

STAT 500: HW6

Jasmine Mou

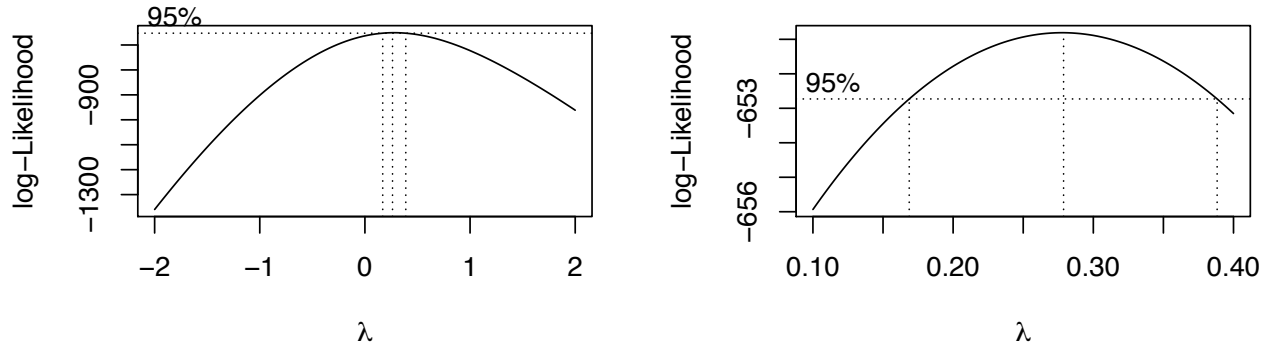
11/7/2017

1. Using the ozone data, fit a model with O3 as the response and temp, humidity and ibh as predictors. Use the Box-Cox method to determine the best transformation on the response.

```
require(MASS)
data(ozone, package="faraway")
lm_ozone <- lm(O3 ~ temp + humidity + ibh, data=ozone)
```

Check whether the response needs transformation and plot the results. The first plot is too broad so we narrow down the range of λ in the second plot.

```
par(mfrow=c(1,2))
boxcox(lm_ozone, plotit=T)
boxcox(lm_ozone, plotit=T, lambda=seq(0.1,0.4,by=0.05))
```



The λ that optimizes the log-Likelihood is picked at some point between 0.25 and 0.30, with 95% confidence interval being $\sim (0.15, 0.4)$, which is far from and excludes 1. Thus we need the transformation. Let's pick $\hat{\lambda}$ being 0.28.

```
lambda_hat <- 0.28
lm_ozone_transformed_response <- lm(O3^lambda_hat ~ temp + humidity + ibh, data=ozone)
summary(lm_ozone)
```

```
##
## Call:
## lm(formula = O3 ~ temp + humidity + ibh, data = ozone)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5291  -3.0137  -0.2249   2.8239  13.9303
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.049e+01  1.616e+00  -6.492 3.16e-10 ***
## temp         3.296e-01  2.109e-02  15.626 < 2e-16 ***
## humidity     7.738e-02  1.339e-02   5.777 1.77e-08 ***
## ibh          -1.004e-03  1.639e-04  -6.130 2.54e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 4.524 on 326 degrees of freedom
## Multiple R-squared:  0.684, Adjusted R-squared:  0.6811
## F-statistic: 235.2 on 3 and 326 DF,  p-value: < 2.2e-16
```

```
summary(lm_ozone_transformed_response)
```

```
##
## Call:
## lm(formula = O3^lambda_hat ~ temp + humidity + ibh, data = ozone)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.63939 -0.12797  0.01051  0.14760  0.58732
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.8693842  0.0739886  11.750  < 2e-16 ***
## temp         0.0156913  0.0009655  16.252  < 2e-16 ***
## humidity     0.0035916  0.0006130   5.859 1.14e-08 ***
## ibh          -0.0000578  0.0000075  -7.706 1.58e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2071 on 326 degrees of freedom
## Multiple R-squared:  0.7162, Adjusted R-squared:  0.7136
## F-statistic: 274.3 on 3 and 326 DF,  p-value: < 2.2e-16
```

