Exploring and Transforming Data

Dr. Randall R. Rojas

Note: To access your textbook resources type the following on the console:

```
#library(car)
#carWeb()
```

I. Univariate Characterizations

In a regression setting we first start by looking at a univariate characterization of the data. This entails looking at the respective statistical summaries, features, and distributions of all the variables we have. Among the popular univariate graphical tools, we will consider histograms, density plots, qqplots, and boxplots.

I.0 Statistical Summary

Looking at the summary statistics of our data allows to identify any potential issues we may encounter when building our regression model. For example, there may NAs, values outside the expected range, etc. Below are two examples using the `pastecs` and `psych` libraries.


```
library(pastecs)
```



```
##
## Attaching package: 'pastecs'

## The following object is masked from 'package:tidyr':
##
##     extract

## The following objects are masked from 'package:xts':
##
##     first, last

## The following objects are masked from 'package:dplyr':
##
##     first, last
```



```
# Summary for 1 variable:
stat.desc(rnorm(100),norm=TRUE)
```

```
##       nbr.val     nbr.null       nbr.na          min          max
## 100.000000000  0.000000000  0.000000000 -2.270031399  3.158588298
##         range          sum       median         mean      SE.mean
##   5.428619697 11.061751029 -0.009273364  0.110617510  0.096806935
##  CI.mean.0.95          var      std.dev     coef.var     skewness
##   0.192085961  0.937158263  0.968069348  8.751501870  0.375694605
##      skew.2SE     kurtosis     kurt.2SE   normtest.W   normtest.p
##   0.778223027  0.009654453  0.010091809  0.984344744  0.284579109
```

```
# Summary for an entire datacube:
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
attach(Prestige)
```

```
## The following object is masked from package:datasets:
##
##     women
```

```r
stat.desc(Prestige, norm=TRUE)
```

```
##                  education       income        women     prestige
## nbr.val       1.020000e+02 1.020000e+02 1.020000e+02  102.0000000
## nbr.null      0.000000e+00 0.000000e+00 5.000000e+00    0.0000000
## nbr.na        0.000000e+00 0.000000e+00 0.000000e+00    0.0000000
## min           6.380000e+00 6.110000e+02 0.000000e+00   14.8000000
## max           1.597000e+01 2.587900e+04 9.751000e+01   87.2000000
## range         9.590000e+00 2.526800e+04 9.751000e+01   72.4000000
## sum           1.095280e+03 6.933860e+05 2.955860e+03 4777.0000000
## median        1.054000e+01 5.930500e+03 1.360000e+01   43.6000000
## mean          1.073804e+01 6.797902e+03 2.897902e+01   46.8333333
## SE.mean       2.701562e-01 4.204089e+02 3.141236e+00    1.7034979
## CI.mean.0.95  5.359173e-01 8.339783e+02 6.231368e+00    3.3792816
## var           7.444408e+00 1.802786e+07 1.006471e+03  295.9943234
## std.dev       2.728444e+00 4.245922e+03 3.172493e+01   17.2044856
## coef.var      2.540915e-01 6.245930e-01 1.094755e+00    0.3673556
## skewness      3.247507e-01 2.129019e+00 8.987765e-01    0.3287011
## skew.2SE      6.791988e-01 4.452731e+00 1.879744e+00    0.6874610
## kurtosis     -1.028406e+00 6.287745e+00 -6.757897e-01  -0.7925793
## kurt.2SE     -1.085203e+00 6.635016e+00 -7.131135e-01  -0.8363533
## normtest.W    9.495767e-01 8.150513e-01 8.157943e-01    0.9719786
## normtest.p    6.772830e-04 5.633526e-10 5.956847e-10    0.0287498
##                     census type
## nbr.val       1.020000e+02   NA
## nbr.null      0.000000e+00   NA
## nbr.na        0.000000e+00   NA
## min           1.113000e+03   NA
## max           9.517000e+03   NA
## range         8.404000e+03   NA
## sum           5.509810e+05   NA
## median        5.135000e+03   NA
## mean          5.401775e+03   NA
## SE.mean       2.618934e+02   NA
## CI.mean.0.95  5.195260e+02   NA
## var           6.995989e+06   NA
## std.dev       2.644993e+03   NA
## coef.var      4.896527e-01   NA
## skewness      1.116090e-01   NA
## skew.2SE      2.334243e-01   NA
## kurtosis     -1.490291e+00   NA
## kurt.2SE     -1.572599e+00   NA
## normtest.W    9.008953e-01   NA
```

```
## normtest.p    1.270617e-06   NA
```

```r
library(psych)
```

```
##
## Attaching package: 'psych'
## The following object is masked from 'Prestige':
##
##     income
## The following object is masked from 'package:car':
##
##     logit
## The following objects are masked from 'package:scales':
##
##     alpha, rescale
```

```r
describe(Prestige)
```

```
##           vars   n    mean      sd  median trimmed     mad     min
## education    1 102   10.74    2.73   10.54   10.63    3.15    6.38
## income       2 102 6797.90 4245.92 5930.50 6161.49 3060.83  611.00
## women        3 102   28.98   31.72   13.60   24.74   18.73    0.00
## prestige     4 102   46.83   17.20   43.60   46.20   19.20   14.80
## census       5 102 5401.77 2644.99 5135.00 5393.87 4097.91 1113.00
## type*        6  98    1.79    0.80    2.00    1.74    1.48    1.00
##                max    range skew kurtosis     se
## education    15.97     9.59 0.32    -1.03   0.27
## income    25879.00 25268.00 2.13     6.29 420.41
## women        97.51    97.51 0.90    -0.68   3.14
## prestige     87.20    72.40 0.33    -0.79   1.70
## census     9517.00  8404.00 0.11    -1.49 261.89
## type*         3.00     2.00 0.40    -1.36   0.08
```

```r
# Note: The flag norm allows us to also include normal
# distribution statistics such as skewness, kurtosis, etc.
```


I.1 Histograms

Histograms should be one our first analysis tool since they can help us learn about each variable's properties.

Things to look for:

- *Outliers: This can influene the regression fit in many ways*
- *Shape: Is the distribution symmetric, skewed, etc?*
- *Range: Is the range of values very large? For exmaple several orders of magnitude?*
- *Density of observaions and/or clustering feautures. Are the data all clustered in certain regions? In high density regions your regression fit may be robust but then become unstable in sless dense ones.*

The near optimal number of bins to use based on the number of observations n is given by:

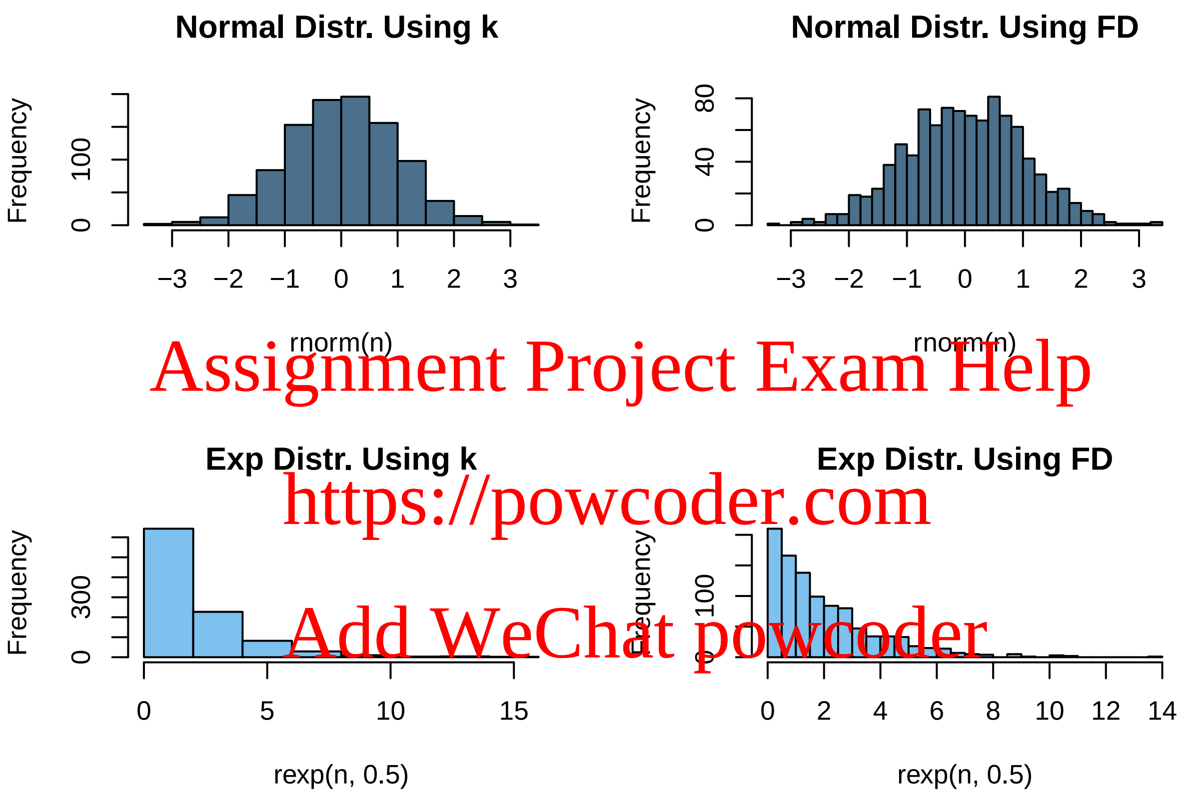
$$k = 1 + \log_2(n)$$

However, more sample-specific formulas exist such as Freedman's and Diaconis (FD)' suggested:

$$\frac{n^{1/3}(max - min)}{2(Q_3 - Q_1)}$$

For example, we can compare the two methods for 2 different data sets:

```r
#Compare histograms using k vs FD:
quartz()
par(mfrow=c(2,2))
n = 1000;k = 1 + log2(n)
hist(rnorm(n),breaks = k,col="skyblue4", main = "Normal Distr. Using k ")
hist(rnorm(n),breaks = "FD",col="skyblue4", main = "Normal Distr. Using FD")
n = 1000;k = 1 + log2(n)
hist(rexp(n,0.5),breaks = k,col="skyblue2", main ="Exp Distr. Using k")
hist(rexp(n,0.5),breaks = "FD",col="skyblue2", main ="Exp Distr. Using FD")
```



I.2 Density Estimation

Histograms are very informative but being able to visualize the density plot, along with the histogram and 1D scatterplot (rug-plot) can be more informative. A common kernel-density (nonparametric) estimate used is given by:

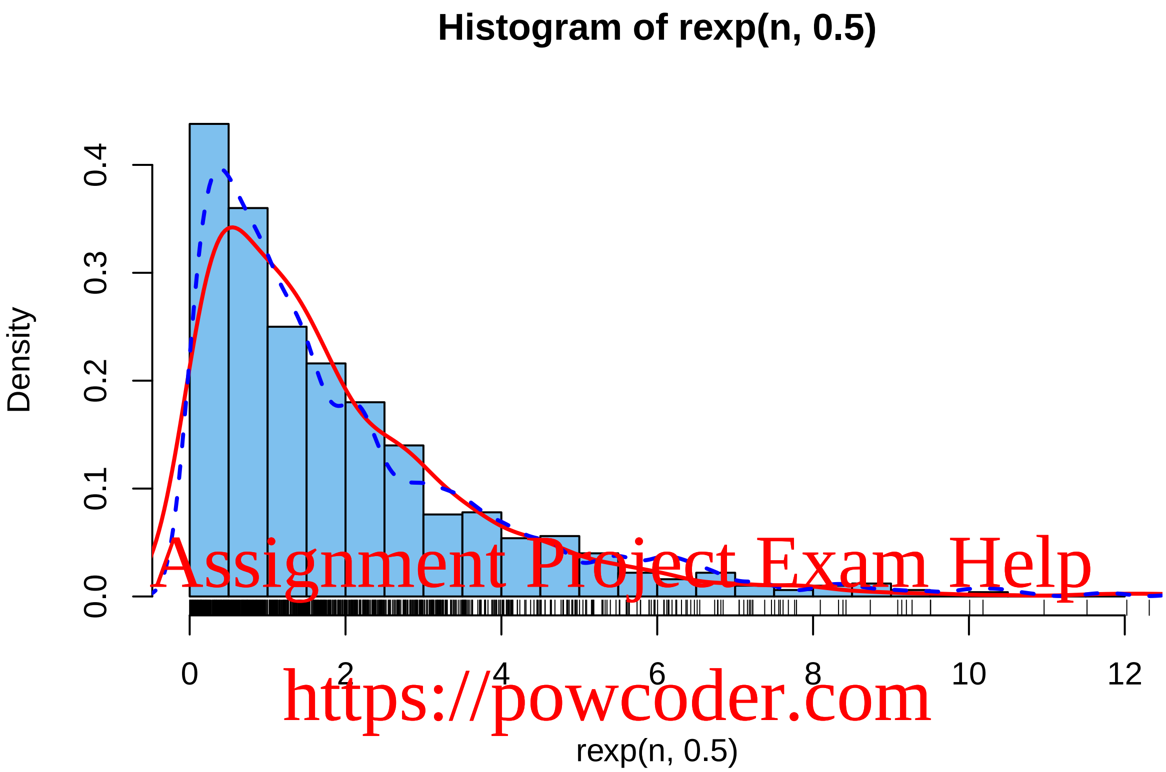
$$\widehat{p}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

where K is kernel function (e.g., Normal distribution), h is the bandwidth (amount of smoothness), and n the number of observations.

```r
quartz()
hist(rexp(n,0.5),breaks ="FD",col="skyblue2", freq = FALSE, ylab = "Density")
lines(density(rexp(n,0.5)),lwd = 2, col ="red")
```

```r
lines(density(rexp(n,0.5), adjust =0.5),lwd = 2, col = "blue", type ='l', lty=2)
rug(rexp(n,0.5))
```

