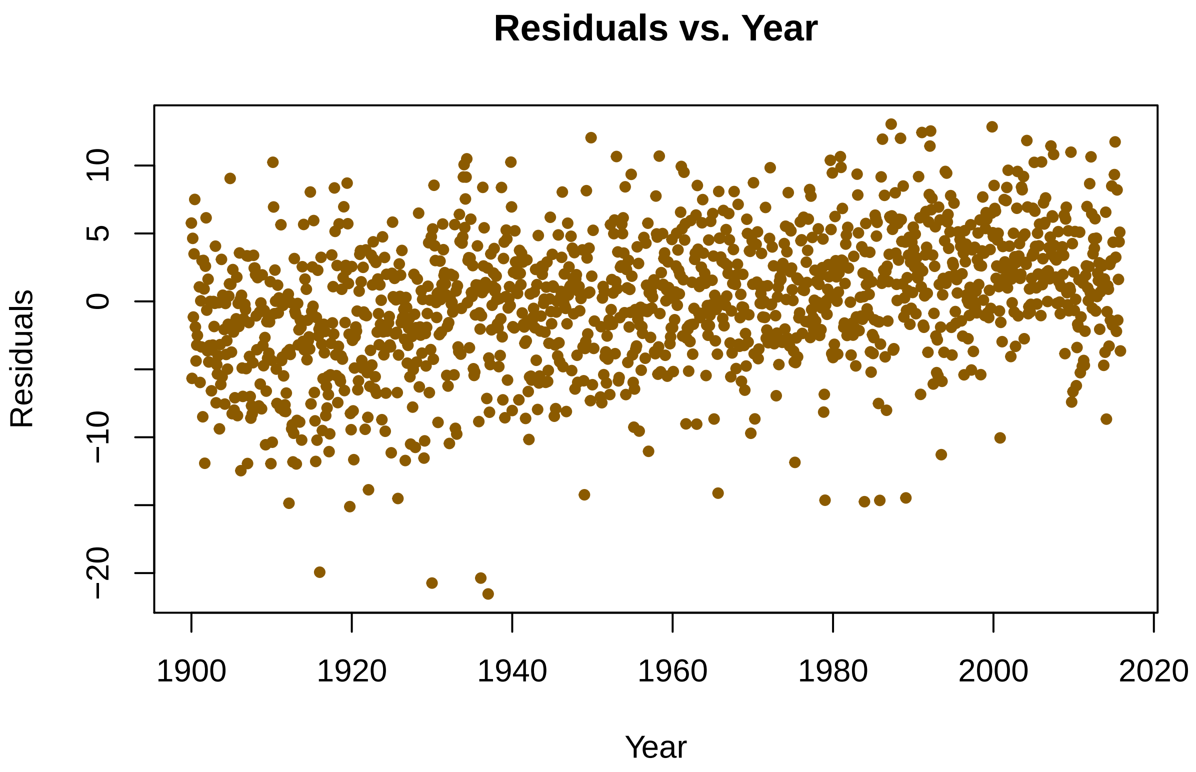
You can alone or in pairs that may or may not be people you worked with before. You can discuss it with old partners but should try work as much as possible with new collaborators. 5% bonus if you find someone completely new to work with - that you did not work with on first two assignments.

I mentioned de-seasonalizing of time series, where the seasonal variation is removed from the series to highlight variation at either higher or lower frequencies. There are a variety of techniques for doing this but the simplest is to just subtract the mean for each month from the observations. And the easiest way to find that is using `lm(y~month,data=...)`.

1) *For the Bozeman temperature data from HW 1 and 2, estimate a model with month only, subtract its fitted values from the responses (or just extract residuals). Plot the residuals vs the fractional `Year` variable and compare the plot of this result to the plot of the original time series.*

```
weather = read.csv("rawbozemandata.csv", header = TRUE)

weather$YEAR = substr(as.character(weather$DATE), 0,nchar(weather$DATE)-2)
weather$MONTH = as.factor(substr(as.character(weather$DATE),5,nchar(weather$DATE)))
weather$year = (floor(weather$DATE/100))
weather$Month<-round((weather$DATE/100-weather$year)*100,1)
weather$Yearfrac<-weather$year+(weather$Month-1)/12 #makes it an integer variable
#fit the linear model
lm.weather = lm(MMXT ~ MONTH, data = weather)
#lm.weather$residuals
plot(weather$Yearfrac, lm.weather$residuals, pch = 20, col = "orange4",
 main = "Residuals vs. Year", xlab = "Year", ylab = "Residuals")
```