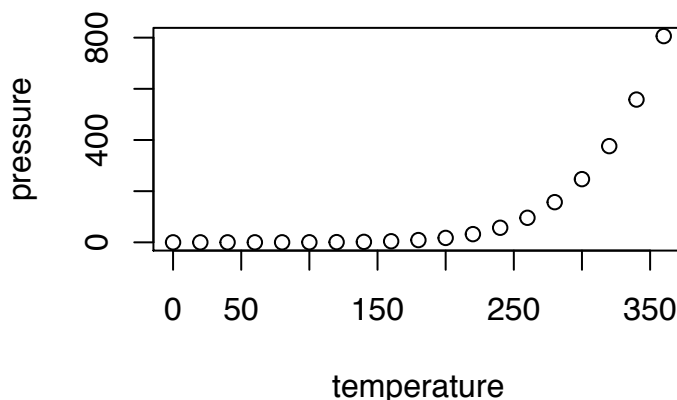
By comparing the summary results of models before and after the transformation, we can see that the residual standard error has been reduced from 4.524 to 0.2071, and the adjusted R^2 has risen from 0.6811 to 0.7136. Thus the transformed model for O_3 is:

$$O_3^{0.28} = -0.1049 + 0.3296 * temp + 0.0774 * humidity - 0.001 * ibh$$

2. Use the `pressure` data to fit a model with `pressure` as the response and `temperature` as the predictor using transformations to obtain a good fit.

Plot *pressure* versus *temperature* and find out there seems existing polynomial relationship between them. Construct orthogonal polynomials of *temperature* up to power of 10.

```
data(pressure)
plot(x=pressure$temperature, y=pressure$pressure, xlab="temperature", ylab="pressure")
```



```
lm_pressure <- lm(pressure ~ poly(temperature,10), data=pressure)
summary(lm_pressure)
```

```
##
## Call:
## lm(formula = pressure ~ poly(temperature, 10), data = pressure)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32109 -0.02985  0.00889  0.04763  0.21745
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      124.33671     0.04097 3034.705 < 2e-16 ***
## poly(temperature, 10)1  722.17059     0.17859 4043.712 < 2e-16 ***
## poly(temperature, 10)2  545.94688     0.17859 3056.967 < 2e-16 ***
## poly(temperature, 10)3  280.65281     0.17859 1571.483 < 2e-16 ***
## poly(temperature, 10)4   97.13691     0.17859  543.907 < 2e-16 ***
## poly(temperature, 10)5   20.07923     0.17859  112.431 4.38e-14 ***
## poly(temperature, 10)6    1.24400     0.17859    6.966 0.000117 ***
## poly(temperature, 10)7   -0.42142     0.17859   -2.360 0.045980 *
## poly(temperature, 10)8   -0.07879     0.17859   -0.441 0.670748
## poly(temperature, 10)9   -0.07262     0.17859   -0.407 0.694952
## poly(temperature, 10)10 -0.18567     0.17859   -1.040 0.328900
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1786 on 8 degrees of freedom
## Multiple R-squared:      1, Adjusted R-squared:      1
## F-statistic: 2.847e+06 on 10 and 8 DF, p-value: < 2.2e-16
```

By checking summary statistics, the p -value of terms to the power below 8 are all significant (<0.05). Thus the fitted model of pressure with temperature will be (P : pressure, T : temperature):

$$P = 124.34 + 722.17 * T + 545.95 * T^2 + 280.65 * T^3 + 97.14 * T^4 + 20.08 * T^5 + 1.24 * T^6 + -0.42 * T^7$$

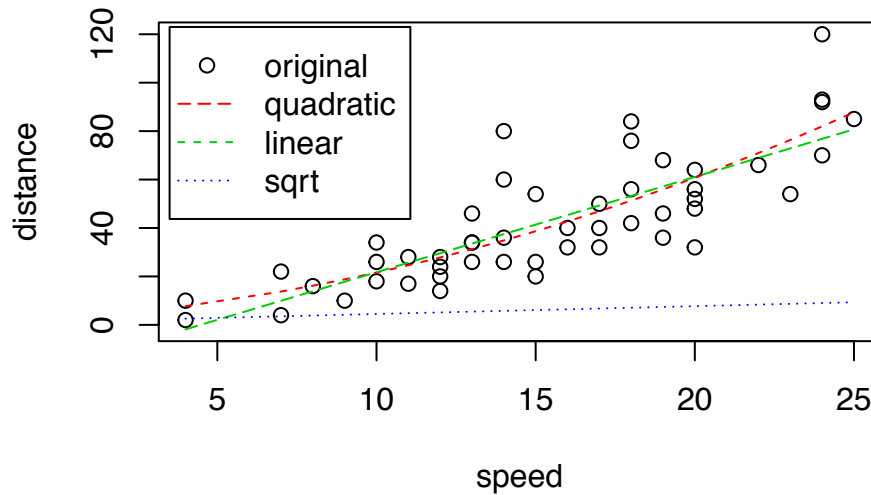
3. Use the `cars` data with distance as the response and speed as the predictor.

