INDIAN STATISTICAL INSTITUTE
Saptarshi Sinha(bs2031)

Acknowledgement

I want to thank Prof. Arnab Chakraborty for providing detailed intuitive explanations about the topic and neat visuals in his web page. I also took help from the web-pages https://www.rdocumentation.org/packages/msme/versions/0.5.3/topics/irls and https://stat.ethz.ch/R-manual/R-patched/library/base/html/chol.html to write about the different R functions in use.
To know more about which decomposition is used in glm() function, I visited https://pages.stat.wisc.edu/ st849-1/Rnotes/ModelMatrices.html–Section:3.2

1 IRLS

In the previous section, we have seen the use of MLE to find the estimates of the unknown parameters. Now, solving MLEs is not easy as it seems since the equations are sometimes too complicated to be solved by hand. In those situations, we take help from iterative methods, specifically the **"Newton-Raphson Method"**. Now this also doesn't make our lives easy, because the Newton-Raphson method also has some "problems".

Newton-Raphson method is very sensitive to the choice of the initial value of the root(term is in *Numerical analysis*), i.e. if the initial guess is close enough to the actual answer, then we expect to get that actual answer(obviously upto a certain precision). But if the initial guess is far away, then we may not get our desired result(problems include divergence, extreme slow convergence or convergence to some other nearer root).

To tackle this problem (in some simpler situations though*), we need some intermediate process which will lead to that "close guess" given any arbitrary guess. Fortunately an algorithm exists which is robust to the initial guess of the answer and quickly gives the initial guess for Newton-Raphson Method. This method is called the **IRLS**(Iterated Re-weighted Least Squares), whose naming will be justified later.

1.1 Setup

The setup is like this:

$$\vec{Y}_{nx1} \;=\; X_{nxp}\vec{\beta}_{px1} + \vec{\epsilon}_{nx1}\ , \qquad \vec{\epsilon} \sim \left(\vec{0}, \sigma^2 R(\vec{\theta})\right) \tag{1}$$

where $R(\vec{\theta})$ is an (n x n) Positive-Definite(PD) matrix, $\sigma > 0$ and the covariance matrix $\Sigma(\vec{\theta})$ is expressible in the above form(this is the simpler situation mentioned earlier). To go ahead, we need two more assumptions:

(i) Given a particular value of $\vec{\theta}$, $(R(\vec{\theta}))^{1/2}$ (this matrix always exist for PD matrices) can be calculated analytically, i.e. no approximate solution(far away from precision range).

(ii) If any realization(numerical value) of $\vec{\epsilon}$ is given, then we can solve for $\vec{\theta}$ analytically, without any further use of iterative methods.

1.2 Detailed Algorithm

The algorithm for finding the initial guess for Newton-Raphson Method from any initial start is given as:

1) Start with any $\vec{\theta}_0$. Thus we have the covariance matrix of the form $\sigma^2 P_0$, where $P_0 = R(\vec{\theta}_0)$ is a known PD matrix, the exact setup to invoke GLS.

2) At the ith iteration, apply GLS to find $\widehat{\vec{\beta}_{(i-1)}}$ and $\widehat{\sigma^2_{(i-1)}}$ in the following ways:

   i) Find $P^{1/2}_{(i-1)} = (R(\vec{\theta}_{i-1}))^{1/2}$ using the Cholesky decomposition or by other methods. [*This comes from the first assumption*]

   ii) Pre-multiply both sides of eqn.(1) by the inverse of $P^{1/2}_{(i-1)}$, i.e. $P^{-1/2}_{(i-1)}$ to reduce it into a Gauss-Markov setup.

iii) Find $\widehat{\vec{\beta}_{(i-1)}}$ from the Gauss-Markov theorem. [*The $\vec{y}$ and the design matrix will be changed in the reduced model*]

iv) Evaluate $\widehat{\sigma^2_{(i-1)}}$ by using the formula $\hat{\sigma^2} = \left\|\vec{y'} - X'\hat{\vec{\beta}}\right\|^2 / (n - rank(X'))$, where $\vec{y'}, X'$ are the obtained values and Design-matrix respectively in the reduced equation.

3) Get a realization of the errors using, $\widehat{\vec{\epsilon_i}} = \vec{y} - X\widehat{\vec{\beta}_{(i-1)}}$

4) We solve for $\vec{\theta_i}$ [*Using second assumption*] and **Goto Step 2**.

Continue these iterations until a sequence of reasonable answers(with very little difference in values) are obtained. Take the last iteration value of $\vec{\beta}$, σ and $\vec{\theta}$. Generally 20 - 30 steps are enough for finding a satisfactory answer.

1.3 Justification

Though the proof of the convergence is outside the scope of this book, we do justify the name of this algorithm in order to remember it firmly. In this method, we are iteratively using GLS, but every time we are changing the $R(\theta)$ matrix which acted as a weight multiplied to eqn.(1). So, we are re-weighting the equation at every iteration to find the least square estimate(s). Hence the name is justified.

R Corner

The **"chol(X, pivot = FALSE, tol = -1, $\cdots$)"**(default values can be omitted while writing the function) returns the Upper-triangular factor of the "Cholesky Decomposition" in R. The default function assumes that X is a symmetric PD matrix and return R such that $R^TR = X$. In case X is not PD, pivot= TRUE with some tolerance level(in the error) is entered as argument.

One great news is that R has a separate functions dedicated for this particular algorithm. The **"irls(formula, data, family, link, tol = 1e-06, $\cdots$)"** is one of the functions which returns the estimates of the parameters. The formula argument contains a *"symbolic representation of the model to be fit"*, data contains the variable(s) in the model. The error distribution is considered in family and the link function(like logit or probit links) is mentioned in link. The tol controls the tolerance level(convergence criterion for the change in deviance between the answers).

Another function that can be used is **"glm(formula, family, data, $\cdots$)"**. The default methods for lm() and glm() functions use the **"QR decomposition"**, created by the qr() function. However, the Chol() functions returns R which is same as the "R_1" in QR decomposition. To know more, visit https://pages.stat.wisc.edu/ st849-1/Rnotes/ModelMatrices.html–Section:3.2