Generalized Linear Models in R
Frequentist and Bayesian Methods

NO ORDER Kenneth K. Lopiano, Garvesh Raskutti, Dan Yang,
Marybeth Broadbent, and Fernando Bonassi

last modified 12 May 2013

1 Background and Data

In practice, researchers are often interested in assessing the relationship between a response variable y and, in the simplest case, a single covariate x. Suppose we observe n pairs of observations $(x_i, y_i), i = 1, \ldots, n$. For illustration, we have created two examples. Each dataset contains 100 observations of y and x. First, using R we read in the two datasets. One dataset will be used in an example of a Gaussian GLM and the other dataset will be used in an example of a Poisson GLM.

```
> ### Set library to location on computer where the example datasets are
> setwd("K:\\short_courses\\SAMSI UG\\lessonsfinal\\r-datasets")
> ### Import the datasets pointozone and mi
> input.data.gaussian=read.csv("example.gaussian.csv")
> input.data.poisson=read.csv("example.poisson.csv")
> head(input.data.gaussian)

         x        y
1 37.90323 72.59429
2 37.96081 71.25487
3 28.29004 49.34230
4 35.31866 67.68882
5 40.08797 77.80367
6 42.48031 78.54409

> head(input.data.poisson)

         x  y
1 39.21345  9
2 32.90769  1
3 44.06213 20
4 40.46214  6
5 44.71557 25
6 39.30153  3

>
```

2 Generalized Linear Models

Generalized linear models (GLMs) are commonly used to model the relationship between two variables. A GLM consists of three parts. Let $\mu = \mathrm{E}(y)$. The first part is called the *linear predictor,*

$$\eta = \beta_0 + \beta_1 x,$$

and the second part is the *link function,*

$$g(\mu) = \eta,$$

where g is a smooth, monotonic function. The linear predictor defines the relationship between η and the covariate x. Here, we assume a linear relationship between η and x where β_0 is the intercept and β_1 is the slope. The link function is a function that *links* the expected value μ of the response variable to the linear predictor η.

The third component is the random or stochastic component. The stochastic component specifies the distibution of the response variable y. The observations $y_1, \ldots, y_n$ are assumed to be independent and it is assumed that the density of y_i is from the exponential family (e.g., Gaussian, Poisson, binomial, gamma, beta, etc.). Our focus will be on the Gaussian GLM and the Poisson GLM.

3 Gaussian Response

Let $(x_i, y_i), i = 1, \ldots, n$ be a set of observations. Assume the y_i follows a Gaussian distribution with mean μ_i and variance σ^2, the identity link function (i.e., $g(\mu_i) = \mu_i$) and linear predictor $\eta_i = \beta_0 + \beta_1 x_i$. That is,

$$y_i \sim \mathrm{N}(\mu_i, \sigma^2) \tag{3.1}$$

$$\mu_i = \beta_0 + \beta_1 x_i. \tag{3.2}$$

In other words, we assume the response y_i is the realization of a Gaussian random variable with mean $\beta_0 + \beta_1 x_i$ and variance σ^2. (Illustration on board)

As mentioned before, the goal is to assess the relationship between y and x. To do so, we need to estimate the regression parameters β_0 and β_1 and the variance parameter σ^2. In this section we will use the `glm` function in `R` to do this. Essentially, the `glm` function is maximizing the likelihood to estimate the parameters.

First we plot the raw data in Figure 1.

Notice the data appear to be noisy around a straight line. This is exactly what is expected for data arising from a GLM with Gaussian data and an identity link.

Next, we review the syntax of the `glm` command, estimate the parameters and discuss the results displayed below

```
> glm.gaussian= glm(y~x,# model of the form response~covariate+covariate+...
+                   family=gaussian,# family - gaussian or poisson
+                   data=input.data.gaussian) # dataset
> summary(glm.gaussian) # summary of the fit

Call:
glm(formula = y ~ x, family = gaussian, data = input.data.gaussian)
```

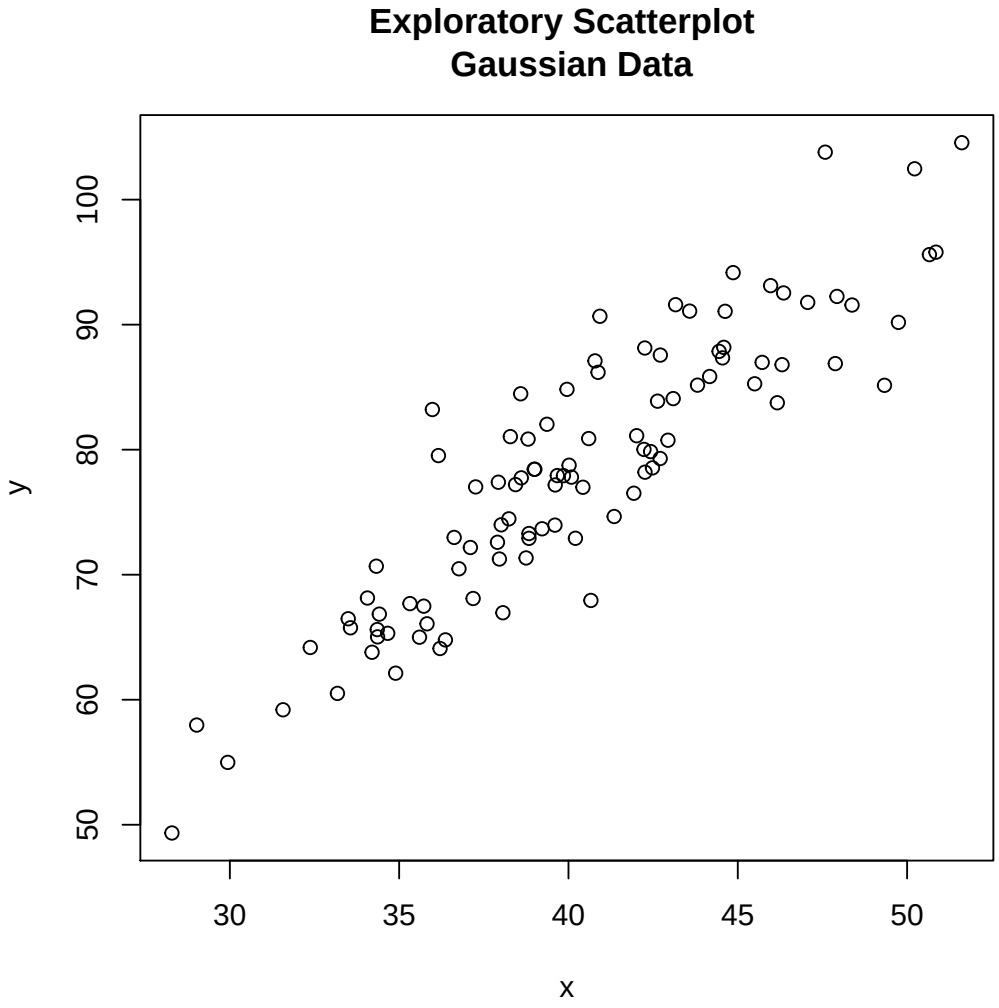


Figure 1: Scatterplot of the raw data from the Gaussian example.

```
Deviance Residuals:
     Min        1Q    Median        3Q       Max
-11.1990   -3.1127   -0.1672    2.5529   13.2658

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.70089    3.79143  -0.185    0.854
x            1.96352    0.09355  20.990   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 21.88232)

    Null deviance: 11785.1  on 99  degrees of freedom
```

```
Residual deviance:  2144.5  on 98  degrees of freedom
AIC: 596.34

Number of Fisher Scoring iterations: 2

> summary(glm.gaussian)$coefficients #summary about the coefficients from the fit

              Estimate Std. Error    t value     Pr(>|t|)
(Intercept) -0.7008861 3.79142754 -0.1848607 8.537203e-01
x            1.9635173 0.09354685 20.9896678 4.866718e-38

> summary(glm.gaussian)$dispersion # dispersion parameter

[1] 21.88232

> coefficients(glm.gaussian)# coefficients from the fit

(Intercept)           x
 -0.7008861   1.9635173
```

Finally, the raw data and the fit are plotted in Figure2.