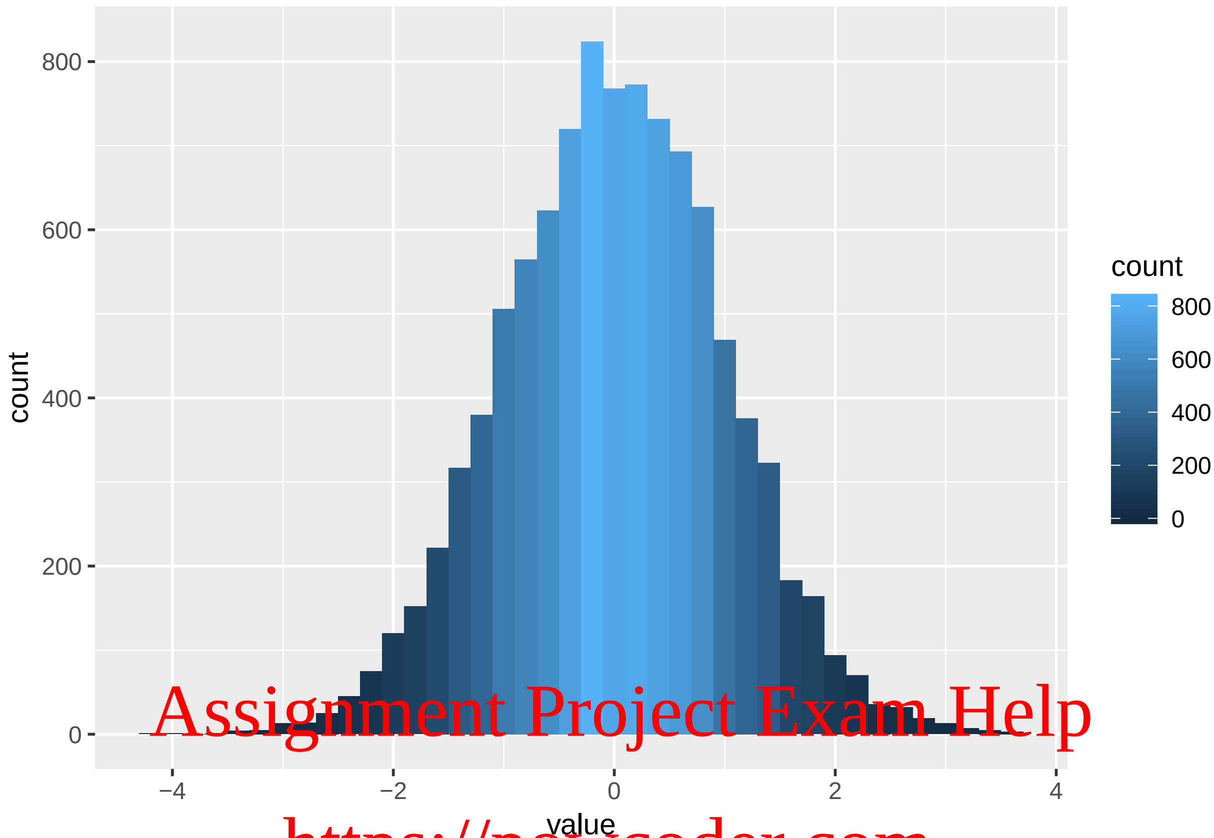
```
## Warning in rug(rexp(n, 0.5)): some values will be clipped
```




```r
#You can also show the denisty with color
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':
##
##     %+%, alpha

## The following object is masked from 'package:NLP':
##
##     annotate
```

```r
data=data.frame(value=rnorm(10000))
ggplot(data, aes(x=value))+geom_histogram(binwidth = 0.2, aes(fill = ..count..) )
```



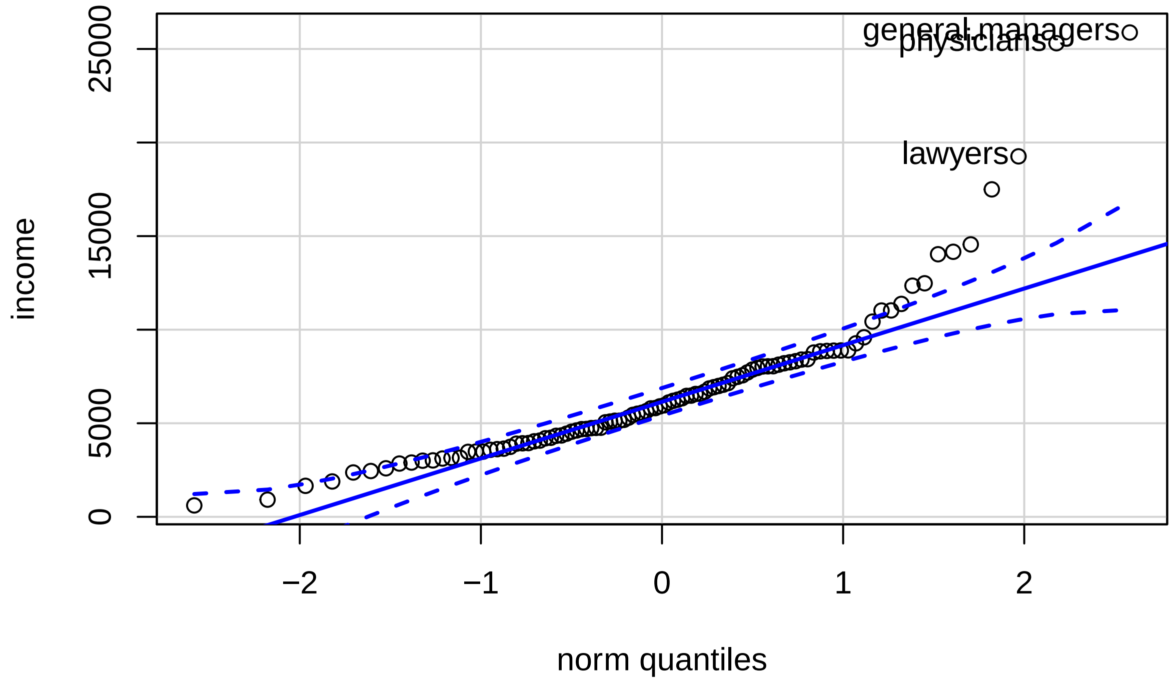
I.3 Quantile Plots

These plots are very useful for comparing the distribution of a variable to a theoretical reference distribution. By default the Normal distribution is used in the function 'qqPlot' (Quantile - Quantile) but we can provide any other distribution.

```
#The dataset Prestige from the car package has 102 observations
library(car)
attach(Prestige)
```

```
## The following object is masked from package:psych:
##
##     income

## The following objects are masked from Prestige (pos = 5):
##
##     census, education, income, prestige, type, women

## The following object is masked from package:datasets:
##
##     women
```

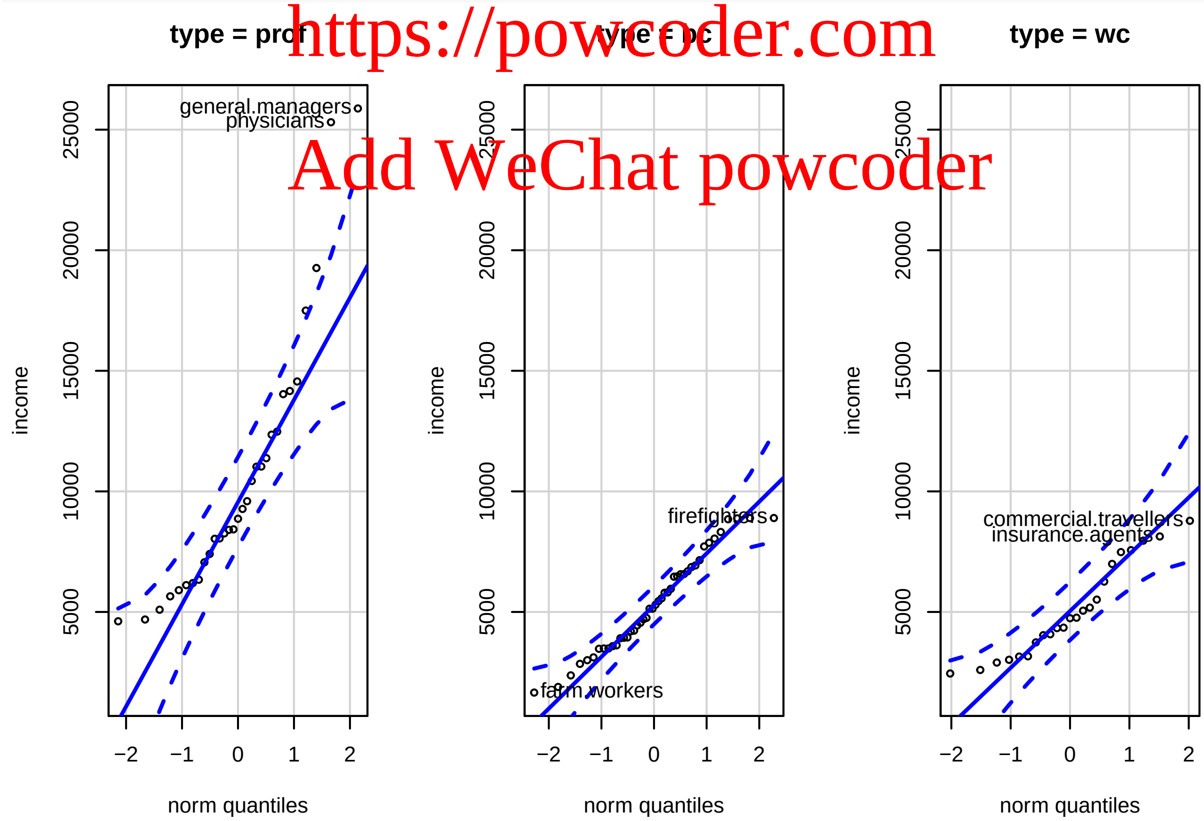
```
qqPlot(~ income, data=Prestige, id=list(n=3))
```




```
## general.managers      physicians        lawyers
##                2             24             17
```

