A2 Project 1

Johnsen & Johnsen

2022-11-15

Setting working directories and loading the data. Data is also normalized.

```
library(ggplot2)
library(numDeriv)
library(betareg)
library(gridExtra)

if (Sys.getenv("LOGNAME") == "mortenjohnsen"){
  setwd("/Users/mortenjohnsen/OneDrive - Danmarks Tekniske Universitet/DTU/9. Semester/02418 - Statisti
} else {
  setwd("~/Documents/02418 Statistical Modelling/Assignments/Assignment 1/Project-1")
  #setwd("~/Documents/02418 Statistical Modelling/Assignments/Assignment 1/Project-1/Project 2")
  #source("testDistribution.R")
}

D <- read.table("tuno.txt", header=TRUE, sep=" ",
                as.is=TRUE)

D$date <- as.Date("2003-01-01")-1+D$r.day
D$pow.obs.norm <- D$pow.obs/5000
```

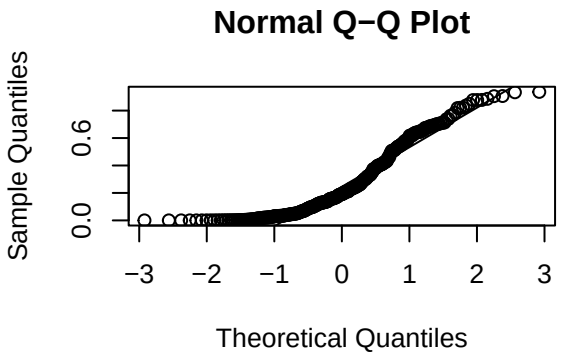
Data is transformed to be normal using transformation 1 in description of assignment 1:

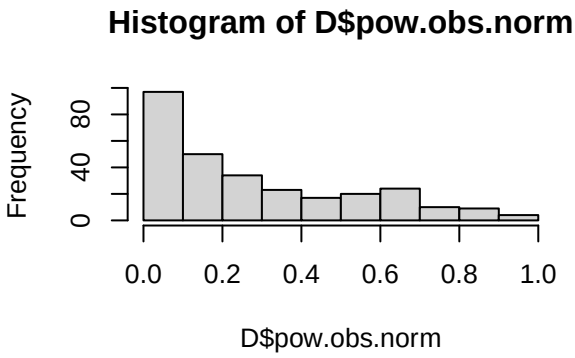
$$y^{(\lambda)} = \frac{1}{\lambda} log\left(\frac{y^{\lambda}}{1-y^{\lambda}}\right); \lambda > 0$$

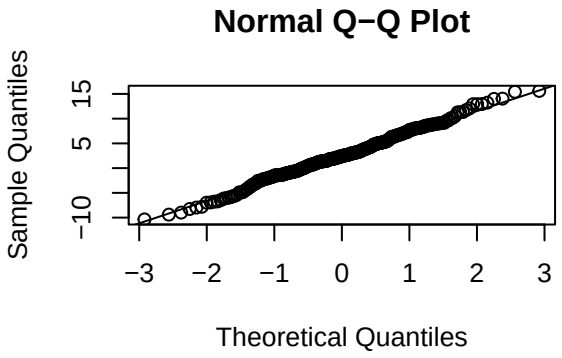
Below can be seen qq-plots comparing the normalized data before and after transformation to allow for a better understanding of the motivation behind the transformation.

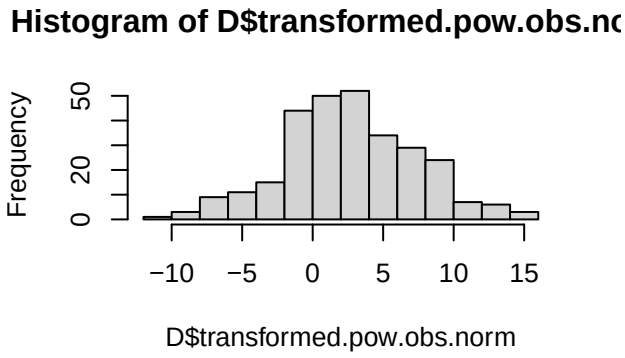
```
#lambdas <- seq(-0.5,0.5,by=0.01)
#library(MASS)
#boxcox(lm(D$pow.obs.norm~1), lambda=lambdas)

#Define transformation function
Trans.eq1 <- function(lambda, y){
  y_lambda <- 1/lambda * log(y^lambda/(1-y^lambda))#, lambda > 0
  return(y_lambda)
}
#Optimization function
#Måske er det bedre at lave nogle undersøgelser selv frem for bare at optimere lambda (Overvej til sene
#se kode fra lecture 4 linje 5-73
lambda_NLL <- function(lambda, x = D$pow.obs.norm){
```

```
  y <- Trans.eq1(lambda, x)
  NLL <- -as.numeric(shapiro.test(y)$statistic)
  return(NLL)
}
lambda.hat <- nlminb(start = 0.2, objective = lambda_NLL)
#round to two decimal points.
lambda <- round(lambda.hat$par, 2)
D$transformed.pow.obs.norm <- Trans.eq1(lambda, D$pow.obs.norm)

#Check qqplot:
par(mfrow = c(2,2))
qqnorm(D$pow.obs.norm)
qqline(D$pow.obs.norm)
hist(D$pow.obs.norm)
qqnorm(D$transformed.pow.obs.norm)
qqline(D$transformed.pow.obs.norm)
hist(D$transformed.pow.obs.norm)
```







