More Remedies for Error Problems

Debopriya Ghosh

2019-07-18

R Libraries

```
library("printr")
library("faraway")
library("car")
library("MASS")
library("rgl")
library("MASS")
library("quantreg")
library("robustbase")
```

Remedy for correlated errors: Generalized Leats Squares

- When errors are not independent or have non-zero correlation, OLS is not suitable
- Generalized Least Squares (GLS) account for dependent errors
- The errors have to obey special rules, but these cover a wide array of circumstances
- Valid inferences still require that errors are normally distributed
- Example: Using *longley* data

```
data(longley)
help(longley)
```

longley R Documentation

Longley's Economic Regression Data

Description

A macroeconomic data set which provides a well-known example for a highly collinear regression.

Usage

```
longley
```

Format

A data frame with 7 economical variables, observed yearly from 1947 to 1962 (*n=16*).

`GNP.deflator`

GNP implicit price deflator (*1954=100*)

`GNP`

Gross National Product.

`Unemployed`

number of unemployed.

`Armed.Forces`

number of people in the armed forces.

`Population`

'noninstitutionalized' population ≥ 14 years of age.

`Year`

the year (time).

`Employed`

number of people employed.

The regression `lm(Employed ~ .)` is known to be highly collinear.

Source

J. W. Longley (1967) An appraisal of least-squares programs from the point of view of the user. *Journal of the American Statistical Association* **62**, 819–841.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

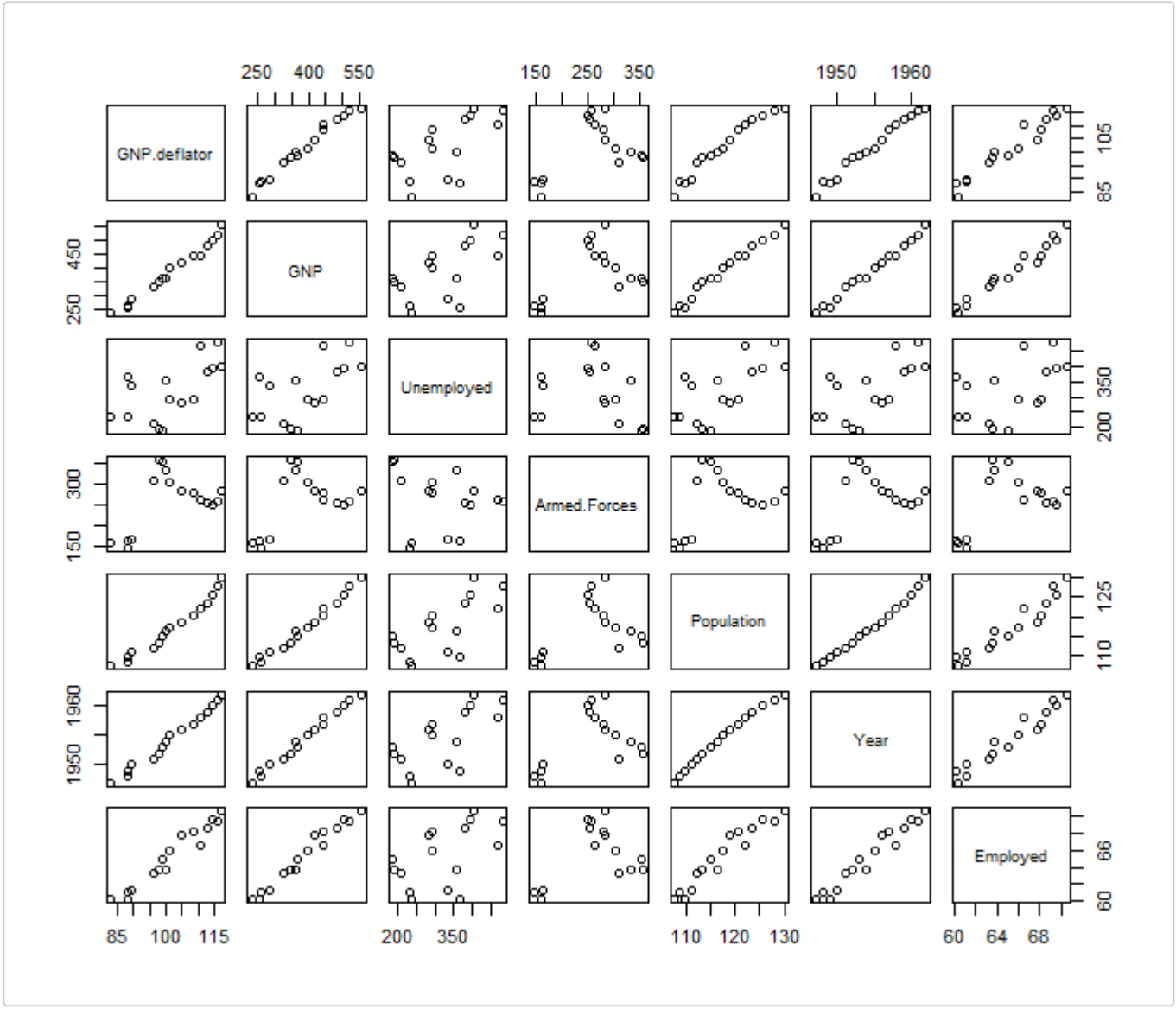
Examples

```
require(stats); require(graphics)
## give the data set in the form it is used in S-PLUS:
longley.x <- data.matrix(longley[, 1:6])
longley.y <- longley[, "Employed"]
pairs(longley, main = "longley data")
summary(fm1 <- lm(Employed ~ ., data = longley))
opar <- par(mfrow = c(2, 2), oma = c(0, 0, 1.1, 0),
            mar = c(4.1, 4.1, 2.1, 1.1))
plot(fm1)
par(opar)
```

Plotting the data

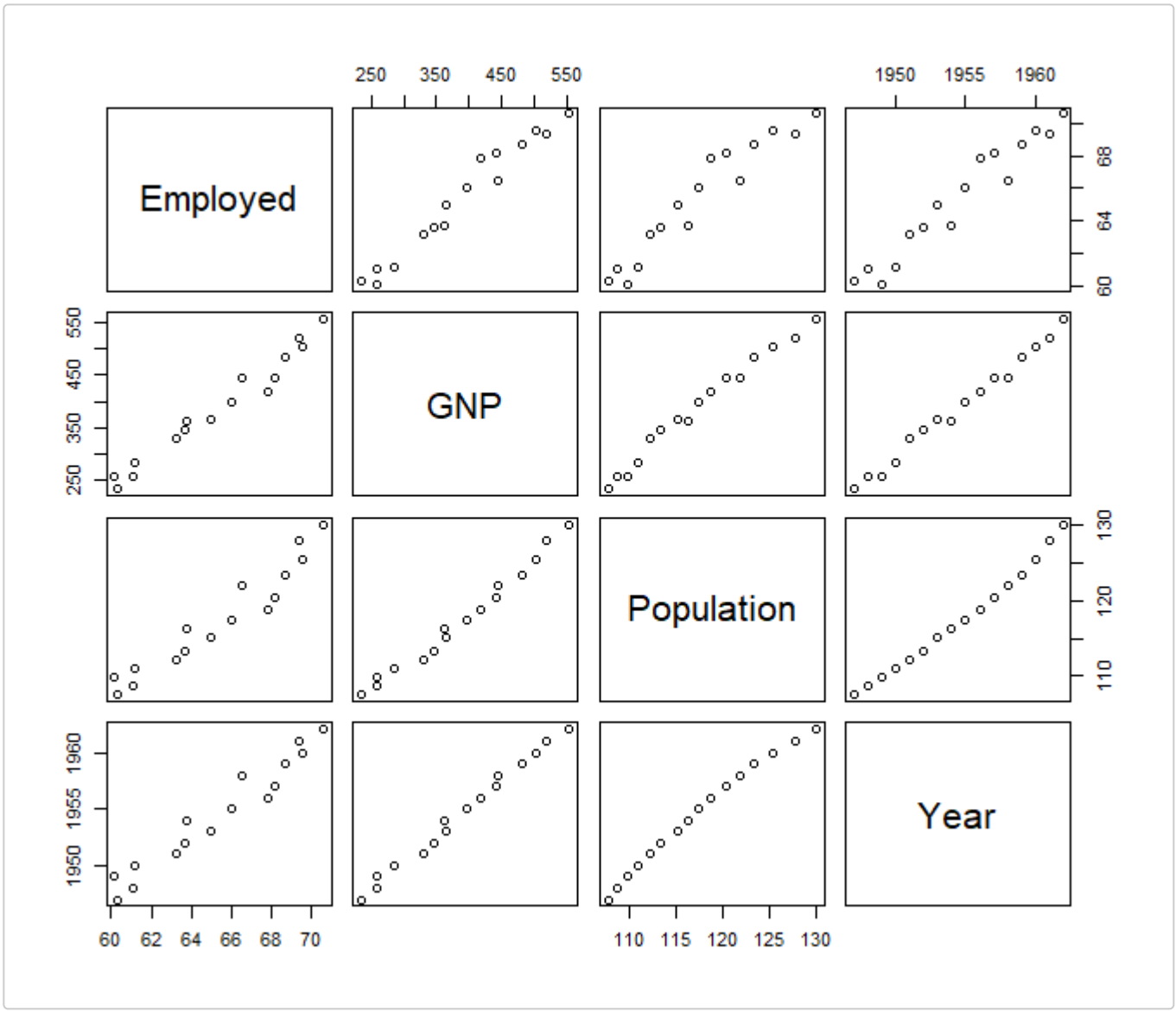
- The aim is to build a model for *Employed* with just the predictors *GNP* and *Population*
- We will also use *Year* as predictor, but we will use it in an indirect way

```
plot(longley)
```



- We can narrow down the set of predictors
- We can use the `pairs()` function as follows

```
pairs(~Employed+GNP+Population+Year, longley)
```



- It appears that all variables are highly correlated with each other.
- Consider the times series *Employed* as a response variable, it appears to have a simple linear dependence on *Year*,
  - but it would be a mistake to think *Employed* has a simple statistical linear relation with *Year*.
  - Rather, *Employed* is a time series, and the value for one year depends on the values for the prior years.
- Can we fit a statistical linear model for the response *Employed* with *GNP* and *Population* as predictors in spite of the collinearity?
- The answer will be yes, but we must also fit a time series model for the errors.
- Before we discuss a time series model for the errors, let's introduce some tools used in time series modelling.

The Auto Correlation Function (ACF)

- The *Auto Correlation Function* is an array of correlations of a time series with itself for k lagged value of time, as $k = 1, \ldots, n-2$
- For example, for $k = 1$, a lag(1) auto correlation is

$$\hat{\rho}_1 = cor\left(y[-1], y[-n]\right)$$

where $y[-1] = \{y_t, t = 2, \ldots n\}$ and $y[-n] = \{y_t, t = 1, \ldots n-1\}$

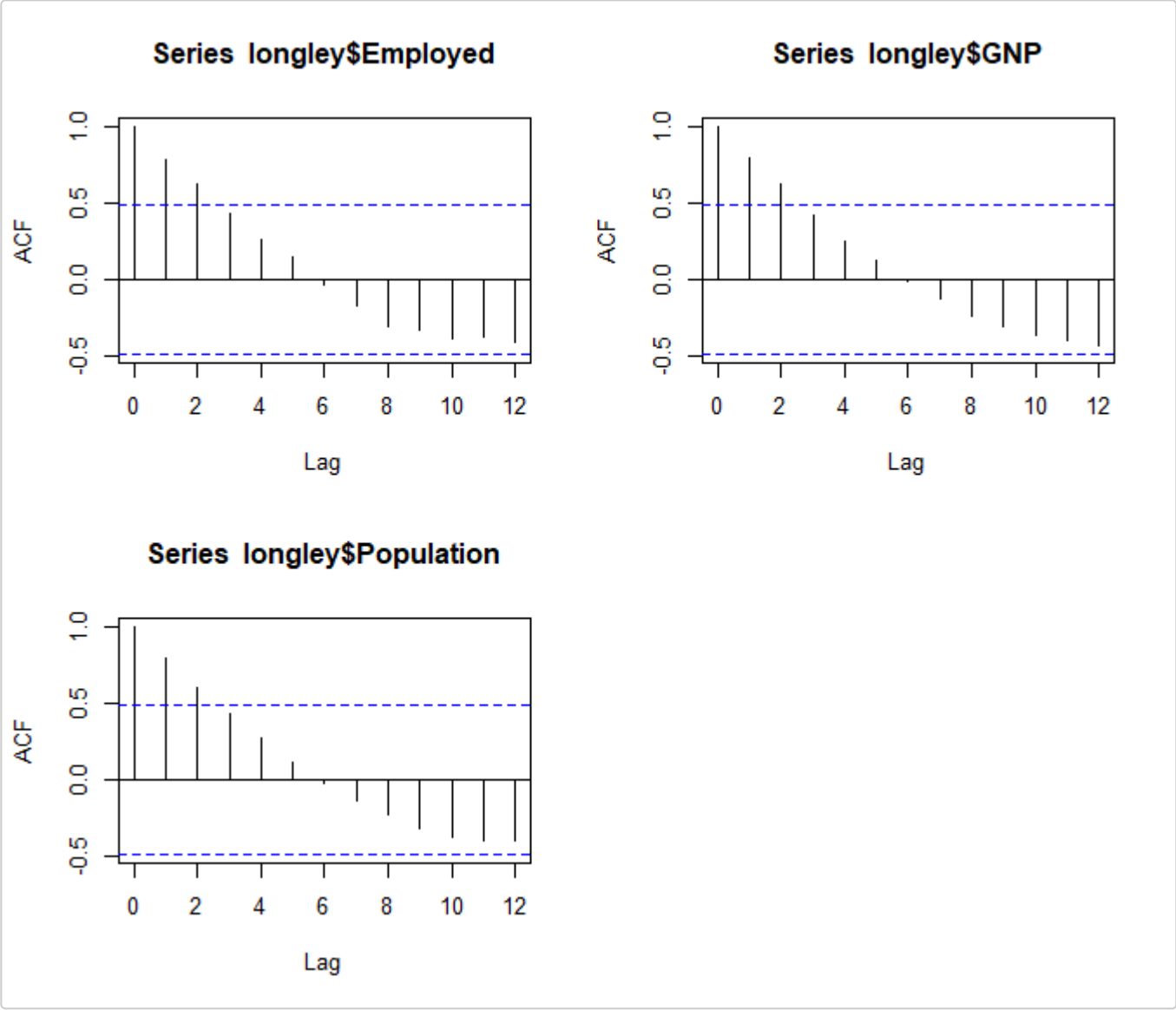
- Define $\hat{\rho}_2, \hat{\rho}_3, \ldots, \hat{\rho}_{n-2}$ similarly
- For example, the lag(1) auto correlation of *Employed* is obtained as follows:

```
with(longley, cor(Employed[-1],Employed[-16]))
```

```
## [1] 0.9315494
```

- A graph of the ACF along with 95% CI's for each $\hat{\rho}_k$ is useful for understanding the structure of a times series
- We graph the ACF's of all the variables

```
par(mfrow=c(2,2))
acf(longley$Employed)
acf(longley$GNP)
acf(longley$Population)
par(mfrow=c(1,1))
```

