SDIO Relaxivity Resampling

Rohan Kapre

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

SDIO Resampling Simulation

In this Markdown, we will resample the data on SDIO Relaxivity to evaluate the various fitting methods. Note that the data is given such that C is the concentration of [Fe] in mM and T2 is in seconds.

```
library(tidyr)
library(mblm)
SDIO = read.table("SDIO_Relaxivity_Resampling_Data", header = T)
SDIOT2 = data.frame(C = SDIO$C, T2 = SDIO$T2, Trial = SDIO$Trial)  #C is in mM, T2 in s
R2 = 1/SDIOT2$T2
C = SDIOT2$C

SDIOT2
```

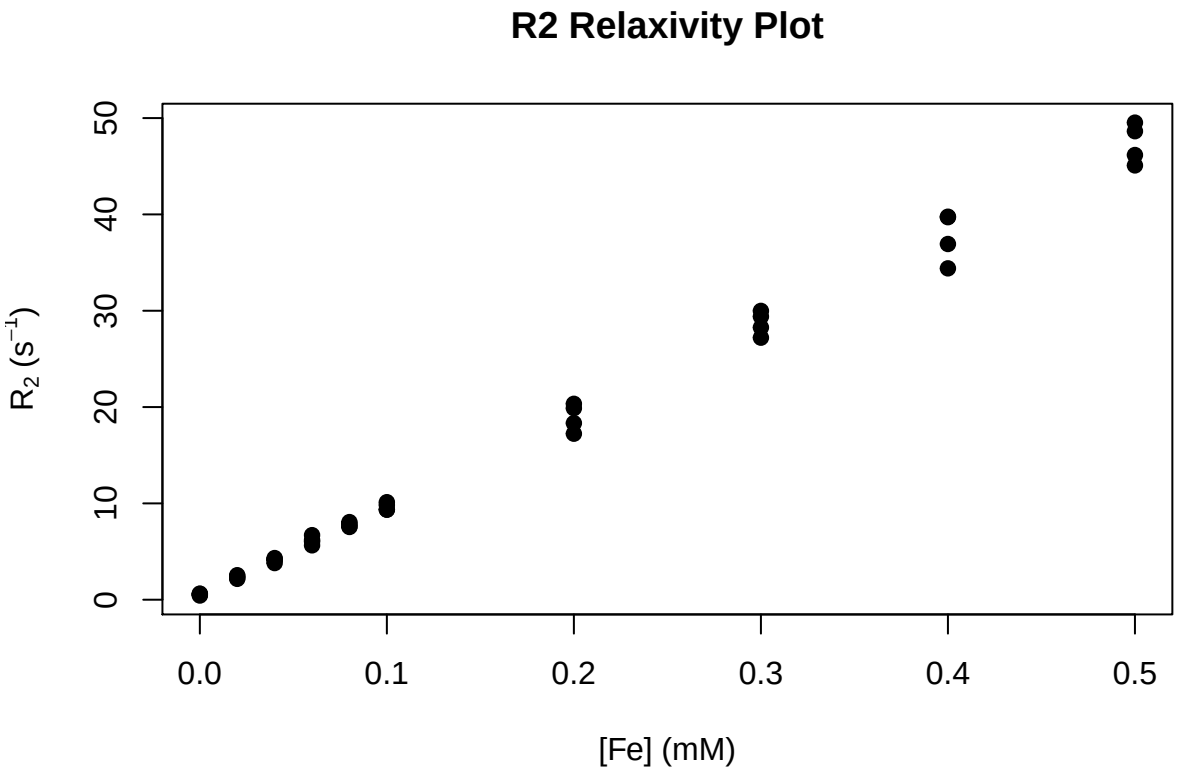
```
##       C      T2 Trial
## 1  0.00 1.67000     1
## 2  0.02 0.41000     1
## 3  0.04 0.24400     1
## 4  0.06 0.16560     1
## 5  0.08 0.12700     1
## 6  0.10 0.10175     1
## 7  0.20 0.05029     1
## 8  0.30 0.03402     1
## 9  0.40 0.02515     1
## 10 0.50 0.02056     1
## 11 0.00 1.91000     2
## 12 0.02 0.39300     2
## 13 0.04 0.23130     2
## 14 0.06 0.16130     2
## 15 0.08 0.12410     2
## 16 0.10 0.09877     2
## 17 0.20 0.04917     2
## 18 0.30 0.03335     2
## 19 0.40 0.02518     2
## 20 0.50 0.02019     2
## 21 0.00 1.59000     3
## 22 0.02 0.44300     3
## 23 0.04 0.26290     3
## 24 0.06 0.17720     3
## 25 0.08 0.13240     3
## 26 0.10 0.10707     3
## 27 0.20 0.05455     3
```

```
## 28 0.30 0.03538     3
## 29 0.40 0.02708     3
## 30 0.50 0.02166     3
## 31 0.00 2.30000     4
## 32 0.02 0.46200     4
## 33 0.04 0.23840     4
## 34 0.06 0.14920     4
## 35 0.08 0.13130     4
## 36 0.10 0.10680     4
## 37 0.20 0.05803     4
## 38 0.30 0.03676     4
## 39 0.40 0.02907     4
## 40 0.50 0.02218     4
```

Quick Plot

First, lets visualize the data with a quick plot. It is clear that the spread is increasing with the mean R2

```
plot(C, R2, main = "R2 Relaxivity Plot", xlab = "[Fe] (mM)",
    ylab = expression(paste(R[2], " ", "(", s^-1, ")")), pch = 19)
```



Full Relaxivity Fitting

Before we do resampling, we need to fit each of the models individually to all of the data points. This gives the relaxivity on the full dataset for each model that we can compare to to obtain the Bias and coverage/Type I Errors for the 95% CIs based on the t-distribution for the resampling simulation

```
OLSfull <- lm(R2 ~ C, data = SDIOT2)
NLSfull <- nls(T2 ~ 1/(a + r2 * C), start = list(a = 0.5, r2 = 100),
    data = SDIOT2)

WLSfull <- lm(R2 ~ C, weights = 1/R2^2, data = SDIOT2)
NWLSfull <- nls(T2 ~ 1/(a + r2 * C), start = list(a = 0.5, r2 = 100),
    weights = 1/T2^2, data = SDIOT2)

LNLSfull <- nls(log(R2) ~ log(a + r2 * C), start = list(a = 0.5,
    r2 = 100), data = SDIOT2)
GGLM.INVfull <- glm(T2 ~ C, family = Gamma(link = inverse), data = SDIOT2)
GGLM.IDfull <- glm(R2 ~ C, family = Gamma(link = identity), data = SDIOT2)

TSfull <- mblm(R2 ~ C)

r2full <- data.frame(OLS = coef(OLSfull)[2], NLS = coef(NLSfull)[2],
    WLS = coef(WLSfull)[2], NWLS = coef(NWLSfull)[2], LNLS = coef(LNLSfull)[2],
    GGLM.INV = coef(GGLM.INVfull)[2], GGLM.ID = coef(GGLM.IDfull)[2],
    TS = coef(TSfull)[2])
rownames(r2full) <- "Relaxivity"

r2full
```

```
##                 OLS      NLS      WLS     NWLS    LNLS GGLM.INV  GGLM.ID
## Relaxivity 93.65493 90.44717 91.75031 92.17208 91.9496  91.8813 92.02091
##                 TS
## Relaxivity 92.0761
```

Convert to Wide Format

For resampling, it is easier to work with the data organized in a Wide Format so that the T2 can be resampled at each concentration

```
SDIOT2.Wide = spread(SDIOT2, Trial, T2)

SDIOT2.Wide
```

```
##       C       1       2       3       4
## 1  0.00 1.67000 1.91000 1.59000 2.30000
## 2  0.02 0.41000 0.39300 0.44300 0.46200
## 3  0.04 0.24400 0.23130 0.26290 0.23840
## 4  0.06 0.16560 0.16130 0.17720 0.14920
## 5  0.08 0.12700 0.12410 0.13240 0.13130
## 6  0.10 0.10175 0.09877 0.10707 0.10680
## 7  0.20 0.05029 0.04917 0.05455 0.05803
## 8  0.30 0.03402 0.03335 0.03538 0.03676
```

```
## 9  0.40 0.02515 0.02518 0.02708 0.02907
## 10 0.50 0.02056 0.02019 0.02166 0.02218
```

Each row above contains the T2 values (in s) at each [Fe] concentration (mM) in the dataset by Trial 1-4.Next, we need to get ready to resample one T2 value in each row.

```
Csamp = SDIOT2.Wide$C  #This extracts the concentrations for resampling which are the same

ONESAMP <- function(x) {

    sample(x, size = 1, replace = F)  #This makes the function to resample 1 object from a vector
}

Csamp
```

```
##  [1] 0.00 0.02 0.04 0.06 0.08 0.10 0.20 0.30 0.40 0.50
```

Relaxivity T2 Resampling Prep

In this section, we first prepare data frames to store resampling results. Data frames must be prepared to store resampled relaxivities as well as their associated variance directly from the fits, which will be used to construct 95% CIs in the end for each run. We will be doing 100K runs.

```
runs = 1e+05

r2resamp = rep(NaN, runs)
r2boot <- data.frame(OLS = r2resamp, NLS = r2resamp, WLS = r2resamp,
    NWLS = r2resamp, LNLS = r2resamp, GGLM.INV = r2resamp, GGLM.ID = r2resamp,
    TS = r2resamp)

Varresamp = rep(NaN, runs)
Varboot <- data.frame(OLS = Varresamp, NLS = Varresamp, WLS = Varresamp,
    NWLS = Varresamp, LNLS = Varresamp, GGLM.INV = Varresamp,
    GGLM.ID = Varresamp, TS = Varresamp)

head(r2boot)
```

```
##   OLS NLS WLS NWLS LNLS GGLM.INV GGLM.ID  TS
## 1 NaN NaN NaN  NaN  NaN      NaN     NaN NaN
## 2 NaN NaN NaN  NaN  NaN      NaN     NaN NaN
## 3 NaN NaN NaN  NaN  NaN      NaN     NaN NaN
## 4 NaN NaN NaN  NaN  NaN      NaN     NaN NaN
## 5 NaN NaN NaN  NaN  NaN      NaN     NaN NaN
## 6 NaN NaN NaN  NaN  NaN      NaN     NaN NaN
```

```
head(Varboot)
```

```
##   OLS NLS WLS NWLS LNLS GGLM.INV GGLM.ID  TS
## 1 NaN NaN NaN  NaN  NaN      NaN     NaN NaN
## 2 NaN NaN NaN  NaN  NaN      NaN     NaN NaN
## 3 NaN NaN NaN  NaN  NaN      NaN     NaN NaN
## 4 NaN NaN NaN  NaN  NaN      NaN     NaN NaN
## 5 NaN NaN NaN  NaN  NaN      NaN     NaN NaN
## 6 NaN NaN NaN  NaN  NaN      NaN     NaN NaN
```