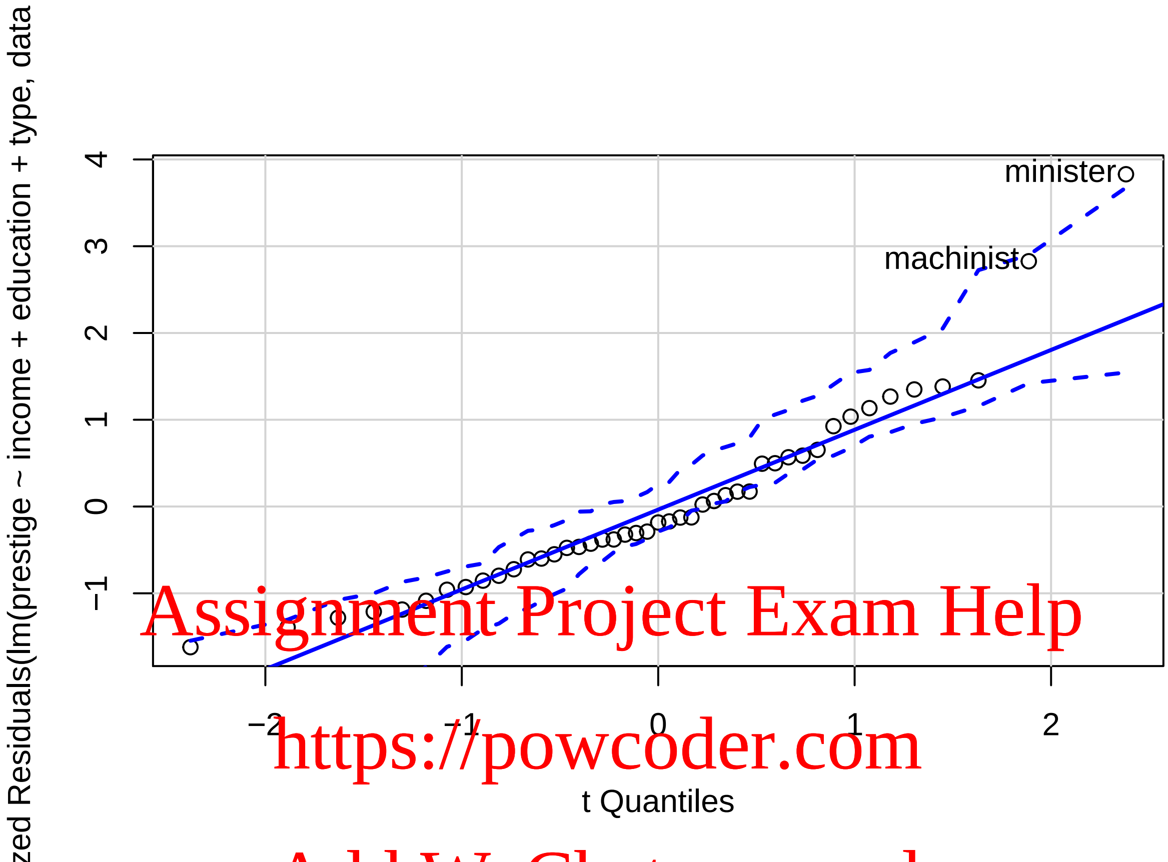


```r
qqPlot(income ~ type, data=Prestige, layout=c(1, 3))
```


```r
# Type = Professional (Prof), White Collar (WC), "Blue Collar (BC)"
```

```r
qqPlot(lm(prestige ~ income + education + type, data=Duncan),
    envelope=.99)
```

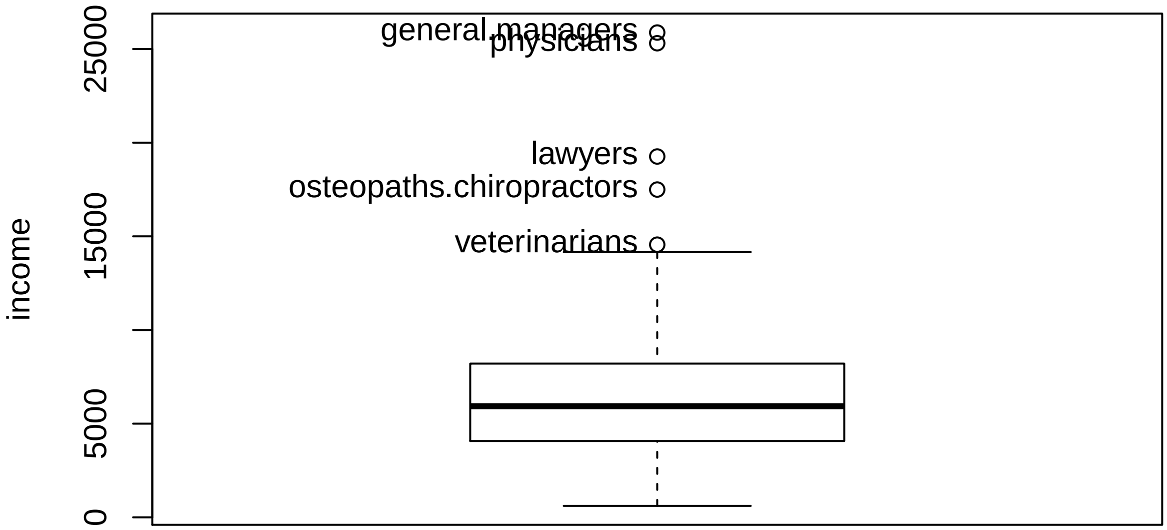



```
##  minister machinist
##         6        28
```

I.4 Boxplots

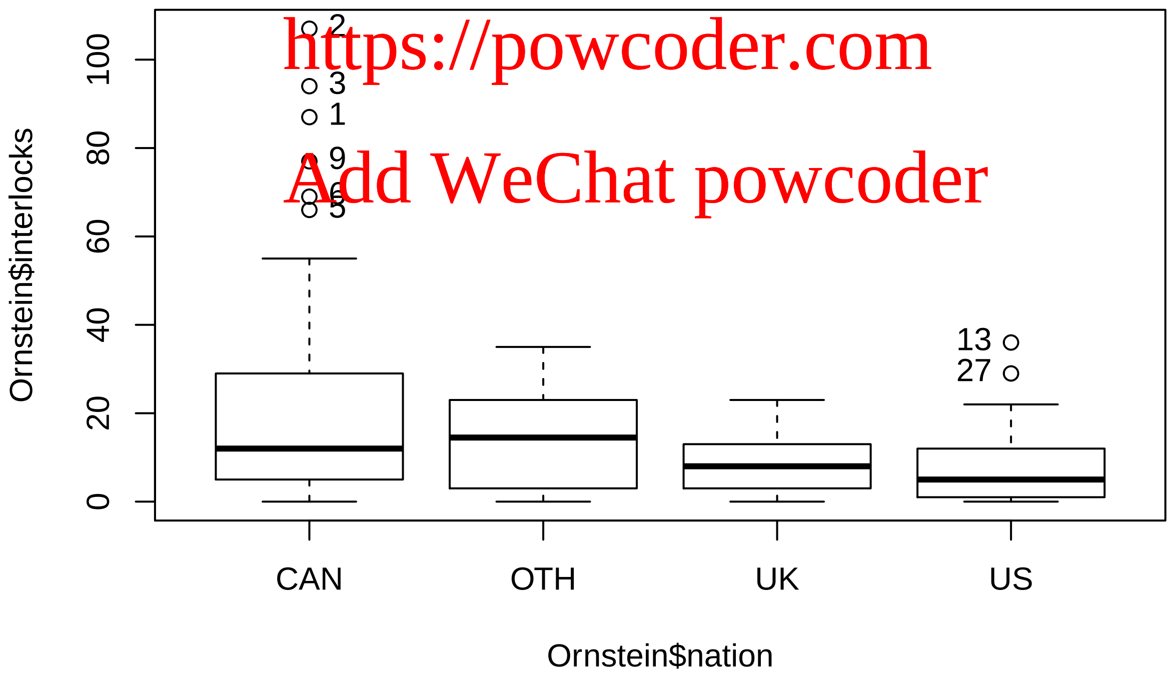
Boxplots allow us know more quantitative aspects of a variables distribution such as the min, max, Q1, Q3, IQR, and Median. Another helpful property is that they are very convenient when comparing many distributions simultaneously and/or conditioning on other variables.

```r
#Boxplot of Income
Boxplot(~income, data=Prestige)
```




```
## [1] "general.managers"          "lawyers"
## [3] "physicians"                "veterinarians"
## [5] "osteopaths.chiropractors"
```

```
# Note: For more choices of boxplots, look at: https://www.r-graph-gallery.com/boxplot/

# These are known as Parallel Boxplots
Boxplot(Ornstein$interlocks~Ornstein$nation)
```


```
## [1] "1"  "2"  "3"  "5"  "6"  "9"  "13" "27"
```

```
# Note: interlocks = Number of interlocking director and executive positions shared with other major fi

library("plotrix")
```

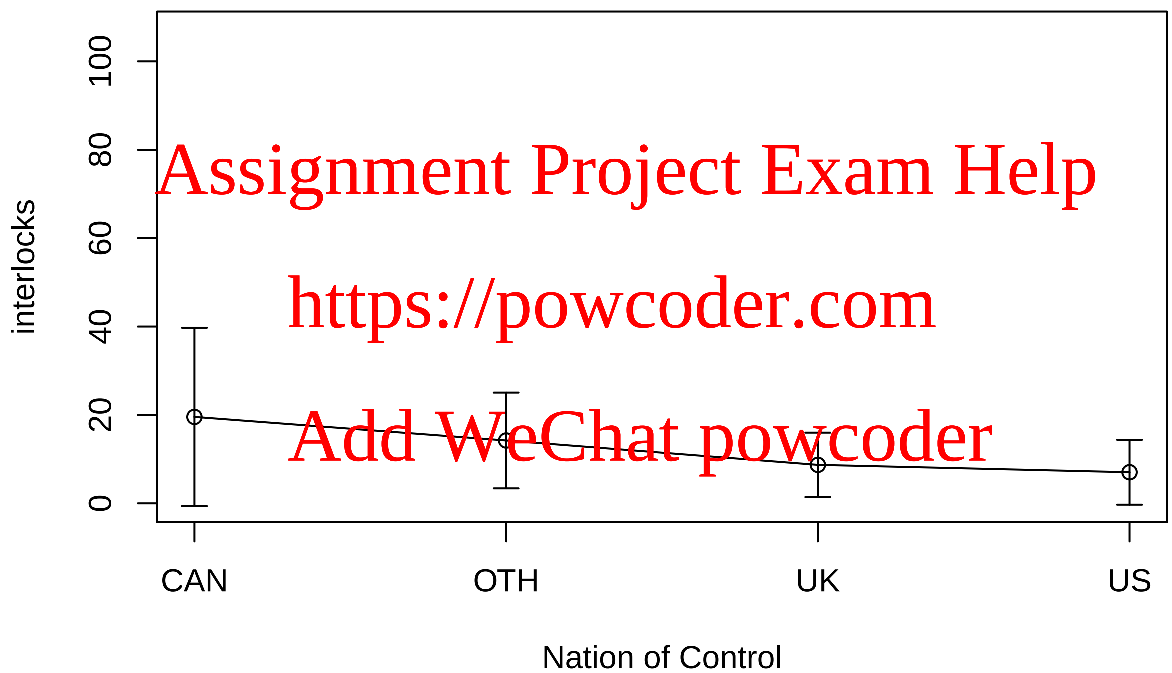
```
##
## Attaching package: 'plotrix'
```

```
## The following object is masked from 'package:psych':
##
##     rescale
```

```
## The following object is masked from 'package:rgl':
##
##     mtext3d
```

```
## The following object is masked from 'package:scales':
##
##     rescale
```

```
means <- Tapply(interlocks ~ nation, mean, data=Ornstein)
sds <- Tapply(interlocks ~ nation, sd, data=Ornstein)
plotCI(1:4, means, sds, xaxt="n", xlab="Nation of Control",
    ylab="interlocks", ylim=range(Ornstein$interlocks))
lines(1:4, means)
axis(1, at=1:4, labels = names(means))
```



II Bivariate Characterizations

Once you have the univariate description of the variables, we can then exam the pairwise associations via e.g., scatterplots

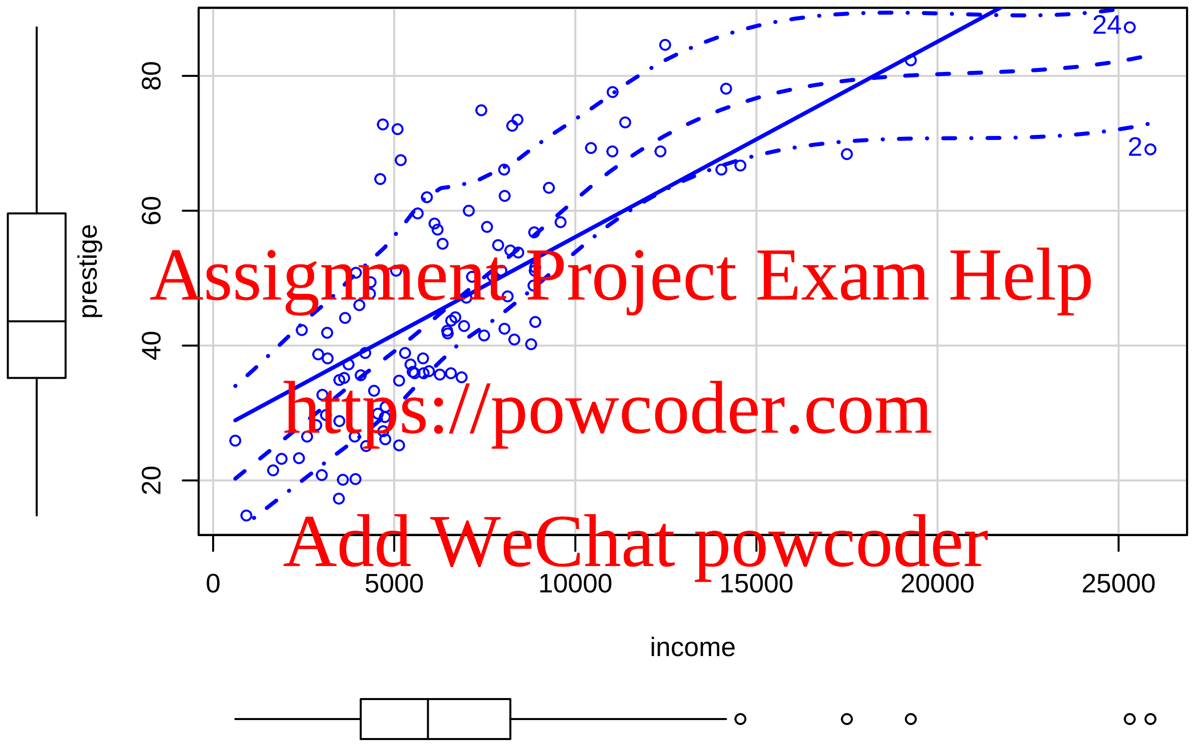
II.1 Scatterplots

```
library(car)
attach(Prestige)
```

```
## The following objects are masked from Prestige (pos = 4):
##
```