4 Poisson Response

Let $(x_i, y_i), i = 1, \ldots, n$ be a set of observations. Assume the y_i follows a Poisson distribution with mean μ_i, the log link function (i.e., $g(\mu_i) = \log(\mu_i)$) and linear predictor $\eta_i = \beta_0 + \beta_1 x_i$. Remember, for the Poisson distribution, the mean is equal to the variance. Our model can be written as

$$y_i \sim \text{Poisson}(\mu_i) \tag{4.1}$$
$$\log(\mu_i) = \beta_0 + \beta_1 x_i. \tag{4.2}$$

In other words, we assume the response y_i is the realization of a Gaussian random variable with mean $\beta_0 + \beta_1 x_i$ and variance σ^2. (Illustration on board)

The goal is to assess the relationship between y and x. To do so, we need to estimate the regression parameters β_0 and β_1. As in the previous example, we use the `glm` function in `R`. Essentially, the `glm` function is maximizing the likelihood to estimate the parameters.

First we plot the raw data in Figure 3.

Notice the data do not appear to be linear. This is expected because the linear relationship is on the link scale. That is, the $\log(\mu)$ should have a linear relationship with y.

Next we fit the Poisson regression model using the `glm` command and discuss the results displayed below

```
> glm.poisson= glm(y~x,# model of the form response~covariate+covariate+...
+                  family=poisson,# family - gaussian or poisson
+                  data=input.data.poisson) # dataset
> summary(glm.poisson) # summary of the fit
```

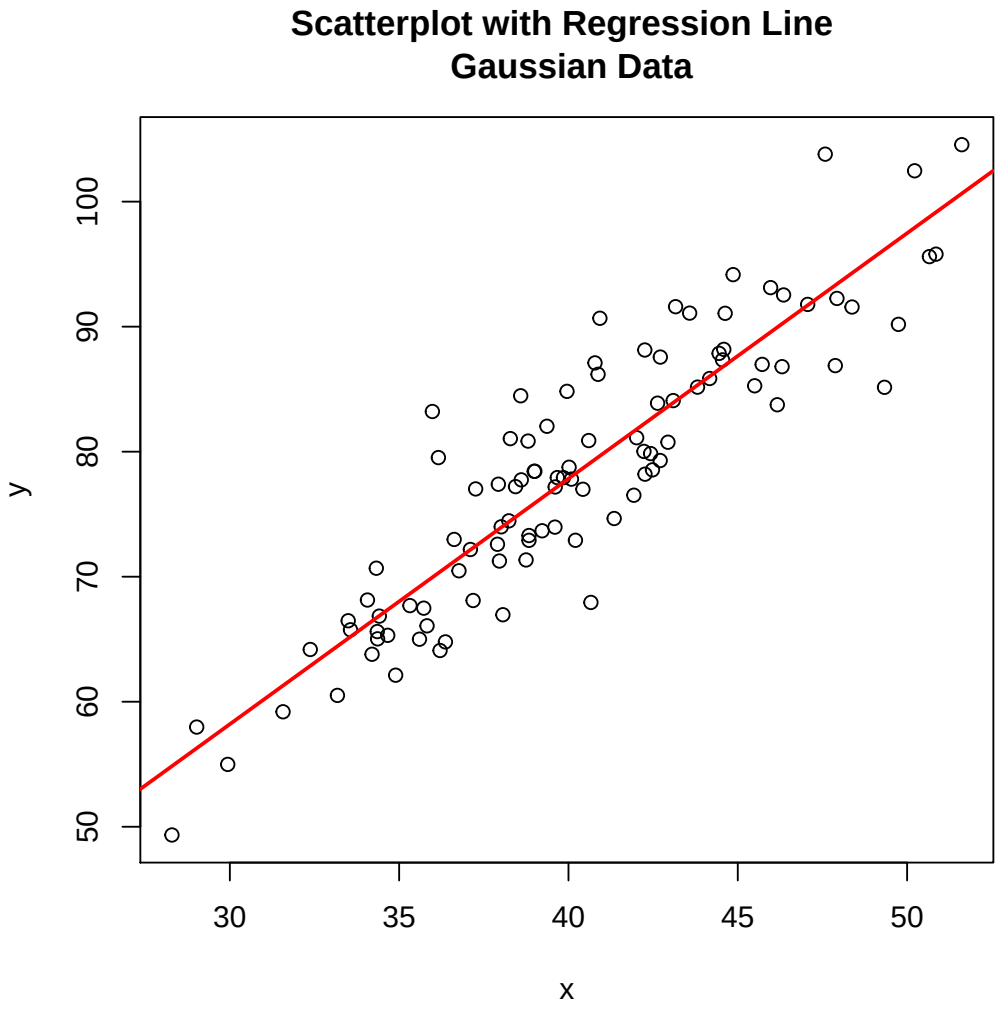


Figure 2: Scatterplot of the raw data from the Gaussian example with the estimated regression line.

```
Call:
glm(formula = y ~ x, family = poisson, data = input.data.poisson)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8384  -0.8599  -0.1023   0.6425   2.5242

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.778652   0.231967  -24.91   <2e-16 ***
x            0.195797   0.005115   38.28   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)
```