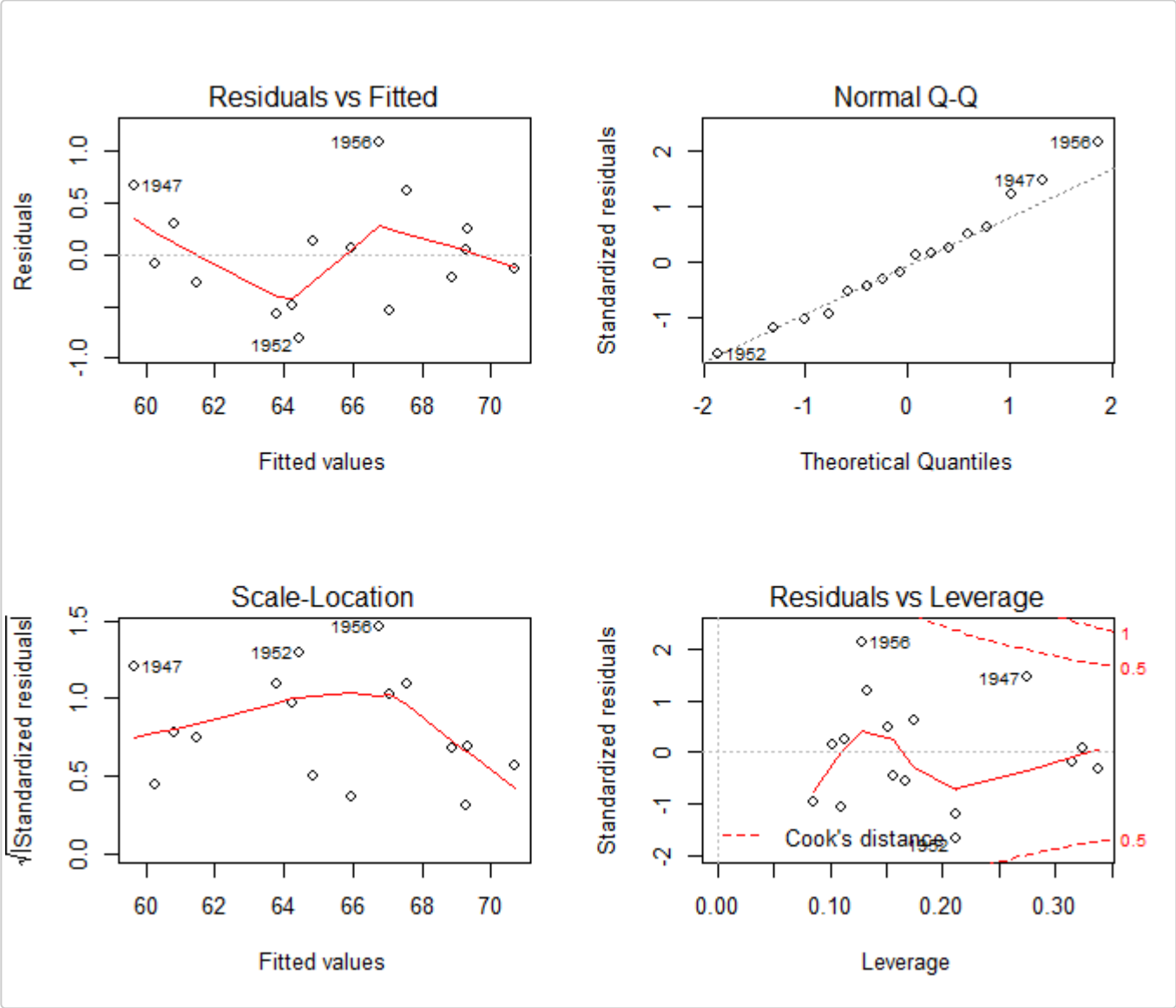


- In the ACF of *Employed*, the first two lagged auto correlations are statistically different from zero
- It appears that both *GNP* and *Population* have the same auto correlation structure as does *Employed*
- This is because all three variables have a strong linear dependence on *Year*

OLS Fit

- Suppose we fit a linear model in *GNP* and *Population*

```
g = lm(Employed~GNP+Population, longley)
summary(g)
shapiro.test(residuals(g))
car::durbinWatsonTest(residuals(g))
car::vif(g)
par(mfrow=c(2,2))
plot(g)
```



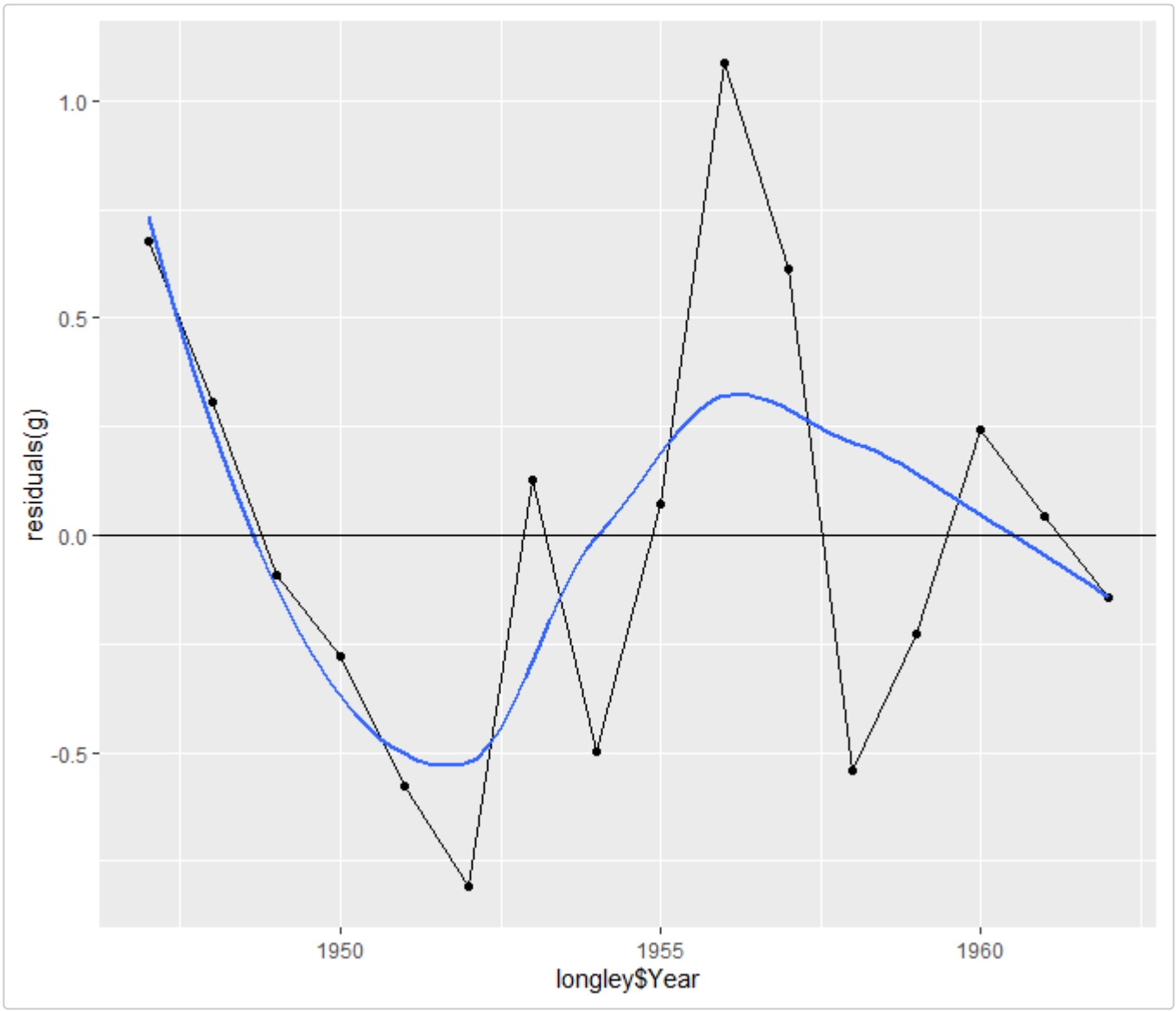

```
par(mfrow=c(1,1))
```

```
##
## Call:
## lm(formula = Employed ~ GNP + Population, data = longley)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.80899 -0.33282 -0.02329  0.25895  1.08800
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 88.93880   13.78503   6.452 2.16e-05 ***
## GNP          0.06317    0.01065   5.933 4.96e-05 ***
## Population  -0.40974    0.15214  -2.693   0.0184 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5459 on 13 degrees of freedom
```

```
## Multiple R-squared:  0.9791, Adjusted R-squared:  0.9758
## F-statistic: 303.9 on 2 and 13 DF,  p-value: 1.221e-11
##
##
##  Shapiro-Wilk normality test
##
## data:  residuals(g)
## W = 0.97515, p-value = 0.9135
##
## [1] 1.301484
##        GNP Population
##   56.36828   56.36828
```

- In the four anomaly plots, we see that there is a sinusoidal pattern in the residuals against the fitted values, which is a clear indication that the residuals are not independent
- Let's plot the residuals against *Year*

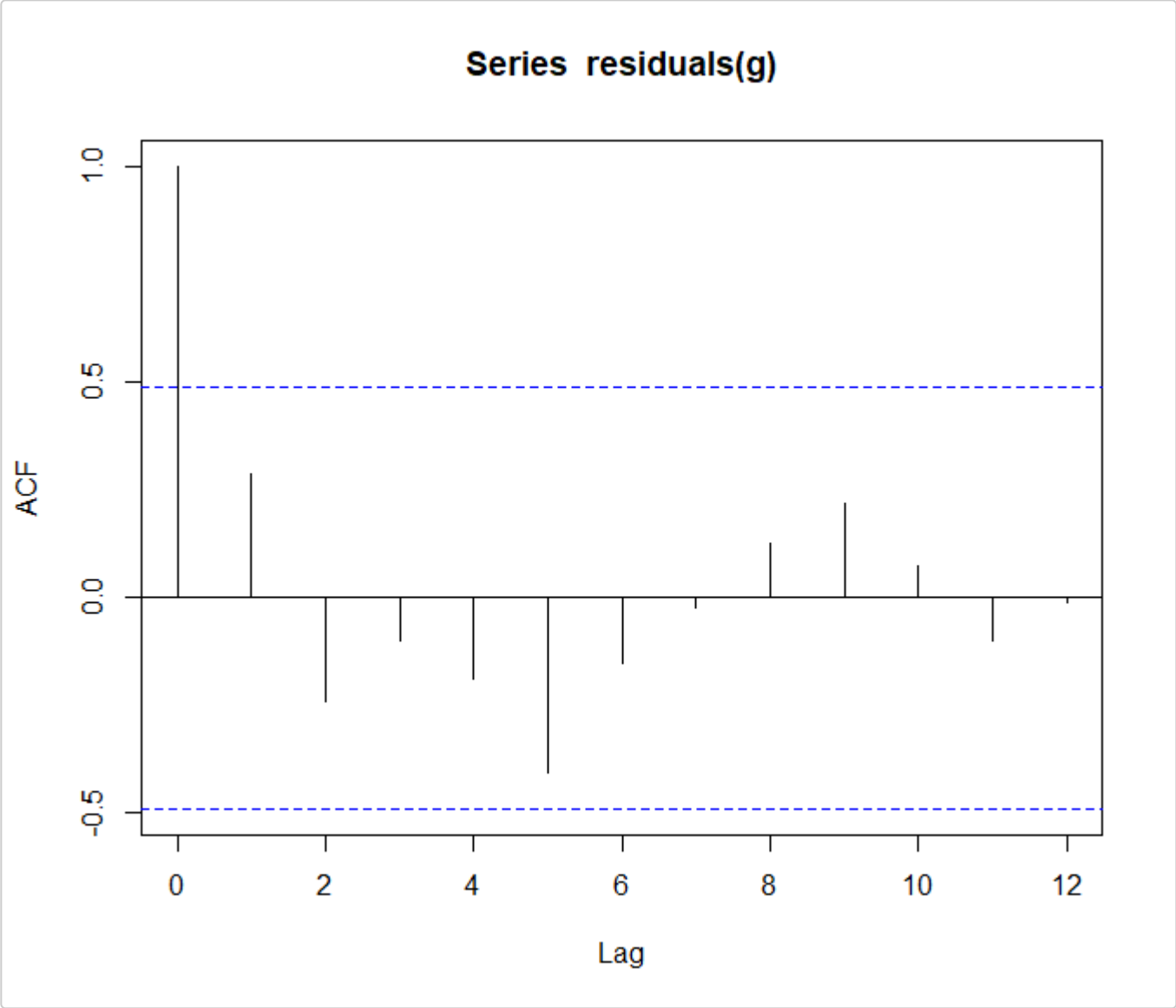
```
library(ggplot2)
qplot(longley$Year, residuals(g)) +
  geom_line() +
  geom_smooth(se=F) +
  geom_hline(yintercept=0)
```



AR(1) Timeseries Model

- Let's have a look at the ACF of the residuals

```
acf(residuals(g))
```



- We see that the lag(1) auto correlation of residuals is almost significant even after controlling for *GNP* and *Population*
- A time series with a strong lag(1) auto correlation can be modeled with an AR(1) *autoregression model*:

$$\epsilon_t = \phi\epsilon_{t-1} + \delta_t$$

$$\delta_t \sim N\left(0, \tau^2\right)$$

$$t = 1, \ldots n$$

GLS Fit using Maximum Likelihood

- A maximum likelihood procedure for fitting a linear model with normal, but correlated, errors following an auto regression model is easily obtained
- It is available in the R Package *nlme* "Linear and Nonlinear Mixed Effects Models"

```
library(nlme)
g1 = gls(Employed~GNP + Population,
```