2) *In the de-seasonalized Bozeman temperature data set, re-assess evidence for the linear trend. Compare the result (test statistic, degrees of freedom and size of p-value) of just fitting a linear time trend in these de-seasonalized responses to the result from our original model with a linear year component and a month adjustment (not the quadratic trend model).*

```r
lm.seasoned <- lm(MMXT ~ Yearfrac + MONTH, data = weather)
lm.deseasoned <- lm(lm.weather$residuals ~ Yearfrac, data = weather)
summary(lm.seasoned)
```

```
##
## Call:
## lm(formula = MMXT ~ Yearfrac + MONTH, data = weather)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20.5022  -2.9005   0.1112   3.0412  12.6950
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -69.290162   7.266158  -9.536  < 2e-16 ***
## Yearfrac      0.051674   0.003706  13.942  < 2e-16 ***
## MONTH02       3.701446   0.604984   6.118 1.23e-09 ***
## MONTH03      11.189381   0.604985  18.495  < 2e-16 ***
## MONTH04      21.977317   0.604986  36.327  < 2e-16 ***
## MONTH05      31.224003   0.606297  51.500  < 2e-16 ***
## MONTH06      39.708393   0.606299  65.493  < 2e-16 ***
## MONTH07      49.564963   0.608994  81.388  < 2e-16 ***
## MONTH08      48.465835   0.607645  79.760  < 2e-16 ***
## MONTH09      37.629365   0.608976  61.791  < 2e-16 ***
## MONTH10      25.822126   0.607627  42.497  < 2e-16 ***
## MONTH11      10.315109   0.607657  16.975  < 2e-16 ***
## MONTH12       1.732846   0.608989   2.845   0.0045 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.597 on 1361 degrees of freedom
## Multiple R-squared:  0.9349, Adjusted R-squared:  0.9343
## F-statistic:  1628 on 12 and 1361 DF,  p-value: < 2.2e-16
```

```r
summary(lm.deseasoned)
```

```
##
## Call:
## lm(formula = lm.weather$residuals ~ Yearfrac, data = weather)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20.4490  -2.9043   0.0912   3.0355  12.7135
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.012e+02  7.229e+00  -13.99   <2e-16 ***
## Yearfrac     5.166e-02  3.691e-03   14.00   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.579 on 1372 degrees of freedom
## Multiple R-squared:  0.1249, Adjusted R-squared:  0.1243
## F-statistic: 195.9 on 1 and 1372 DF,  p-value: < 2.2e-16
```

After de-seasonalization, there is strong evidence of a linear trend in MMXT through time ($t_{1372} = 14.00$, p-value < 0.0001). This is very similar to the result when when month is included as a factor, where the estimated year term has $t_{1361} = 13.94$ with p-value < 0.0001. It seems like the degrees of freedom for the de-seasonalized model is wrong because it doesn't account for estimating the monthly means.

3) *I briefly discussed the HADCRUT data set in class. We will consider the HADCRUT4 series of temperature anomalies for the Nothern Hemisphere. The fully up to date data set is available at:* *`http://www.metoffice.gov.uk/hadobs/hadcrut4/data/current/time_series/HadCRUT.4.4.0.0.monthly_nh.tx`*

   Download the ensemble median monthly northern hemisphere temperature data. We will use the entire time series that is currently available (January 1850 to July 2016). You might want to read *`http://www.metoffice.gov.uk/hadobs/hadcrut4/data/current/series_format.html`* *for more information on the columns in the data set.*

   Because the time series is complete over the time frame under consideration, you can use `ts()` to create a time variable instead of messing around with their `Year/Month` variable.

   Make a plot versus time of the ensemble medians and use that as your response variable in the following questions. Discuss trend, seasonality, outliers, and variability.

4) *Our main focus with these data will be on estimating the long-term trend, starting with polynomial trend models. But first, check for seasonality in a model that accounts for a linear time trend. Use our previous fractional year for the trend. Report an `effects` plot and a test for the month model component.*

   Note: when you use `time()` to generate the `Year` variable from a time series object it retains some time series object information that can cause conflicts later on. Create a new variable in your data.frame that uses something like `as.vector(time(tsdataname))`.

5) *Check the residuals versus fitted values for any evidence of nonlinearity in the residuals vs fitted that was missed by the model with a linear trend and month component. Also note any potential issues with the constant variance assumption.*

6) *You can add higher order polynomial terms to models using `x1+I(x1^2)+I(x1^3)...` or using the `poly` function, such as `poly(x1,3,raw=T)` for a cubic polynomial that includes the linear and quadratic components (we want this!). The `raw=T` keeps the variables in their raw or input format. Estimate the same model but now using polynomial trends that steps up from linear (poly(time,1,raw=T)) and stop when you get a failure to estimate a part of the model. Briefly discuss what happened.*

7) *If we center or, even better, make the polynomial functions orthogonal to one another, we can avoid the issue in the previous question. Use `poly(x1,?,raw=F)` and step up the polynomial order for time until the p-value for the last coefficient (use `summary()`) is "large", reporting the single test result for each step in the building process. Then drop back one order and re-fit the model. Report the `effects` plot of the resulting model and describe the estimated trend. Note: aside from access to orthogonal polynomials the `poly` function interfaces with `Anova` and the `effects` package really nicely.*

8) *Check the diagnostic plots from your final model. Does anything improve from the first version. Also plot the residuals vs time and compare that plot to residuals vs fitted.*

9) *Run the following code so I can see what version of R you are now using:*

Documenting R version

```r
sessionInfo()$R.version$nickname
```

```
## [1] "Bug in Your Hair"
```

```r
getRversion()
```

```
## [1] '3.3.1'
```