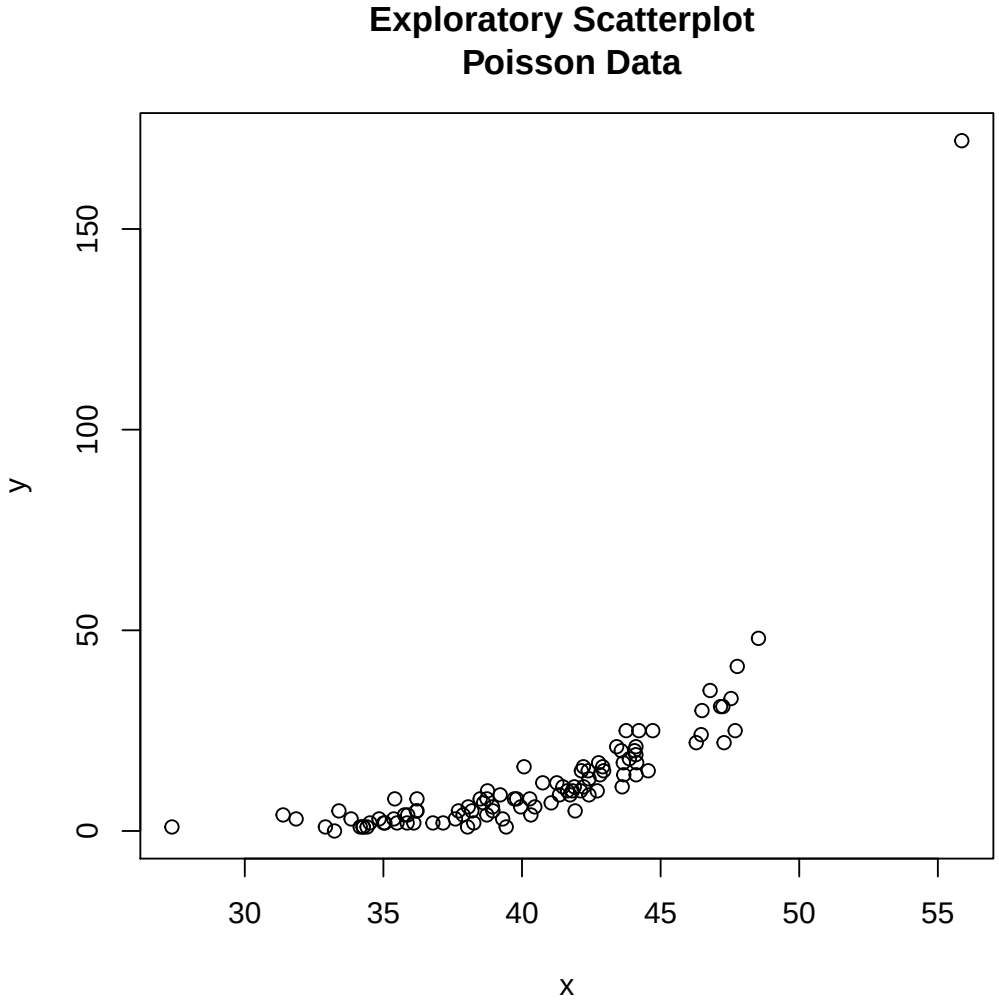


Figure 3: Scatterplot of the raw data from the Poisson example.

```
    Null deviance: 1363.50  on 99  degrees of freedom
Residual deviance:  109.73  on 98  degrees of freedom
AIC: 500.6

Number of Fisher Scoring iterations: 4
```

```
> summary(glm.poisson)$coefficients #summary about the coefficients from the fit
```

```
             Estimate Std. Error   z value      Pr(>|z|)
(Intercept) -5.7786525 0.23196677 -24.91155 5.577161e-137
x            0.1957967 0.00511461  38.28184  0.000000e+00
```

```
> coefficients(glm.poisson)# coefficients from the fit
```

```
(Intercept)           x
 -5.7786525   0.1957967
```

Finally, the raw data and the fit are plotted in Figure 4. The Poisson GLM assumes that a response y is the realization of a Poisson random variable with mean $\exp(\beta_0 + \beta_1 * x)$. The estimate of $\exp(\beta_0 + \beta_1 * x)$ is the red line in Figure 4.

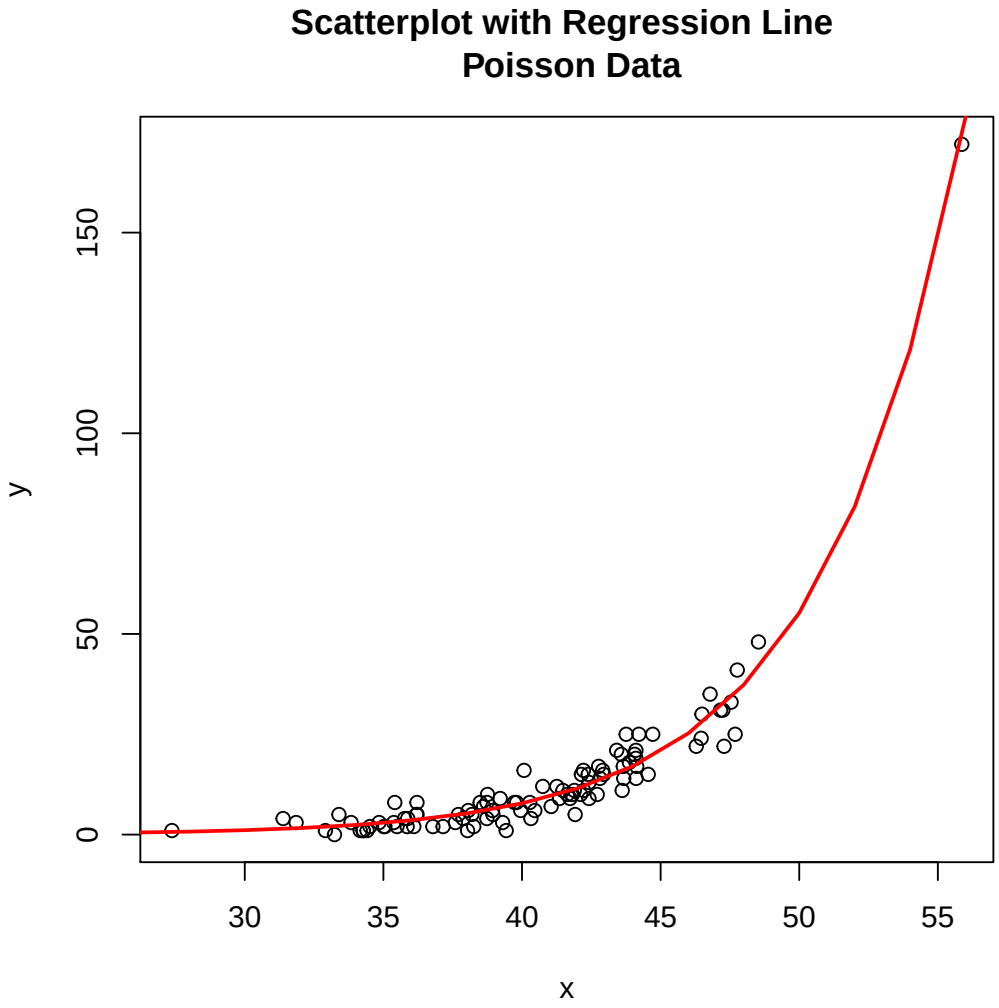


Figure 4: Scatterplot of the raw data from the Poisson example with the estimated regression line.

5 Predictions from `glm` fits

In practice, researchers are ultimately interested in using GLMs to predict y for new values of x. To do this in `R` we use the `predict`. In reality, the `predict` function estimates the expected value of y for the x value of interest.

```
> x.pred = data.frame(x=seq(25,60,1))
> predict.gaussian = predict(glm.gaussian,
+                            type="response",
+                            newdata=x.pred,
```

```
+                              se.fit=TRUE)
> head(predict.gaussian$fit)

       1        2        3        4        5        6
48.38705 50.35056 52.31408 54.27760 56.24112 58.20463

> head(predict.gaussian$se.fit)

       1        2        3        4        5        6
1.498665 1.410094 1.322209 1.235156 1.149123 1.064360

> predict.poisson = predict(glm.poisson,
+                              type="response",
+                              newdata=x.pred,
+                              se.fit=TRUE)
> head(predict.poisson$fit)

        1         2         3         4         5         6
0.4132366 0.5026113 0.6113159 0.7435310 0.9043417 1.0999323

> head(predict.poisson$se.fit)

          1           2           3           4           5           6
0.04387267 0.05088649 0.05889411 0.06800233 0.07831986 0.08995430
```

6 Bayesian Methods for GLMs

In the previous sections, we reviewed what are called Frequentist methods for estimating the parameters in GLMs and obtaining predicted values. Here, we present an alternative approach referred to as a Bayesian approach. The distinction lies in the assumption about the parameters.