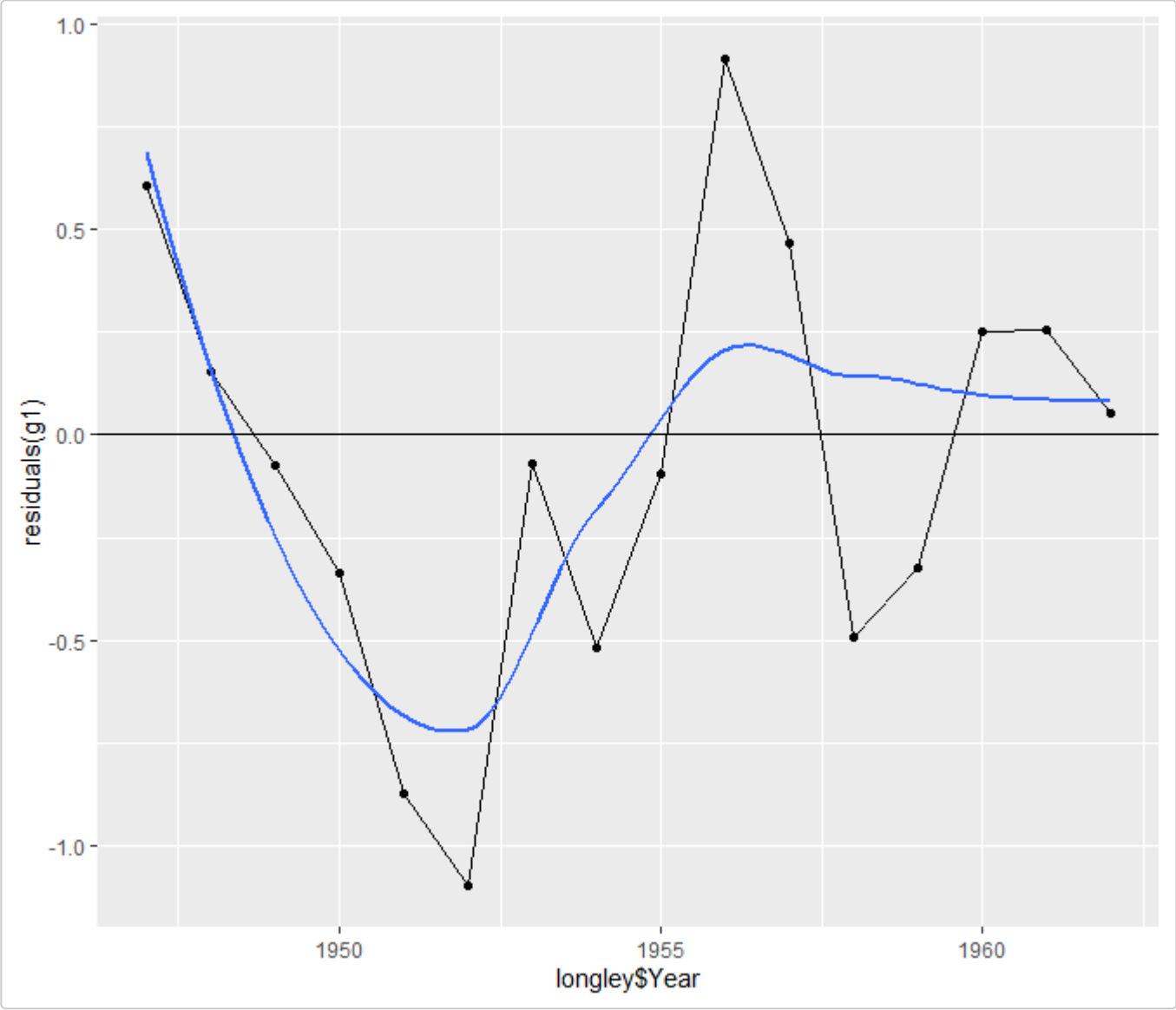
```
          correlation=corAR1(form= ~Year),
          data=longley)
summary(g1)
shapiro.test(residuals(g1))
intervals(g1)
compareCoefs(g, g1)
```

```
## Generalized least squares fit by REML
##   Model: Employed ~ GNP + Population
##   Data: longley
##        AIC      BIC    logLik
##   44.66377 47.48852 -17.33188
##
## Correlation Structure: AR(1)
##  Formula: ~Year
##  Parameter estimate(s):
##       Phi
## 0.6441692
##
## Coefficients:
##                Value Std.Error   t-value p-value
## (Intercept) 101.85813 14.198932  7.173647  0.0000
## GNP           0.07207  0.010606  6.795485  0.0000
## Population   -0.54851  0.154130 -3.558778  0.0035
##
##  Correlation:
##            (Intr) GNP
## GNP         0.943
## Population -0.997 -0.966
##
## Standardized residuals:
##        Min         Q1        Med         Q3        Max
## -1.5924564 -0.5447822 -0.1055401  0.3639202  1.3281898
##
## Residual standard error: 0.689207
## Degrees of freedom: 16 total; 13 residual
##
##  Shapiro-Wilk normality test
##
## data:  residuals(g1)
## W = 0.98927, p-value = 0.9988
##
## Approximate 95% confidence intervals
##
##  Coefficients:
##                   lower         est.       upper
## (Intercept) 71.18320460 101.85813305 132.5330615
## GNP          0.04915865   0.07207088   0.0949831
## Population  -0.88149053  -0.54851350  -0.2155365
## attr(,"label")
## [1] "Coefficients:"
##
##  Correlation structure:
```

```
##          lower      est.     upper
## Phi -0.4432383 0.6441692 0.9645041
## attr(,"label")
## [1] "Correlation structure:"
##
##  Residual standard error:
##     lower      est.     upper
## 0.2477527 0.6892070 1.9172599
## Calls:
## 1: lm(formula = Employed ~ GNP + Population, data = longley)
## 2: gls(model = Employed ~ GNP + Population, data = longley,
##   correlation = corAR1(form = ~Year))
##
##             Model 1 Model 2
## (Intercept)    88.9   101.9
## SE             13.8    14.2
##
## GNP          0.0632  0.0721
## SE           0.0106  0.0106
##
## Population   -0.410  -0.549
## SE            0.152   0.154
##
```

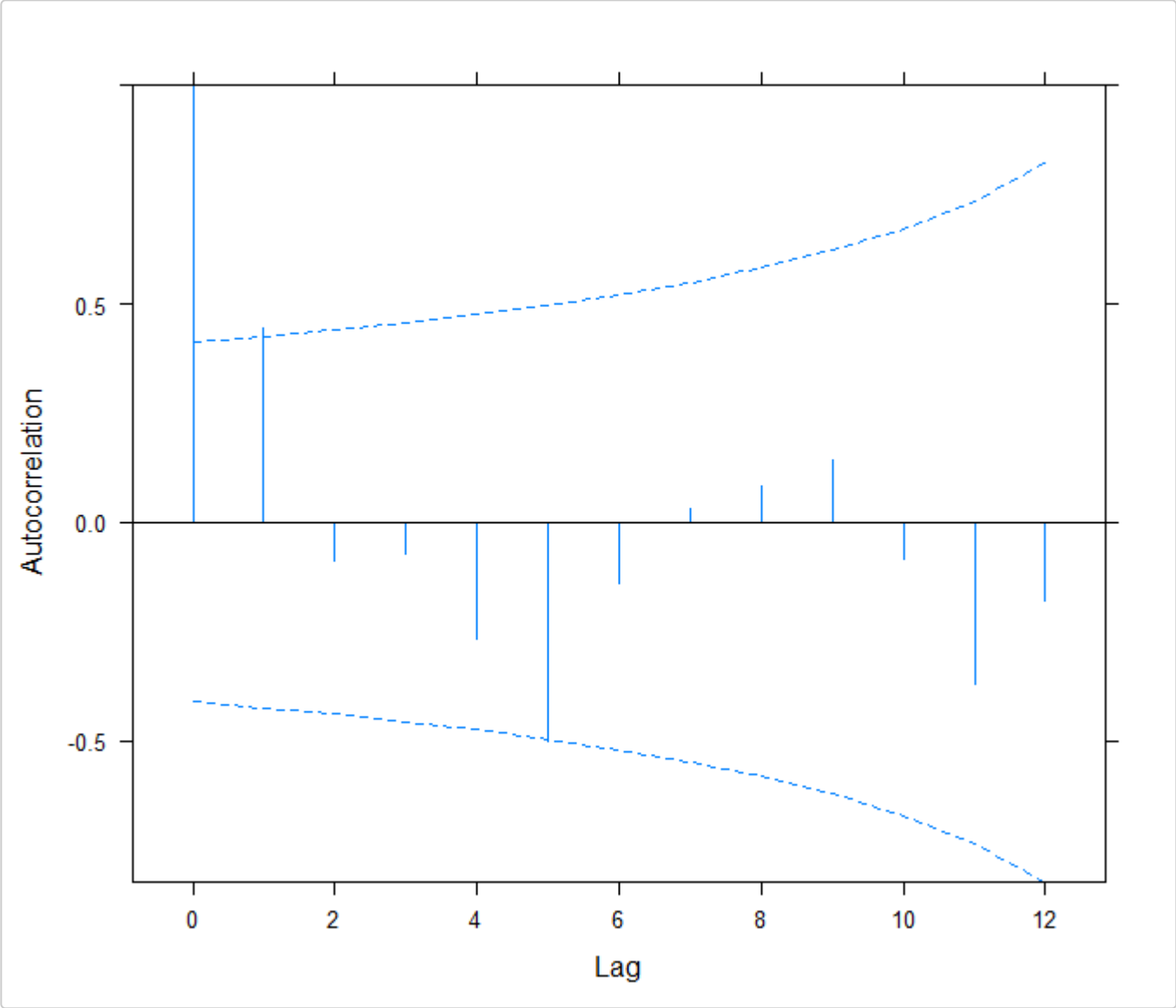
- We see that the GLS estimated model is different from OLS model
- A plot of the GLS residuals against *Year* reveals the same pattern as OLS residuals

```
qplot(longley$Year, residuals(g1)) +
  geom_line() +
  geom_smooth(se=F) +
  geom_hline(yintercept=0)
```



- A plot of the ACF of the GLS residuals reveals the same structure as the OLS residuals
- Note the R nlme package, the function *ACF()* is different than the standard *acf()* function and needs to be plotted

```
plot(ACF(g1), alpha = 0.10)
```



Remedy for non-constant error variance: Weighted Least Squares

- When the errors are uncorrelated but have non constant variance, we can use *weighted least squares*.
- Errors must be normal.
- The WLS estimates of the coefficients with model (design) matrix $\boldsymbol{X}$ is readily obtained by modifying the OLS method:

$$\sum_{i=1}^{n} w_i (y_i - \mu_i)^2 = (\boldsymbol{y} - \boldsymbol{\mu})^T \boldsymbol{W} (\boldsymbol{y} - \boldsymbol{\mu})$$

$$\boldsymbol{W} = \begin{pmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & w_n \end{pmatrix}$$

$$\boldsymbol{\mu} = \boldsymbol{X}\boldsymbol{\beta}$$

$$\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{X}^T\boldsymbol{W}\boldsymbol{y}$$

$$\hat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{y}$$

Example: 1981 French Presedential Election regression data

- We will demonstrate with the *fpe* data in the *faraway* package

```
data(fpe)
?fpe
```

fpe R Documentation

1981 French Presidential Election

Description

Elections for the French presidency proceed in two rounds. In 1981, there were 10 candidates in the first round. The top two candidates then went on to the second round, which was won by Francois Mitterand over Valery Giscard-d'Estaing. The losers in the first round can gain political favors by urging their supporters to vote for one of the two finalists. Since voting is private, we cannot know how these votes were transferred, we might hope to infer from the published vote totals how this might have happened. Data is given for vote totals in every fourth department of France:

Usage

```
data(fpe)
```

Format

This dataframe contains the following columns (vote totals are in thousands)

EI

Electeur Inscrits (registered voters)

A

Voters for Mitterand in the first round

B

Voters for Giscard in the first round

C

Voters for Chirac in the first round

D

Voters for Communists in the first round

E

Voters for Ecology party in the first round

F

Voters for party F in the first round

G

Voters for party G in the first round

H

Voters for party H in the first round

I

Voters for party I in the first round

J

Voters for party J in the first round

K

Voters for party K in the first round

A2

Voters for Mitterand in the second round

B2

Voters for party Giscard in the second round

N

Difference between the number of voters in the second round and in the first round

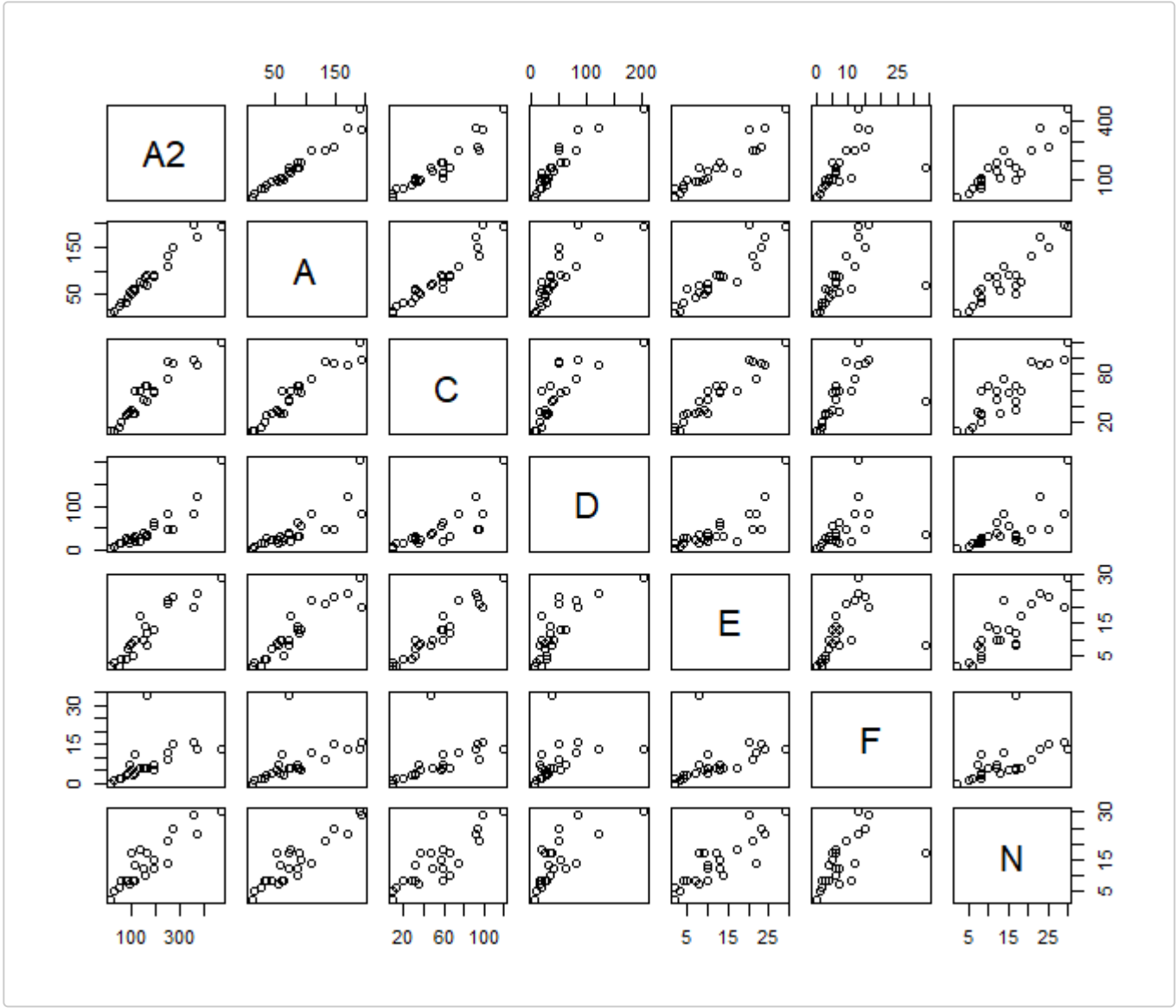
Source

"The Teaching of Practical Statistics" by C.W. Anderson and R.M. Loynes, Wiley,1987

Modeling the data

- Here we want to build a model to predict the number of votes the leading candidate (Mitterrand) gets in the second round
- A matrix scatterplot with $A2$ votes for Mitterrand in the second round as the output:

```
pairs(~A2+A+C+D+E+F+N, fpe)
```



- The unit of measurement is the French voting district
- We do not want to include the intercept in the model because we already have a good idea what it is
- Think of the variables as the market share for each candidate, so one candidate's loss is another candidate's gain
- In the second round of voting, the sum of Mitterrand's and Gescard's votes $A2 + B2$ is the number that voted in the second round on a per district basis
- And in the first round, the sum of all candidates votes is the number voting in the first round on a per district basis, $A + B + C + D + E + F + G + H + J + K$
- But N is the difference between the number voting in the second round and in the first round, so

$$N = (A2 + B2) - (A + B + C + D + E + F + G + H + J + K)$$

- Therefore

$$A2 = B2 - (A + B + C + D + E + F + G + H + J + K) - N$$

- Now we

- spread Gescard's second round vote $B2$ over those voting for him in the first round
- plus those who switched from Mitterrand
- plus those he picked up from the other candidates who got dropped from the second round