Relaxivity Resampling

The seed will be set to the randomly determined value of 2628 to ensure reproducibility of the RNG.The coefficients of each fit in each run will be stored in the r2boot dataframe while the variance of the relaxivity from the variance-covariance matrix (4th element) will be stored in the Varboot dataframe.For nonlinear fitting methods, starting values were set to 0.5 for the water relaxation rate and 100 for relaxivity to ensure convergence in the 100K iterations. Starting values in practice for nonlinear relaxivity fits can always be obtained via preliminary OLS fits.

```
set.seed(2628)
for (i in 1:runs) {

    T2resamp <- apply(SDIOT2.Wide[2:5], 1, ONESAMP)
    R2resamp = 1/T2resamp

    SDIO.Resamp <- data.frame(C = Csamp, T2 = T2resamp, R2 = R2resamp)

    OLSboot <- lm(R2 ~ C, data = SDIO.Resamp)
    NLSboot <- nls(T2 ~ 1/(a + r2 * C), start = list(a = 0.5,
        r2 = 100), data = SDIO.Resamp)

    WLSboot <- lm(R2 ~ C, weights = 1/R2^2, data = SDIO.Resamp)
    NWLSboot <- nls(T2 ~ 1/(a + r2 * C), start = list(a = 0.5,
        r2 = 100), weights = 1/T2^2, data = SDIO.Resamp)

    LNLSboot <- nls(log(R2) ~ log(a + r2 * C), start = list(a = 0.5,
        r2 = 100), data = SDIO.Resamp)
    GGLM.INVboot <- glm(T2 ~ C, family = Gamma(link = inverse),
        data = SDIO.Resamp)
    GGLM.IDboot <- glm(R2 ~ C, family = Gamma(link = identity),
        data = SDIO.Resamp)

    TSboot <- mblm(R2 ~ C, data = SDIO.Resamp)

    r2boot$OLS[i] = coef(OLSboot)[2]
    r2boot$NLS[i] = coef(NLSboot)[2]

    r2boot$WLS[i] = coef(WLSboot)[2]
    r2boot$NWLS[i] = coef(NWLSboot)[2]

    r2boot$LNLS[i] = coef(LNLSboot)[2]
    r2boot$GGLM.INV[i] = coef(GGLM.INVboot)[2]
    r2boot$GGLM.ID[i] = coef(GGLM.IDboot)[2]

    r2boot$TS[i] = coef(TSboot)[2]


    Varboot$OLS[i] = vcov(OLSboot)[4]
    Varboot$NLS[i] = vcov(NLSboot)[4]

    Varboot$WLS[i] = vcov(WLSboot)[4]
    Varboot$NWLS[i] = vcov(NWLSboot)[4]

    Varboot$LNLS[i] = vcov(LNLSboot)[4]
```

```
    Varboot$GGLM.INV[i] = vcov(GGLM.INVboot)[4]
    Varboot$GGLM.ID[i] = vcov(GGLM.IDboot)[4]

    Varboot$TS[i] = vcov(TSboot)[4]
}

head(r2boot)
```

```
##        OLS      NLS      WLS     NWLS     LNLS GGLM.INV  GGLM.ID       TS
## 1 98.21917 86.51313 91.52818 92.41976 91.98484 91.83463 92.13209 96.78383
## 2 97.70004 82.69907 90.41718 91.57427 91.01034 90.81521 91.20146 96.98389
## 3 97.86533 98.08751 93.55147 94.68762 94.13182 93.94031 94.31953 96.06452
## 4 92.28653 94.17228 90.50279 91.43066 90.96985 90.81453 91.12412 91.78010
## 5 90.36578 87.74189 92.28316 92.86164 92.56853 92.47265 92.66551 93.88965
## 6 90.40462 84.66077 89.42773 89.97732 89.69829 89.60733 89.79053 91.08128
```

```
head(Varboot)
```

```
##         OLS      NLS      WLS     NWLS     LNLS GGLM.INV  GGLM.ID
## 1 3.1247731 3.291423 4.039914 3.809757 3.925950 4.029528 3.818238
## 2 0.5608704 1.908415 5.790286 5.358000 5.570650 5.715835 5.410068
## 3 3.6063350 3.149941 5.008803 4.834354 4.926196 5.071626 4.777734
## 4 5.0857258 8.410739 4.144717 4.152192 4.148065 4.182500 4.111046
## 5 2.0021158 4.339258 2.789250 2.846647 2.814198 2.781157 2.850787
## 6 1.7167249 1.904544 2.517178 2.570758 2.540092 2.499195 2.585851
##          TS
## 1 3.5580631
## 2 0.6267436
## 3 4.2078365
## 4 5.1251859
## 5 3.9269640
## 6 1.7967006
```

Now that the resampling simulation is complete, we need to calculate the results such as the mean resampled relaxivity, bias relative to the full relaxivity, mean predicted SE of the relaxivity fit, actual SD of the resampled relaxivity, and Type I Error rates.

The t-distribution with 10-2=8 degrees of freedom will be used to construct 95% CIs for the relaxivities in each fit. The full relaxivity is expected ideally to be in 95% of the constructed CIs, so the ideal Type I Error rate is 5%.

Results

```
r2boot.bar = apply(r2boot, 2, mean)

r2boot.SD = apply(r2boot, 2, sd)

SEboot = sqrt(Varboot)

meanSE = apply(SEboot, 2, mean)
```

```
tcrit = qt(0.975, 8)
r2boot.high = r2boot + tcrit * SEboot
r2boot.low = r2boot - tcrit * SEboot

head(r2boot.high)  #High end of 95% CI for each fit (first 6 rows only)
```

```
##         OLS       NLS      WLS     NWLS     LNLS GGLM.INV  GGLM.ID
## 1 102.29550  90.69675 96.16315 96.92076 96.55395 96.46363 96.63810
## 2  99.42704  85.88471 95.96612 96.91206 96.45302 96.32836 96.56512
## 3 102.24451 102.18022 98.71239 99.75786 99.25001 99.13349 99.36000
## 4  97.48693 100.85999 95.19748 96.12958 95.66644 95.53058 95.79971
## 5  93.62868  92.54550 96.13443 96.75233 96.43699 96.31833 96.55903
## 6  93.42604  87.84317 93.08635 93.67468 93.37352 93.25286 93.49872
##          TS
## 1 101.13361
## 2  98.80949
## 3 100.79483
## 4  97.00063
## 5  98.45936
## 6  94.17228
```

```
head(r2boot.low)  #Low end of 95% CI for each fit (first 6 rows only)
```

```
##        OLS      NLS      WLS     NWLS     LNLS GGLM.INV  GGLM.ID       TS
## 1 94.14284 82.32951 86.89322 87.91876 87.41572 87.20563 87.62609 92.43406
## 2 95.97305 79.51344 84.86824 86.23648 85.56765 85.30205 85.83780 95.15829
## 3 93.48615 93.99479 88.39055 89.61737 89.01364 88.74712 89.27906 91.33421
## 4 87.08613 87.48457 85.80809 86.73173 86.27325 86.09849 86.44854 86.55957
## 5 87.10287 82.93828 88.43189 88.97094 88.70008 88.62697 88.77199 89.31994
## 6 87.38321 81.47836 85.76911 86.27997 86.02306 85.96181 86.08234 87.99029
```

Above is the data frame with the high and low end of the calculated 95% CI for relaxivity with each fit. Next, we need to determine MSE and Type I Error rates. PSESD represents the proportion of times the fit SE underestimates the actual resampled SD. Ideally it should have an equal probability (50%) of being above or below.

```
Bias = r2boot.bar - r2full

MSE = Bias^2 + r2boot.SD^2

Type1Err.OLS = 1 - mean(r2full$OLS < r2boot.high$OLS & r2full$OLS >
    r2boot.low$OLS)
Type1Err.NLS = 1 - mean(r2full$NLS < r2boot.high$NLS & r2full$NLS >
    r2boot.low$NLS)

Type1Err.WLS = 1 - mean(r2full$WLS < r2boot.high$WLS & r2full$WLS >
    r2boot.low$WLS)
Type1Err.NWLS = 1 - mean(r2full$NWLS < r2boot.high$NWLS & r2full$NWLS >
    r2boot.low$NWLS)

Type1Err.LNLS = 1 - mean(r2full$LNLS < r2boot.high$LNLS & r2full$LNLS >
```

```
    r2boot.low$LNLS)
Type1Err.GGLM.INV = 1 - mean(r2full$GGLM.INV < r2boot.high$GGLM.INV &
    r2full$GGLM.INV > r2boot.low$GGLM.INV)
Type1Err.GGLM.ID = 1 - mean(r2full$GGLM.ID < r2boot.high$GGLM.ID &
    r2full$GGLM.ID > r2boot.low$GGLM.ID)

Type1Err.TS = 1 - mean(r2full$TS < r2boot.high$TS & r2full$TS >
    r2boot.low$TS)

Type1Err <- data.frame(OLS = Type1Err.OLS, NLS = Type1Err.NLS,
    WLS = Type1Err.WLS, NWLS = Type1Err.NWLS, LNLS = Type1Err.LNLS,
    GGLM.INV = Type1Err.GGLM.INV, GGLM.ID = Type1Err.GGLM.ID,
    TS = Type1Err.TS)

PSESD.OLS = mean(SEboot$OLS < r2boot.SD[1])
PSESD.NLS = mean(SEboot$NLS < r2boot.SD[2])

PSESD.WLS = mean(SEboot$WLS < r2boot.SD[3])
PSESD.NWLS = mean(SEboot$NWLS < r2boot.SD[4])

PSESD.LNLS = mean(SEboot$LNLS < r2boot.SD[5])
PSESD.GGLM.INV = mean(SEboot$GGLM.INV < r2boot.SD[6])
PSESD.GGLM.ID = mean(SEboot$GGLM.ID < r2boot.SD[7])

PSESD.TS = mean(SEboot$TS < r2boot.SD[8])


PSESD <- data.frame(OLS = PSESD.OLS, NLS = PSESD.NLS, WLS = PSESD.WLS,
    NWLS = PSESD.NWLS, LNLS = PSESD.LNLS, GGLM.INV = PSESD.GGLM.INV,
    GGLM.ID = PSESD.GGLM.ID, TS = PSESD.TS)
```