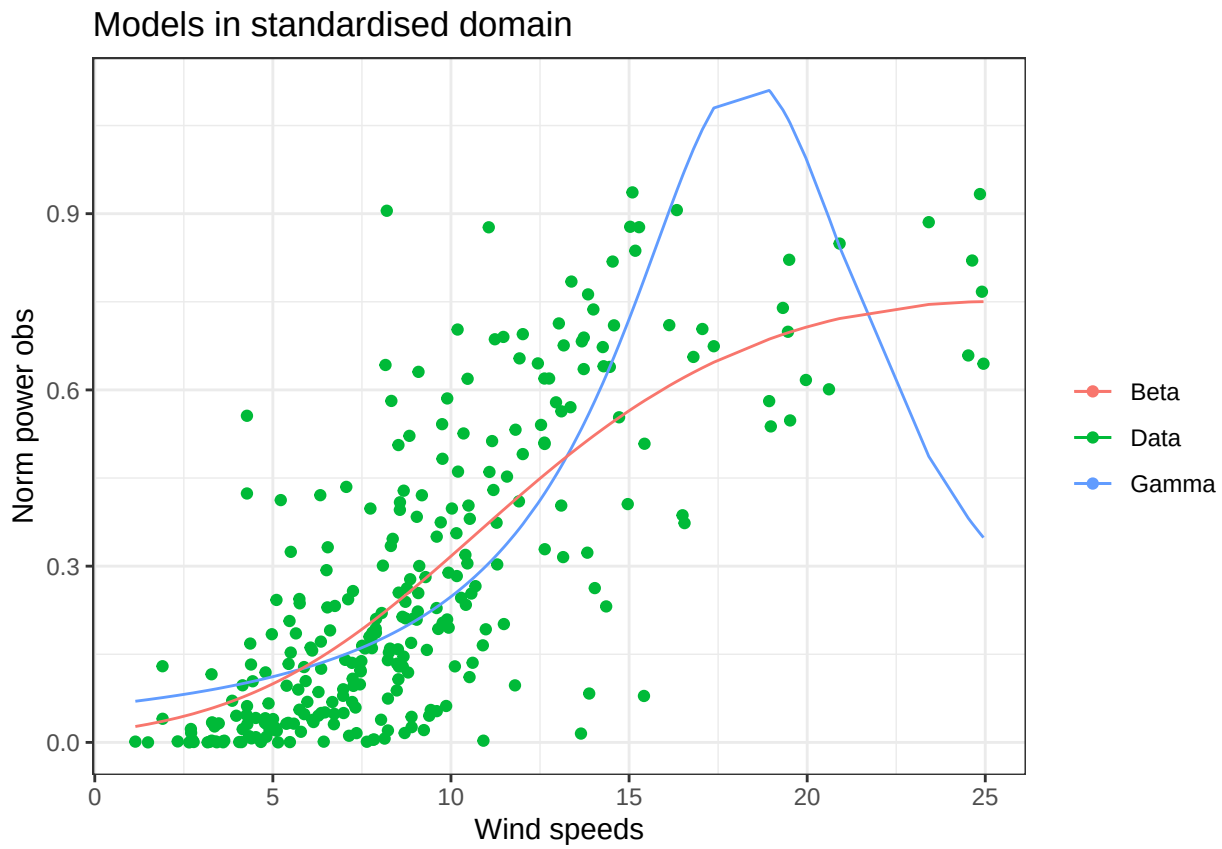


Two models will be produced: a non-normal beta or gamma regression model or the normalized data and a normal linear regression model of the transformed, normalized data.