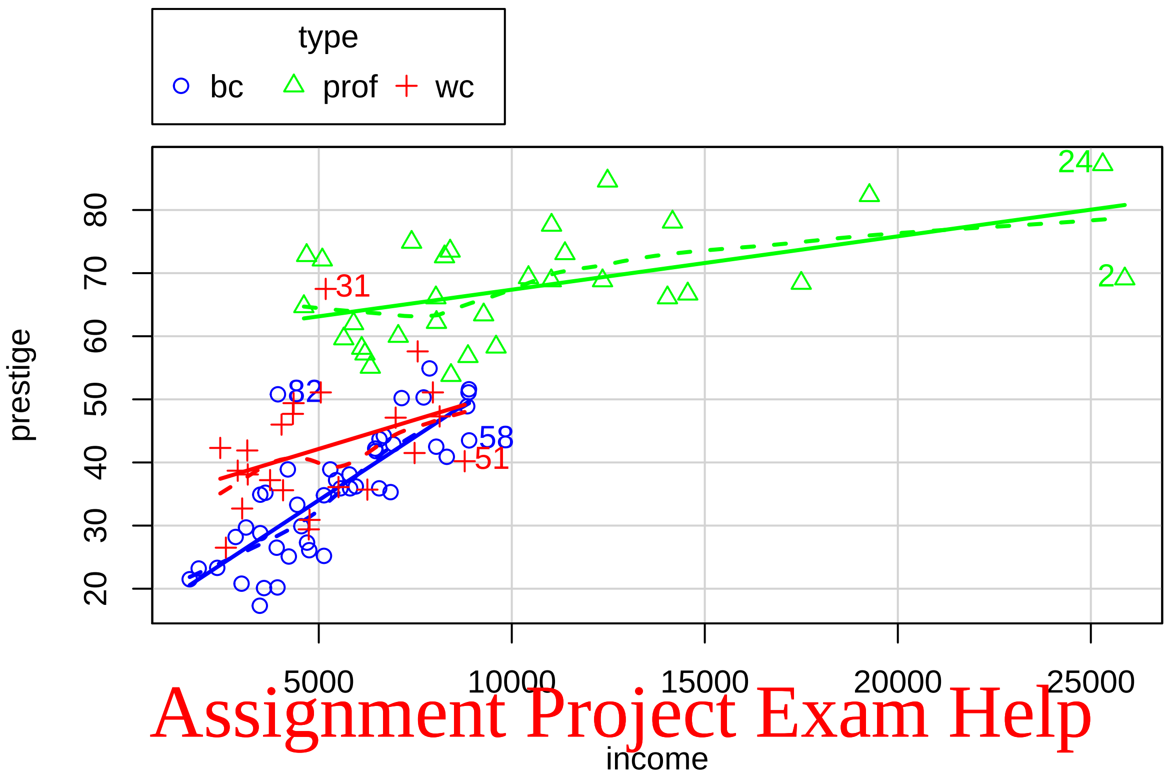
```
##     census, education, income, prestige, type, women

## The following object is masked from package:psych:
##
##     income

## The following objects are masked from Prestige (pos = 7):
##
##     census, education, income, prestige, type, women

## The following object is masked from package:datasets:
##
##     women
```

```r
scatterplot(prestige ~income, lwd=3, id=TRUE)
```




```
## [1]  2 24
```

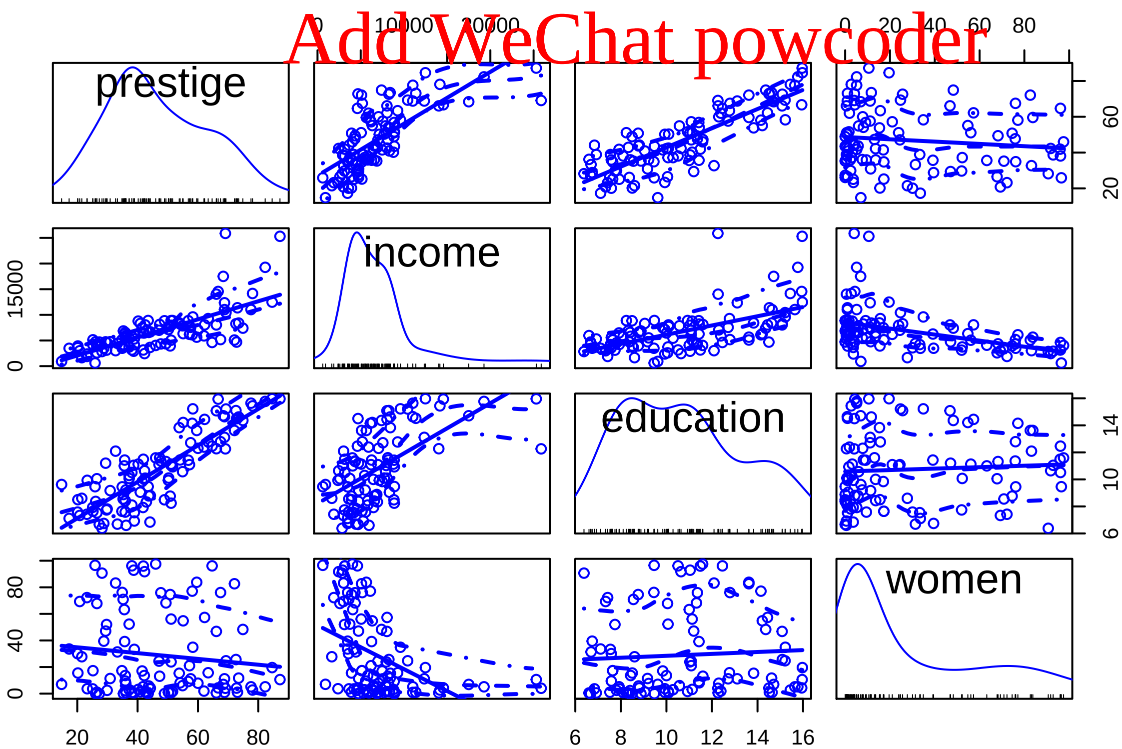
```r
#Note: Lowess smoother is a 'locally weighted smoother'

#We may want to condition on other variables, for exmaple, on type:
scatterplot(prestige ~income|type, lwd=3,col=c("blue", "green", "red"),id=TRUE)
```




```
## 31     58  51     82
##  1   2   3  19  24  25
```



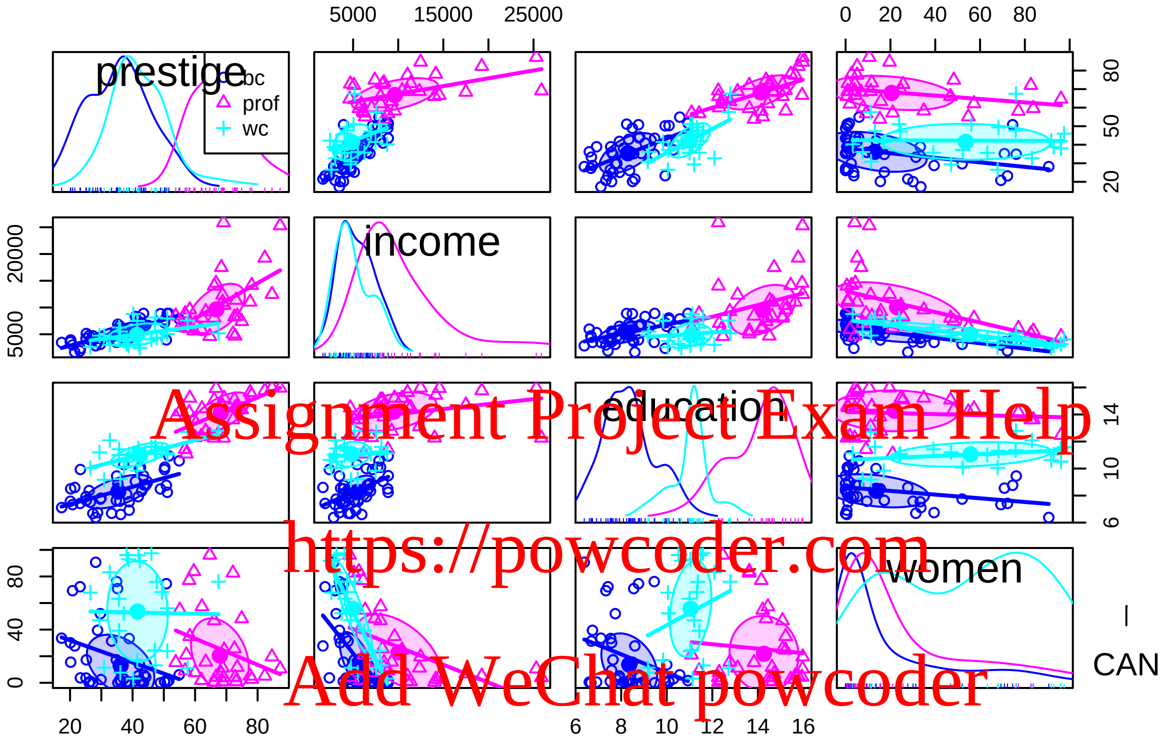
```
#If have sevaral predictors in our model, it is more practical to simply look at as scatterplot matrix o
scatterplotMatrix(~ prestige + income + education + women)
```

```r
#We can also look at all the variables conditioned on type:
scatterplotMatrix(~ prestige + income + education + women | type,smooth=FALSE, ellipse=list(levels=0.5))

#We can also look at 3D visualizations
scatter3d(prestige ~ income + education)
```

```
## Loading required namespace: mgcv
```

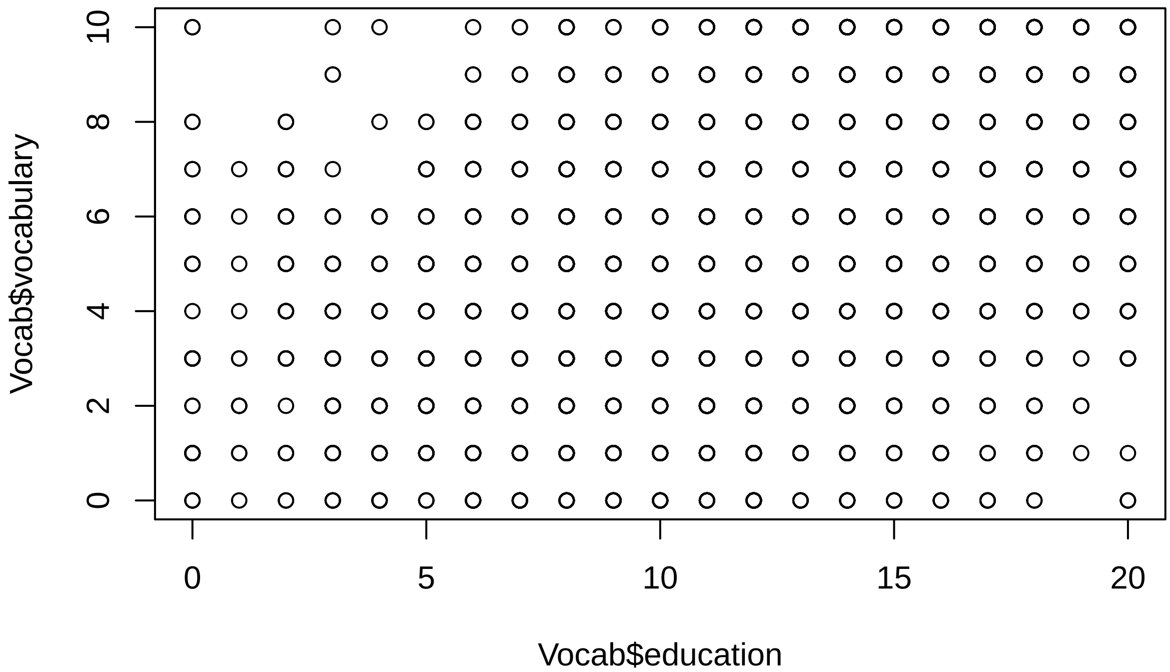
```r
axis(1, at=1:4, labels = names(means))
```