- Plot `dist` against `speed`.
- Show a linear fit to the data on the plot.
- Show a quadratic fit to the data on the plot.
- Now use `sqrt(dist)` as the response and fit a linear model. Show the fit on the same plot.

```
data(cars)
lm_cars_linear <- lm(dist~speed, data=cars)
lm_cars_quad <- lm(dist~speed+I(speed^2), data=cars)
lm_cars_sqrt <- lm(sqrt(dist)~speed, data=cars)

bind<-cbind(cars$dist, lm_cars_quad$fit, lm_cars_linear$fit, lm_cars_sqrt$fit)
matplot(cars$speed, bind, type="p", pch=1, lty=c(1,2,5,3),
        xlab="speed", ylab="distance", main="Speed v.s. Distance")
legend(x=3.5,y=123, legend=c("original", "quadratic", "linear", "sqrt"),
       lty=c(NA,5,2,3), pch=c(1,NA,NA,NA), col=seq_len(ncol(bind)))
```

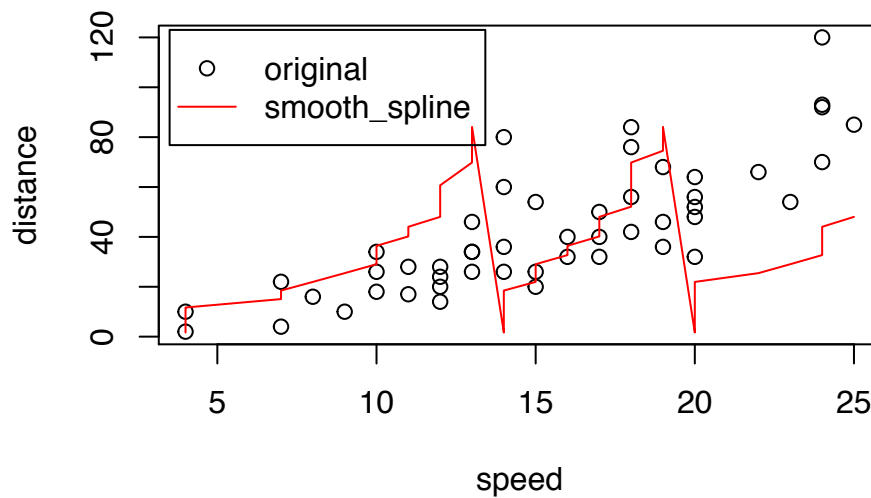
Speed v.s. Distance



- (e) Compute the default smoothing spline fit to the plot and display on a fresh plot of the data. How does it compare to the previous fits?

```
library(splines)
ssf <- smooth.spline(cars$speed, cars$dist)
bind_smooth <- cbind(cars$dist, ssf$y)
matplot(cars$speed, bind_smooth, type="pl", pch=1, lty=1,
        xlab="speed", ylab="distance", main="Speed v.s. Distance")
legend(x=3.5, y=123, legend=c("original", "smooth_spline"),
       lty=c(NA, 1), pch=c(1, NA), col=seq_len(ncol(bind_smooth)))
```

Speed v.s. Distance



Compared to the previous fits, though smoothing spline fit captures the points of inflexion, the valleys before $speed=15$ and after $speed=20$ is fit too roughly.

Attachment: RMarkdown Codes

title: 'STAT 500: HW6'

author: "Jasmine Mou"

date: "11/7/2017"

output: pdf_document

1. Using the `ozone` data, fit a model with `O3` as the response and `temp`, `humidity` and `ibh` as predictors. Use the Box-Cox method to determine the best transformation on the response.

```
```{r message=FALSE}
```

```
require(MASS)
```

```
data(ozone, package="faraway")
```

```
lm_ozone <- lm(O3 ~ temp + humidity + ibh, data=ozone)
```

```
```
```

*Check whether the response needs transformation and plot the results. The first plot is too broad so we narrow down the range of λ in the second plot. *

```
```{r, fig.width=8, fig.height=3}
```