In the previous sections, we assumed the response y was random and the parameters β_0, β_1, and σ^2 were fixed and unknown. A Bayesian approach assumes that both the response and the parameters are random. Because the parameters are assumed to be random, we have to assign a distribution to the parameters. The distribution assigned to the parameters is called the *prior distribution*. Any parameters that define the prior distribution are called *hyperparameters*.

6.1 Bayesian GLM with Gaussian Data

A Bayesian version of the Gaussian GLM with an identity link can be described as follows. Let $(x_i, y_i), i = 1, \ldots, n$ be a set of observations. Assume the y_i follows a Gaussian distribution with mean μ_i and variance σ^2, the identity link function (i.e., $g(\mu_i) = \mu_i$) and linear predictor $\eta_i = \beta_0 + \beta_1 x_i$. That is,

$$
\begin{aligned}
y_i &\sim \mathrm{N}(\mu_i, \sigma^2) && (6.1)\\
\mu_i &= \beta_0 + \beta_1 x_i && (6.2)\\
\beta_0 &\sim f_1 && (6.3)\\
\beta_1 &\sim f_2 && (6.4)\\
\sigma^2 &\sim f_3 && (6.5)
\end{aligned}
$$

Notice the model is the same as before, however, now we have assumed β_0, β_1 and σ^2 are realizations from the distributions f_1, f_2, and f_3 respectively. Discussing prior distributions is beyond the scope of today's lesson. Such a model specification is appealing because the *posterior distribution* of the parameters β_0, β_1, and σ^2 given the data y and x can be calculated. Inference about the parameters is done using the posterior distribution.

To implement such a model in `R` we use integrated nested laplace approximations via the `inla` function found in the INLA package. First we load the package.

```
> library(INLA)
```

Then we fit the model using the `inla` command. We will use the default prior distributions in the `INLA` package. Below we fit the model and review the results.

```
> form.inla = y~x # We specify the form of the model here and call it later
> inla.gaussian = inla(form.inla, # the form of the model
+      data=input.data.gaussian, # the dataset
+      family="gaussian") # the family the response comes from
> summary(inla.gaussian)

Call:
"inla(formula = form.inla, family = \"gaussian\", data = input.data.gaussian)"

Time used:
 Pre-processing    Running inla Post-processing           Total
      0.3910010       0.3432000       0.2028010       0.9370019

Fixed effects:
                  mean         sd 0.025quant   0.5quant 0.975quant          kld
(Intercept) -0.7001985 3.77197738  -8.106633 -0.7002333   6.708713 1.950797e-18
x            1.9635002 0.09306695   1.780759  1.9634996   2.146302 3.697785e-32

The model has no random effects

Model hyperparameters:
                                          mean     sd       0.025quant 0.5quant
Precision for the Gaussian observations 0.046573 0.006595 0.034775   0.046190
                                          0.975quant
Precision for the Gaussian observations 0.060619

Expected number of effective parameters(std dev): 2.00(9.669e-06)
Number of equivalent replicates : 50.00

Marginal Likelihood:  -314.86

> inla.gaussian$summary.fixed

                  mean         sd 0.025quant   0.5quant 0.975quant          kld
(Intercept) -0.7001985 3.77197738  -8.106633 -0.7002333   6.708713 1.950797e-18
x            1.9635002 0.09306695   1.780759  1.9634996   2.146302 3.697785e-32
```

```
> inla.gaussian$summary.hyper
```

```
                                                mean          sd 0.025quant
Precision for the Gaussian observations 0.04657298 0.006594991 0.03477503
                                          0.5quant 0.975quant
Precision for the Gaussian observations 0.04618993 0.06061865
```

The raw data and the fit are plotted in Figure 5.

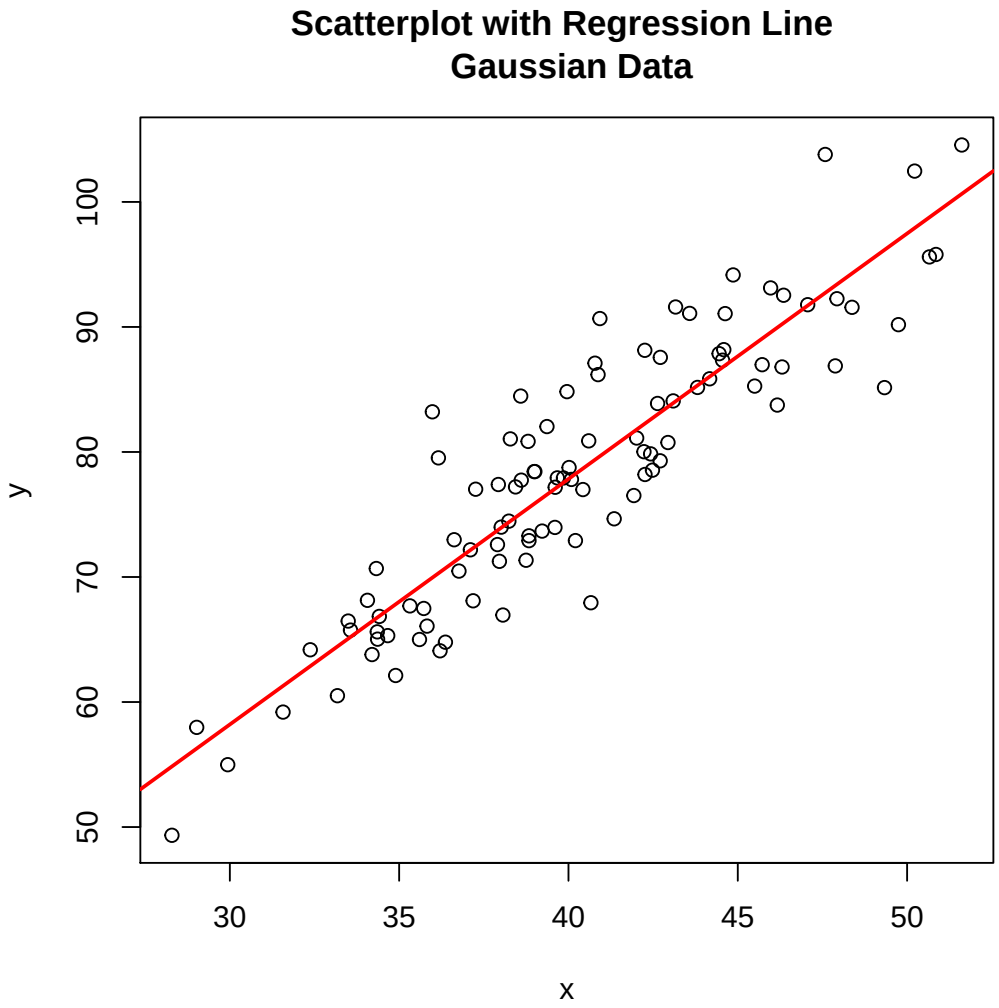


Figure 5: Scatterplot of the raw data from the Gaussian example with the estimated regression line derived from `inla`.