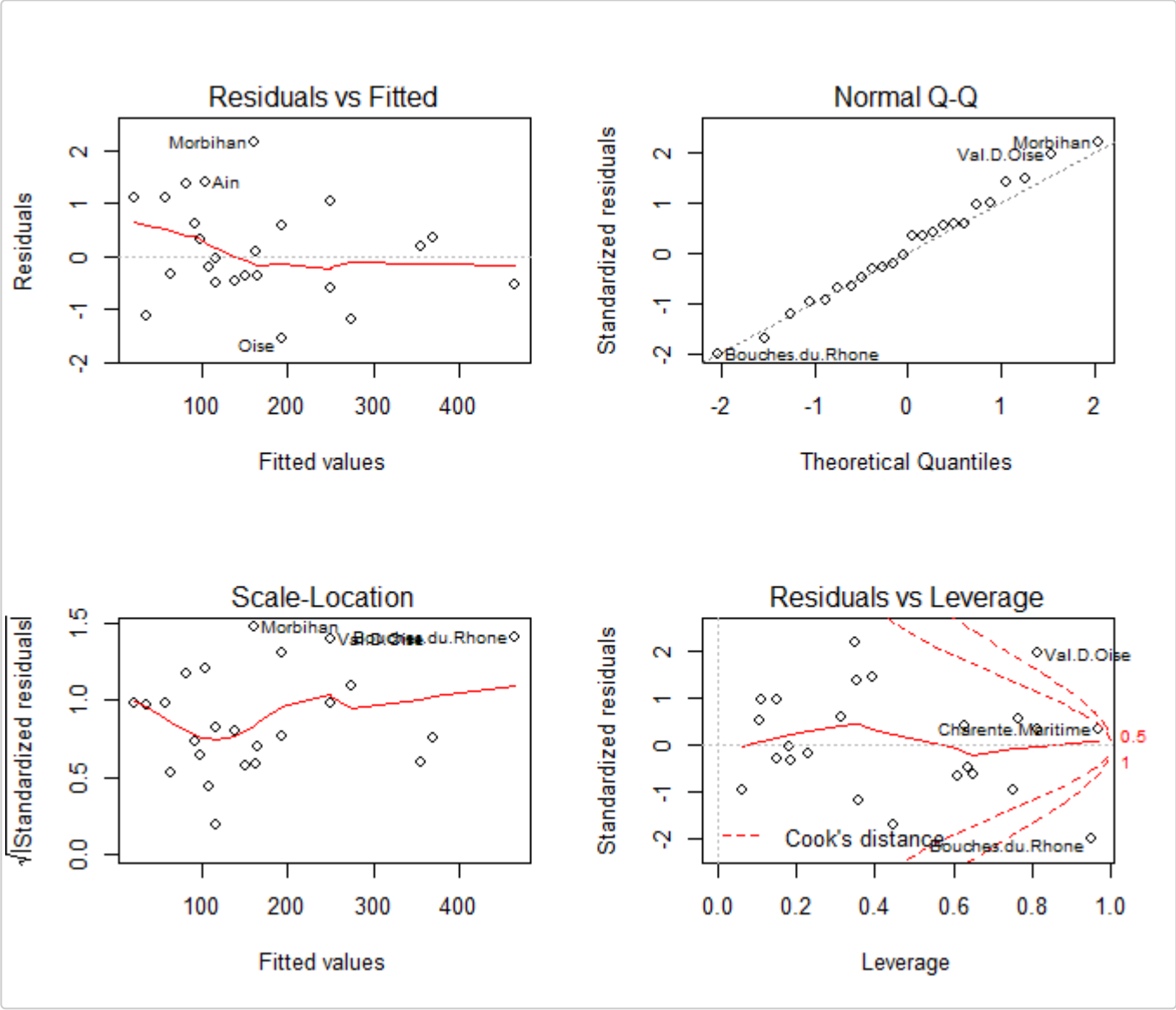
- as we arrive at the following model:

$$A2 = \beta_A A + \beta_B B + \beta_C C + \beta_D D + \beta_E E + \beta_F F + \beta_G G + \beta_H H + \beta_J J + \beta_K K + \beta_N N + \epsilon$$

- In the above model we absorb the unknown shifts in votes from first to second round into the coefficients and the errors

- So if Mitterrand gets all the votes in one district, then the vote variables of all the other candidates must be zero, the fitted value would be roughly the number of actual voters in that district (assuming everyone voted)
- Fit a model without an intercept using OLS
- In R, this is done using the notation `-1`

```
g1 = lm(A2 ~ A + B + C + D + E + F + G + H + J + K + N -1, fpe)
par(mfrow=c(2,2))
plot(g1)
```




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

- Fit a model without an intercept using WLS
- Use $weight = 1/EI$
- Compare coefficients

```
g2 = lm(A2 ~ A+B+C+D+E+F+G+H+J+K+N-1, fpe, weight=1/EI)
compareCoefs(g1, g2)
```

```
## Calls:
## 1: lm(formula = A2 ~ A + B + C + D + E + F + G + H + J + K + N - 1,
##   data = fpe)
## 2: lm(formula = A2 ~ A + B + C + D + E + F + G + H + J + K + N - 1,
##   data = fpe, weights = 1/EI)
##
##    Model 1 Model 2
## A   1.0751  1.0671
```

```
## SE  0.0353  0.0356
##
## B  -0.1246 -0.1051
## SE  0.0261  0.0340
##
## C   0.2574  0.2460
## SE  0.0570  0.0692
##
## D   0.9045  0.9262
## SE  0.0174  0.0221
##
## E    0.671   0.249
## SE   0.293   0.283
##
## F   0.7825  0.7551
## SE  0.0507  0.0577
##
## G    2.166   1.972
## SE   0.221   0.279
##
## H   -0.854  -0.566
## SE   0.468   0.506
##
## J    0.144   0.612
## SE   0.620   0.579
##
## K    0.518   1.211
## SE   0.480   0.507
##
## N   0.5583  0.5294
## SE  0.0912  0.0942
##
```

- Model fit is the same for all sets of weights satisfying the same proportionality

```
g3 = lm(A2 ~ A+B+C+D+E+F+G+H+J+K+N-1, fpe, weight=53/EI)
compareCoefs(g1, g2, g3)
```

```
## Calls:
## 1: lm(formula = A2 ~ A + B + C + D + E + F + G + H + J + K + N - 1,
##   data = fpe)
## 2: lm(formula = A2 ~ A + B + C + D + E + F + G + H + J + K + N - 1,
##   data = fpe, weights = 1/EI)
## 3: lm(formula = A2 ~ A + B + C + D + E + F + G + H + J + K + N - 1,
##   data = fpe, weights = 53/EI)
##
##    Model 1 Model 2 Model 3
## A   1.0751  1.0671  1.0671
## SE  0.0353  0.0356  0.0356
##
## B  -0.1246 -0.1051 -0.1051
## SE  0.0261  0.0340  0.0340
```

```
##
## C   0.2574  0.2460  0.2460
## SE  0.0570  0.0692  0.0692
##
## D   0.9045  0.9262  0.9262
## SE  0.0174  0.0221  0.0221
##
## E    0.671   0.249   0.249
## SE   0.293   0.283   0.283
##
## F   0.7825  0.7551  0.7551
## SE  0.0507  0.0577  0.0577
##
## G    2.166   1.972   1.972
## SE   0.221   0.279   0.279
##
## H   -0.854  -0.566  -0.566
## SE   0.468   0.506   0.506
##
## J    0.144   0.612   0.612
## SE   0.620   0.579   0.579
##
## K    0.518   1.211   1.211
## SE   0.480   0.507   0.507
##
## N   0.5583  0.5294  0.5294
## SE  0.0912  0.0942  0.0942
##
```

- Constrain the coefficients:
  - Set coefficients for *B* and *H* equal to zero.
  - Set coefficients for *A*, *G*, and *K* equal to one.

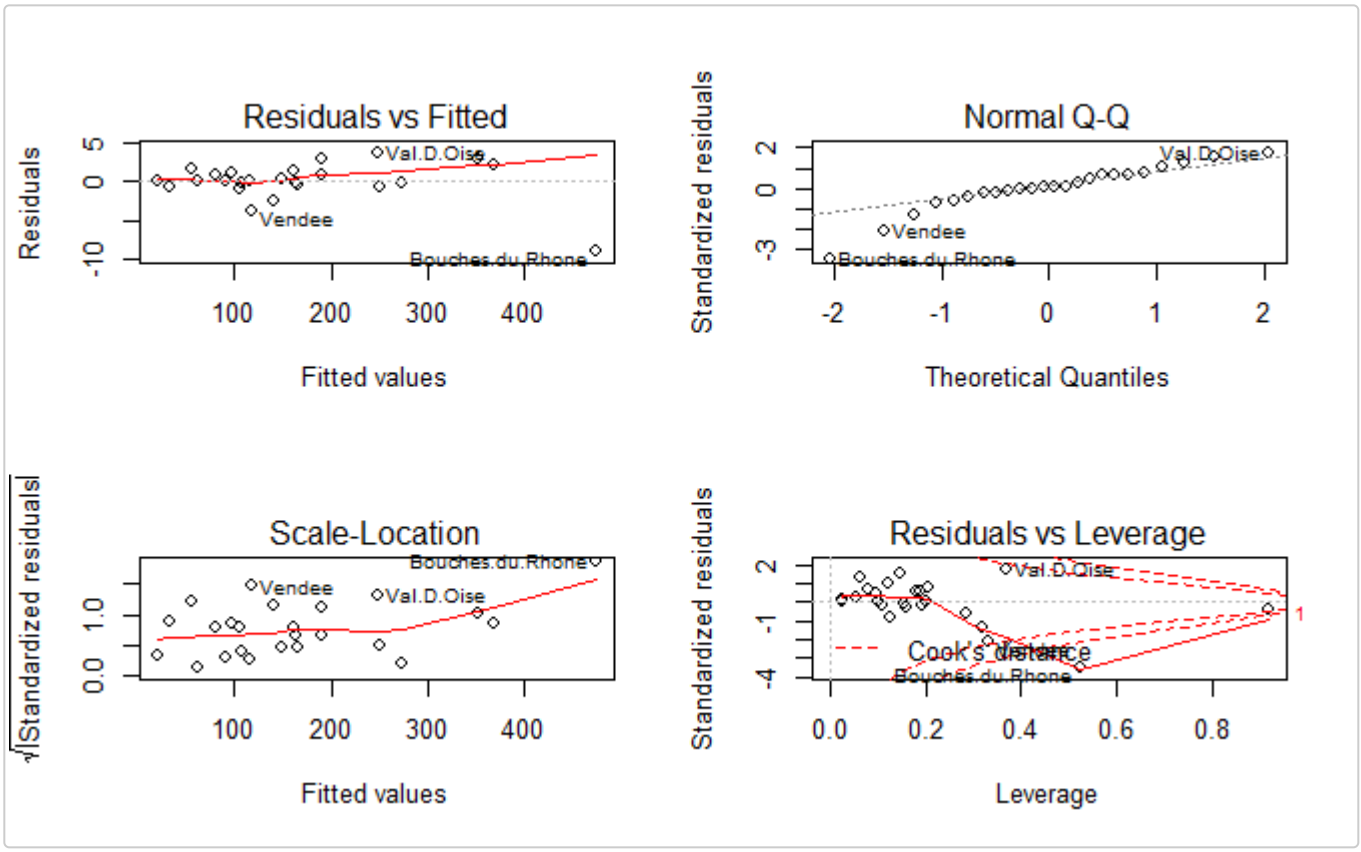
```
g4 = lm(A2 ~ offset(A+G+K)+C+D+E+F+N-1,
        fpe,
        weight=1/EI)

compareCoefs(g1, g2, g3, g4)
```

```
## Calls:
## 1: lm(formula = A2 ~ A + B + C + D + E + F + G + H + J + K + N - 1,
##   data = fpe)
## 2: lm(formula = A2 ~ A + B + C + D + E + F + G + H + J + K + N - 1,
##   data = fpe, weights = 1/EI)
## 3: lm(formula = A2 ~ A + B + C + D + E + F + G + H + J + K + N - 1,
##   data = fpe, weights = 53/EI)
## 4: lm(formula = A2 ~ offset(A + G + K) + C + D + E + F + N - 1, data =
##   fpe, weights = 1/EI)
##
##    Model 1 Model 2 Model 3 Model 4
## A   1.0751  1.0671  1.0671
## SE  0.0353  0.0356  0.0356
```

```
##
## B  -0.1246 -0.1051 -0.1051
## SE  0.0261  0.0340  0.0340
##
## C   0.2574  0.2460  0.2460  0.2258
## SE  0.0570  0.0692  0.0692  0.0553
##
## D   0.9045  0.9262  0.9262  0.9700
## SE  0.0174  0.0221  0.0221  0.0233
##
## E    0.671   0.249   0.249   0.390
## SE   0.293   0.283   0.283   0.225
##
## F   0.7825  0.7551  0.7551  0.7442
## SE  0.0507  0.0577  0.0577  0.0812
##
## G    2.166   1.972   1.972
## SE   0.221   0.279   0.279
##
## H   -0.854  -0.566  -0.566
## SE   0.468   0.506   0.506
##
## J    0.144   0.612   0.612
## SE   0.620   0.579   0.579
##
## K    0.518   1.211   1.211
## SE   0.480   0.507   0.507
##
## N   0.5583  0.5294  0.5294  0.6085
## SE  0.0912  0.0942  0.0942  0.1202
##
```

- Get the diagnotic plots:

```
par(mfrow=c(2,2))
plot(g4)
```




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

- Test the residuals for normality:

```
shapiro.test(residuals(g4))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(g4)
## W = 0.82641, p-value = 0.0008225
```

WLS Fit using the *savings* data

- Use the *savings* data from the *faraway* package.
- Use weights two ways: proportional to
  - $\frac{1}{pop15}$
  - $\frac{1}{pop15^2}$
- Are there any differences in the estimated coefficients?

```
library(faraway)
data(savings)
g1 = lm(sr ~ pop15 + pop75 + dpi + ddpi, savings)
g2 = lm(sr ~ pop15 + pop75 + dpi + ddpi, savings,
        weight=1/pop15)
```

```
g3 = lm(sr ~ pop15 + pop75 + dpi + ddpi, savings,
        weight=1/(pop15^2))
compareCoefs(g1, g2, g3)
```

```
## Calls:
## 1: lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = savings)
## 2: lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = savings,
##   weights = 1/pop15)
## 3: lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = savings,
##   weights = 1/(pop15^2))
##
##               Model 1   Model 2   Model 3
## (Intercept)     28.57     28.60     28.29
## SE               7.35      6.78      6.39
##
## pop15          -0.461    -0.468    -0.467
## SE              0.145     0.134     0.128
##
## pop75          -1.691    -1.671    -1.645
## SE              1.084     0.975     0.890
##
## dpi         -0.000337 -0.000395 -0.000424
## SE           0.000931  0.000828  0.000752
##
## ddpi            0.410     0.466     0.536
## SE              0.196     0.199     0.204
##
```

Remedy for Incorrect Model Structure: Test of Lack of Fit

- We have previously presented use of the *partial regression plots* to diagnose problems with the model structure.
- We present a statistical hypothesis test about whether or not the model actually fits the data.
- The method is called *testing for lack of fit*.
- It cannot be performed with just any dataset.
- The dataset must contain some *replication* of outcomes with the same configuration of input variables.
- Replication can happen by chance, or can be built into the design of an experiment.

Testing for Lack of Fit

- The following works if we have replications of the response variables at several values of the input variables.
- In our example, we will see that even though a model may enjoy a high value of R^2, the model still may not fit the data.

- The statistical method relies on the estimate of the error variance σ^2 that comes from the residuals of the fitted model.
- The estimate of this variance is the mean squared error $MSE = \frac{SSE}{n-p}$.
- If the model does not fit the data, its MSE will over-estimate σ^2.
- If we have replication in the dataset, we can compute another independent estimate of σ^2.
- Then the MSE will be compared to this independent estimate of σ^2 that comes from replicating the observations.
- In experimental design, we called this a *replication* experiment, and it means that the response is observed several times for the same values of the predictor variable.
- In case there are multiple predictors, the response is observed several times for the same combination of values for the predictor variables.

Example: Corrosion loss in Cu-Ni alloys

- Here is an example involving the test of fit of a straight line when the model should be an unspecified curved line.

```
data(corrosion)
?corrosion
```

corrosion R Documentation

Corrosion loss in Cu-Ni alloys

Description

Data consist of thirteen specimens of 90/10 Cu-Ni alloys with varying iron content in percent. The specimens were submerged in sea water for 60 days and the weight loss due to corrosion was recorded in units of milligrams per square decimeter per day.