First the non-normal:

```
#gamma's pretty bad. Maybe beta would be better?
gammaModel <- glm( (pow.obs.norm ~ ws30 + I(ws30^2)), data = D, family = Gamma)
betaModel <- betareg(formula = (pow.obs.norm ~ ws30 + I(ws30^2)), data = D)
D$gammaModel.pow <- gammaModel$fitted.values
```

```
D$betaModel.pow <- betaModel$fitted.values

ggplot(data = D)+
  geom_point(aes(x=ws30, y=pow.obs.norm, colour="Data"))+
  geom_line(aes(x=ws30, y=gammaModel.pow, colour="Gamma"))+
  geom_line(aes(x=ws30, y=betaModel.pow, colour="Beta"))+
  labs(x = "Wind speeds", y="Norm power obs", colour = "")+
  theme_bw()+
  ggtitle("Models in standardised domain")
```



This can also be done by hand. See p. 163-65 for reference. Exponential family regression models estimate the log-likelihood contribution of an outcome y_i as seen below:

$$logL(\theta_i, \phi) = \frac{y_i\theta_i - A(\theta_i)}{\phi} + c(y_i, \phi)$$

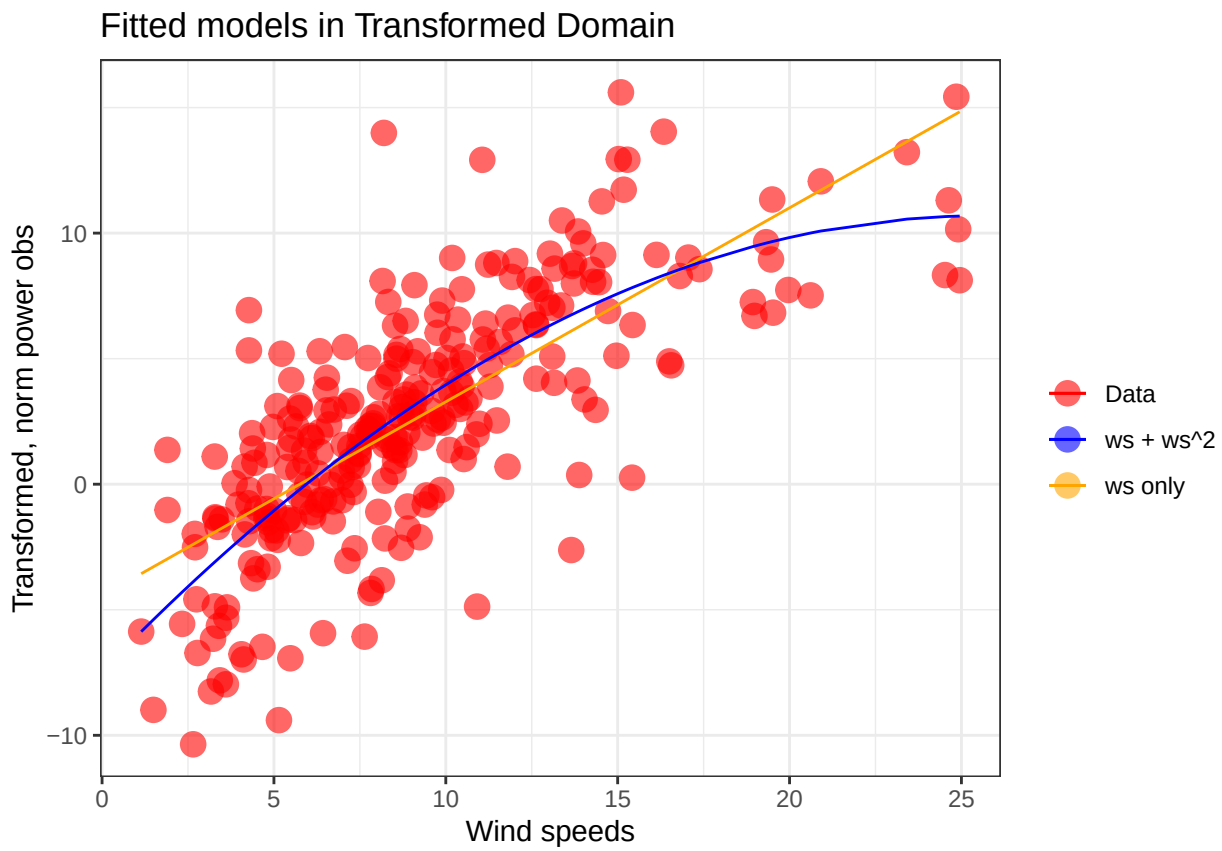
Here, $A(\theta_i)$ is defined as such that its derivative is the mean $E_{y_1} = A'(\theta_i) = \mu_i$. ϕ is the dispersion parameter, where $var(y_i) = \phi A''(\theta_i) = \phi v(\mu_i)$. To use this general exponential family for regression analysis we only have to specify $A(\theta_i)$ and a link function $h(\mu_i)$. The reason as to why $c(y_i, \phi)$ can be neglected is that the term disappears when determining the score function anyways.??? KOMMENTAR