6.2 Bayesian GLM with Poisson Data

Similarly, the Bayesian version of the Poisson GLM described previously can be written as

$$\begin{aligned} y_i &\sim \text{Poisson}(\mu_i) && (6.6)\\ \log(\mu_i) &= \beta_0 + \beta_1 x_i && (6.7)\\ \beta_0 &\sim f_1 && (6.8)\\ \beta_1 &\sim f_2 && (6.9) \end{aligned}$$

We fit the model using the `inla` command. We will use the default prior distributions in the `INLA` package. Below we fit the model and review the results.

```
> form.inla = y~x # We specify the form of the model here and call it later
> inla.poisson = inla(form.inla, # the form of the model
+      data=input.data.poisson, # the dataset
+      family="poisson") # the family the response comes from
> summary(inla.poisson)

Call:
"inla(formula = form.inla, family = \"poisson\", data = input.data.poisson)"

Time used:
 Pre-processing    Running inla Post-processing           Total
      0.1726000       0.2496011       0.0467999       0.4690011

Fixed effects:
                  mean          sd 0.025quant  0.5quant 0.975quant          kld
(Intercept) -5.7783383 0.232132806  -6.233144 -5.778483 -5.3226225 1.425522e-06
x            0.1957985 0.005118827   0.185726  0.195811  0.2058021 1.330064e-10

The model has no random effects

The model has no hyperparameters

Expected number of effective parameters(std dev): 2.019(0.00)
Number of equivalent replicates : 49.53

Marginal Likelihood:  -259.69

> inla.poisson$summary.fixed

                  mean          sd 0.025quant  0.5quant 0.975quant          kld
(Intercept) -5.7783383 0.232132806  -6.233144 -5.778483 -5.3226225 1.425522e-06
x            0.1957985 0.005118827   0.185726  0.195811  0.2058021 1.330064e-10
```

The raw data and the fit are plotted in Figure 6.

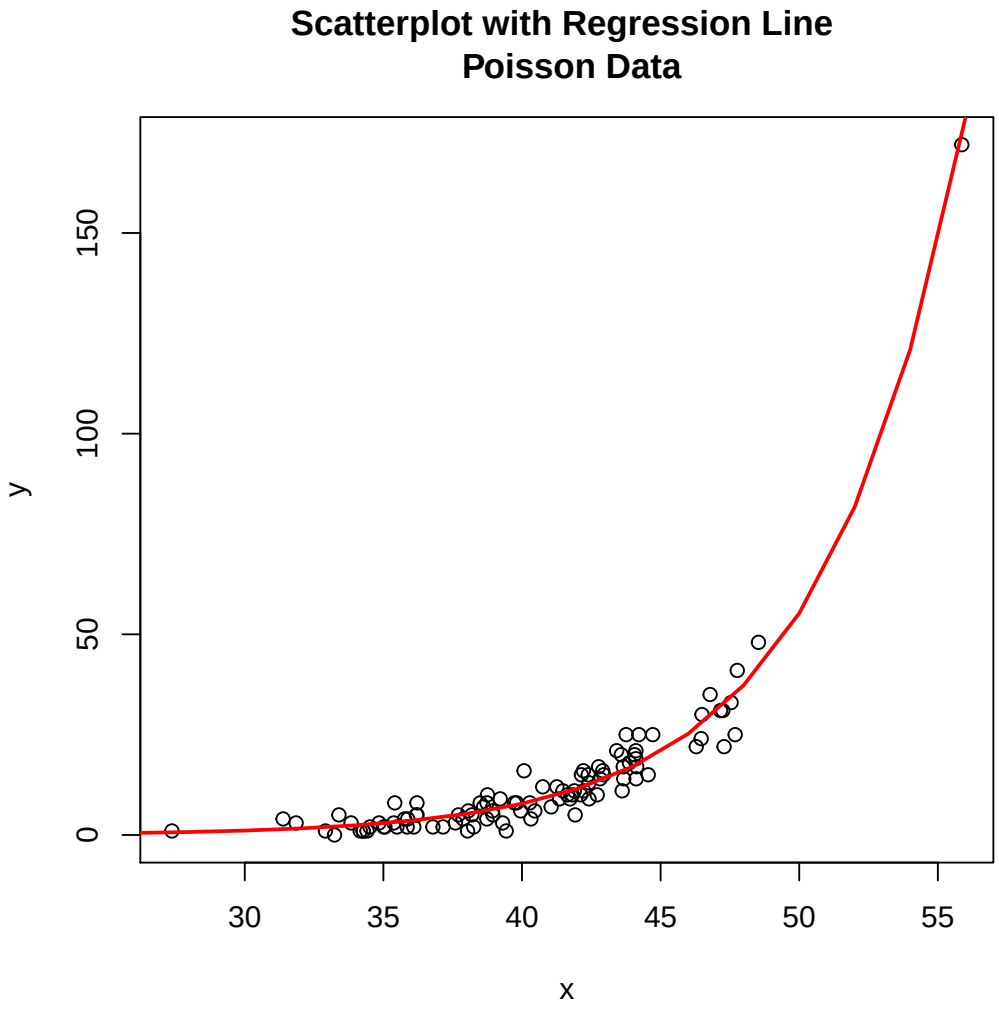


Figure 6: Scatterplot of the raw data from the Poisson example with the estimated regression line derived from `inla`.

6.3 Predictions using `inla`

```
> form.inla = y~x # We specify the form of the model here and call it later
> pred.data.gaussian = rbind(input.data.gaussian,
+                            data.frame(y=rep(NA,31),x=seq(25,55,1)))
> inla.gaussian.pred = inla(form.inla, # the form of the model
+                       data=pred.data.gaussian, # the dataset
+                       family="gaussian",# the family the response comes from
+                       control.predictor=list(compute=TRUE,link=1))
> head(inla.gaussian.pred$summary.fitted.values[-c(1:100),])

                          mean       sd 0.025quant 0.5quant 0.975quant
fitted.predictor.101 48.38751 1.491189   45.45986 48.38740   51.31618
fitted.predictor.102 50.35100 1.403061   47.59637 50.35090   53.10659
fitted.predictor.103 52.31449 1.315615   49.73154 52.31439   54.89834
```

```
fitted.predictor.104 54.27798 1.228997   51.86509 54.27789   56.69171
fitted.predictor.105 56.24147 1.143395   53.99664 56.24138   58.48707
fitted.predictor.106 58.20496 1.059055   56.12571 58.20488   60.28492
```

```
> form.inla = y~x # We specify the form of the model here and call it later
> pred.data.poisson = rbind(input.data.poisson,
+                           data.frame(y=rep(NA,31),x=seq(25,55,1)))
> inla.poisson.pred = inla(form.inla, # the form of the model
+                      data=pred.data.poisson, # the dataset
+                      family="poisson",# the family the response comes from
+                      control.predictor=list(compute=TRUE,link=1))
> head(inla.poisson.pred$summary.fitted.values[-c(1:100),])
```

```
                          mean         sd 0.025quant  0.5quant 0.975quant
fitted.predictor.101 0.4157191 0.04429762  0.3354417 0.4134427  0.5089273
fitted.predictor.102 0.5053714 0.05134128  0.4119329 0.5028670  0.6130393
fitted.predictor.103 0.6143735 0.05937933  0.5058479 0.6116326  0.7384773
fitted.predictor.104 0.7469049 0.06851861  0.6211474 0.7439224  0.8896207
fitted.predictor.105 0.9080486 0.07886812  0.7626889 0.9048241  1.0717534
fitted.predictor.106 1.1039864 0.09053599  0.9364274 1.1005252  1.2912523
```

7 Model Selection

7.1 GLM with Multiple Predictors

Suppose there are p predictors,

$$\eta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p.$$

Throughout this part, we consider an subset of the hurricane dataset and use 5 variables as predictors as an example. For Gaussian or Poisson GLMs, the models are as following respectively.