Usage

```
data(corrosion)
```

Format

This dataframe contains the following columns

Fe

Iron content in percent

loss

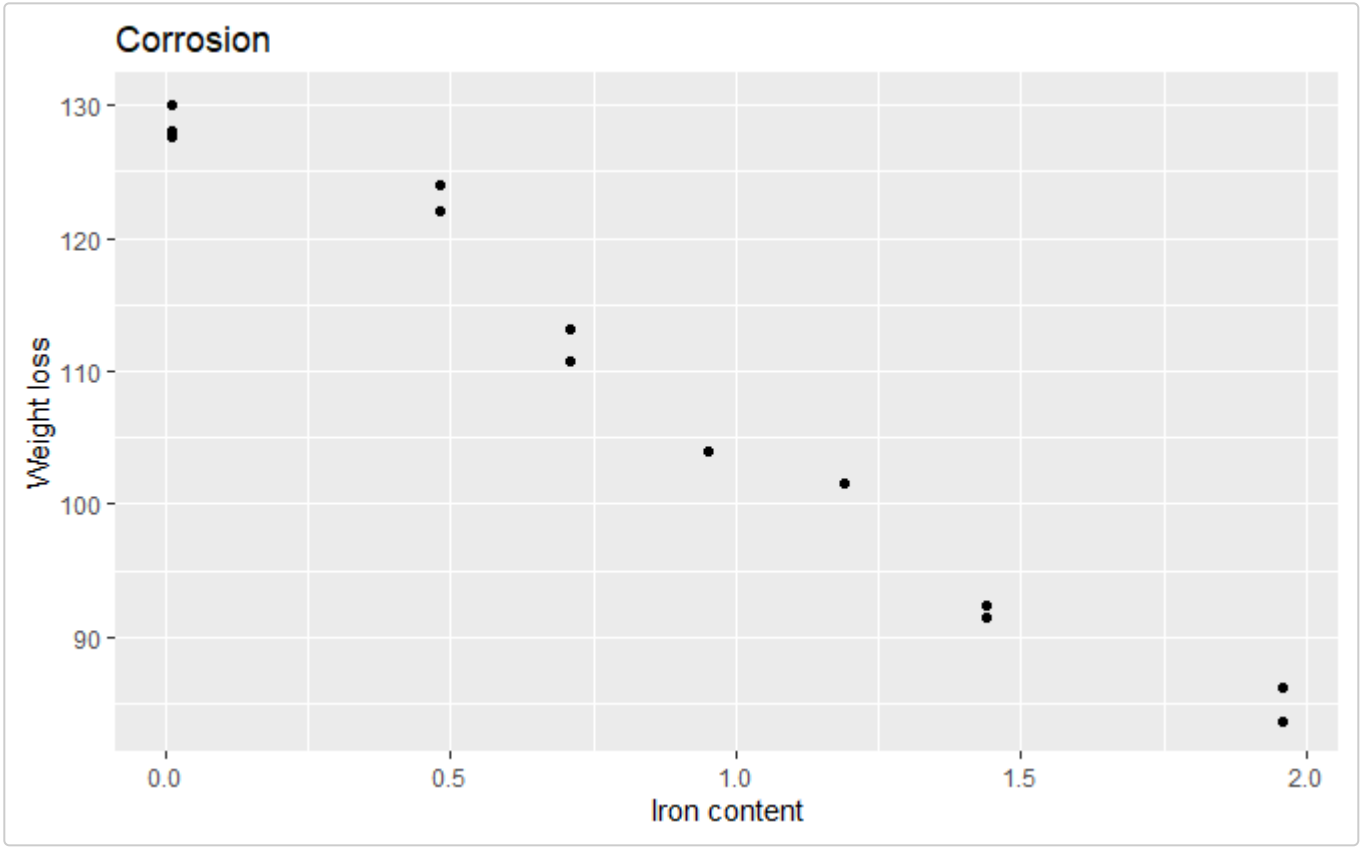
Weight loss in mg per square decimeter per day

Source

"Applied Regression Analysis" by N. Draper and H. Smith, Wiley, 1998

- A plot of the data reveals several values of the predictor where two or more replications of the response have been taken

```
(p <- qplot(Fe, loss, data=corrosion) +
  labs(x="Iron content",
       y="Weight loss",
       title="Corrosion"))
```

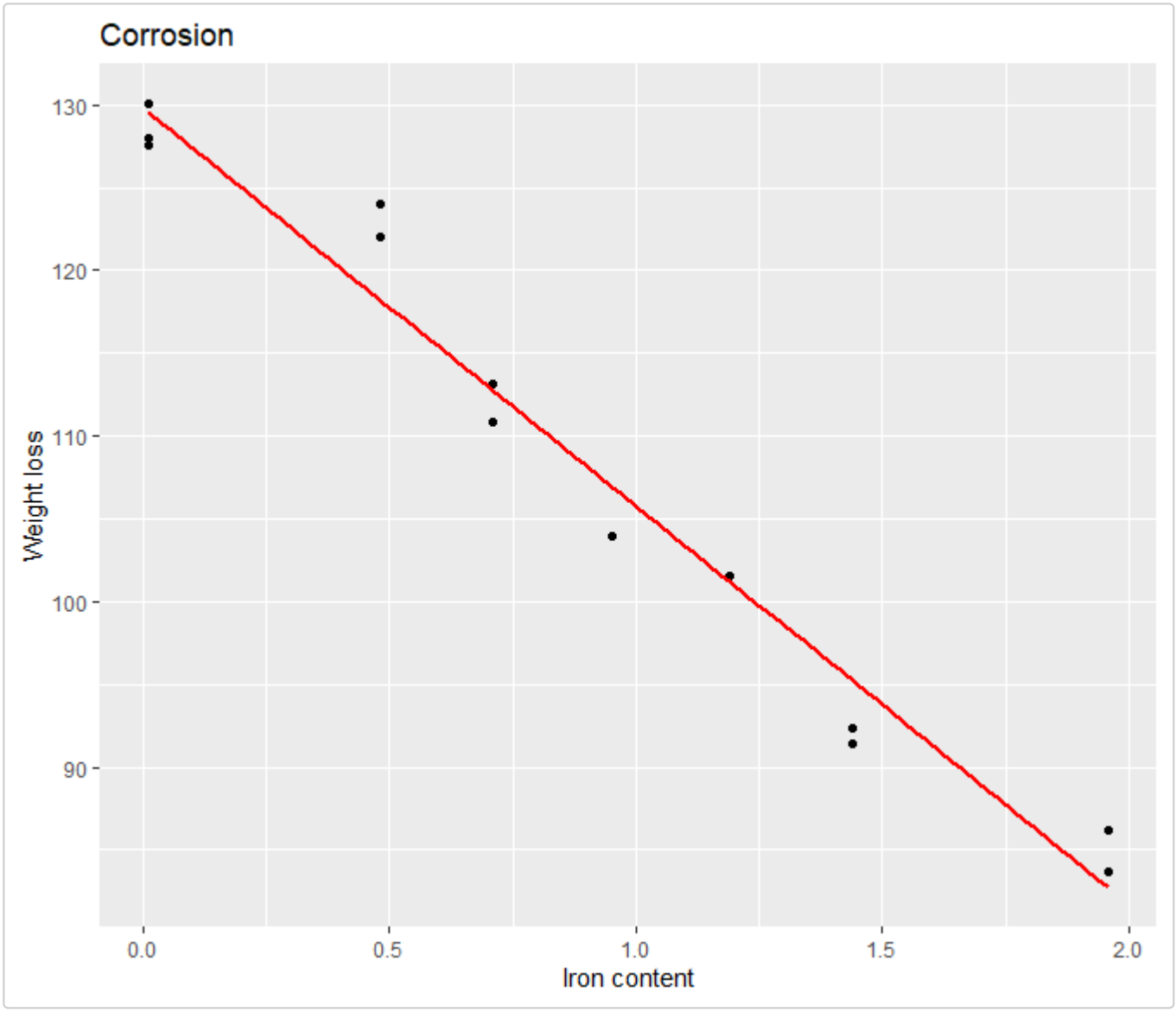


- We fit a simple linear model to the data

```
g1 = lm(loss ~ Fe, corrosion)
sum.g1 = summary(g1)
```

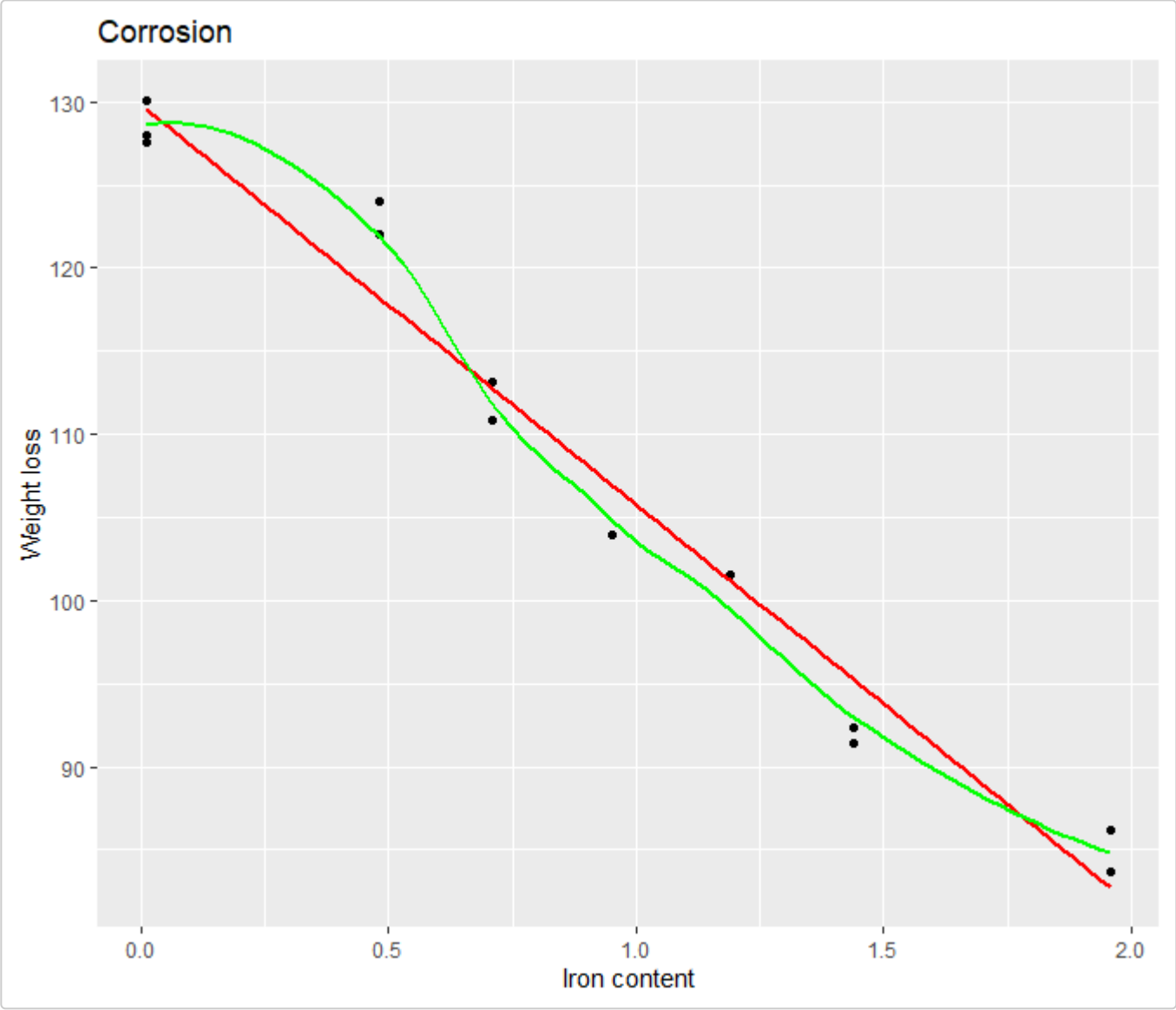
- We see that for a one percent increase in *Iron content*, there will be a 24.0198934 decrease in *Weight loss*
- Moreover, the R^2 is equal to 0.97
- The *MSE* is the square of the "Residual standard error": 3.0577804
- We graph a straight line fit to the data

```
(p + geom_smooth(method = "lm", formula = y~x, colour="red", se=F))
```



- For comparison, we fit a smooth nonparametric curve to the data
- It appears that the “green line” does not lie along the straight line

```
(p  + geom_smooth(method = "lm",
                  formula = y~x, colour="red",   se=F) +
      geom_smooth(method = "loess",
                  formula = y~x, colour="green", se=F))
```



- Now to get an independent estimate of σ^2, we make use of the replications to fit the *group means model* to each group

$$y_{ij} = \mu_j + \epsilon_{ij} \; i = 1, \ldots n_j$$

$$\epsilon_{ij} \sim N\left(0, \sigma^2\right)$$

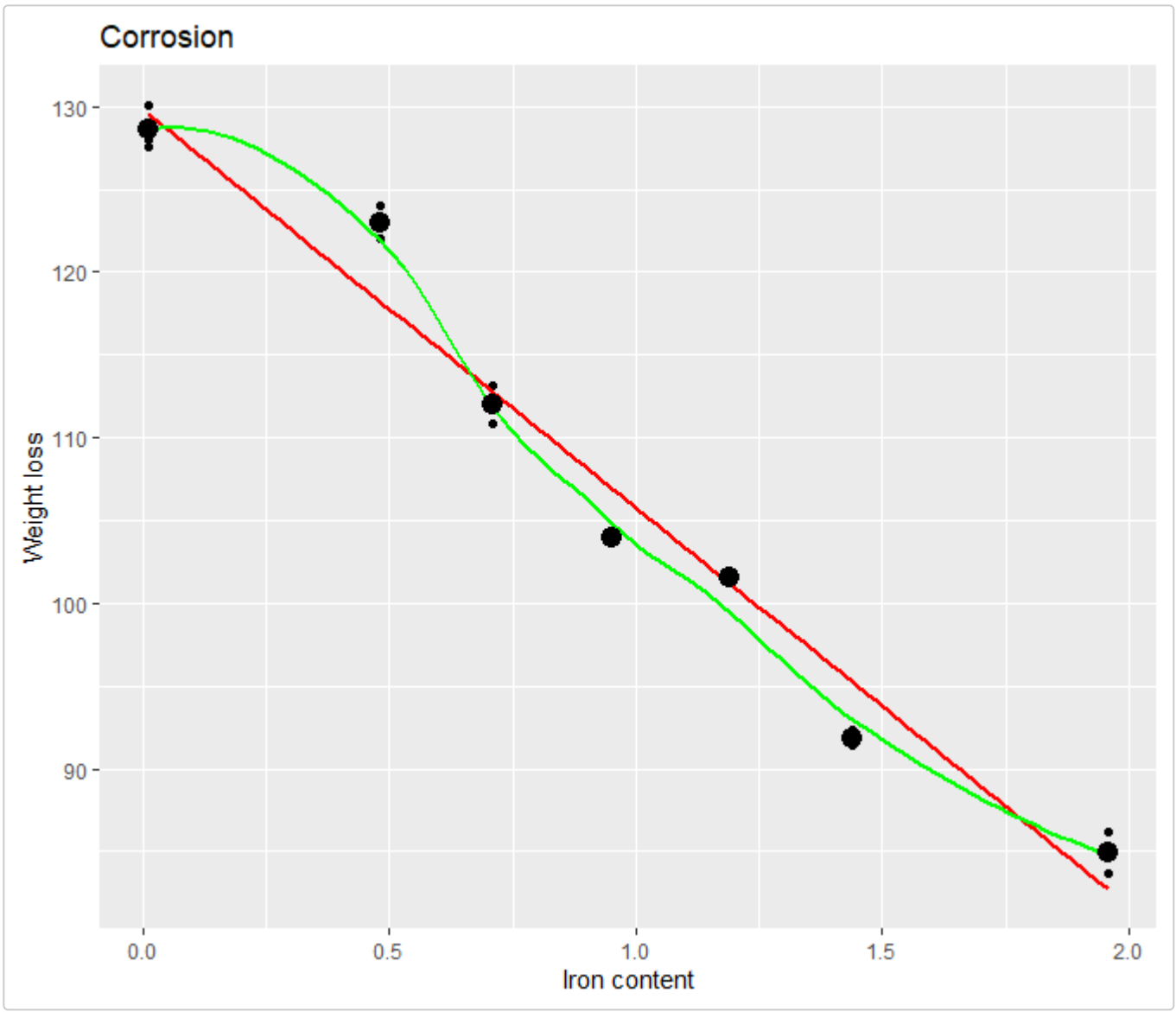
- This model is extremely easy to fit in R
- Since the predictor `Fe` has just a few unique values, it is safe to treat it as a categorical variable

```
ga = lm(loss ~ factor(Fe), corrosion)
summary(ga)$sigma
```

```
## [1] 1.401289
```