Now to the normal linear model:

```
ws.only <- lm( (D$transformed.pow.obs.norm ~ ws30), data = D ) #good for comparison transfit1
ws.and.ws.squared <- lm( (D$transformed.pow.obs.norm ~ ws30 + I(ws30^2)), data = D) #brilliant transfit
ws.squared.only <- lm( (D$transformed.pow.obs.norm ~ I(ws30^2)), data = D ) #1. jeg har ladt disse to s
ws.ws2.and.wd <- lm( (D$transformed.pow.obs.norm ~ ws30 + I(ws30^3) + cos(wd30)), data = D ) #2.
summary(ws.ws2.and.wd)
```

```
##
## Call:
## lm(formula = (D$transformed.pow.obs.norm ~ ws30 + I(ws30^3) +
##     cos(wd30)), data = D)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.9103 -1.5611  0.1013  1.7783 10.9988
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.4832169  0.5872795 -11.039  < 2e-16 ***
## ws30         1.1029703  0.0805139  13.699  < 2e-16 ***
## I(ws30^3)   -0.0006824  0.0001463  -4.664 4.78e-06 ***
## cos(wd30)   -0.8496758  0.2597539  -3.271   0.0012 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.006 on 284 degrees of freedom
## Multiple R-squared:  0.6012, Adjusted R-squared:  0.597
## F-statistic: 142.7 on 3 and 284 DF,  p-value: < 2.2e-16
```

```
#transfitS <- lm( (D$transformed.pow.obs.norm ~ ws30 + I(ws30^2) + Season), data = D )
#summary(ws.only)
#summary(ws.and.ws.squared)

D$ws.pred <- ws.only$coefficients[1] + ws.only$coefficients[2] * D$ws30
D$ws.ws2.pred <- ws.and.ws.squared$coefficients[1] + ws.and.ws.squared$coefficients[2] * D$ws30 + ws.an
D$ws.ws2.wd.pred <- ws.ws2.and.wd$coefficients[1] + ws.ws2.and.wd$coefficients[2] * D$ws30 +
  ws.ws2.and.wd$coefficients[3] * D$ws30^2 + ws.ws2.and.wd$coefficients[4] * cos(D$wd30)

#D$transformed.predS <- transfitS$coefficients[1] + transfitS$coefficients[2] * D$ws30 + transfitS$coef
#summary(transfitS)

ggplot(data = D)+
  geom_point(aes(x=ws30, y=transformed.pow.obs.norm, colour="Data"), size = 4, alpha = 0.6)+
  geom_line(aes(x=ws30, y=ws.pred, colour="ws only"))+
  geom_line(aes(x=ws30, y=ws.ws2.pred, colour="ws + ws^2"))+
  scale_colour_manual(values = c("red", "blue", "orange"))+
  #geom_line(aes(x=ws30, y=ws.ws2.wd.pred, colour="ws + ws^2 + cos(wd)"))+
  labs(x = "Wind speeds", y="Transformed, norm power obs", colour = "")+
  theme_bw()+
  ggtitle("Fitted models in Transformed Domain")
```