Final Results and Histograms

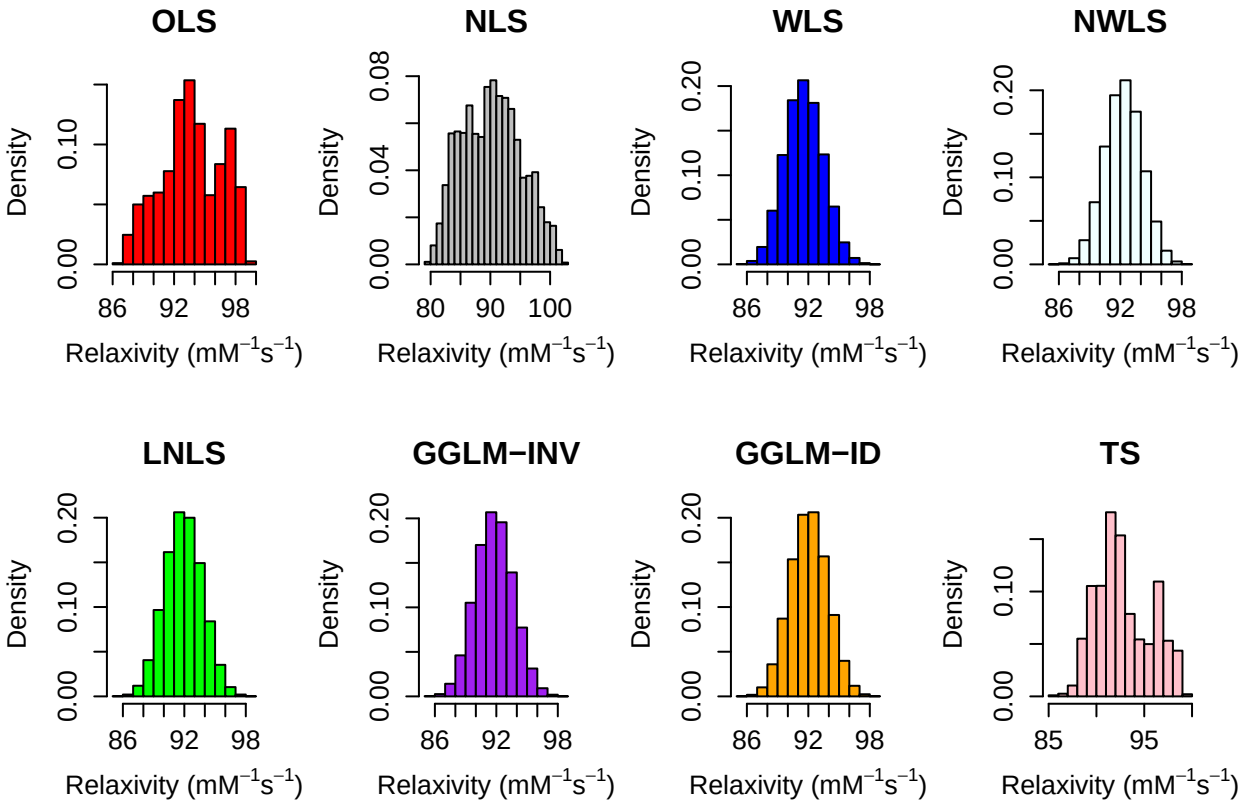
All of the calculated results can be displayed in a final nicely organized data frame. Histograms for the relaxivity are also shown

```
Result <- rbind(Full = r2full, Boot = r2boot.bar, SE = meanSE,
    SD = r2boot.SD, Bias = Bias, MSE = MSE, Type1Err = Type1Err,
    PSESD = PSESD)

t(Result)
```

```
##              Full     Boot       SE       SD         Bias       MSE
## OLS      93.65493 93.66418 1.808774 2.922434  0.009250334  8.540704
## NLS      90.44717 90.25954 1.695549 4.890053 -0.187637459 23.947822
## WLS      91.75031 91.55600 1.775579 1.855705 -0.194314129  3.481400
## NWLS     92.17208 92.28409 1.767747 1.819233  0.112015753  3.322157
## LNLS     91.94960 91.92156 1.771406 1.834298 -0.028040942  3.365437
## GGLM.INV 91.88130 91.79988 1.776218 1.840667 -0.081421758  3.394683
## GGLM.ID  92.02091 92.04278 1.766362 1.828502  0.021872237  3.343897
## TS       92.07610 92.83014 2.110942 2.869174  0.754041819  8.800741
##          Type1Err   PSESD
```

```
## OLS       0.28158 0.94254
## NLS       0.49350 1.00000
## WLS       0.05472 0.58547
## NWLS      0.05344 0.55444
## LNLS      0.05389 0.56655
## GGLM.INV  0.05389 0.56872
## GGLM.ID   0.05380 0.56563
## TS        0.18960 0.84357
```

```
par(mfrow = c(2, 4))
hist(r2boot$OLS, cex.main = 1.5, cex.axis = 1.2, freq = F, cex.lab = 1.3,
    col = "red", main = "OLS", xlab = expression(paste("Relaxivity",
        " ", "(", mM^-1, s^-1, ")")))
hist(r2boot$NLS, cex.main = 1.5, cex.axis = 1.2, freq = F, cex.lab = 1.3,
    col = "grey", main = "NLS", xlab = expression(paste("Relaxivity",
        " ", "(", mM^-1, s^-1, ")")))
hist(r2boot$WLS, cex.main = 1.5, cex.axis = 1.2, freq = F, cex.lab = 1.3,
    col = "blue", main = "WLS", xlab = expression(paste("Relaxivity",
        " ", "(", mM^-1, s^-1, ")")))
hist(r2boot$NWLS, cex.main = 1.5, cex.axis = 1.2, freq = F, cex.lab = 1.3,
    col = "azure", main = "NWLS", xlab = expression(paste("Relaxivity",
        " ", "(", mM^-1, s^-1, ")")))
hist(r2boot$LNLS, cex.main = 1.5, cex.axis = 1.2, freq = F, cex.lab = 1.3,
    col = "green", main = "LNLS", xlab = expression(paste("Relaxivity",
        " ", "(", mM^-1, s^-1, ")")))
hist(r2boot$GGLM.INV, cex.main = 1.5, cex.axis = 1.2, freq = F,
    cex.lab = 1.3, col = "purple", main = "GGLM-INV", xlab = expression(paste("Relaxivity",
        " ", "(", mM^-1, s^-1, ")")))
hist(r2boot$GGLM.ID, cex.main = 1.5, cex.axis = 1.2, freq = F,
    cex.lab = 1.3, col = "orange", main = "GGLM-ID", xlab = expression(paste("Relaxivity",
        " ", "(", mM^-1, s^-1, ")")))
hist(r2boot$TS, cex.main = 1.5, cex.axis = 1.2, freq = F, cex.lab = 1.3,
    col = "pink", main = "TS", xlab = expression(paste("Relaxivity",
        " ", "(", mM^-1, s^-1, ")")))
```



Conclusion

Using OLS leads to high Type I Error rates of 25-30%, which can lead to erroneous conclusions when using linear models with MRI Relaxometry Data. Furthermore, the relaxivities from repeated sampling tend to follow asymmetric, bimodal distributions. The results suggest that using GGLMs, LNLS, NWLS, or WLS lead to more consistent relaxivity estimates and Type I Error rates near the ideal 5%.

Monte Carlo Relaxivity Simulation: No Conc Error

Kapre et al.

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Setting True r2 Relaxivity

For this simulation, the true r2 relaxivity will be set to $r_2 = 100\ mM^{-1}s^{-1}$. Concentrations of [Fe] ranging from 0-0.08 mM spaced by 0.02 mM and 0.1-0.5 mM spaced by 0.1 mM will be tested. The CV for T_2^p will be set to 2.5% across all values. We will also initiatlize the data frame which will store the relaxivities calculated via each fitting method

```
library(mblm)

r2=100

R20=0.5

C=c(seq(0,0.08,0.02),seq(0.1,0.5,0.1))

n=length(C)

CV=0.025

runs=100000

a1=rep(NaN,runs)

r2all<-data.frame("OLS"=a1,"NLS"=a1,"WLS"=a1,"NWLS"=a1,"LNLS"=a1,"GGLMINV"=a1,"GGLMID"=a1,"TS"=a1)

r2var=r2all
```