```
par(mfrow=c(1,2))
```

```
boxcox(lm_ozone, plotit=T)
```

```
boxcox(lm_ozone, plotit=T, lambda=seq(0.1,0.4,by=0.05))
```

```
```
```

*The λ that optimizes the log-Likelihood is picked at some point between 0.25 and 0.30, with 95% confidence interval being $\sim (0.15, 0.4)$, which is far from and excludes 1. Thus we need the transformation. Let's pick $\hat{\lambda}$ being 0.28. *

```
```{r}
```

```
lambda_hat <- 0.28
```

```
lm_ozone_transformed_response <- lm(O3^lambda_hat ~ temp + humidity +
ibh, data=ozone)
```

```
summary(lm_ozone)
```

```
summary(lm_ozone_transformed_response)
```

```
```
```

*By comparing the summary results of models before and after the transformation, we can see that the residual standard error has been reduced from 4.524 to 0.2071, and the adjusted R^2 has risen from 0.6811 to 0.7136. Thus the transformed model for `O3` is: $Y = -0.1049 + 0.3296 * \text{temp} + 0.0774 * \text{humidity} - 0.001 * \text{ibh}$ *

2. Use the `pressure` data to fit a model with `pressure` as the response and `temperature` as the predictor using transformations to obtain a good fit.

Plot `pressure` versus `temperature` and find out there seems existing polynomial relationship between them. Construct orthogonal polynomials of `temperature` up to power of 10.

```
```{r, fig.width=4, fig.height=3}
```

```
data(pressure)
```

```
plot(x=pressure$temperature, y=pressure$pressure, xlab="temperature",
ylab="pressure")
lm_pressure <- lm(pressure ~ poly(temperature,10), data=pressure)
summary(lm_pressure)
```

```

*By checking summary statistics, the p-value of terms to the power below 8 are all significant (<0.05). Thus the fitted model of pressure with temperature will be (P: `pressure`, T: `temperature`):

```
$$P = `r round(lm_pressure$coefficients[1],2)` +
`r round(lm_pressure$coefficients[2],2)`*T +
`r round(lm_pressure$coefficients[3],2)`*T^2 +
`r round(lm_pressure$coefficients[4],2)`*T^3 +
`r round(lm_pressure$coefficients[5],2)`*T^4 +
`r round(lm_pressure$coefficients[6],2)`*T^5 +
`r round(lm_pressure$coefficients[7],2)`*T^6 +
`r round(lm_pressure$coefficients[8],2)`*T^7 $$*
```

3. Use the `cars` data with distance as the response and speed as the predictor.

- (a) Plot `dist` against `speed`.
- (b) Show a linear fit to the data on the plot.
- (c) Show a quadratic fit to the data on the plot.
- (d) Now use `sqrt(dist)` as the response and fit a linear model. Show the fit on the same plot.

```
```{r, fig.width=5, fig.height=3.5}
data(cars)
lm_cars_linear <- lm(dist~speed, data=cars)
lm_cars_quad <- lm(dist~speed+I(speed^2), data=cars)
lm_cars_sqrt <- lm(sqrt(dist)~speed, data=cars)

bind<-cbind(cars$dist, lm_cars_quad$fit, lm_cars_linear$fit,
lm_cars_sqrt$fit)
matplot(cars$speed, bind, type="p", pch=1, lty=c(1,2,5,3),
 xlab="speed", ylab="distance", main="Speed v.s. Distance")
legend(x=3.5,y=123, legend=c("original", "quadratic", "linear", "sqrt"),
 lty=c(NA,5,2,3), pch=c(1,NA,NA,NA), col=seq_len(ncol(bind)))
```
```

- (e) Compute the default smoothing spline fit to the plot and display on a fresh plot of the data. How does it compare to the previous fits?

```
```{r, warning=FALSE, fig.width=5, fig.height=3.5}
library(splines)
ssf <- smooth.spline(cars$speed, cars$dist)
bind_smooth <- cbind(cars$dist, ssf$y)
matplot(cars$speed, bind_smooth, type="pl", pch=1, lty=1,
 xlab="speed", ylab="distance", main="Speed v.s. Distance")
legend(x=3.5,y=123, legend=c("original", "smooth_spline"),
 lty=c(NA,1), pch=c(1,NA), col=seq_len(ncol(bind_smooth)))
```
```

Compared to the previous fits, though smoothing spline fit captures the points of inflexion, the valleys before `speed`=15 and after `speed`=20 is fit too roughly.