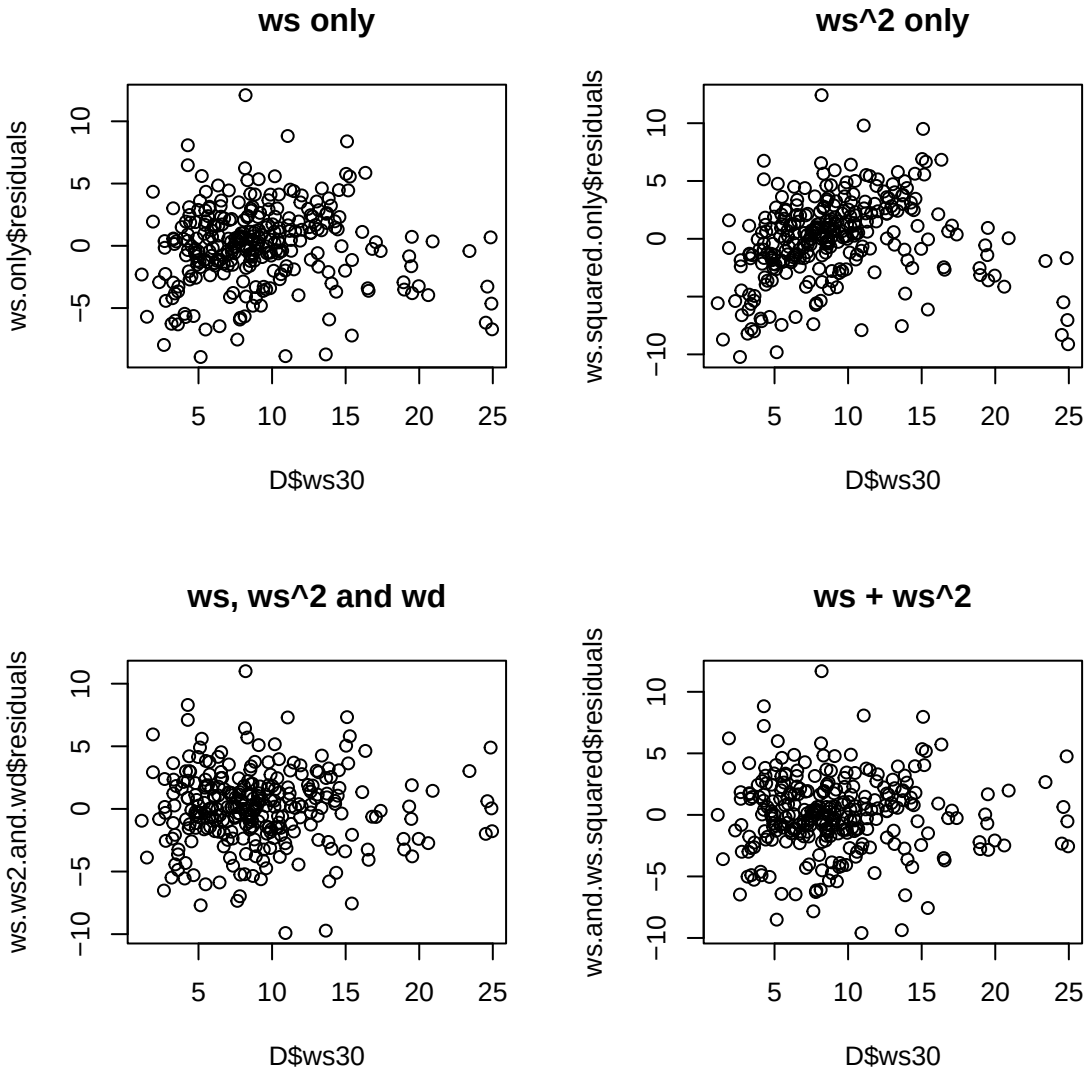

```
summary(lm( (D$transformed.pow.obs.norm ~ ws30 + I(ws30^2) + wd30), data = D)) #simply a check with the
```

```
##
## Call:
## lm(formula = (D$transformed.pow.obs.norm ~ ws30 + I(ws30^2) +
##     wd30), data = D)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.6060 -1.4739  0.0307  1.7897 11.6787
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.496543   0.801711  -9.351  < 2e-16 ***
## ws30         1.416925   0.139440  10.162  < 2e-16 ***
## I(ws30^2)   -0.027728   0.005729  -4.840 2.14e-06 ***
## wd30         0.016252   0.107570   0.151     0.88
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.058 on 284 degrees of freedom
## Multiple R-squared:  0.5872, Adjusted R-squared:  0.5828
## F-statistic: 134.6 on 3 and 284 DF,  p-value: < 2.2e-16
```

```
#Outcommented so it is possible to create a PDF
# library(plotly)
#
# fig <- plot_ly(mtcars, x = D$ws30, y = cos(D$wd30), z = D$transformed.pow.obs.norm
#                , color = "Observed Power Production (Normalised)", colors = "red")
# fig <- fig %>% add_markers()
# fig <- fig %>% layout(scene = list(xaxis = list(title = 'Pos.Obs'),
#                                    yaxis = list(title = 'cos(WD)'),
#                                    zaxis = list(title = 'WS')))
# add_trace(type = 'mesh3d',
#           p = fig,
#           x = D$ws30,
#           y = cos(D$wd30),
#           z = D$ws.ws2.wd.pred,
#           facecolor = rep("blue", length(D$ws30)),
#           opacity = 0.1
# )
```