Actual Simulation

For this Monte Carlo Simulation, we will have no concentration errors considered.The SD for T_2 will be chosen to be $CV * T_2^p$ where $CV = 0.025$. $p \sim N(1.18, 0.12^2)$ and $T_2 \sim N(T_2^{true}, (CV * T_2^{true})^2)$. Technically, $R_2 = 1/T_2$ will not be normal after the inverse transformation.

```
set.seed(9177)
for(i in 1:runs){


Ctrue=C #For this simulation, the true concentration is exactly the set concentration

T2true=1/(r2*Ctrue+R20)
```

```r
p=rnorm(n,1.18,0.12)  #Generate a power p to account for deviations from "perfect" CV in T2

T2obs=rnorm(n,T2true,CV*T2true^p)  #This is to test normal distribution with SD=CV*T2true^p


R2obs=1/T2obs


OLS<-lm(R2obs~C)

NLS<-nls(T2obs~(b0+b1*C)^(-1),start=list(b0=0.4,b1=95))

WLS<-lm(R2obs~C,weights=1/R2obs^2)

NWLS<-nls(T2obs~(b0+b1*C)^(-1),start=list(b0=0.4,b1=95),weights=1/T2obs^2)

LNLS<-nls(log(R2obs)~log(b0+b1*C),start=list(b0=0.4,b1=95))

GGLMINV<-glm(T2obs~C,family=Gamma(link=inverse))

GGLMID<-glm(R2obs~C,family=Gamma(link=identity))

TS<-mblm(R2obs~C,repeated = T)

r2all$OLS[i]=coef(OLS)[2]
r2all$NLS[i]=coef(NLS)[2]

r2all$WLS[i]=coef(WLS)[2]
r2all$NWLS[i]=coef(NWLS)[2]

r2all$LNLS[i]=coef(LNLS)[2]
r2all$GGLMINV[i]=coef(GGLMINV)[2]
r2all$GGLMID[i]=coef(GGLMID)[2]

r2all$TS[i]=coef(TS)[2]

r2var$OLS[i]=vcov(OLS)[4]
r2var$NLS[i]=vcov(NLS)[4]

r2var$WLS[i]=vcov(WLS)[4]
r2var$NWLS[i]=vcov(NWLS)[4]

r2var$LNLS[i]=vcov(LNLS)[4]
r2var$GGLMINV[i]=vcov(GGLMINV)[4]
r2var$GGLMID[i]=vcov(GGLMID)[4]

r2var$TS[i]=vcov(TS)[4]

}

head(r2all)
```

```
##          OLS       NLS       WLS      NWLS      LNLS   GGLMINV    GGLMID
```

```
## 1  99.81842 101.30002 100.52370 100.59982 100.56129 100.54867 100.57404
## 2 100.19155  98.19790  99.76997  99.80178  99.78602  99.78069  99.79130
## 3 100.21543 102.05965  99.86187  99.99344  99.92858  99.90652  99.95037
## 4 100.33897  99.78897 100.03135 100.05100 100.04119 100.03792 100.04447
## 5  99.24889 101.56001  99.72159  99.76415  99.74275  99.73568  99.74986
## 6 102.08126  98.59848  99.65444  99.77323  99.71350  99.69375  99.73335
##         TS
## 1 99.80551
## 2 99.87388
## 3 99.30771
## 4 99.84513
## 5 99.10060
## 6 99.18735
```

```
head(r2var)
```

```
##          OLS        NLS       WLS       NWLS       LNLS    GGLMINV
## 1 0.20992555 0.18292478 0.3426852 0.35397298 0.34824922 0.34321496
## 2 0.07635732 0.45201402 0.1770739 0.17182331 0.17441623 0.17677800
## 3 0.62838166 0.42442266 0.6189641 0.60093382 0.60975025 0.62098784
## 4 0.07133741 0.02375678 0.0873872 0.08695213 0.08716735 0.08735933
## 5 0.07366146 0.20901859 0.2121427 0.21615767 0.21413930 0.21267097
## 6 1.17749968 0.34629031 0.5278173 0.53589333 0.53184429 0.52743219
##      GGLMID         TS
## 1 0.3534223 0.22096929
## 2 0.1720936 0.09488868
## 3 0.5989607 0.73313280
## 4 0.0869810 0.12322453
## 5 0.2155995 0.07641307
## 6 0.5363115 2.47478659
```

The first few relaxivities and associated variances (from the fit itself) are displayed.

Results

Now that we have obtained the relaxivities for all runs, we go on to calculate mean, SD, bias, MSE, and Type I error rates.

```
r2bar<-apply(r2all,2,mean)
r2sd<-apply(r2all,2,sd) ##True SD from all the fits##

r2SE<-sqrt(r2var) #Estimated fitting SE

Bias=r2bar-r2

MSE=Bias^2+r2sd^2

bool_SESD<-matrix(rep(NaN,dim(r2SE)[1]*dim(r2SE)[2]),nrow=dim(r2SE)[1],ncol=dim(r2SE)[2])
Type1Err=rep(NaN,length(r2bar))


df=n-2 # t-distribution degrees of freedom
```

```r
tcrit=qt(0.975,df) #critical t value at alpha=0.05 2 sided

r2UPPER=r2all+tcrit*r2SE
r2LOWER=r2all-tcrit*r2SE #Forming 95% CIs




for(i in 1:8){

bool_SESD[,i]=r2SE[,i]<=r2sd[i]

Type1Err[i]=mean((r2>r2UPPER[i])|r2<r2LOWER[i]) #Check if r2=100 is contained in 95% CI

}

P_SESD=apply(bool_SESD,2,mean)

Result<-rbind(Bias,r2sd,MSE,P_SESD,Type1Err)
t(Result)
```

```
##                 Bias      r2sd       MSE  P_SESD Type1Err
## OLS       0.020854492 1.1151937 1.2440918 0.94053  0.21679
## NLS       0.010049707 1.6416494 2.6951139 0.99638  0.44176
## WLS      -0.031275232 0.6391708 0.4095175 0.59040  0.04345
## NWLS      0.056566596 0.6422441 0.4156773 0.59550  0.04401
## LNLS      0.012591610 0.6398242 0.4095336 0.59153  0.04344
## GGLMINV -0.002040896 0.6394191 0.4088610 0.59053  0.04336
## GGLMID    0.027239186 0.6404407 0.4109063 0.59238  0.04355
## TS       -0.002200604 0.7379476 0.5445715 0.65565  0.08854
```

Monte Carlo Relaxivity Simulation: 10% Conc Error

Kapre et al.

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Setting True r2 Relaxivity

For this simulation, the true r2 relaxivity will be set to $r_2 = 100\ mM^{-1}s^{-1}$. Concentrations of [Fe] ranging from 0-0.08 mM spaced by 0.02 mM and 0.1-0.5 mM spaced by 0.1 mM will be tested. The CV for T_2^p will be set to 2.5% across all values. We will also initiatlize the data frame which will store the relaxivities calculated via each fitting method

```
library(mblm)

r2=100

R20=0.5

C=c(seq(0,0.08,0.02),seq(0.1,0.5,0.1))

n=length(C)

CV=0.025

runs=100000

a1=rep(NaN,runs)

r2all<-data.frame("OLS"=a1,"NLS"=a1,"WLS"=a1,"NWLS"=a1,"LNLS"=a1,"GGLMINV"=a1,"GGLMID"=a1,"TS"=a1)

r2var=r2all
```

Actual Simulation

For this Monte Carlo Simulation, we will have no concentration errors considered.The SD for T_2 will be chosen to be $CV * T_2^p$ where $CV = 0.025$. $p \sim N(1.18, 0.12^2)$ and $T_2 \sim N(T_2^{true}, (CV * T_2^{true})^2)$. Technically, $R_2 = 1/T_2$ will not be normal after the inverse transformation. In this simulation, we will also be accounting for the results of each method under 10% concentration error.

```
set.seed(9177)
for(i in 1:runs){

ErrC=rnorm(n,0,0.1*C)  #10% concentration error
Ctrue=C+ErrC #For this simulation, the true concentration is within +-10% SD of what we think it is

T2true=1/(r2*Ctrue+R20)
```

```
p=rnorm(n,1.18,0.12)  #Generate a power p to account for deviations from "perfect" CV in T2

T2obs=rnorm(n,T2true,CV*T2true^p)  #This is to test normal distribution with SD=CV*T2true^p


R2obs=1/T2obs


OLS<-lm(R2obs~C)

NLS<-nls(T2obs~(b0+b1*C)^(-1),start=list(b0=0.4,b1=95))

WLS<-lm(R2obs~C,weights=1/R2obs^2)

NWLS<-nls(T2obs~(b0+b1*C)^(-1),start=list(b0=0.4,b1=95),weights=1/T2obs^2)

LNLS<-nls(log(R2obs)~log(b0+b1*C),start=list(b0=0.4,b1=95))

GGLMINV<-glm(T2obs~C,family=Gamma(link=inverse))

GGLMID<-glm(R2obs~C,family=Gamma(link=identity))

TS<-mblm(R2obs~C,repeated = T)

r2all$OLS[i]=coef(OLS)[2]
r2all$NLS[i]=coef(NLS)[2]

r2all$WLS[i]=coef(WLS)[2]
r2all$NWLS[i]=coef(NWLS)[2]

r2all$LNLS[i]=coef(LNLS)[2]
r2all$GGLMINV[i]=coef(GGLMINV)[2]
r2all$GGLMID[i]=coef(GGLMID)[2]

r2all$TS[i]=coef(TS)[2]

r2var$OLS[i]=vcov(OLS)[4]
r2var$NLS[i]=vcov(NLS)[4]

r2var$WLS[i]=vcov(WLS)[4]
r2var$NWLS[i]=vcov(NWLS)[4]

r2var$LNLS[i]=vcov(LNLS)[4]
r2var$GGLMINV[i]=vcov(GGLMINV)[4]
r2var$GGLMID[i]=vcov(GGLMID)[4]

r2var$TS[i]=vcov(TS)[4]

}

head(r2all)
```

```
##         OLS       NLS       WLS      NWLS      LNLS   GGLMINV    GGLMID
## 1 101.55486  95.09850 101.24808 103.27194 102.25767 101.91975 102.59616
## 2  96.44401  96.85618  94.82621  97.74511  96.39212  95.89111  96.86196
## 3  91.31116 102.99400  96.41860  98.24379  97.32125  97.01756  97.62701
## 4  99.27657 103.48008  99.59955 101.13293 100.38935 100.12968 100.64188
## 5 102.31764 106.78508 101.09553 104.30036 102.74483 102.20289 103.27224
## 6  98.05783  93.25185  99.16377 101.17157 100.14049  99.80926 100.47947
##          TS
## 1 100.48494
## 2  95.29685
## 3  95.54239
## 4 104.68769
## 5 102.88477
## 6 101.64456
```

```
head(r2var)
```

```
##         OLS       NLS       WLS      NWLS      LNLS   GGLMINV    GGLMID
## 1 11.605933 13.829515  9.851366  9.997401  9.932285  9.917331  9.912286
## 2  6.571391 31.658620 14.970986 12.750176 13.768872 15.232582 12.528992
## 3  5.147226  5.974230  7.963181  8.420547  8.192843  8.124536  8.251248
## 4 14.938868  2.578446  7.057156  6.670774  6.878240  7.169316  6.578021
## 5  7.829521 23.568676 15.771932 15.229659 15.475422 16.112628 14.912519
## 6 10.240363  8.534509  9.119854  9.956821  9.523240  9.172743  9.877458
##          TS
## 1 12.184843
## 2  7.369554
## 3  8.172548
## 4 23.113340
## 5  7.877431
## 6 12.395553
```

The first few relaxivities and associated variances (from the fit itself) are displayed.