$$\begin{aligned} y_i &\sim \mathrm{N}(\mu_i, \sigma^2) && (7.1)\\ \mu_i &= \beta_0 + \beta_1 x_{i1} + ... + +\beta_5 x_{i5}, && (7.2)\\ y_i &\sim \mathrm{Poisson}(\mu_i) && (7.3)\\ \log(\mu_i) &= \beta_0 + \beta_1 x_{i1} + ... + +\beta_5 x_{i5}. && (7.4) \end{aligned}$$

We load the data by the following command

```
> rm(list=ls())
> final.data = read.csv("K:\\short_courses\\SAMSI UG\\lessonsfinal\\r-datasets\\example.h
> #subset to only five predictors for illustration
> final.data = final.data[,c("count","amon.csv","limsta.csv",
+                            "dm.csv","ggst.csv","ammsst.csv")]
> head(final.data)
```

```
  count amon.csv limsta.csv dm.csv ggst.csv ammsst.csv
1     4   -0.044     -0.449  -0.02      -15       0.21
2     1    0.433     -0.440   0.52       -1       2.87
3     2    0.384     -0.522   0.24        7       2.36
4     6    0.355     -0.505   0.25        0       1.45
5     7   -0.033     -0.880   0.47      -17       0.39
6     5    0.280     -0.637   0.39       -8       2.70
```

Before any modelling takes place, one should visualized the data first; see Figure 7 for a matrix of scatterplots and Figure 8 for a histogram of the response variable. Note that some of the variables are highly correlated. For instance, amon and ammsst, or dm and ammsst. Further note that the range of the response variable is nonnegative, which prohibits us from using Gaussian model on the counts directly. We should either run the GLM on the counts with Poisson model or on the transformed counts with Gaussian model. For most of the time, logarithm transformation is recommended when dealing with nonnegative variables, which maps the range $[0, \infty)$ to $(-\infty, \infty)$ and makes Guassian model sensible. In this session, we will stick with Poisson regression and leave the Gaussian experiment to you.

7.2 What is Model Selection and Why Model Selection?

What is model selection? Simply put, one wants to select the best model among several choices based on evaluation of the performance of the models.

There are mainly two principles for model selection, or to judge which one is best:

- Goodness of fit, i.e., how close is the theory (model) close to the reality (data).
- Parsimony, i.e., simplicity of the model which essentially boils down to the number of variables in the model for GLM.

One immediate question is how to measure goodness of fit? There are many choices. The simplest one is residual sum of squares

$$\sum (y_i - \hat{y}_i)^2$$

, where y_i is the observed response and $\hat{y}_i$ is the fitted value. This measure is also called mean squared error, and commonly used as the default measure. For GLM, *deviance* is a quality of fit for a model that depends on the likelihood. We will skip the details and the underlying theory of the definition, but pointing out that the statistical software can compute the quantity automatically.

The next question is why we should consider model selection? There are several reasons. First, people tend to believe or can understand simpler models with fewer predictors and less complicated structure. Second, one can certainly add more and more features into the model without screening and get better and better fit, till perfect fit, but the problem is overfitting. Remember that we want to find the best-predicting model not the best fitting model. See Figure **??** for an intuitive explanation.

Now that we understand the significance of model selection, how do we perform it? We will introduce two methods, AIC and CV.

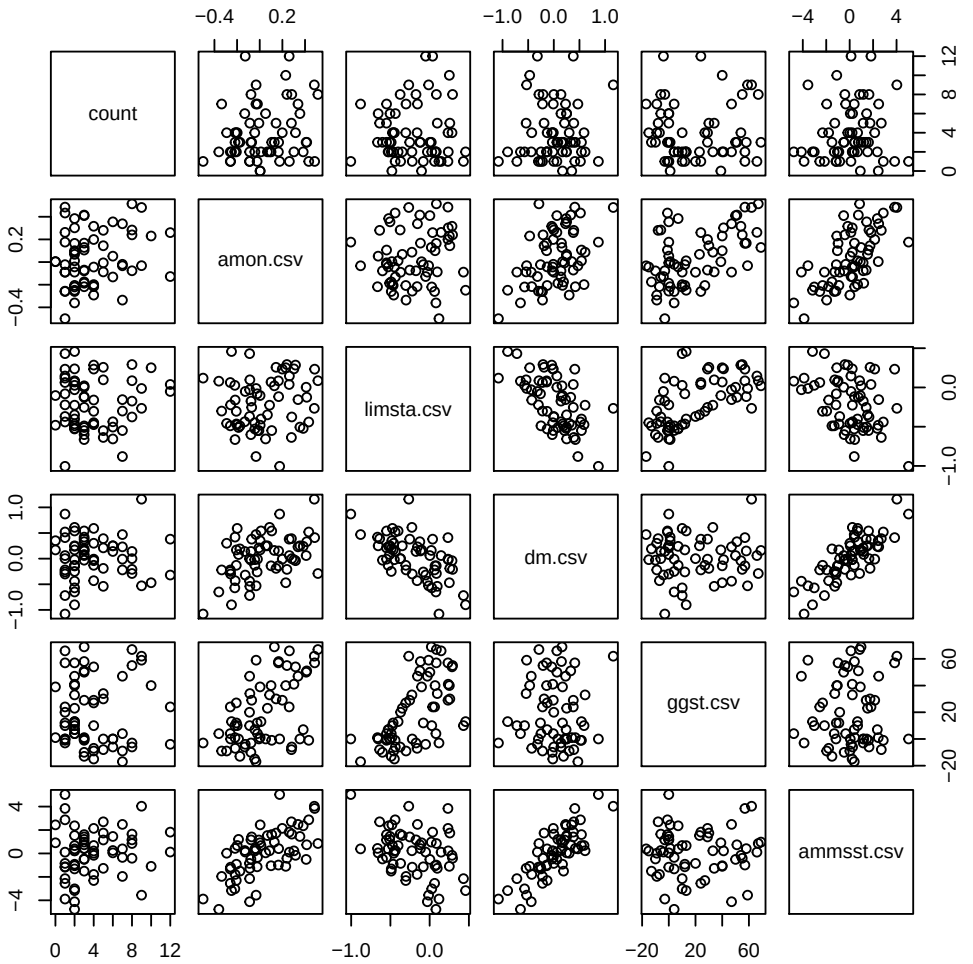


Figure 7: Scatterplot of the hurricane.

7.3 AIC

Akaike Information Criterion (AIC) as a method of model selection.

$$AIC = -2 \log f(y|\hat{\theta}) + 2K$$

, where the first term is the goodness of fit and the second term is model complexity. The model that minimizes AIC should be preferred.

Bayesian Information Criterion (BIC) is similar to AIC, but differs in the way for penalizing model complexity

$$AIC = -2 \log f(y|\hat{\theta}) + K \log n$$

, where n is number of observations.

Simply looking at the definitions of AIC and BIC, one can conclude that BIC will choose models more parsimonious that AIC since it penalized the model complexity more.