Tror ikke at det giver mening at have wind direction med, det ligner ikke rigtig at det giver så meget? Men AIC er dog bedre... Ligeledes er det også statistisk signifikant, hvis man tilføjer måned på året.

Tjek af modellerne:



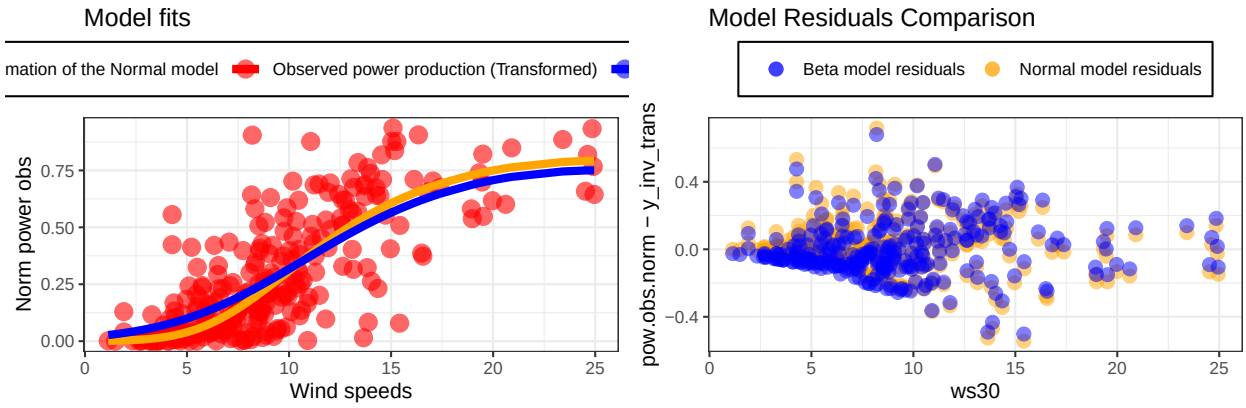
Normal linear model by hand and by optimisation:

```
D$base <- 1 #slide 15 for all of this
X <- as.matrix(data.frame(D$base, D$ws30, D$ws30^2))
Y <- D$transformed.pow.obs.norm
betas <- solve( (t(X)%*%X) ) %*% t(X) %*% Y

NLL.norm.lin <- function(p, regr1, regr2, response){
  mu <- p[1] + p[2]*regr1 + p[3]*regr2
  sigma <- sqrt( 1/length(response) * sum( (response - mu)^2  ) )
  return( -sum( dnorm( x = response, mean = mu, sd = sigma, log = T ) )  )
}
norm.lin.par <- nlminb(c(1,1,1), objective = NLL.norm.lin, regr1 = D$ws30, regr2 = D$ws30^2, response =

round( rbind(ws.and.ws.squared$coefficients,betas[1:3],norm.lin.par$par), digits=6)
```

```
##      (Intercept)     ws30 I(ws30^2)
```

```
## [1,]   -7.450337 1.418150 -0.027717
## [2,]   -7.450337 1.418150 -0.027717
## [3,]   -7.450346 1.418151 -0.027717
```

```
y_p <- D$ws.ws2.pred
#y_pS <- D$transformed.predS
D$y_inv_trans <- 1/( exp(y_p*lambda)+1 )^(1/lambda) * exp(y_p)
#D$Season_inv_trans <- 1/( exp(y_pS*lambda)+1 )^(1/lambda) * exp(y_pS)

#D$Season <- 0
#D$Season[D$month > 5] <- 1

fitplot <- ggplot(data = D)+
  geom_point(aes(x=ws30, y=pow.obs.norm, colour = "Observed power production (Transformed)"), size = 4,
  geom_line(aes(x=ws30, y=y_inv_trans, colour="Inverse Transformation of the Normal model"), size = 2)+
  geom_line(aes(x=ws30, y=betaModel.pow, colour="The directly fitted beta model"), size = 2)+
  #geom_line(aes(x=ws30, y=Season_inv_trans, colour="InvS"))+
  labs(x = "Wind speeds", y="Norm power obs", colour = "")+
  theme_bw()+
  labs(colour = "")+
  ggtitle("Model fits")+
  theme(legend.background = element_rect(colour = "black"), legend.position = "top")+
  scale_colour_manual(values = c("orange", "red", "blue"))
#scale_shape_manual(values = c(1:10))+
#scale_colour_manual(values = c("blue", "yellow", "black"))

resplot <- ggplot(data = D)+
  geom_point(aes(x = ws30, y = pow.obs.norm - y_inv_trans, colour = "Normal model residuals"), size = 3
  geom_point(aes(x = ws30, y = pow.obs.norm - betaModel.pow, colour = "Beta model residuals"), size = 3
  theme_bw()+
  scale_colour_manual(values = c("blue", "orange"))+
  ggtitle("Model Residuals Comparison")+
  labs(colour = "")+
  theme(legend.position = "top", legend.background = element_rect(colour = "black"))

grid.arrange(fitplot,resplot, nrow = 1)
```




```
#plot(D$ws30, D$pow.obs.norm-D$y_inv_trans)
#abline(h=0)
```

```
AIC(betaModel)
```

```
## [1] -480.2231
```

```
AIC(ws.and.ws.squared)
```

```
## [1] 1465.213
```