- Just like any linear model, we have fitted values, `predict(ga)`.
- The fitted values for the *group means model* are the group means.
- We add the group means to the graph

```
(p +  geom_smooth(method = "lm",
                  formula = y~x, colour="red",
                  se=F) +
      geom_smooth(method = "loess",
                  formula = y~x, colour="green",
                  se=F) +
      geom_point(x=corrosion$Fe,
                 y=predict(ga),
                 size = 4))
```



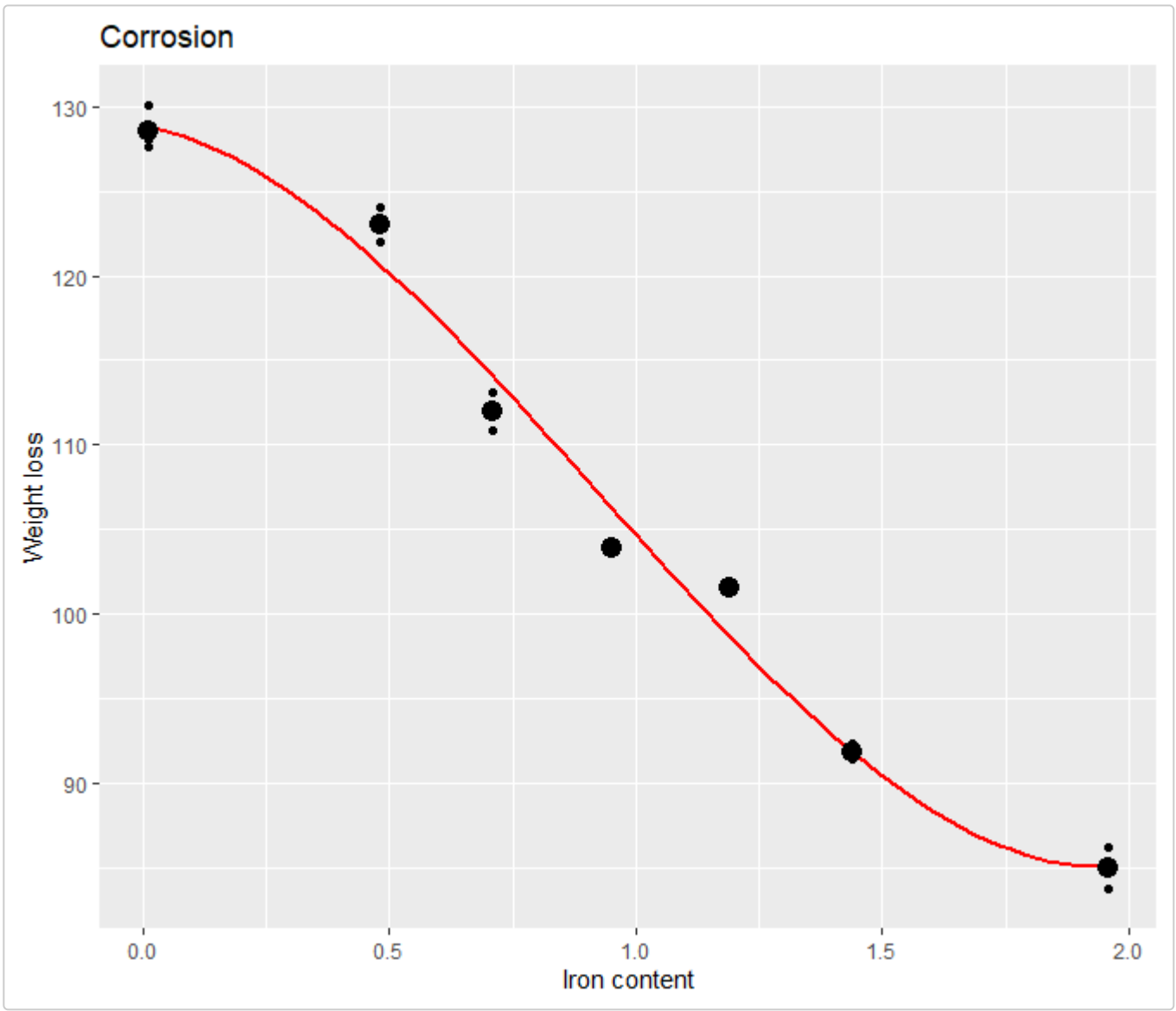
- Now test the fit of the *straight line model* (small model) with the *group means model* model

```
anova(g1,ga)
```

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
11	102.85023	NA	NA	NA	NA
6	11.78167	5	91.06857	9.275621	0.0086228

- The small model is the null hypothesis: $H_0 : E(loss) = \beta_0 + \beta_1 Fe$
- Conclusion of the test of fit: Reject the H_0 : straight line model, since the P-value is less than $\alpha = 0.05$
- Now we test the fit of a polynomial model order 3

```
(p  + geom_smooth(method = "lm",
                  formula = y ~ poly(x,3),
                  colour="red",
                  se=F) +
      geom_point(x=corrosion$Fe,
                 y=predict(ga),
                 size = 4))
```

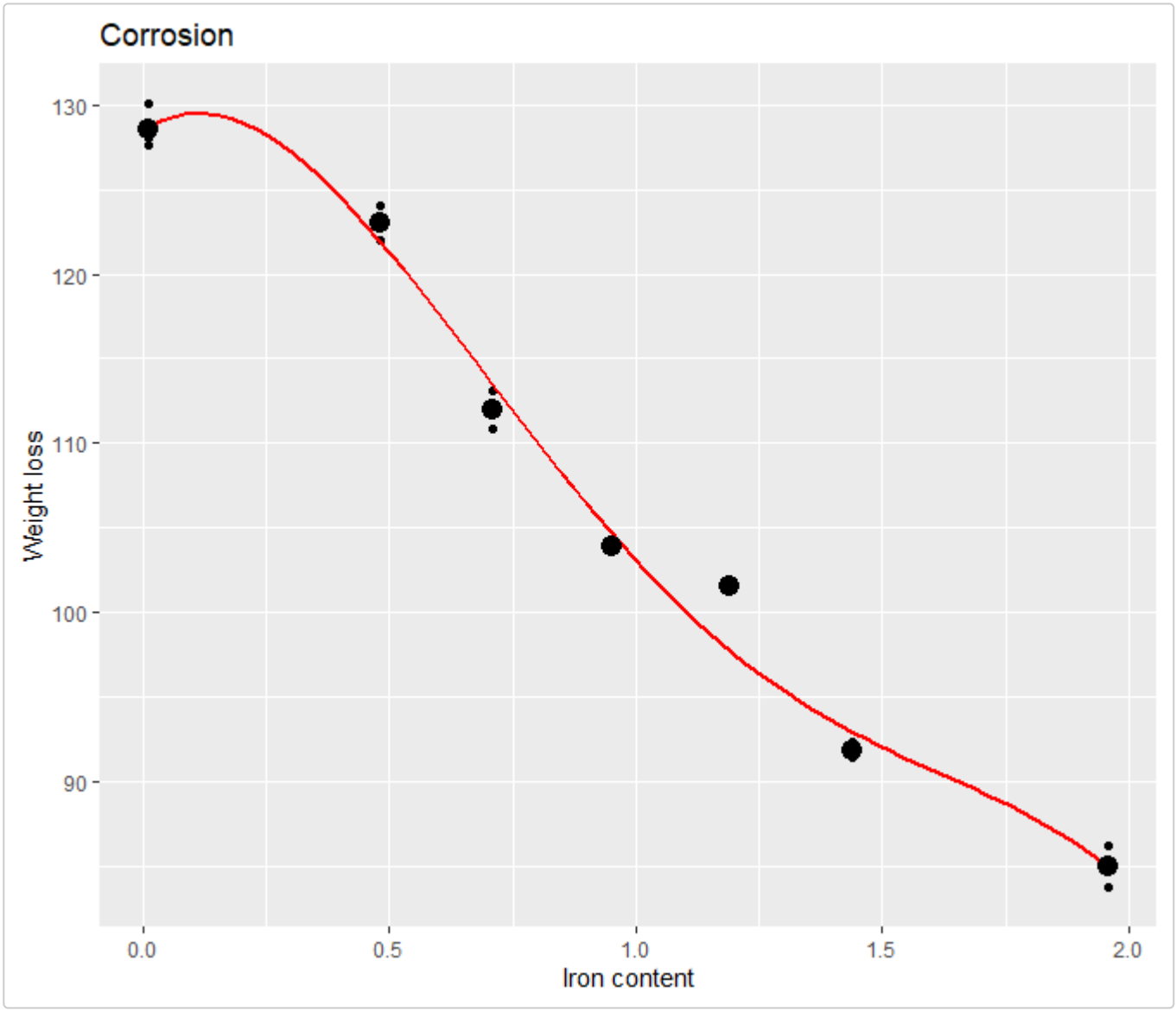



```
g3 = lm(loss ~ poly(Fe,3), corrosion)
anova(g3,ga)
```

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
9	45.05070	NA	NA	NA	NA
6	11.78167	3	33.26903	5.647593	0.0350625

- Conclusion of the test of fit: Reject the polynomial(3) model
- Now test the fit a polynomial model order 4

```
(p  + geom_smooth(method = "lm",
                  formula = y ~ poly(x,4),
                  colour="red",
                  se=F) +
     geom_point(x=corrosion$Fe,
                y=predict(ga),
                size = 4))
```




```
g4 = lm(loss ~ poly(Fe,4), corrosion)
anova(g4,ga)
```

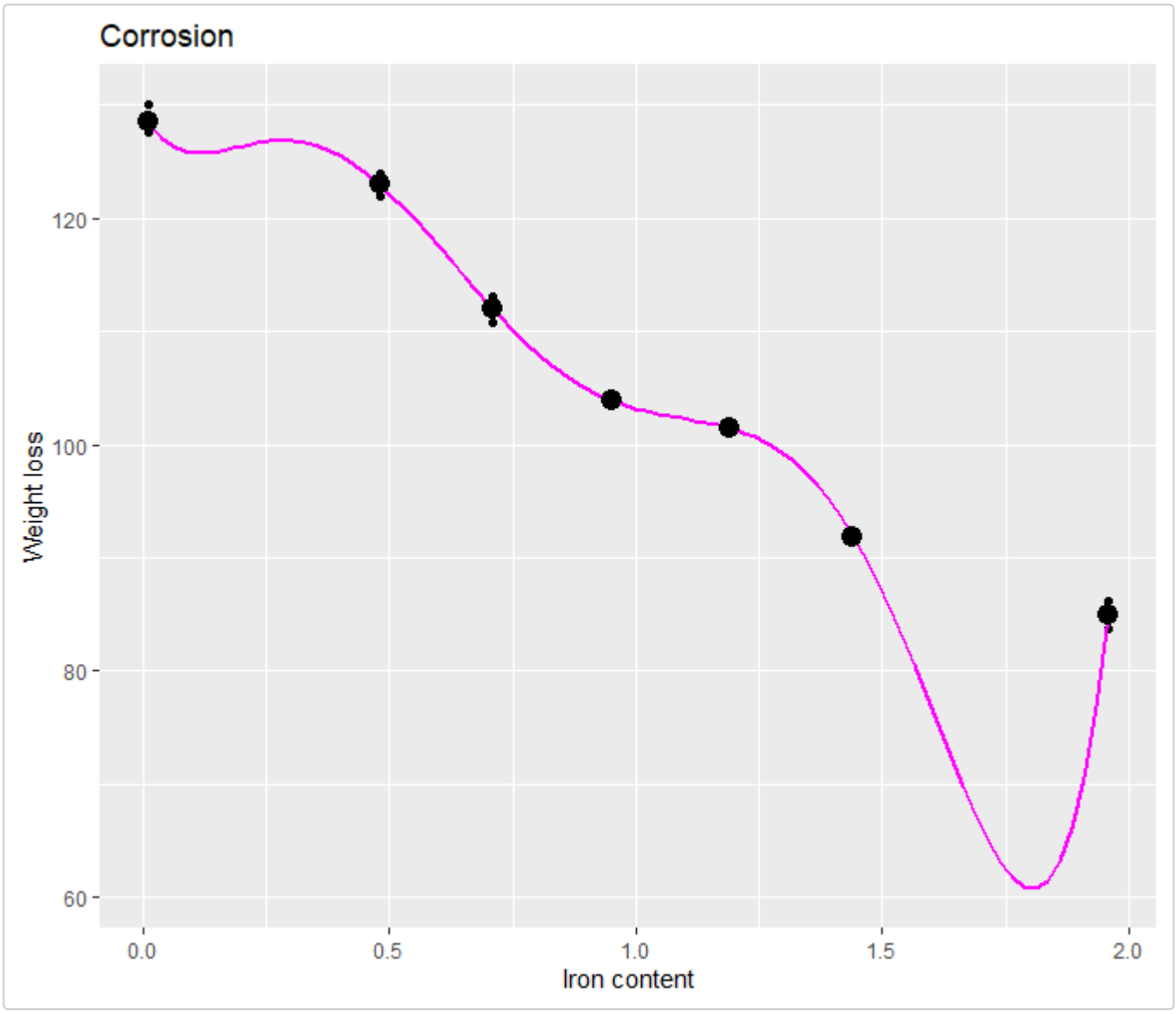
Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
8	34.97534	NA	NA	NA	NA
6	11.78167	2	23.19367	5.905871	0.0382239

- Conclusion of the test of fit: Reject the polynomial(4) model
- We could keep going, but what is the point of a higher order polynomial when it would clearly not be predictive of future responses?
- We should take a clue from the nonparametric fit, as it seems to suggest a low degree sinusoidal model
- We leave this task to another day, but we have learned an important lesson about over-fitting and about R^2

R^2 is not a measure of model fit

- Even though all models enjoy high R^2, they are not adequate fits to the data
- R^2 is only good for comparing models
- It is possible to fit a model through the group means perfectly and get an R^2 almost one, but such a model is useless for predicting new responses
- Here we have 7 groups, so a model with 7 parameters will fit the group means perfectly
- A polynomial of degree 6, has 7 parameters
- A plot of the polynomial reveals that, while it fits the group means perfectly, the model is not useful for prediction or interpolation

```
(p  + geom_smooth(method = "lm",
                  formula = y ~ poly(x,6),
                  colour=6,
                  se=F) +
      geom_point(x=corrosion$Fe,
                 y=predict(ga),
                 size = 4))
```



Remedy for Influential Data: Robust Regression

- When there are high leverage data points and these are also influential, we have seen that they will distort a linear model fit.
- There are several methods for fitting models under these circumstances
- These methods fall under the topic of *robust regression*
- In the following we explore some of these methods

M-estimation

- To overcome distortions from influential observations, Huber (1961) introduced the M-estimator.
- The OLS method minimizes the sum of squared residuals
- In case of OLS, the *loss function* is the square function of the error in fitting the response observations

General loss function and M-estimation

- Suppose $\rho()$ is a given "loss function", the M-estimate of $\boldsymbol{\beta}$ is obtain from the solution to:

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \rho\left(y_i - \sum_{k=0}^{p-1} x_{i,k}\beta_k\right)$$

- $\rho(u) = u^2$ is just ordinary least squares loss (OLS).
- $\rho(u) = |u|$ is least absolute deviation loss (LAD). Also called L_1 regression.
- We can see that large residuals are squared in OLS but are just their absolute values in LAD.
- Therefore one would think that outliers would have less effect on the estimates from LAD versus OLS loss functions.
- There are several loss functions that have been proposed and studied as follows:

Huber's loss function

$$\rho(u) = \begin{cases} \frac{u^2}{2} & |u| \le c \\ c|u| - \frac{c^2}{2} & |u| > c \end{cases}$$

- Huber's M-estimate uses a loss function that is a combination of OLS and LAD loss functions.
- It is like OLS for small residuals, but like LAD for large residuals.
- c should be a robust estimate of σ, a value proportional to the median of $|e_i|$ is recommended.