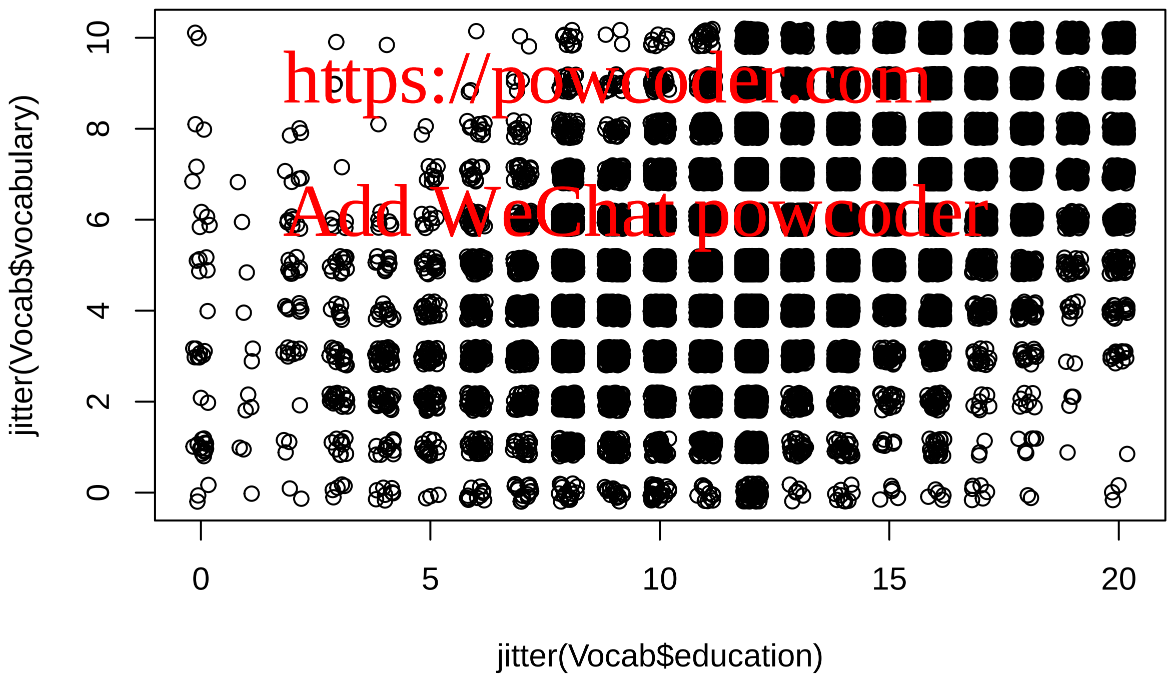
II.2 Jittering Scatterplots

Jittering the data (adding noise) helps separate overplotted observations

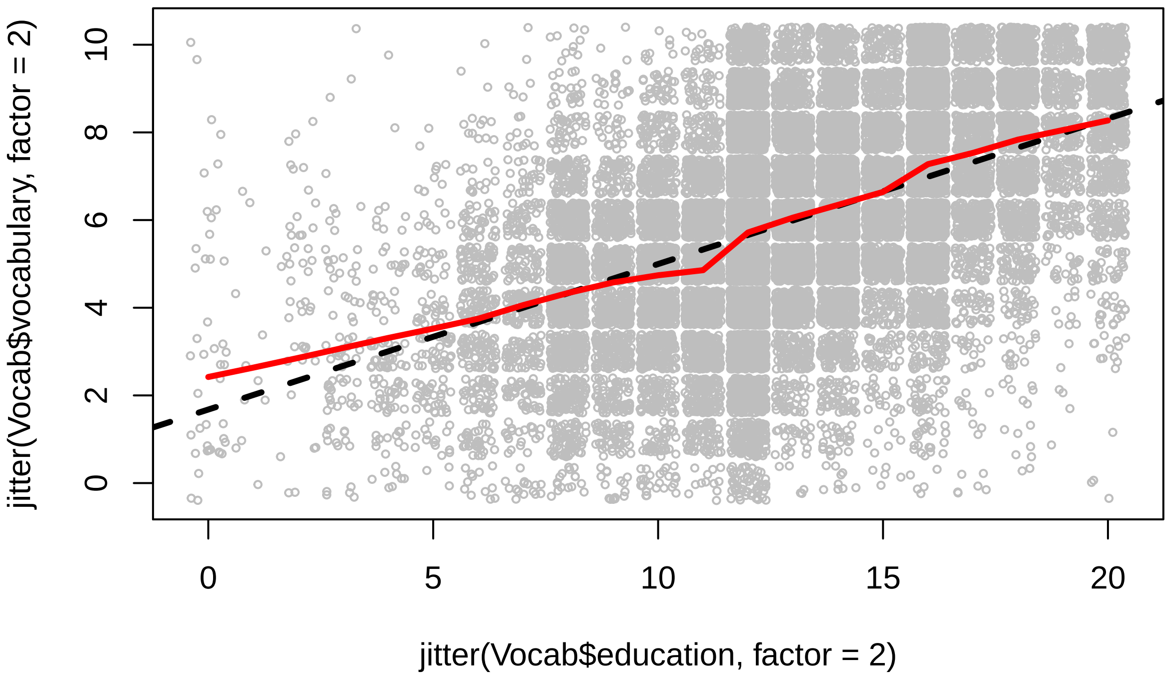
```r
plot(Vocab$vocabulary~Vocab$education)
```




```
#Add some random noise to the variables -> jitter the data
plot(jitter(Vocab$vocabulary)~jitter(Vocab$education))
```


```
# Increse the amount of jitter
plot(jitter(Vocab$vocabulary,factor =2)~jitter(Vocab$education, factor =2),col="gray",cex=0.5)
abline(lm(Vocab$vocabulary~ Vocab$education),lwd= 3, lty="dashed")
lines(lowess(Vocab$education,Vocab$vocabulary,f=0.2), lwd =3, col="red")
```

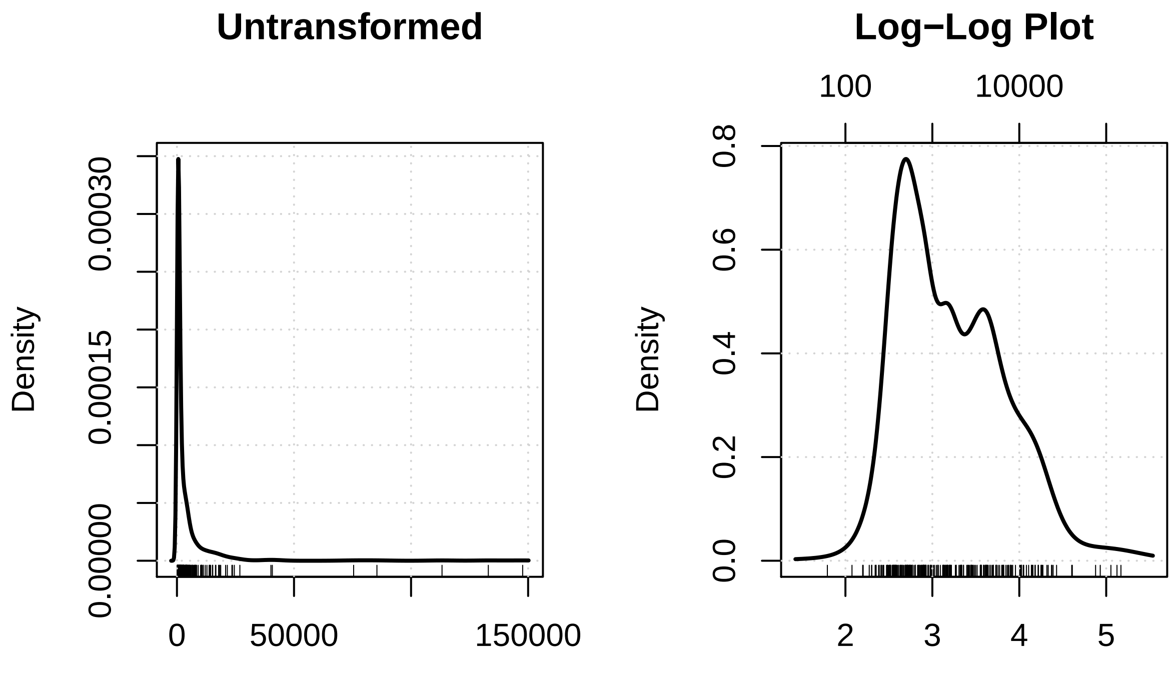


III. Transforming Data

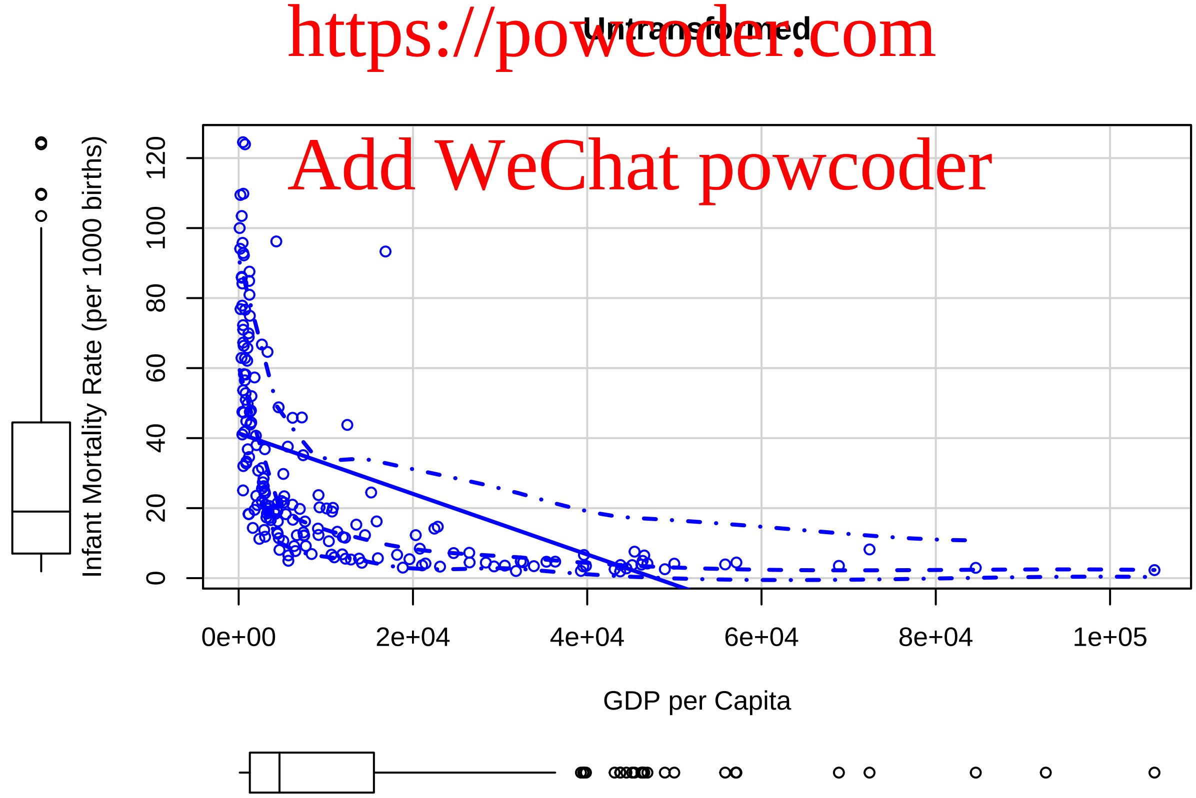
III.1 Logarithms

In econometrics logarithms are very useful when dealing with variables that have span different orders of magnitude yet must be included in the regression model. For example consider a model that includes the predictors GDP (10^{10}) and real interest rates (10^{-2}). Logarithms can help us mitigate violations to the constant variance assumption often used in Least Squares. By default it uses natural logs when you use the `log` function.