Essentially, we want to select the simplest model that describes the data sufficiently well. AIC and BIC try to estimate the model's ability to fit all "future" unseen data samples from the same

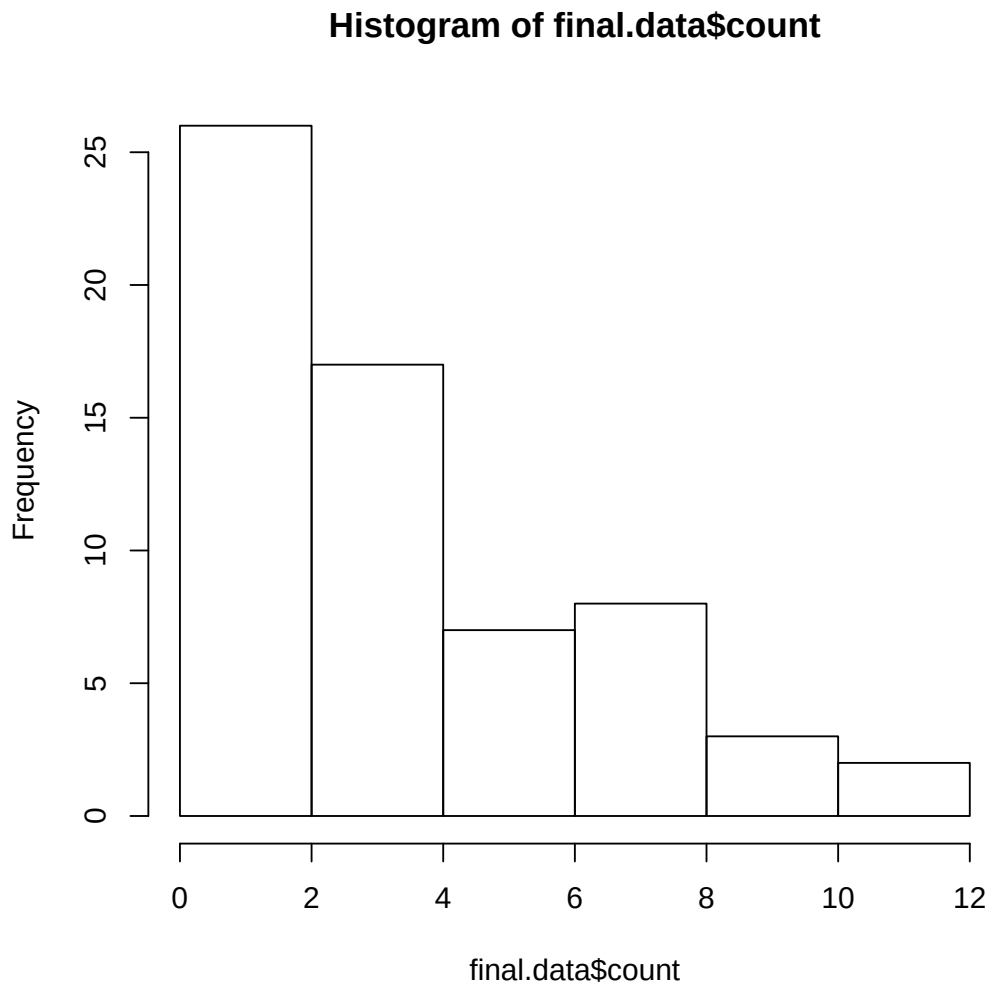


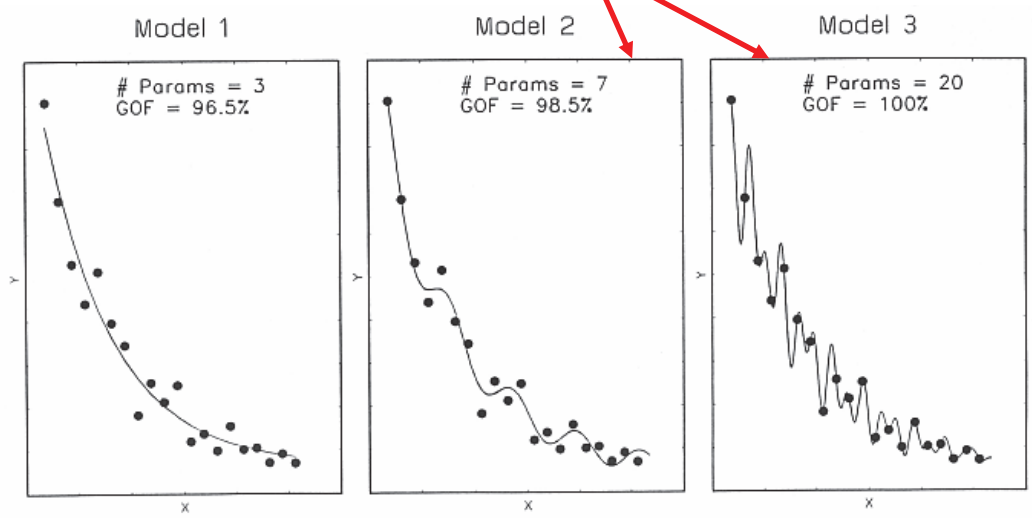
Figure 8: Histogram of the response variable in the hurricane dataset example.

underlying process, not just the current data sample, i.e., it focuses on the prediction accuracy. See Figure **??** for a comparision of the goodness of fit and predictive accuracy.

One key issue is there are many models for us to choose from in reality. For a problem with 5 predictors, there are $2^5 = 32$ models to select among. In general, 2^p models for a problem with p predictors, which grows exponentially fast and prevents us from looping through all the possible combinations. There are three main approaches to get a solution, that is probably not optimal, but computationally feasible. Those approaches are all stepwise procedure, which seeks a local optimal at each step.

- Forward selection, which involves starting with no variables in the model, and keeps adding variables that lowers AIC the most, until none of them improves the model.
- Backward elimination, which starts with all the candidate variables, and deletes one variable at a time if that the current model without that variable has the lowest AIC.
- Bidirectional elimination, a combination of the above two, at each step, decides whether to

Fit improves with more parameters (i.e, over-fitting)



Model 1: $Y = ae^{-bX} + c$

Model 2: $Y = ae^{-bX} + c + dX^{-e} \cdot \sin(f \cdot X + g)$

include or exclude one variable.

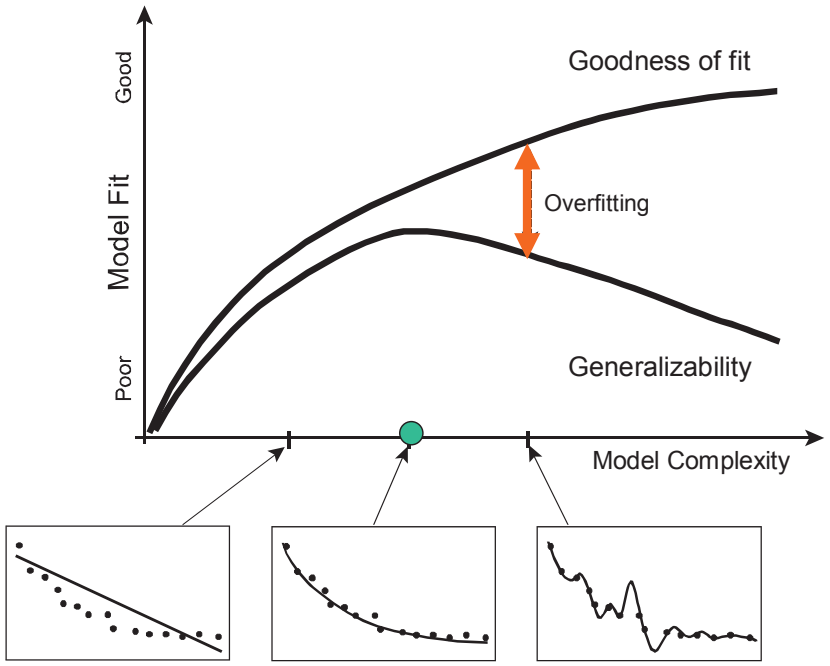
The stepwise procedure can be implemented in R by using stepAIC.

```
> library(MASS)
> temp.glm = glm(count~.,data=final.data,
+                family=poisson)
> stepAIC(temp.glm,
+         k=log(nrow(final.data)),
+         direction="both")

Start:  AIC=329.93
count ~ amon.csv + limsta.csv + dm.csv + ggst.csv + ammsst.csv

             Df Deviance    AIC
- ammsst.csv  1   118.89 325.79
- limsta.csv  1   118.92 325.82
```