Tukey's Bisquare loss function

$$\rho(u) = \begin{cases} \frac{c^2}{6}\left(1 - \left[1 - \left(\frac{u}{c}\right)^2\right]^3\right) & |u| \le c \\ \frac{c^2}{6} & |u| > c \end{cases}$$

- Tukey's M-estimator uses a loss function is a combination of OLS for small residuals and a constant for large residuals.
- This does not let a large residual overly effect the estimate.
- In the following we will reformulate M-estimators as weighted least squares estimators (WLS)
- We will show why Tukey's M-estimator belongs to the special class *redescending M-estimator*.
- Redescending type estimators are robust because they completely reject gross outliers, but do not completely ignore moderately large outliers.

Weight functions

- Taking derivatives of the "sum of loss-errors" function and setting to 0, we solve these equations for the estimates:

$$\sum_{i=1}^{n} \rho'\left(y_i - \sum_{k=0}^{p-1} x_{ik}\beta_k\right) x_{ik} = 0 \quad k = 0, \ldots, p-1$$

- Now using the residual errors definition:

$$e_i = y_i - \sum_{k=0}^{p-1} x_{ik}\beta_k$$

- The normal equations become

$$\sum_{i=1}^{n} \frac{\rho'(e_i)}{e_i} x_{ik}\left(y_i - \sum_{k=0}^{p-1} x_{ik}\beta_k\right) = 0 \quad k = 0, \ldots, p-1$$

- This is a version of *Weighted Least Squares* with the weight function:

$$w(u) = \frac{\rho'(u)}{u}$$

- The weight function takes different forms depending on the method of estimation
  - LS:

$$w(u) = 1$$

+ LAD:

$$w(u) = |u|$$

+ Huber:

$$w(u) = \begin{cases} 1 & |u| \le c \\ \frac{c}{u} & |u| > c \end{cases}$$

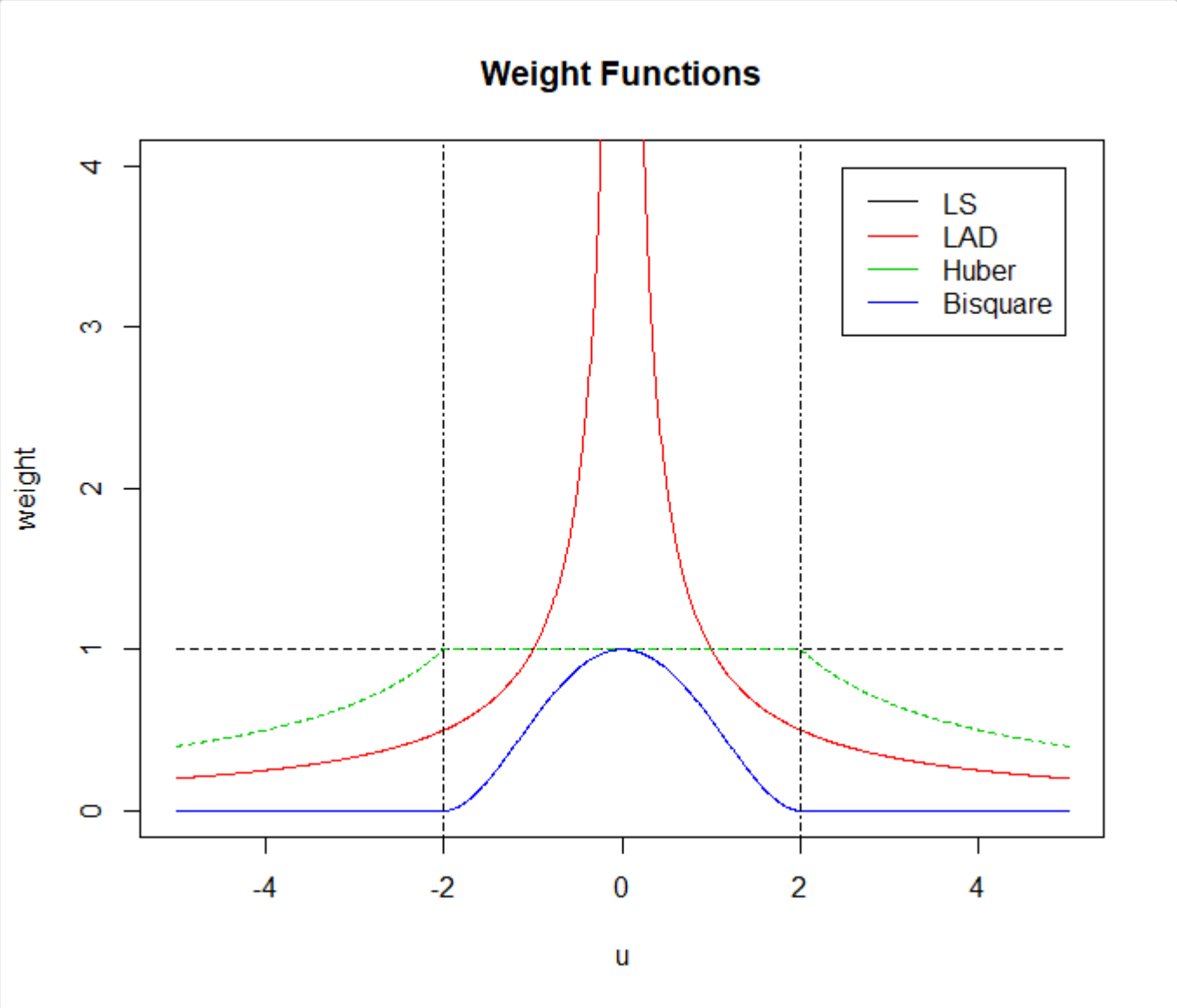
+ Tukey Bi-square:

$$w(u) = \begin{cases} \left[1 - \left(\frac{u}{c}\right)^2\right]^2 & |u| \le c \\ 0 & |u \rhd c \end{cases}$$

- Here is a graph of the weight functions.
- We choose $c = 2$ for illustration of the Huber and Tukey weight function.

```
grid = seq(-5,5,len=1000)

w.LS = function(u)1

w.LAD = function(u) 1/abs(u)

w.Huber = function(u) ifelse(abs(u)<=2, 1, 2/abs(u))

w.Bisquare = function(u) ifelse(abs(u)<=2, (1-(u/2)^2)^2, 0)

LS = sapply(grid, w.LS)
```

```
LAD = sapply(grid, w.LAD)

Huber = sapply(grid, w.Huber)

Bisquare = sapply(grid, w.Bisquare)

matplot(grid, cbind(LS, LAD, Huber, Bisquare), lty = c(2,1,2,1), type="l", ylim=c(0,4), ylab =
"weight", xlab = "u", main = "Weight Functions")

abline(v=c(-2,2), lty=4)

legend("topright",
       c("LS", "LAD", "Huber", "Bisquare"),
       col=c(1,2,3,4),
       inset = .04,
       lty=1)
```



Breakdown Point

- We know that the bad property of the least square estimators is that OLS estimators can be destroyed by moving just one data point (or $1/n$ proportion of data points) to infinity.
- Any estimator that can be moved to infinity by moving just one data point to $+\infty$ or $-\infty$, no matter have large the number of observations, n, is said to have a 0 *breakdown point*.
- If it takes moving at least p proportion of the data points to + or - infinity to move the estimator to infinity, the estimator is said to have a p breakdown point.
- For example, you have to move at least $\frac{1}{2}$ of the observations to infinity in order to move the median estimate to infinity.

Breakdown Points of the Robust M-Estimates

- The LS estimators are have a 0 breakdown point.
- The LAD and the Huber estimators also have 0 breakdown point.
- The LAD and the Huber estimators reduce, however, the effects of outliers when compared to the LS estimators.
- The Tukey Bisquare estimator has a large breakdown point and is preferred to LAD and Huber.
- Another robust method, to be introduced later, is *Least Trimmed Squares* which also has a large breakdown point.

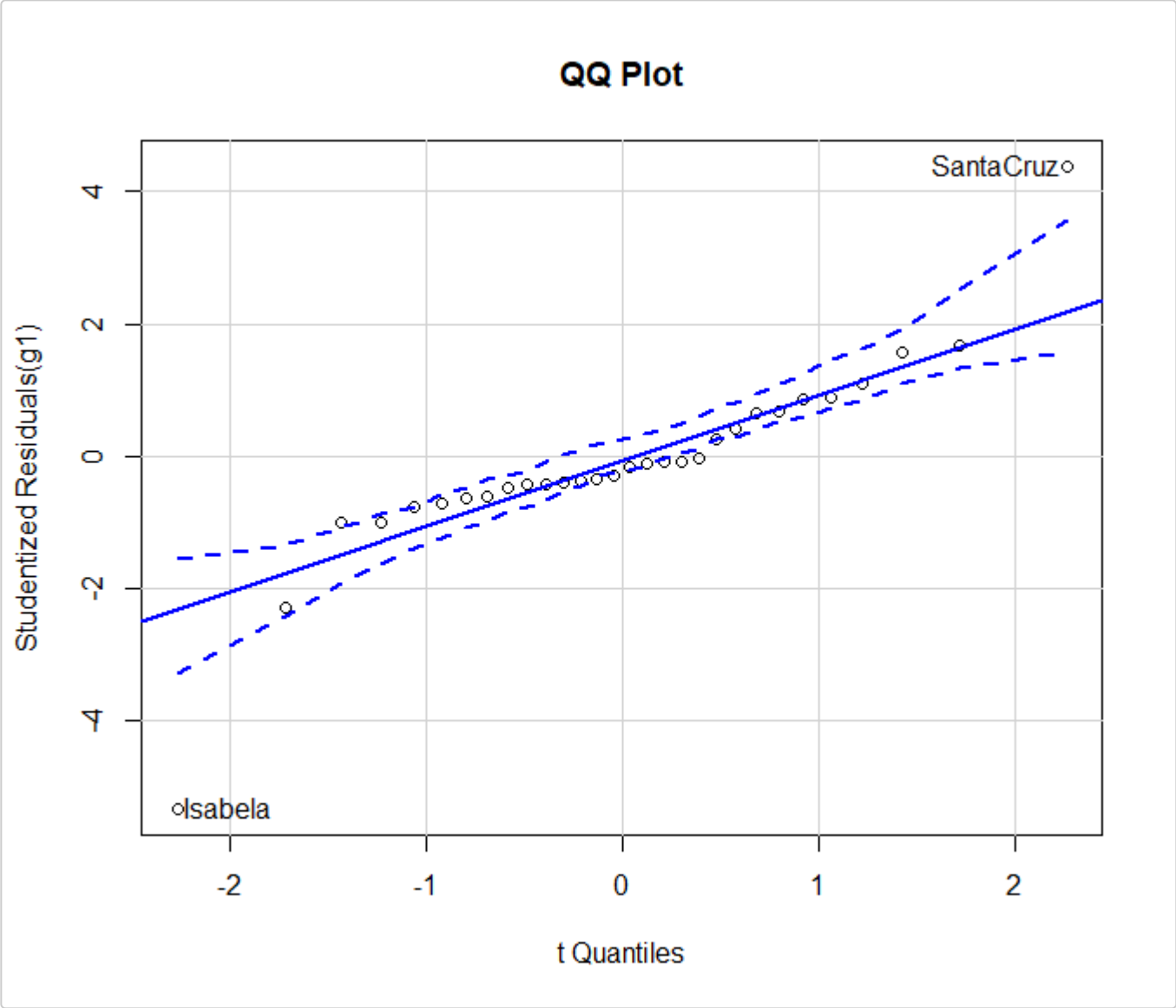
Galapagos data

- Consider the Galapagos data.

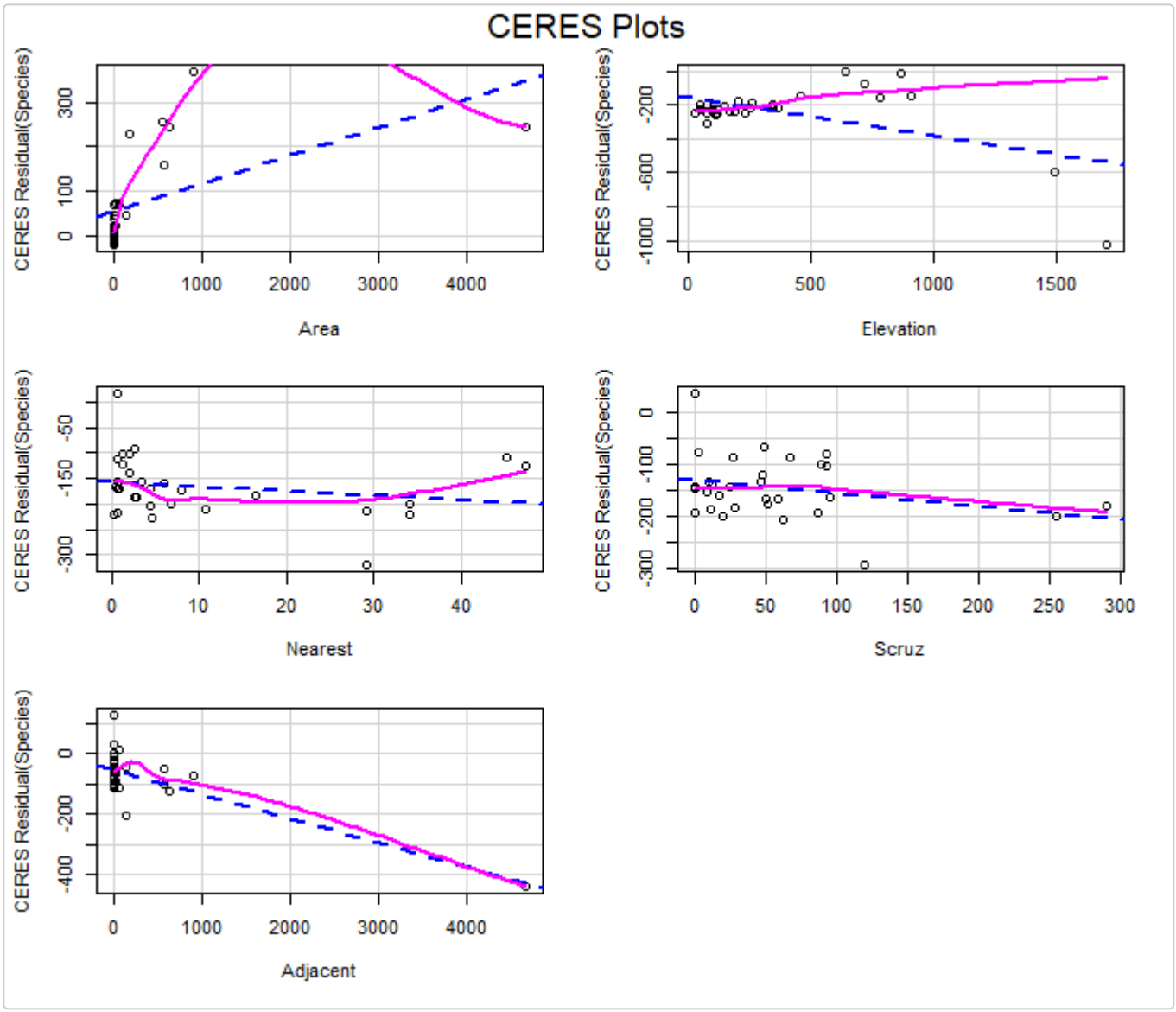
```
data(gala)
g1 = lm(Species ~
           Area+Elevation+Nearest+Scruz+Adjacent,
         gala)
```