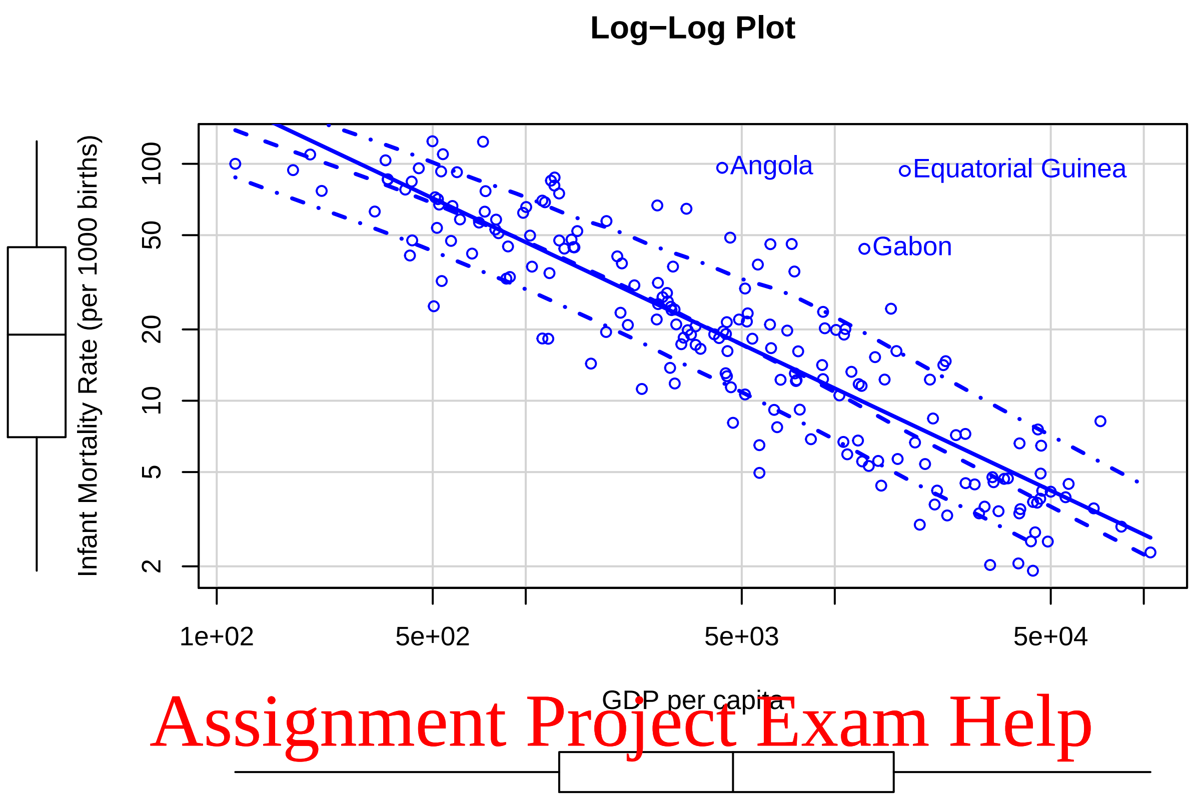

```
# Here we look at the Ornstein data which includes financial assets of 248 Canadian companies.
par(mfrow=c(1, 2), mar=c(5, 4, 6, 2) + 0.1)
densityPlot(~ assets, data=Ornstein, xlab="assets", main="Untransformed")
densityPlot(~ log10(assets), data=Ornstein, adjust=0.65,
    xlab=expression(log[10]~"(assets)"), main="Log-Log Plot")
basicPowerAxis(0, base=10, side="above", at=10^(2:5),
    axis.title="")
```




```
scatterplot(infantMortality ~ ppgdp, data=UN, xlab="GDP per Capita",
    ylab="Infant Mortality Rate (per 1000 births)", main="Untransformed")
```


```
scatterplot(infantMortality ~ ppgdp, data=UN, xlab="GDP per capita",
    ylab="Infant Mortality Rate (per 1000 births)", main="Log-Log Plot",
    log="xy", id=list(n=3))
```




```
##            Angola Equatorial Guinea             Gabon
##                 4                54                62
```

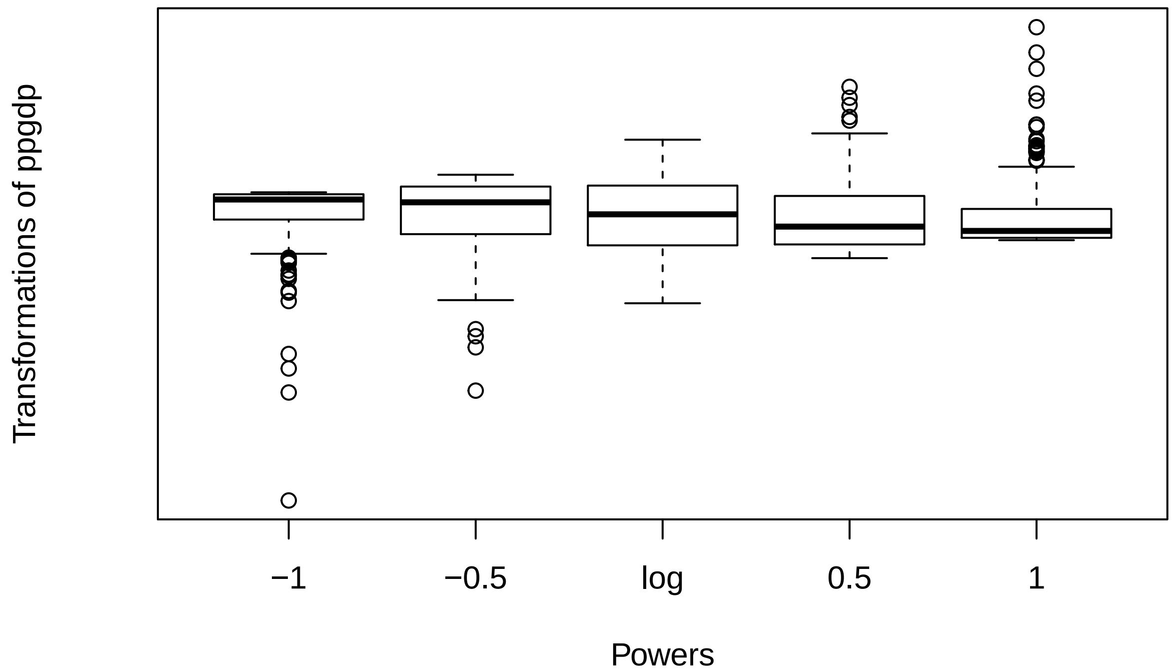

III.2 Power Transformations

Power transformations of the predictors and/or response variable(s) can help improve their respective distributions for modeling/estimation purposes (e.g., Least Squares estimates of regression coefficients) and sampling/robustness (e.g., Bootstrapping and Bayesian inference). A popular transformation is the Box-Cox scale power transformation given by:

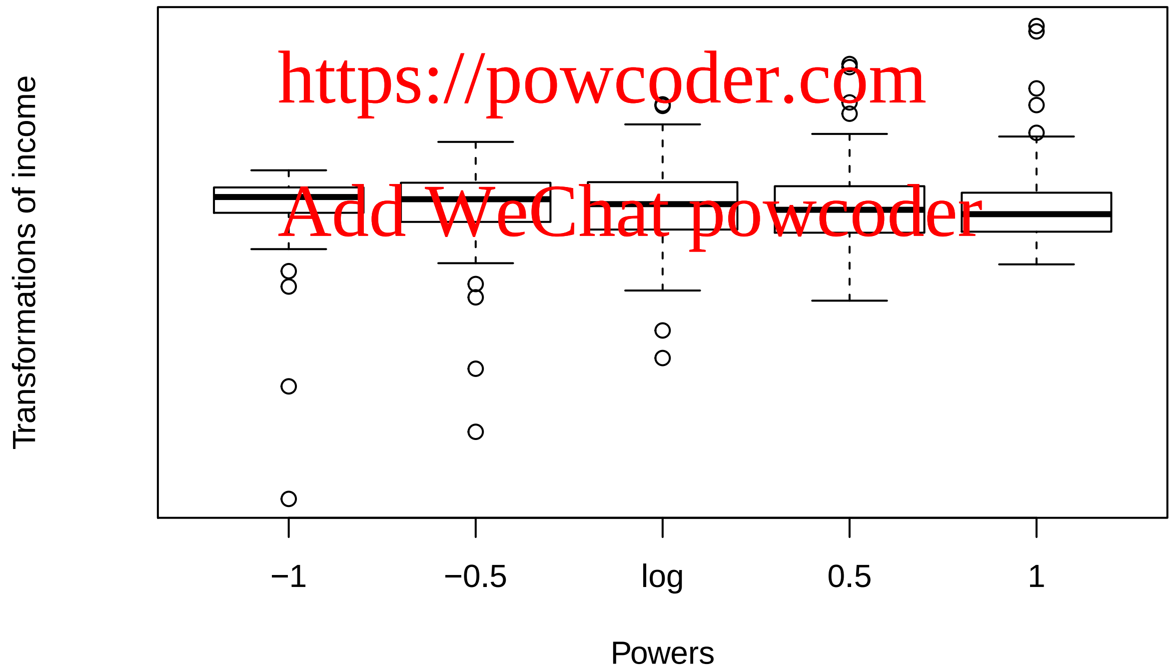
$$T_{BC}(x, \lambda) = x^{(\lambda)} = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{when } \lambda \neq 0; \\ log_e x & \text{when } \lambda = 0. \end{cases}$$

The R function `symbox` automatically produces boxplots of the variable to transform for different values of λ to quickly gauge which power-law is more appropriate. For example, we can use this function on the Per Person GDP (`ppgdp`) as follows:

```
library(car)
symbox(~ppgdp,data=UN)
```




```
# Or we can look at income from the Prestige dataset
symbox(~ income, data=Prestige)
```

Often times our variables take on negative values (e.g., S&P 500 returns), yet transformations such as logs would not work. Instead we can use the Yeo-Johnson transformations $T_{BC}(x+1,\lambda)$ for nonnegative values of x and $T_{BC}(-x+1, 2-\lambda)$ for negative values of x.

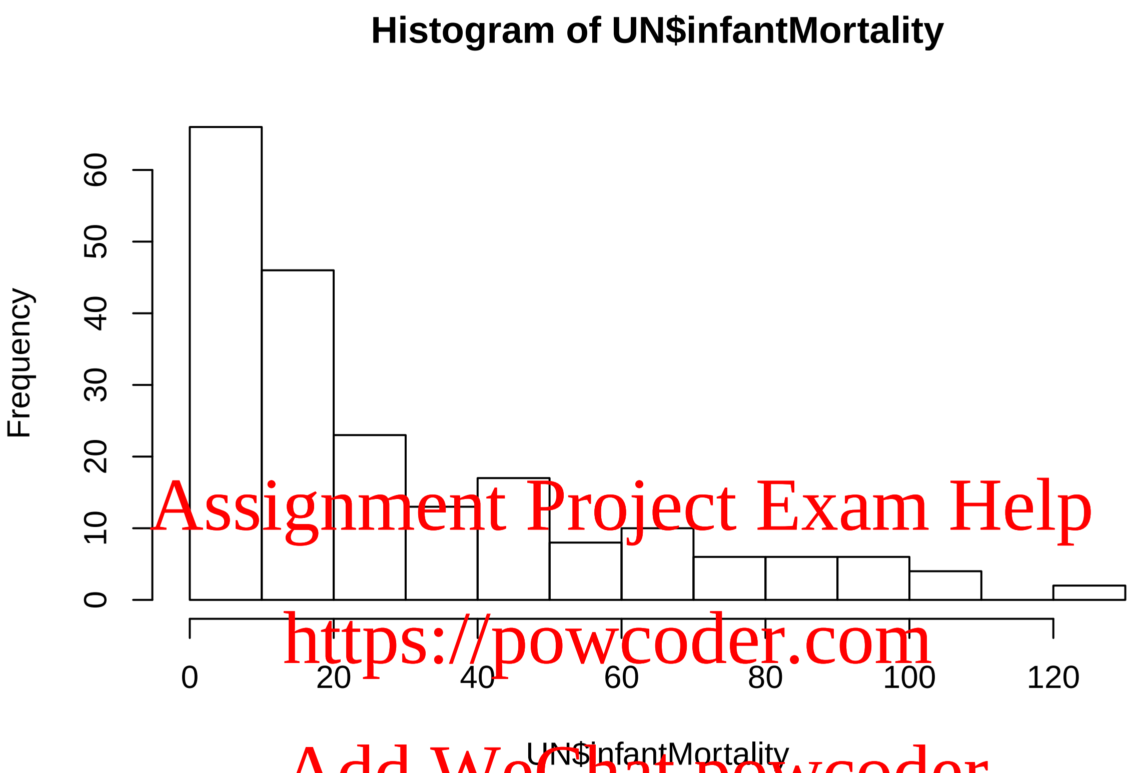
```
# The function yjPower computes takes as inputs your variable (x) and lambda, and outputs the respectiv
x=-5:5
yjPower(x, 3)
```

```
##  [1] -0.8333333 -0.8000000 -0.7500000 -0.6666667 -0.5000000  0.0000000
```

```
## [7]  2.3333333  8.6666667 21.0000000 41.3333333 71.6666667
```