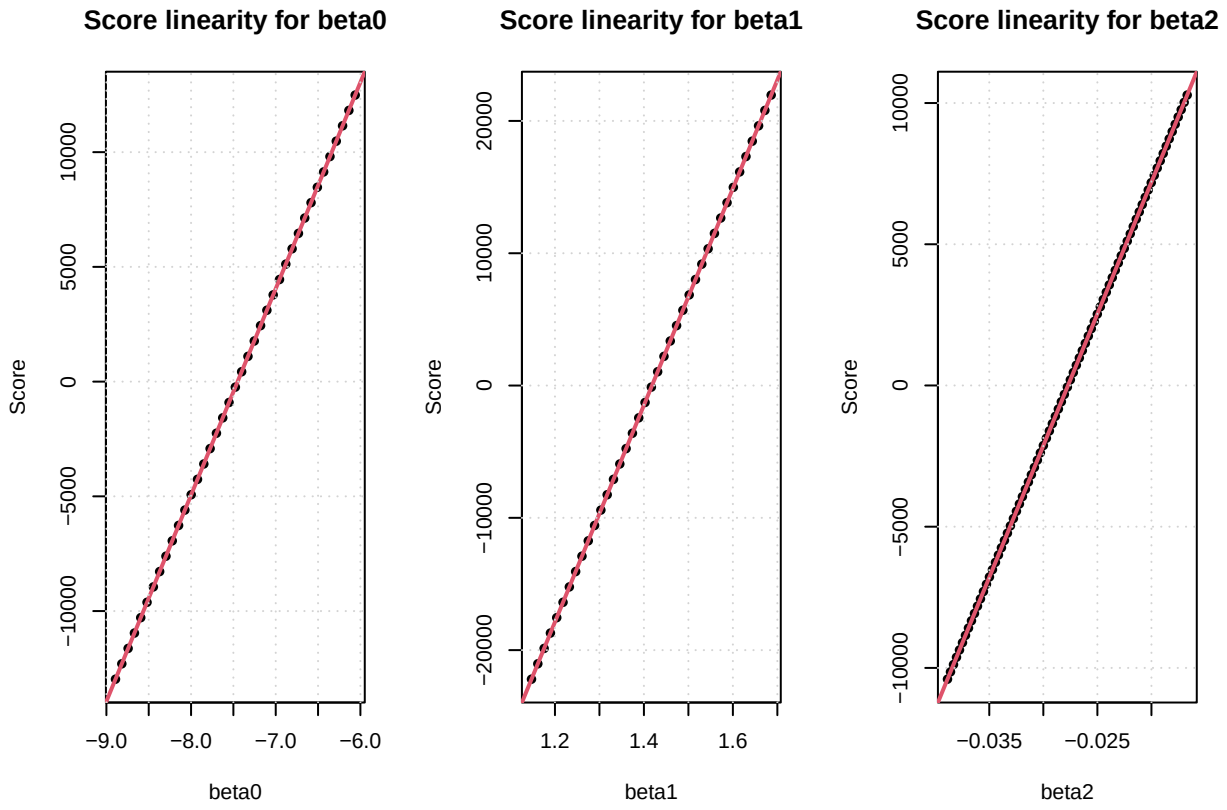
Uncertainties of parameters (in transformed domain)

```
norm.lin.sds <- sqrt(diag(solve(hessian( func = NLL.norm.lin, x = norm.lin.par$par, regr1 = D$ws30, regr
Wald.CI.upper <- norm.lin.par$par + qnorm(0.975) * norm.lin.sds
Wald.CI.lower <- norm.lin.par$par - qnorm(0.975) * norm.lin.sds

cat("beta_0 = ", norm.lin.par$par[1], "95% CI [",Wald.CI.lower[1],", ",Wald.CI.upper[1],"]"
    ,"\nbeta_1 = ", norm.lin.par$par[2], "95% CI [",Wald.CI.lower[2],", ",Wald.CI.upper[2],"]"
    ,"\nbeta_2 = ", norm.lin.par$par[3], "95% CI [",Wald.CI.lower[3],", ",Wald.CI.upper[3],"]"
)
```

```
## beta_0 =  -7.450346 95% CI [ -8.892549 ,  -6.008142 ]
## beta_1 =  1.418151 95% CI [ 1.147252 ,  1.68905 ]
## beta_2 =  -0.02771658 95% CI [ -0.03886543 ,  -0.01656773 ]
```

```
#Test regularity
score.nll.norm <- function(theta, regr1, regr2, response, sigma){

  beta0 <- theta[1]
  beta1 <- theta[2]
  beta2 <- theta[3]
  n <- length(response)

  theta <- beta0 + beta1 * regr1 + beta2 * regr2
  #sigma <- sqrt( 1/length(response) * sum( (response - theta)^2 ) )
  score <- n/sigma^2 * (mean(response) - theta)#p. 213
  return(-sum(score))
}

beta0.test.range <- seq(Wald.CI.lower[1], Wald.CI.upper[1], abs(0.01*norm.lin.par$par[1]))
beta1.test.range <- seq(Wald.CI.lower[2], Wald.CI.upper[2], abs(0.01*norm.lin.par$par[2]))
beta2.test.range <- seq(Wald.CI.lower[3], Wald.CI.upper[3], abs(0.01*norm.lin.par$par[3]))

sigma.s <- sqrt( 1/dim(D)[1] * sum ( (D$transformed.pow.obs.norm - ( norm.lin.par$par[1] + norm.lin.par

res0 <- apply(cbind(beta0.test.range, norm.lin.par$par[2], norm.lin.par$par[3]), MARGIN = 1, FUN = score