Results

Now that we have obtained the relaxivities for all runs, we go on to calculate mean, SD, bias, MSE, and Type I error rates.

```
r2bar<-apply(r2all,2,mean)
r2sd<-apply(r2all,2,sd) ##True SD from all the fits##

r2SE<-sqrt(r2var) #Estimated fitting SE

Bias=r2bar-r2

MSE=Bias^2+r2sd^2

bool_SESD<-matrix(rep(NaN,dim(r2SE)[1]*dim(r2SE)[2]),nrow=dim(r2SE)[1],ncol=dim(r2SE)[2])
Type1Err=rep(NaN,length(r2bar))
```

```r
df=n-2 # t-distribution degrees of freedom
tcrit=qt(0.975,df) #critical t value at alpha=0.05 2 sided

r2UPPER=r2all+tcrit*r2SE
r2LOWER=r2all-tcrit*r2SE #Forming 95% CIs




for(i in 1:8){

bool_SESD[,i]=r2SE[,i]<=r2sd[i]

Type1Err[i]=mean((r2>r2UPPER[i])|r2<r2LOWER[i]) #Check if r2=100 is contained in 95% CI

}

P_SESD=apply(bool_SESD,2,mean)

Result<-rbind(Bias,r2sd,MSE,P_SESD,Type1Err)
t(Result)
```

```
##                Bias     r2sd      MSE  P_SESD Type1Err
## OLS       0.04752557 7.165476 51.34630 0.96251  0.26254
## NLS      -0.49495168 6.263218 39.47287 0.99212  0.34969
## WLS      -1.73018677 3.604720 15.98755 0.64912  0.06710
## NWLS      0.88395260 3.433761 12.57208 0.55387  0.05004
## LNLS     -0.41312806 3.450415 12.07604 0.56969  0.05007
## GGLMINV -0.85169644 3.486304 12.87970 0.58444  0.05324
## GGLMID    0.02179621 3.430327 11.76762 0.56426  0.04895
## TS       -0.01026642 4.976708 24.76773 0.64780  0.10125
```

R1 Repeatability Data Analysis from Waterton et al.

Kapre et al.

November 20, 2019

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

R1 Repeatability Study

This dataset originated from a study on the repeatability of R1 values in MRI Relaxometry across different MRI Facilities. The original paper can be found here:

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6477178/

DOI: 10.1016/j.mri.2019.03.008.

For this analysis, we will be analyzing only the subset of their dataset where measurements were done at 7T Field Strength. This field strength contains the greatest number of different MRI Facilities (5): A, C, E, F, G1. The initial step is extracting just the 7T data and cleaning up the column names. We also will only be considering the isocenter "O" in the dataset

```
data=read.table("Ni_Agarose_R1data_meta.txt",header=T)
data$B0=as.factor(data$B0)
data7T=data[which(data$B0=="7"),]
colnames(data7T)[1]<-"Facility"

data7T=droplevels(data7T) # This drops irrelevant field strengths which may still be in the dataset

data7T$Day=as.factor(data7T$Day) #Change Day to a Factor type (categorical)

head(data7T) #Shows the organization of the data
```

```
##   Facility B0   mM Day     O     X     Z    eO    eX    eZ     T
## 1        A  7 0.50   1 0.820 0.840 0.893 0.028 0.039 0.067 20.78
## 2        A  7 0.50   2 0.781 0.787 0.794 0.006 0.008 0.009 20.70
## 3        A  7 1.04   1 1.267 1.274 1.258 0.015 0.017 0.024 19.70
## 4        A  7 1.04   2 1.302 1.297 1.332 0.012 0.013 0.015 20.80
## 5        A  7 2.02   1 2.079 2.110 2.066 0.046 0.037 0.077 21.03
## 6        A  7 2.02   2 2.101 2.128 2.141 0.024 0.043 0.036 20.70
```

```
str(data7T) #Examine type of each variable
```

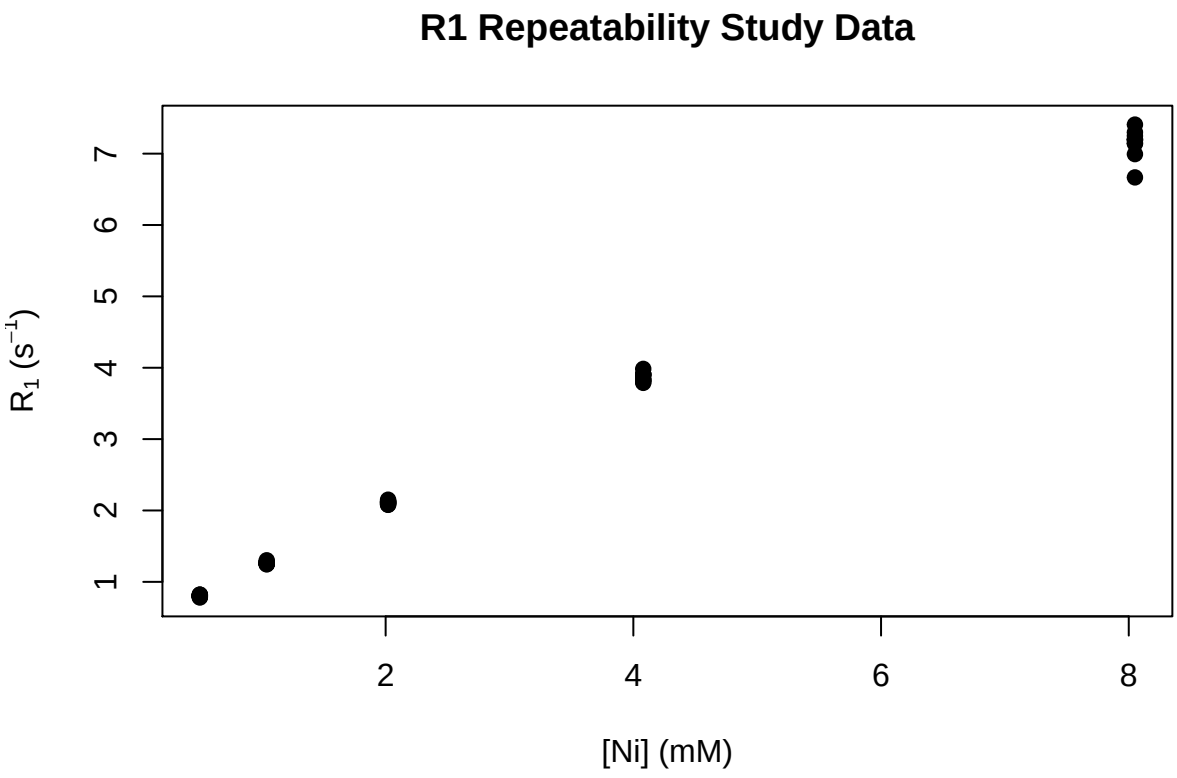
```
## 'data.frame':    50 obs. of  11 variables:
##  $ Facility: Factor w/ 5 levels "A","C","E","F",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ B0      : Factor w/ 1 level "7": 1 1 1 1 1 1 1 1 1 1 ...
##  $ mM      : num  0.5 0.5 1.04 1.04 2.02 2.02 4.08 4.08 8.05 8.05 ...
##  $ Day     : Factor w/ 2 levels "1","2": 1 2 1 2 1 2 1 2 1 2 ...
##  $ O       : num  0.82 0.781 1.267 1.302 2.079 ...
```

```
##  $ X       : num  0.84 0.787 1.274 1.297 2.11 ...
##  $ Z       : num  0.893 0.794 1.258 1.332 2.066 ...
##  $ e0      : num  0.028 0.006 0.015 0.012 0.046 0.024 0.129 0.033 0.782 0.104 ...
##  $ eX      : num  0.039 0.008 0.017 0.013 0.037 0.043 0.152 0.038 0.653 0.175 ...
##  $ eZ      : num  0.067 0.009 0.024 0.015 0.077 0.036 0.136 0.05 0.274 0.093 ...
##  $ T       : num  20.8 20.7 19.7 20.8 21 ...
```