For regression purposes, transformations for normality, linearity and/or constant variance can be easily implemented and tested statistically with the `powerTransform` function. For example, consider the variable infant mortality from the UN dataset.

```
hist(UN$infantMortality)
```




```
# The histogram shows that the distribution is left-skewed. Therefore, we can try and (1) test if a tra
p1 <- powerTransform(infantMortality ~ 1, data=UN, family="bcPower")
summary(p1)
```

```
## bcPower Transformation to Normality
##    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## Y1    0.0468           0      -0.0879       0.1814
##
## Likelihood ratio test that transformation parameter is equal to 0
##  (log transformation)
##                              LRT df    pval
## LR test, lambda = (0) 0.4644634  1 0.49555
##
## Likelihood ratio test that no transformation is needed
##                             LRT df       pval
## LR test, lambda = (1) 172.8143  1 < 2.22e-16
```

```
testTransform(p1, lambda=1/2)
```

```
##                                LRT df       pval
## LR test, lambda = (0.5) 41.95826  1 9.3243e-11
```

In the case of Multiple Regression, we may need to transform more than one variable at the same time (each with its own power-law). For example, we can look the Multivariate Box Cox Highway1 data. The following example if from the bcPower document ion:

```r
# Multivariate Box Cox uses Highway1 data
summary(powerTransform(cbind(len, adt, trks, sigs1) ~ 1, Highway1))
```

```
## bcPower Transformations to Multinormality
##       Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## len      0.1439           0      -0.2728       0.5606
## adt      0.0876           0      -0.1712       0.3464
## trks    -0.6954           0      -1.9046       0.5139
## sigs1   -0.2654           0      -0.5575       0.0267
##
## Likelihood ratio test that transformation parameters are equal to 0
##  (all log transformations)
##                                  LRT df    pval
## LR test, lambda = (0 0 0 0) 6.014218  4 0.19809
##
## Likelihood ratio test that no transformations are needed
##                                  LRT df       pval
## LR test, lambda = (1 1 1 1) 127.7221  4 < 2.22e-16
```

```r
# Multivariate transformation to normality within levels of 'htype'
summary(a3 <- powerTransform(cbind(len, adt, trks, sigs1) ~ htype, Highway1))
```

```
## bcPower Transformations to Multinormality
##       Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## len      0.1451        0.00      -0.2733       0.5636
## adt      0.2396        0.33       0.0255       0.4536
## trks    -0.7336        0.00      -1.9408       0.4735
## sigs1   -0.2959       -0.50      -0.5511      -0.0408
##
## Likelihood ratio test that transformation parameters are equal to 0
##  (all log transformations)
##                                 LRT df    pval
## LR test, lambda = (0 0 0 0) 13.1339  4 0.01064
##
## Likelihood ratio test that no transformations are needed
##                                  LRT df       pval
## LR test, lambda = (1 1 1 1) 140.5853  4 < 2.22e-16
```

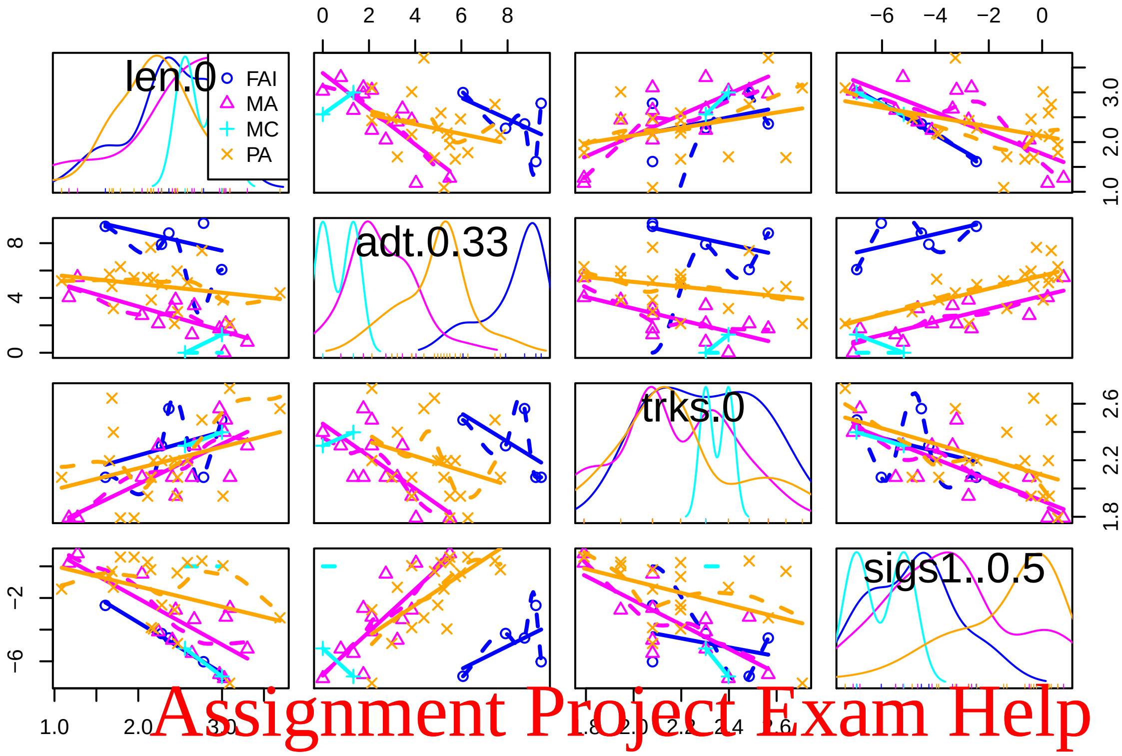


```r
# test lambda = (0 0 0 -1)
testTransform(a3, c(0, 0, 0, -1))
```

```
##                                   LRT df       pval
## LR test, lambda = (0 0 0 -1) 31.12644  4 2.8849e-06
```

```r
# save the rounded transformed values, plot them with a separate
# color for each highway type
transformedY <- bcPower(with(Highway1, cbind(len, adt, trks, sigs1)),
                        coef(a3, round=TRUE))
scatterplotMatrix( ~ transformedY|htype, Highway1)
```

```
## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit smooth
```

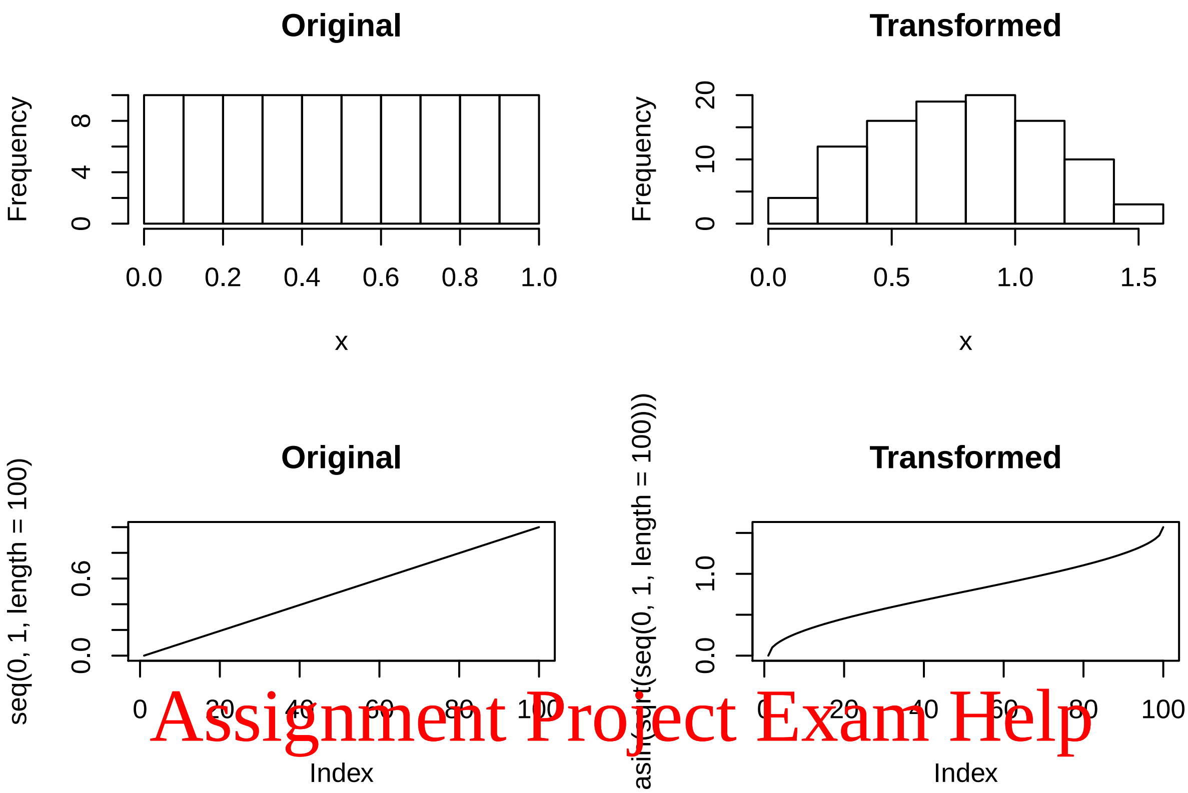


III.3 Transforming Restricted-Range Variables

In regression models such as probit and logistic where the response variable is a probability, their range is restricted to the interval $[0, 1]$, however, this can present problems for such models because values can cluster near the end points. Therefore, transformations that can spread the values such as the *arcsine square-root* are good choices for these cases. This transformation is known for its variance stabilizing properties and is given by:

$$T_{asnisqrt}(x) = \sin^{-1}(\sqrt{x})$$

```
par(mfrow=c(2,2))
hist(seq(0,1,length=100),main="Original", xlab="x")
hist(asin(sqrt(seq(0,1,length=100))),main="Transformed", xlab="x")
plot(seq(0,1,length=100),main="Original",type='l')
plot(asin(sqrt(seq(0,1,length=100))),main="Transformed", type='l')
```



Another popular transformation choice is the *logit* transformation:

$$T_{logit} = \text{logit}(x) = \ln\left(\frac{x}{1-x}\right)$$

```
#library(car)
#attach(Prestige)
#par(mfrow=c(1, 3))
#densityPlot(~women,main="(a) Untransformed")
#densityPlot(~logit(women),main="(b) Logit")
#densityPlot(~asin(sqrt(women/100)),main="(c) Arcsine square-root")
```

III.4 Transformations to Equalize Spread

Consider again the Ornstein dataset showing the number of interlocks by country.

```
Boxplot(interlocks ~ nation, data=Ornstein)
```