- Normal Q-Q Plot shows non-normality of errors.

```
outlierTest(g1) # Bonferonni p-value for most extreme obs

qqPlot(g1, main="QQ Plot") # qq plot for studentized resid
```




```
shapiro.test(residuals(g1))
# The Shapiro-Wilk test rejects normality of errors

ceresPlots(g1, terms = ~ .) # CERES plots
```

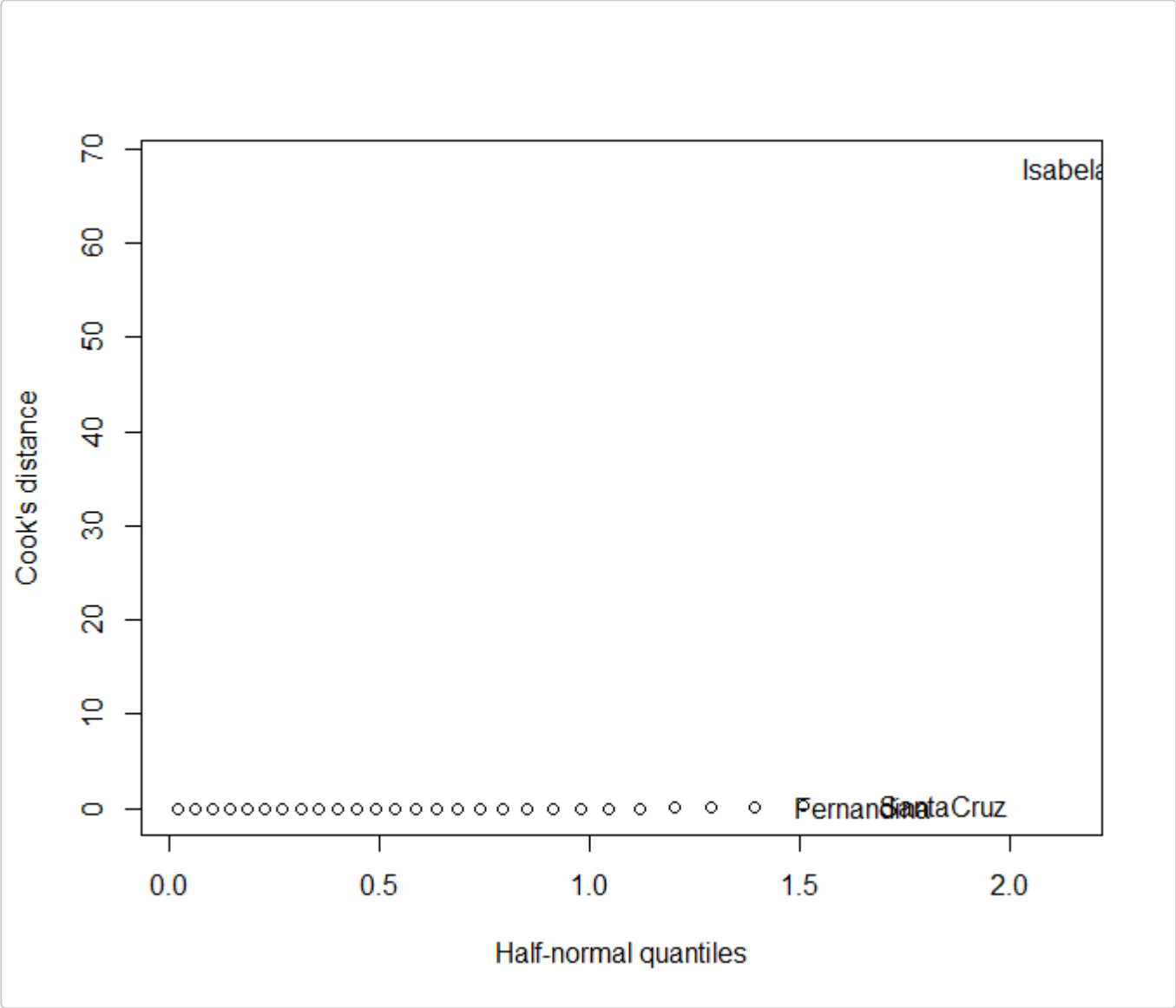



```
##            rstudent unadjusted p-value Bonferroni p
## Isabela   -5.333694         2.0464e-05   0.00061392
## SantaCruz  4.378143         2.1937e-04   0.00658110
##   Isabela SantaCruz
##        16        25
##
##  Shapiro-Wilk normality test
##
## data:  residuals(g1)
## W = 0.91351, p-value = 0.01826
```

- The Shapiro-Wilk test rejects normality of errors.
- Isabella Island has the highest Cook's Distance.

```
islands = row.names(gala)

halfnorm(cooks.distance(g1),
         3,
```

```
          labs=islands,
          ylab="Cook's distance")
```



Huber M-estimation

```
g2 = rlm(Species ~
           Area+Elevation+Nearest+Scruz+Adjacent,
         psi = psi.huber,
         gala)
```

Tukey Bisquare M-estimation

```
g3 = rlm(Species ~
           Area+Elevation+Nearest+Scruz+Adjacent,
         psi = psi.bisquare,
         init="lts",
```

```
          maxit=100,
          gala)
```

Hample M-estimation

```
g4 = rlm(Species ~
           Area+Elevation+Nearest+Scruz+Adjacent,
         psi = psi.hampel,
         init="lts",
         maxit=100,
         gala)
```

Least Trimmed Squares (LTS)

```
g5 = ltsReg(Species ~
              Area+Elevation+Nearest+Scruz+Adjacent,
            data=gala)

g6 = ltsReg(Species ~
              Area+Elevation+Nearest+Scruz+Adjacent,
            data=gala,
            nsamp="exact")
```

```
## Computing all  593775  subsets of size  6  out of  30
##  This may take a very long time!
```

Least Absolution Deviation (LAD)

```
g7 = rq(Species ~
          Area+Elevation+Nearest+Scruz+Adjacent,
        data=gala)
```

- Compare coefficients

```
coefs = compareCoefs(g1, g2, g3, g4,
                     se = FALSE)

colnames(coefs) = c("OLS", "Huber", "Bisquare",
                    "Hample"
                    )

coefs
```

```
## Calls:
## 1: lm(formula = Species ~ Area + Elevation + Nearest + Scruz +
```

```
##   Adjacent, data = gala)
## 2: rlm(formula = Species ~ Area + Elevation + Nearest + Scruz +
##   Adjacent, data = gala, psi = psi.huber)
## 3: rlm(formula = Species ~ Area + Elevation + Nearest + Scruz +
##   Adjacent, data = gala, psi = psi.bisquare, init = "lts", maxit = 100)
## 4: rlm(formula = Species ~ Area + Elevation + Nearest + Scruz +
##   Adjacent, data = gala, psi = psi.hampel, init = "lts", maxit = 100)
##
##              Model 1  Model 2  Model 3  Model 4
## (Intercept)     7.07     6.36    12.38    12.94
## Area        -0.02394 -0.00611  1.53871  0.99789
## Elevation     0.3195   0.2476   0.0209   0.1035
## Nearest      0.00914  0.35916  0.77344 -0.66858
## Scruz       -0.24052 -0.19524 -0.11989 -0.00687
## Adjacent    -0.07480 -0.05455 -0.19675 -0.00181
```

	OLS	Huber	Bisquare	Hample
(Intercept)	7.0682207	6.3610606	12.3846627	12.9441297
Area	-0.0239383	-0.0061112	1.5387096	0.9978879
Elevation	0.3194648	0.2475620	0.0208787	0.1035103
Nearest	0.0091440	0.3591556	0.7734382	-0.6685801
Scruz	-0.2405242	-0.1952416	-0.1198915	-0.0068681
Adjacent	-0.0748048	-0.0545535	-0.1967502	-0.0018112

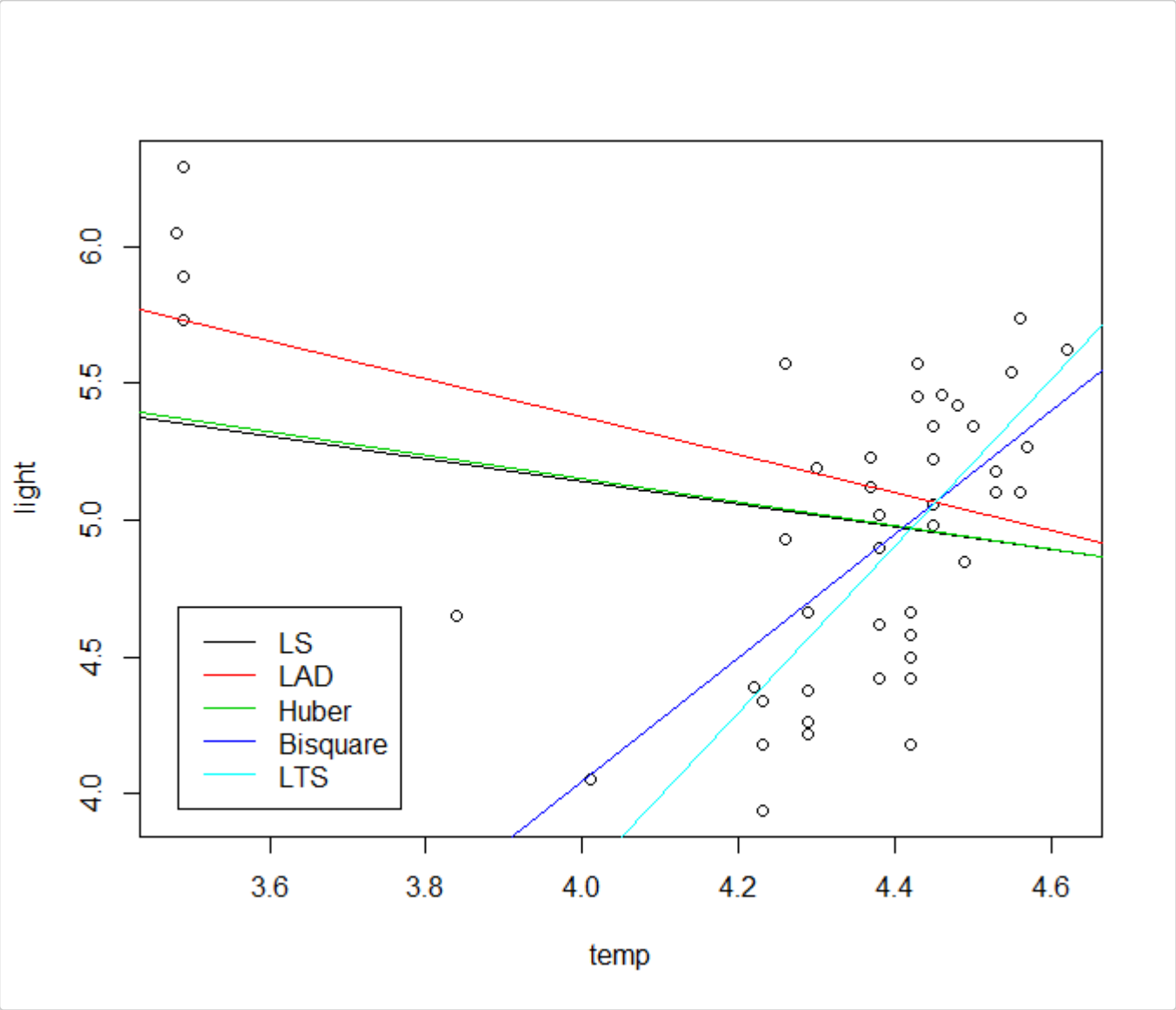
- We see that LTS and LTS-exact appear to agree with each other and both are very different different from OLS.
- All three M-estimation methods, Huber, Bisquare, and Hample are different from each other, and different from OLS and both LTS's.
- LAD is similar to OLS.
- LTS is recommended since it has the best breakdown.
- LTS-exact may take a lot of CPU time for even a moderate size dataset.

Another comparison of several regression methods

- Using the *star* data from the *faraway* package.

```
data(star)

plot(light ~ temp, star)

abline(lm(light ~ temp, star)$coef, col=1) # LS

abline(rq(star$light ~ star$temp)$coef, col=2) # LAD

abline(rlm(light ~ temp, psi=psi.huber, init="lts", star)$coef, col=3) # Huber
```

```
abline(rlm(light ~ temp,  psi = psi.bisquare, init="lts", star)$coef, col=4) #Tukey Bi-square

abline(ltsReg(light ~ temp, star)$coef, col=5) # LTS

legend("bottomleft",
       c("LS", "LAD", "Huber", "Bisquare", "LTS"),
       col=c(1,2,3,4,5),
       inset = .04,
       lty=1)
```



- Print estimated coefficients in a table for several regression methods.

```
g1 <- lm(light ~ temp, data=star)
g2 <- rlm(light ~ temp, psi=psi.huber, data=star)
g4 <- rlm(light ~ temp,  psi=psi.bisquare, init="lts", data=star)
g3 <- rlm(light ~ temp, psi = psi.hampel, init = "lts", data=star)
g5 <- ltsReg(light ~ temp, data=star) # LTS
g6 <- rq(light ~ temp, data=star) # LAD
coefs <- compareCoefs(g1, g2, g3, g4, se = FALSE)
colnames(coefs) <- c("OLS", "Huber", "Bisquare", "Hample")
coefs
```

```
## Calls:
## 1: lm(formula = light ~ temp, data = star)
## 2: rlm(formula = light ~ temp, data = star, psi = psi.huber)
## 3: rlm(formula = light ~ temp, data = star, psi = psi.hampel, init =
##   "lts")
## 4: rlm(formula = light ~ temp, data = star, psi = psi.bisquare, init =
##   "lts")
##
##             Model 1 Model 2 Model 3 Model 4
## (Intercept)    6.79    6.87    6.79   -4.99
## temp         -0.413  -0.429  -0.413   2.257
```

	OLS	Huber	Bisquare	Hample
(Intercept)	6.7934673	6.8658852	6.7934673	-4.986551
temp	-0.4133039	-0.4285228	-0.4133039	2.257037

Exercises

Using the uswages data from the *faraway* package

Exercise 1

- Create factors, remove NAs
- Process the data to make factors with named levels for race, smsa, and pt
- Replace the indicator variables ne, mw, so, and we with one factor variable called region
- Remove all NAs

Exercise 2

- Show the summary of your final data

Exercise 3

- Compute the OLS fit to model log(wage)
- Perform Shapiro-Wilk test of Normality for the residuals, what is the conclusion?

Exercise 4

- Compute WLS fit to model log(wage) and weights = 1/(1+educ)
- Perform Shapiro-Wilk test of Normality for the residuals, what is the conclusion?

Exercise 5

- Compute Robust fit to model log(wage), using Huber, Hampel, Bisquare, LTS and LAD
- Compare coefficients of the above fits using OLS, WLS, Huber, Hampel, Biquare, LTS, and LAD
- Which would you recoomend?