res1 <- apply(cbind(norm.lin.par$par[1], beta1.test.range, norm.lin.par$par[3]), MARGIN = 1, FUN = score

res2 <- apply(cbind(norm.lin.par$par[1], norm.lin.par$par[2], beta2.test.range), MARGIN = 1, FUN = score

par(mfrow = c(1,3))
plot(beta0.test.range, res0, main = "Score linearity for beta0"
```

```
     ,xlab = "beta0", ylab = "Score", pch = 16)
abline(lm(res0 ~ beta0.test.range), col = 2, lwd = 2)
grid()

plot(beta1.test.range, res1, main = "Score linearity for beta1"
     ,xlab = "beta1", ylab = "Score", pch = 16)
abline(lm(res1 ~ beta1.test.range), col = 2, lwd = 2)
grid()

plot(beta2.test.range, res2, main = "Score linearity for beta2"
     ,xlab = "beta2", ylab = "Score", pch = 16)
abline(lm(res2 ~ beta2.test.range), col = 2, lwd = 2)
grid()
```



Interpretation of parameters

```
ws.and.ws.squared$coefficients
```

```
## (Intercept)        ws30   I(ws30^2)
## -7.45033702  1.41814978 -0.02771652
```

```
beta_0 <- ws.and.ws.squared$coefficients[1]
( offset_inv <- 1/( exp(beta_0*lambda)+1 )^(1/lambda) * exp(beta_0) )
```

```
##  (Intercept)
```

```
## 0.0003463096
```

The regression model used is a normal model with data transformation. The first parameter is the intercept with the y-axis and can be transformed back to the original domain yielding an offset value of 0.00035. This is as expected since there should be no power production at 0 wind speed. The other two parameters are slope coefficients. The first is positive and describes the relationship between the wind speed and the power production. According to this model it should increase linearly (in the transformed domain). The second coefficient is negative and implies a negative proportionality to the square of the wind speed (in the transformed domain).