Relationship between Goodness of Fit and Generalizability




```
- dm.csv      1   119.14 326.04
- ggst.csv    1   120.23 327.13
<none>            118.88 329.93
- amon.csv    1   123.95 330.86

Step:  AIC=325.79
count ~ amon.csv + limsta.csv + dm.csv + ggst.csv

             Df Deviance    AIC
- limsta.csv  1   118.92 321.67
- dm.csv      1   119.39 322.15
- ggst.csv    1   120.44 323.19
<none>            118.89 325.79
- amon.csv    1   126.79 329.55
+ ammsst.csv  1   118.88 329.93
```

```
Step:  AIC=321.67
count ~ amon.csv + dm.csv + ggst.csv

             Df Deviance    AIC
- dm.csv      1   120.32 318.94
- ggst.csv    1   121.14 319.76
<none>            118.92 321.67
+ limsta.csv  1   118.89 325.79
+ ammsst.csv  1   118.92 325.82
- amon.csv    1   127.24 325.86

Step:  AIC=318.94
count ~ amon.csv + ggst.csv

             Df Deviance    AIC
- ggst.csv    1   121.67 316.14
<none>            120.32 318.94
+ dm.csv      1   118.92 321.67
- amon.csv    1   127.43 321.90
+ limsta.csv  1   119.39 322.15
+ ammsst.csv  1   119.58 322.34

Step:  AIC=316.14
count ~ amon.csv

             Df Deviance    AIC
<none>            121.67 316.14
- amon.csv    1   127.53 317.86
+ ggst.csv    1   120.32 318.94
+ dm.csv      1   121.14 319.76
+ ammsst.csv  1   121.63 320.25
+ limsta.csv  1   121.66 320.28

Call:  glm(formula = count ~ amon.csv, family = poisson, data = final.data)

Coefficients:
(Intercept)     amon.csv
      1.328        0.639

Degrees of Freedom: 62 Total (i.e. Null);  61 Residual
Null Deviance:            127.5
Residual Deviance: 121.7         AIC: 311.9
```

The first argument, temp.glm, is the initial model in the stepwise search, which we take to be the full model in this example. One can certainly try out others, since for these methods, initial point matters. Furthermore, we are setting k to be the logarithm of the number of observations, which

actually implements BIC although the function is called stepAIC. Last, the direction argument can also be set as "backward" or "forward".

One little suggestion before we move on to the next method for model selection. Although looping through all the 2^p models is intimidating, but one could try to search over all the 2^q models when q is moderate and q is the number of variables that are selected by some methods at first.

7.4 CV

Cross-Validation (CV) is a technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. There is no explicit measure of model complexity, unlike AIC.

One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.

In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k ??? 1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data.

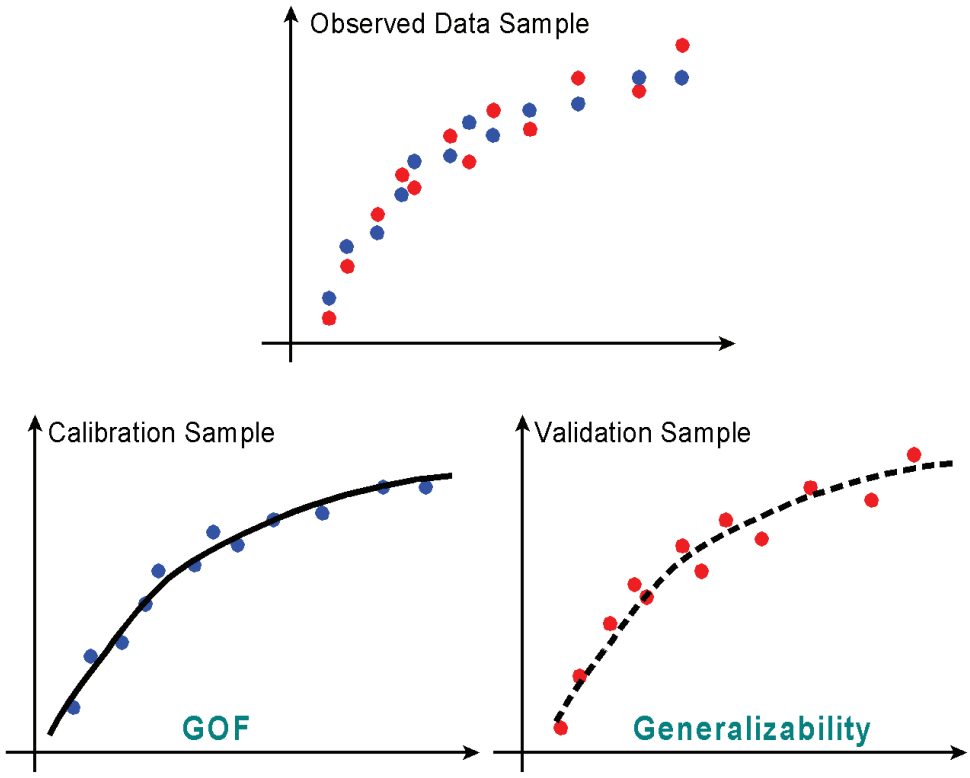
More specifically in GLM, suppose we separate the data into two parts $(x_{training}, y_{training})$ and $(x_{validation}, y_{validation})$. For a given model, We first estimate the parameters β's based on $x_{training}, y_{training})$, then we use β's and $x_{validation}$ to get prediction $\hat{y}_{validation}$. Finally, we measure how close the predicted values $\hat{y}_{validation}$ and the observed values $y_{validation}$ are. See Figure **??** for an illustration.

How do we use CV to select a model? We compute the cross-validated error for each candidate model and choose the one with smallest error.

Note that CV is another way to prevent overfitting because it tries to minimize the prediction errors for holdout data.

There is a function in R, which is included in the package called, boot, that can do CV on GLM for us. It can be used as following.

```
> temp.glm1 = glm(count~.,data=final.data,
+                 family=poisson)
> temp.glm2 = glm(count~amon.csv,data=final.data,
+                 family=poisson)
> temp.glm3 = glm(count~1,data=final.data,
+                 family=poisson)
> library(boot)
> set.seed(410)
> cost <- function(y,yhat,eps=0.0001) mean((log(y+eps) - log(yhat+eps))^2)
> cv.out1 = cv.glm(final.data,
+                  temp.glm1,
+                  cost,
+                  K=10)$delta[1]
```



Amsterdam04 31

```
> cv.out2 = cv.glm(final.data,
+                  temp.glm2,
+                  cost,
+                  K=10)$delta[1]
> cv.out3 = cv.glm(final.data,
+                  temp.glm3,
+                  cost,
+                  K=10)$delta[1]
> print(c(cv.out1,cv.out2,cv.out3))

[1] 4.094255 4.077484 4.139473
```

Here we considered three candidate models, one with all five predictors, one with only one predictor that is selected based on BIC, and one with no predictors. The second model produces smallest CV error and the null model with nothing performs the worst. Note that if one does the experiment

again, the result might be different from what we have seen because of the randomness of the random splitting of the data.