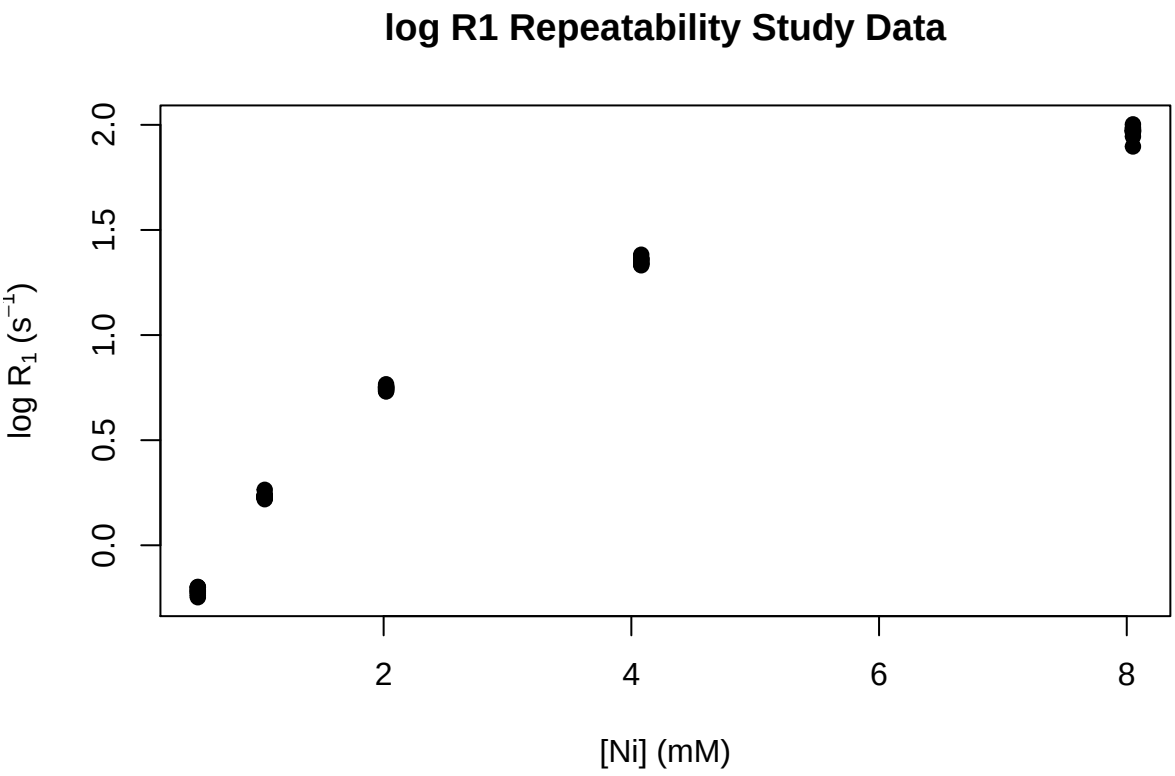
Preliminary Plot

First, visualize all of the data via a simple scatterplot. The scatterplot shows linearity for R1 vs C and nonlinearity for log(R1) vs C. The error SD is clearly increasing with R1 in the R1 vs C plot.

```
par(mfrow=c(1,1))
plot(data7T$mM,data7T$0,main="R1 Repeatability Study Data",xlab="[Ni] (mM)",ylab=expression(paste(R[1],
                                                                                                "(",s
```




```
plot(data7T$mM,log(data7T$0),main="log R1 Repeatability Study Data",xlab="[Ni] (mM)",ylab=expression(pas
                                                                                                "(",s
```



Model Fitting and ANOVA

First, we will include Day as a factor though to simplify the analysis and number of variables, this was removed for both the OLS Model and GGLM-ID/LNLS models.

```
#Full models including Day. This was not shown in the paper for simplicity#
OLSfull<-lm(O~mM*Facility*Day,data=data7T)

GGLM.IDfull<-glm(O~mM*Facility*Day,family=Gamma(link=identity),data=data7T)

#Excluding Day#

OLSNew<-lm(O~mM*Facility,data=data7T)

GGLM.IDNew<-glm(O~mM*Facility,family=Gamma(link=identity),data=data7T)

##Basic Null Model Excluding all except Concentration#

OLSBase<-lm(O~mM,data=data7T)

GGLM.IDBase<-glm(O~mM,family=Gamma(link=identity),data=data7T)

##We can compare AIC of each model here and also do an ANOVA F-test of the "New" vs "Base".
```

```
AIC_OLS<-c(AIC(OLSfull),AIC(OLSNew),AIC(OLSBase))
AIC_GGLM.ID<-c(AIC(GGLM.IDfull),AIC(GGLM.IDNew),AIC(GGLM.IDBase))

#AIC Table#

AICTable<-data.frame("OLS"=AIC_OLS,"GGLM-ID"=AIC_GGLM.ID)
rownames(AICTable)<-c("AIC Full with Day","AIC Facility","AIC Null Model")
AICTable
```

```
##                           OLS   GGLM.ID
## AIC Full with Day -114.74919 -157.3944
## AIC Facility      -107.33224 -170.0940
## AIC Null Model     -90.22609 -172.2329
```

The above suggests that across the board, GGLM-ID has a lower AIC than OLS, indicating it has a better model fit. Furthermore, AIC within GGLM-ID suggests the model can be simplified to the null model. For OLS, however, it is indicating that Day and Facility are significant to be included in the model.

We can also do ANOVA with respect to the Null Model. This will be done for the Facility model (dropping Day for both for simplicity).

```
anova(OLSNew,OLSBase,test="F")
```

```
## Analysis of Variance Table
##
## Model 1: O ~ mM * Facility
## Model 2: O ~ mM
##   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
## 1     40 0.22036
## 2     48 0.42725 -8   -0.2069 4.6945 0.0004165 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(GGLM.IDNew,GGLM.IDBase,test="F")
```

```
## Analysis of Deviance Table
##
## Model 1: O ~ mM * Facility
## Model 2: O ~ mM
##   Resid. Df Resid. Dev Df   Deviance      F Pr(>F)
## 1        40   0.012272
## 2        48   0.016192 -8 -0.0039201 1.6177 0.1504
```

The ANOVA p-values also confirm the analysis with AIC above. OLS conclusion is that Facility is statistically significant enough to be included in the model while GGLM-ID conclusion is that there is no significant effect of Facility at $\alpha = 0.05$. This may affect the conclusion of a study for example which compares the cellular uptake of multiple contrast agents by MRI Relaxometry.

Dummy Variable Coding

In the above section, we made use of R's default factor coding. That will be used in the post-hoc Bonferroni contrasts For interpretation of the summary() output, however, it is easier to use -1/0/1 coding. This makes it easier to fit the LNLS (log-transform nonlinear least squares) model and examine its summary.

Mathematically, this section is equivalent to above. It is only done for interpretation of the summary output and to facilitate the nonlinear fitting, as nls() in R does not take Factor variables as is. The interpretation is easier as the main effect relaxivity is effectively the mean relaxivity across all Facilities when coded in this way. However, for the contrasts later on, we will still use the default factor coding in the above block.

```
IA=rep(0,dim(data7T)[1])
IA[which(data7T$Facility=="A")]=1
IA[which(data7T$Facility=="G1")]=-1

IC=rep(0,dim(data7T)[1])
IC[which(data7T$Facility=="C")]=1
IC[which(data7T$Facility=="G1")]=-1

IE=rep(0,dim(data7T)[1])
IE[which(data7T$Facility=="E")]=1
IE[which(data7T$Facility=="G1")]=-1

IF=rep(0,dim(data7T)[1])
IF[which(data7T$Facility=="F")]=1
IF[which(data7T$Facility=="G1")]=-1

New7T=cbind(data7T,"IA"=IA,"IC"=IC,"IE"=IE,"IF"=IF)

head(New7T)
```

```
##   Facility B0   mM Day     0     X     Z    e0    eX    eZ     T IA IC IE
## 1        A  7 0.50   1 0.820 0.840 0.893 0.028 0.039 0.067 20.78  1  0  0
## 2        A  7 0.50   2 0.781 0.787 0.794 0.006 0.008 0.009 20.70  1  0  0
## 3        A  7 1.04   1 1.267 1.274 1.258 0.015 0.017 0.024 19.70  1  0  0
## 4        A  7 1.04   2 1.302 1.297 1.332 0.012 0.013 0.015 20.80  1  0  0
## 5        A  7 2.02   1 2.079 2.110 2.066 0.046 0.037 0.077 21.03  1  0  0
## 6        A  7 2.02   2 2.101 2.128 2.141 0.024 0.043 0.036 20.70  1  0  0
##   IF
## 1  0
## 2  0
## 3  0
## 4  0
## 5  0
## 6  0
```

```
tail(New7T)
```

```
##    Facility B0   mM Day     0     X     Z    e0    eX    eZ     T IA IC IE
## 45       G1  7 2.02   1 2.114 2.164 2.117 0.048 0.054 0.053 18.40 -1 -1 -1
## 46       G1  7 2.02   2 2.112 2.182 2.096 0.053 0.053 0.051 18.40 -1 -1 -1
## 47       G1  7 4.08   1 3.909 3.956 3.881 0.097 0.109 0.099 18.50 -1 -1 -1
## 48       G1  7 4.08   2 3.905 3.904 3.902 0.100 0.114 0.101 18.35 -1 -1 -1
## 49       G1  7 8.05   1 7.193 7.456 7.345 0.234 0.249 0.261 18.45 -1 -1 -1
## 50       G1  7 8.05   2 7.197 7.196 7.144 0.227 0.209 0.216 18.15 -1 -1 -1
##    IF
## 45 -1
## 46 -1
## 47 -1
```

```
## 48 -1
## 49 -1
## 50 -1
```