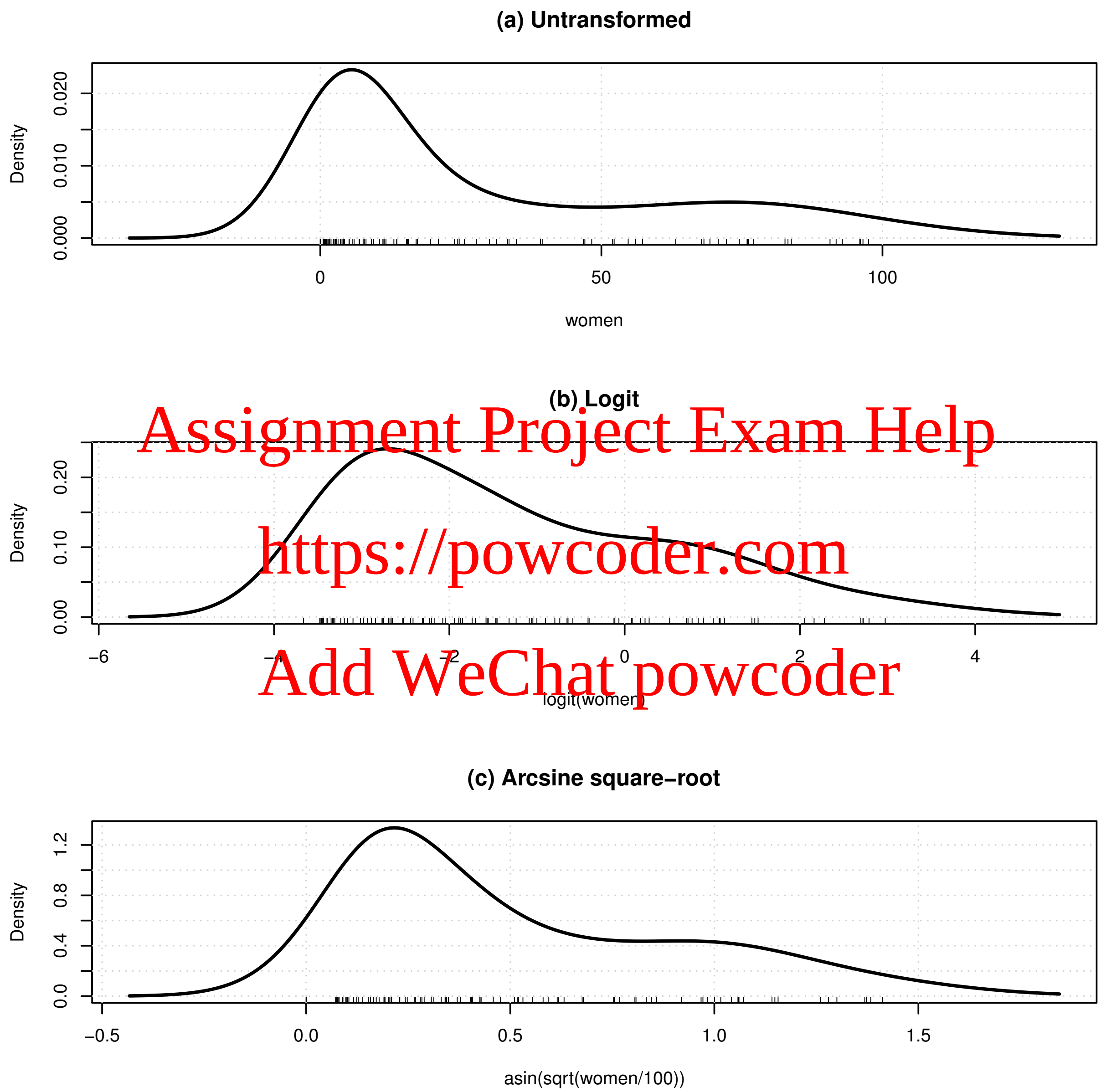
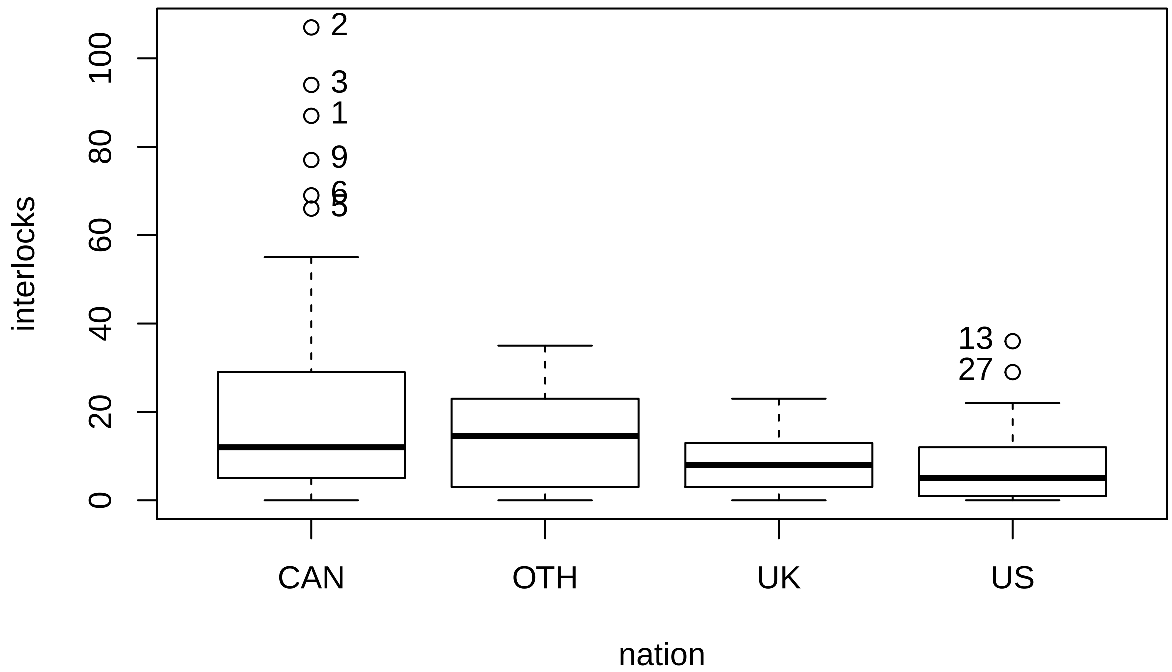


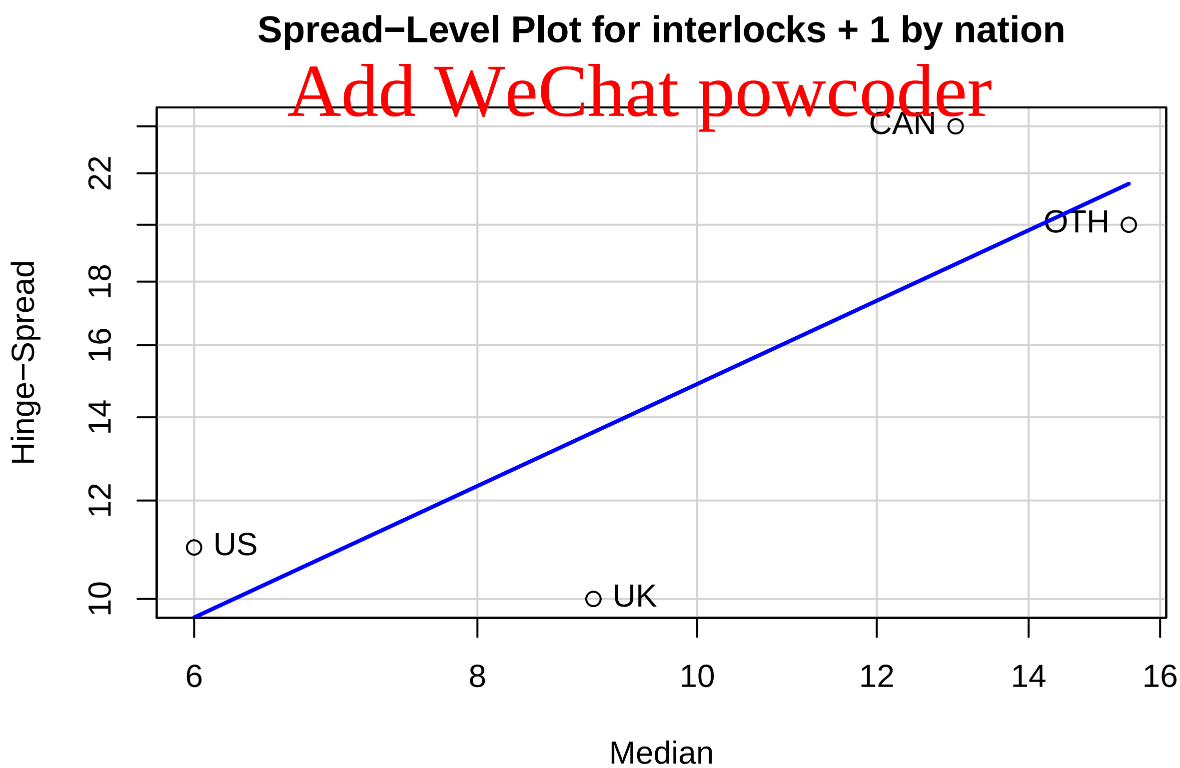
Figure 1: The bulge points in the South-West direction: $p = 2$, $q = 1$




```
## [1] "1"  "2"  "3"  "5"  "6"  "9"  "13" "27"
```


From the figure we can see that it would be helpful to even out the observed spread across countries. A measure of the spread-level can easily be obtain with Tukey's Spread-Level plot which also provides a suggested power-law transformation to stabilize the variance:


```
spreadLevelPlot(interlocks+1 ~ nation, Ornstein)
```

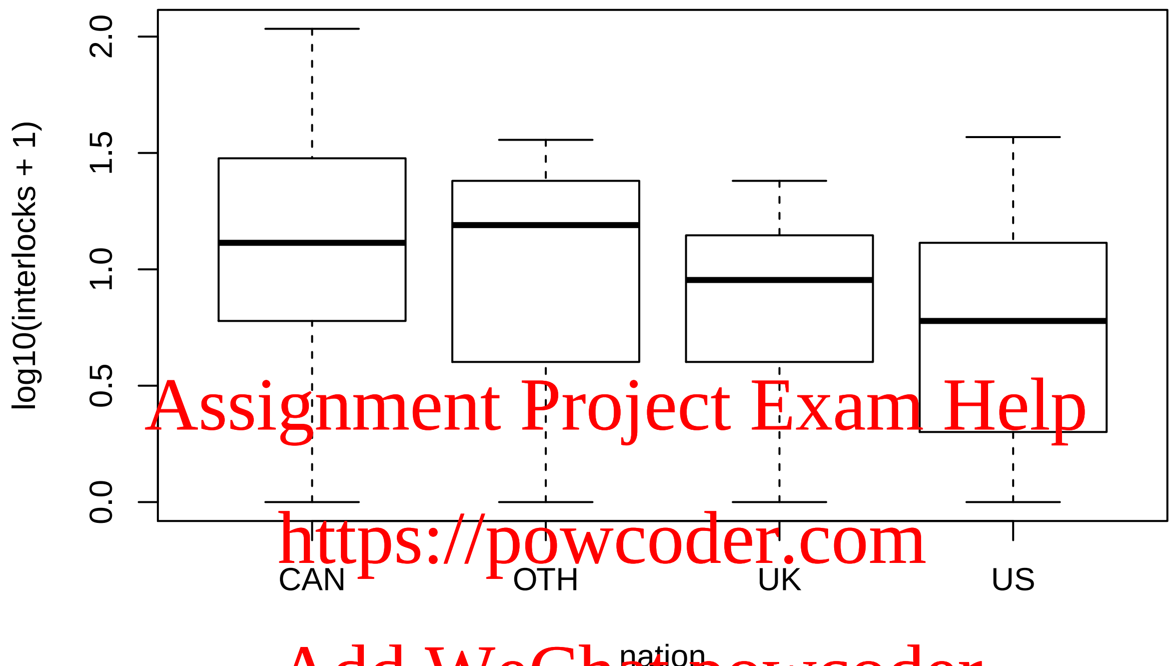



```
##      LowerHinge Median UpperHinge Hinge-Spread
## US            2    6.0         13           11
```

```
## UK            4    9.0          14            10
## CAN           6   13.0          30            24
## OTH           4   15.5          24            20
##
## Suggested power transformation:  0.1534487
```

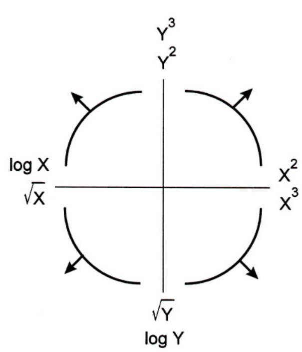
According to the output, the choice of $\lambda = 0.15$ would be an appropriate one. Since this value is close to 0, which would correspond to a log function, we can try a log to see if it works:

```
Boxplot(log10(interlocks+1) ~ nation, data=Ornstein)
```



III.5 Transformations Toward Linearity

We can consider transforming both x and y in an effort to linearize the relationship between. For example, we can find the values of the postie exponents p and q such that the linear regression model $Y_i^q = \beta_0 + \beta_1 X_i^q + \varepsilon$ exhibits a more linear relationship. The choice of these exponents can be guided by Mosteller & Tukey's Bulging Rule for Linearization as shown in the figure below.



For example, below is a visualization of this rule take from (https://dzone.com/articles/tukey-and-mosteller\T1\textquoterights-bulging)

```
fakedataMT=function(p=1,q=1,n=99,s=.1){
  set.seed(1)
  X=seq(1/(n+1),1-1/(n+1),length=n)
  Y=(5+2*X^p+rnorm(n,sd=s))^(1/q)
  return(data.frame(x=X,y=Y))}
par(mfrow=c(2,2))
plot(fakedataMT(p=.5,q=2),main="(p=1/2,q=2)")
plot(fakedataMT(p=3,q=-5),main="(p=3,q=-5)")
plot(fakedataMT(p=.5,q=-1),main="(p=1/2,q=-1)")
plot(fakedataMT(p=3,q=5),main="(p=3,q=5)")
```