Fit the (mathematically identical) models with the dummy variables and obtain summary()

```
Dummy.New.OLS7T<-lm(O~mM*(IA+IC+IE+IF),data=New7T)
Dummy.New.GGLMID7T<-glm(O~mM*(IA+IC+IE+IF),family=Gamma(link=identity),data=New7T)
Dummy.New.LNLS7T<-nls(log(O)~log(R1.0+r1*mM+bA*IA+bC*IC+bE*IE+bF*IF+rA*mM*IA+rC*mM*IC+rE*mM*IE+rF*mM*IF
                           ,start=list(R1.0=0,r1=1,bA=0,bC=0,bE=0,bF=0,rA=0,rC=0,rE=0,rF=0),data=New7T)

Dummy.Base.OLS7T<-lm(O~mM,data=New7T)
Dummy.Base.GGLMID7T<-glm(O~mM,family=Gamma(link=identity),data=New7T)
Dummy.Base.LNLS7T<-nls(log(O)~log(R1.0+r1*mM),start=list(R1.0=0,r1=1),data=New7T)


summary(Dummy.New.OLS7T)
```

```
##
## Call:
## lm(formula = O ~ mM * (IA + IC + IE + IF), data = New7T)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.30638 -0.02033 -0.00178  0.01650  0.22403
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.401404   0.015949  25.169  < 2e-16 ***
## mM           0.840547   0.003826 219.669  < 2e-16 ***
## IA           0.056134   0.031897   1.760 0.086080 .
## IC          -0.019100   0.031897  -0.599 0.552680
## IE          -0.003256   0.031897  -0.102 0.919197
## IF          -0.032291   0.031897  -1.012 0.317448
## mM:IA       -0.031126   0.007653  -4.067 0.000217 ***
## mM:IC        0.008910   0.007653   1.164 0.251203
## mM:IE       -0.010925   0.007653  -1.428 0.161163
## mM:IF        0.026721   0.007653   3.492 0.001186 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07422 on 40 degrees of freedom
## Multiple R-squared:  0.9992, Adjusted R-squared:  0.999
## F-statistic:  5366 on 9 and 40 DF,  p-value: < 2.2e-16
```

```
summary(Dummy.New.GGLMID7T)
```

```
##
## Call:
## glm(formula = O ~ mM * (IA + IC + IE + IF), family = Gamma(link = identity),
##     data = New7T)
##
```

```
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.066619  -0.005922   0.001689   0.006832   0.045426
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.381162   0.004912  77.595   <2e-16 ***
## mM           0.849239   0.003705 229.203   <2e-16 ***
## IA           0.011888   0.009839   1.208   0.2341
## IC          -0.010001   0.009786  -1.022   0.3129
## IE           0.001460   0.009765   0.150   0.8819
## IF          -0.009798   0.009850  -0.995   0.3258
## mM:IA       -0.012150   0.007366  -1.650   0.1069
## mM:IC        0.005629   0.007422   0.758   0.4526
## mM:IE       -0.012563   0.007342  -1.711   0.0948 .
## mM:IF        0.016869   0.007486   2.254   0.0298 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.0003029071)
##
##     Null deviance: 29.519949  on 49  degrees of freedom
## Residual deviance:  0.012272  on 40  degrees of freedom
## AIC: -170.09
##
## Number of Fisher Scoring iterations: 4
```

```
summary(Dummy.New.LNLS7T)
```

```
##
## Formula: log(O) ~ log(R1.0 + r1 * mM + bA * IA + bC * IC + bE * IE + bF *
##     IF + rA * mM * IA + rC * mM * IC + rE * mM * IE + rF * mM *
##     IF)
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## R1.0  0.381185   0.004959  76.864   <2e-16 ***
## r1    0.849091   0.003740 227.002   <2e-16 ***
## bA    0.012045   0.009932   1.213   0.2323
## bC   -0.010050   0.009880  -1.017   0.3152
## bE    0.001451   0.009859   0.147   0.8838
## bF   -0.009851   0.009944  -0.991   0.3278
## rA   -0.012609   0.007433  -1.696   0.0976 .
## rC    0.005747   0.007493   0.767   0.4476
## rE   -0.012456   0.007413  -1.680   0.1007
## rF    0.016990   0.007557   2.248   0.0302 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01757 on 40 degrees of freedom
##
## Number of iterations to convergence: 4
## Achieved convergence tolerance: 5.852e-06
```

```
#Confirm that the OLS and GGLMID are the same models as earlier by looking at AIC#
AICOLSDummy<-c(AIC(Dummy.New.OLS7T),AIC(Dummy.Base.OLS7T))
AICGGLMDummy<-c(AIC(Dummy.New.GGLMID7T),AIC(Dummy.Base.GGLMID7T))

AICDummyTable<-data.frame("OLS"=AICOLSDummy,"GGLM-ID"=AICGGLMDummy)
rownames(AICDummyTable)<-c("AIC Facility","AIC Null Model")
AICDummyTable
```

```
##                        OLS   GGLM.ID
## AIC Facility   -107.33224 -170.0940
## AIC Null Model  -90.22609 -172.2329
```

As seen above, the estimates, standard errors, and even p-values are similar between GGLM-ID and LNLS. Next perform ANOVA with LNLS.

Below, the AIC is also lower for the model without Facility (base model) in LNLS. Note that one cannot compare AIC for LNLS to that of OLS or GGLM-ID since LNLS uses a different (log-transformed) response variable Practically, LNLS is hard to use as it requires the manual assignment of dummy variables to perform the ANOVA, which does not happen automatically within the nls() function in R.

```
AIC(Dummy.New.LNLS7T)
```

```
## [1] -251.4059
```

```
AIC(Dummy.Base.LNLS7T) #AIC is lower for base model
```

```
## [1] -253.4244
```

```
anova(Dummy.New.LNLS7T,Dummy.Base.LNLS7T,test="F")
```

```
## Analysis of Variance Table
##
## Model 1: log(O) ~ log(R1.0 + r1 * mM + bA * IA + bC * IC + bE * IE + bF * IF + rA * mM * IA + rC * ml
## Model 2: log(O) ~ log(R1.0 + r1 * mM)
##   Res.Df Res.Sum Sq Df     Sum Sq F value Pr(>F)
## 1     40   0.012352
## 2     48   0.016337 -8 -0.0039851  1.6132 0.1517
```

Bonferroni Corrected Pairwise Comparisons

The ANOVA from earlier told us that Facility is not statistically significant within the GGLM-ID framework but is within the OLS framework. We can also do pairwise contrasts between the Facilities' r1 relaxivities using the emmeans package in R. In an in-vivo (non-relaxivity) MRI relaxometry study, all the factors would be categorical (qualitative) variables and we would be comparing R1 (between different factor levels) in s^{-1} directly with the emmeans() function.

However, in this example, concentration of [Ni] (mM) is a continuous variable and the relevant comparison is between relaxivities which are slopes in $mM^{-1}\ s^{-1}$. To do this, the emtrends() function within the package must be used.

For the purposes of this, we will be performing the comparisons using only the OLS and GGLM-ID models with Facility (but not Day). No pairwise comparisons will be performed for LNLS, as this is difficult for nls models with emmeans, which is also a practical reason to use GGLM-ID over LNLS.

The Bonferroni correction (for all 10 possible pairwise comparisons) will be used. Multiple testing corrections are important for reproducibility in an in-vivo study where many comparisons are performed.

(Note: Technically, the pairwise comparisons for GGLM-ID with the emmeans package use Wald Z tests while T-Tests are used for OLS models)

OLS Pairwise Comparisons

```
library(emmeans)

emmOLS<-emtrends(OLSNew,pairwise~Facility,var="mM",adjust="bonferroni") #var specifies the continuous c

emmOLS
```

```
## $emtrends
##  Facility mM.trend      SE df lower.CL upper.CL
##  A           0.809 0.00856 40    0.792    0.827
##  C           0.849 0.00856 40    0.832    0.867
##  E           0.830 0.00856 40    0.812    0.847
##  F           0.867 0.00856 40    0.850    0.885
##  G1          0.847 0.00856 40    0.830    0.864
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast estimate     SE df t.ratio p.value
##  A - C    -0.04004 0.0121 40 -3.309  0.0199
##  A - E    -0.02020 0.0121 40 -1.669  1.0000
##  A - F    -0.05785 0.0121 40 -4.781  0.0002
##  A - G1   -0.03755 0.0121 40 -3.103  0.0351
##  C - E     0.01984 0.0121 40  1.639  1.0000
##  C - F    -0.01781 0.0121 40 -1.472  1.0000
##  C - G1    0.00249 0.0121 40  0.206  1.0000
##  E - F    -0.03765 0.0121 40 -3.111  0.0343
##  E - G1   -0.01735 0.0121 40 -1.433  1.0000
##  F - G1    0.02030 0.0121 40  1.678  1.0000
##
## P value adjustment: bonferroni method for 10 tests
```

Using OLS can lead one to conclude (erroneously) that A-F, A-G1, and E-F have statistically significant differences in relaxivity.

GGLM-ID Pairwise Comparisons

```
emmGGLM.ID<-emtrends(GGLM.IDNew,pairwise~Facility,var="mM",adjust="bonferroni")

emmGGLM.ID #degrees of freedom df are Inf due to wald Z test
```

```
## $emtrends
##  Facility mM.trend      SE  df asymp.LCL asymp.UCL
##  A           0.837 0.00822 Inf     0.821     0.853
##  C           0.855 0.00830 Inf     0.839     0.871
##  E           0.837 0.00818 Inf     0.821     0.853
##  F           0.866 0.00840 Inf     0.850     0.883
##  G1          0.851 0.00832 Inf     0.835     0.868
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast  estimate     SE  df z.ratio p.value
##  A - C    -0.017779 0.0117 Inf -1.522  1.0000
##  A - E     0.000413 0.0116 Inf  0.036  1.0000
##  A - F    -0.029020 0.0117 Inf -2.470  0.1352
##  A - G1   -0.014365 0.0117 Inf -1.228  1.0000
##  C - E     0.018192 0.0117 Inf  1.561  1.0000
##  C - F    -0.011241 0.0118 Inf -0.952  1.0000
##  C - G1    0.003414 0.0118 Inf  0.290  1.0000
##  E - F    -0.029433 0.0117 Inf -2.510  0.1207
##  E - G1   -0.014778 0.0117 Inf -1.266  1.0000
##  F - G1    0.014654 0.0118 Inf  1.240  1.0000
##
## P value adjustment: bonferroni method for 10 tests
```

Confidence Intervals: Practical vs Statistical Significance

In many situations, it is possible to have a comparison be statistically significant, but not practically significant. One can evaluate this by examining the confidence intervals of the comparison and determining whether the confidence interval lies within the practically significant range. OLS however will give erroneous 95% CIs which do not have the correct coverage. In this section, we calculate the 95% CIs for the relaxivities and comparisons

OLS 95% CI

```
confint(emmOLS)
```

```
## $emtrends
##  Facility mM.trend      SE df lower.CL upper.CL
##  A           0.809 0.00856 40    0.792    0.827
##  C           0.849 0.00856 40    0.832    0.867
##  E           0.830 0.00856 40    0.812    0.847
##  F           0.867 0.00856 40    0.850    0.885
##  G1          0.847 0.00856 40    0.830    0.864
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast estimate     SE df lower.CL upper.CL
##  A - C    -0.04004 0.0121 40  -0.0760 -0.00408
```

```
##  A - E    -0.02020 0.0121 40  -0.0562  0.01575
##  A - F    -0.05785 0.0121 40  -0.0938 -0.02190
##  A - G1   -0.03755 0.0121 40  -0.0735 -0.00159
##  C - E     0.01984 0.0121 40  -0.0161  0.05579
##  C - F    -0.01781 0.0121 40  -0.0538  0.01814
##  C - G1    0.00249 0.0121 40  -0.0335  0.03844
##  E - F    -0.03765 0.0121 40  -0.0736 -0.00169
##  E - G1   -0.01735 0.0121 40  -0.0533  0.01861
##  F - G1    0.02030 0.0121 40  -0.0157  0.05625
##
## Confidence level used: 0.95
## Conf-level adjustment: bonferroni method for 10 estimates
```

GGLM-ID 95% CI

```
confint(emmGGLM.ID)
```

```
## $emtrends
##  Facility mM.trend      SE  df asymp.LCL asymp.UCL
##  A           0.837 0.00822 Inf     0.821     0.853
##  C           0.855 0.00830 Inf     0.839     0.871
##  E           0.837 0.00818 Inf     0.821     0.853
##  F           0.866 0.00840 Inf     0.850     0.883
##  G1          0.851 0.00832 Inf     0.835     0.868
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast  estimate     SE  df asymp.LCL asymp.UCL
##  A - C    -0.017779 0.0117 Inf   -0.0506   0.01501
##  A - E     0.000413 0.0116 Inf   -0.0321   0.03297
##  A - F    -0.029020 0.0117 Inf   -0.0620   0.00396
##  A - G1   -0.014365 0.0117 Inf   -0.0472   0.01847
##  C - E     0.018192 0.0117 Inf   -0.0145   0.05091
##  C - F    -0.011241 0.0118 Inf   -0.0444   0.02191
##  C - G1    0.003414 0.0118 Inf   -0.0296   0.03641
##  E - F    -0.029433 0.0117 Inf   -0.0623   0.00348
##  E - G1   -0.014778 0.0117 Inf   -0.0475   0.01799
##  F - G1    0.014654 0.0118 Inf   -0.0185   0.04784
##
## Confidence level used: 0.95
## Conf-level adjustment: bonferroni method for 10 estimates
```

Bootstrap

One way to perform robust hypothesis testing is through the bootstrap resampling *with replacement*. This is also one way to determine whether the conclusions from OLS are reproducible under different realizations of the data. In the following bootstrap, we will assign for every concentration at each Facility a random R1 value selected (with replacement) from all the observed R1 values at that concentration among all Facilities.

This artifically creates the Null situation where no Facility will have statistically significant differences. The observed t (or z for GGLM-ID) values for each comparison can then be compared to the bootstrap t and z distributions obtained through this resampling to obtain p-values.

Preparing Bootstrap Data Frame and T/Z value matrices

```
library(tidyr)


ord=order(data7T$mM)
orgdata7T<-data7T[ord,] #Order by mM

BootData<-orgdata7T
BootData<-BootData[,-(6:11)]
BootData$O<-NaN  #This is the bootstrap data frame, set all R1s to NaN for now

head(orgdata7T)
```

```
##    Facility BO  mM Day     O     X     Z    eO    eX    eZ     T
## 1         A  7 0.5   1 0.820 0.840 0.893 0.028 0.039 0.067 20.78
## 2         A  7 0.5   2 0.781 0.787 0.794 0.006 0.008 0.009 20.70
## 11        C  7 0.5   1 0.788 0.807 0.806 0.006 0.005 0.011 20.80
## 12        C  7 0.5   2 0.803 0.816 0.824 0.006 0.008 0.007 21.45
## 21        E  7 0.5   1 0.800 0.826 0.826 0.013 0.021 0.016 17.80
## 22        E  7 0.5   2 0.794 0.813 0.813 0.015 0.015 0.020 18.50
```

```
head(BootData)
```

```
##    Facility BO  mM Day   O
## 1         A  7 0.5   1 NaN
## 2         A  7 0.5   2 NaN
## 11        C  7 0.5   1 NaN
## 12        C  7 0.5   2 NaN
## 21        E  7 0.5   1 NaN
## 22        E  7 0.5   2 NaN
```

```
runs=10000
tstar.OLS=matrix(rep(NaN,runs*10),nrow=10,ncol=runs) #Each row is each comparison
zstar.GGLM=tstar.OLS
```

Bootstrap (Resampling w/ Replacement)

In this section, we resample and essentially repeat the same procedure as for the actual observed data. Obtain t/z values and store them into a matrix. The objects from emmeans have to be converted to data frames to extract the values.

```
set.seed(9453)

for(i in 1:runs){
```

```
BootData[BootData$mM==0.50,]$O=sample(orgdata7T[orgdata7T$mM==0.50,]$O,replace=T)
BootData[BootData$mM==1.04,]$O=sample(orgdata7T[orgdata7T$mM==1.04,]$O,replace=T)
BootData[BootData$mM==2.02,]$O=sample(orgdata7T[orgdata7T$mM==2.02,]$O,replace=T)
BootData[BootData$mM==4.08,]$O=sample(orgdata7T[orgdata7T$mM==4.08,]$O,replace=T)
BootData[BootData$mM==8.05,]$O=sample(orgdata7T[orgdata7T$mM==8.05,]$O,replace=T)

Boot.OLS<-lm(O~Facility*mM,data=BootData)
Boot.GGLM<-glm(O~Facility*mM,family=Gamma(link=identity),data=BootData)


BOLSEMM<-emtrends(Boot.OLS,pairwise~Facility,var="mM",adjust="none")
BGGLMEMM<-emtrends(Boot.GGLM,pairwise~Facility,var="mM",adjust="none")


BOLSComps<-as.data.frame(BOLSEMM$contrasts)
BGGLMComps<-as.data.frame(BGGLMEMM$contrasts)


tstar.OLS[,i]=BOLSComps$t.ratio
zstar.GGLM[,i]=BGGLMComps$z.ratio


}

tstar.OLS[1:10,1:5] #Show first 5 columns
```

```
##             [,1]       [,2]        [,3]        [,4]       [,5]
##  [1,]  0.4174279 -0.3864856 -0.68781588  2.70062961  0.8667043
##  [2,] -0.1837581  0.6579509 -0.82388859  1.81039307 -0.8266384
##  [3,] -0.3350836  3.0446074  0.06511325  0.69849098 -0.6543079
##  [4,]  0.2136208  1.3935606 -0.03852905 -0.07808133 -2.0059528
##  [5,] -0.6011860  1.0444365 -0.13607271 -0.89023655 -1.6933427
##  [6,] -0.7525115  3.4310930  0.75292914 -2.00213863 -1.5210122
##  [7,] -0.2038071  1.7800462  0.64928683 -2.77871094 -2.8726571
##  [8,] -0.1513255  2.3866565  0.88900185 -1.11190208  0.1723306
##  [9,]  0.3973789  0.7356097  0.78535954 -1.88847439 -1.1793144
## [10,]  0.5487045 -1.6510468 -0.10364231 -0.77657231 -1.3516450
```

```
zstar.GGLM[1:10,1:5]
```

```
##             [,1]       [,2]        [,3]       [,4]       [,5]
##  [1,] -0.1668765 -1.4224074 -0.87725434  1.5311078  0.5323953
##  [2,]  0.6049238 -1.2526081 -0.17362441  1.0267802 -0.4393095
##  [3,] -0.4533745  0.8679801  0.06074426  0.5152245 -0.7678269
##  [4,]  0.3771024 -0.4431380 -0.47660149 -0.3618718 -1.5631331
##  [5,]  0.7731311  0.1700531  0.70081739 -0.5020917 -0.9712595
##  [6,] -0.2869522  2.2892039  0.93686916 -1.0138629 -1.3003827
##  [7,]  0.5445744  0.9800650  0.40055216 -1.8914487 -2.0946239
##  [8,] -1.0602457  2.1197020  0.23394535 -0.5109546 -0.3277481
##  [9,] -0.2269851  0.8100923 -0.30145702 -1.3871091 -1.1232772
## [10,]  0.8310629 -1.3112418 -0.53672015 -0.8761136 -0.7970658
```

Extract Observed t and z values (and p values)

```
ObsOLSComps=as.data.frame(emmOLS$contrasts)
ObsGGLMComps=as.data.frame(emmGGLM.ID$contrasts)


tOLS<-ObsOLSComps$t.ratio
zGGLM<-ObsGGLMComps$z.ratio



pOLS<-ObsOLSComps$p.value
pGGLM<-ObsGGLMComps$p.value

tOLS
```

```
##  [1] -3.3087302 -1.6694613 -4.7807014 -3.1029397  1.6392688 -1.4719713
##  [7]  0.2057905 -3.1112401 -1.4334784  1.6777618
```

```
zGGLM
```

```
##  [1] -1.52195335  0.03561153 -2.46988008 -1.22817003  1.56056001
##  [6] -0.95193573  0.29039589 -2.51020101 -1.26611203  1.23950253
```

Obtain Bootstrap p-values

We must now compare the observed t/z values to the t/z value matrix (which is the null distribution) by row and do a 2 tailed comparison. We also perform the Bonferroni adjustment in the end for 10 tests.

In the end, it appears the bootstrap resampling suggests the initial OLS conclusion is not reproducible. For GGLM-ID, the results are more consistent.

```
Boot.pOLS<-rep(NaN,10)
Boot.pGGLM<-Boot.pOLS


for(j in 1:10){


  Boot.pOLS[j]=mean(tstar.OLS[j,]<=-abs(tOLS[j]))+mean(tstar.OLS[j,]>=abs(tOLS[j]))
  Boot.pGGLM[j]=mean(zstar.GGLM[j,]<=-abs(zGGLM[j]))+mean(zstar.GGLM[j,]>=abs(zGGLM[j]))

}


#Bonferroni adjustment

BONF.Boot.pOLS=p.adjust(Boot.pOLS,method="bonferroni")
BONF.Boot.PGGLM=p.adjust(Boot.pGGLM,method="bonferroni")
```

```r
#Final Result#

FinalResult<-data.frame("contrast"=ObsOLSComps$contrast,"pOLS"=pOLS,"pGGLM"=pGGLM,
                        "pOLSBoot"=BONF.Boot.pOLS,"pGGLMBoot"=BONF.Boot.PGGLM)
FinalResult
```

```
##    contrast         pOLS     pGGLM pOLSBoot pGGLMBoot
## 1     A - C 0.0198966708 1.0000000    0.842     1.000
## 2     A - E 1.0000000000 1.0000000    1.000     1.000
## 3     A - F 0.0002375104 0.1351584    0.182     0.322
## 4    A - G1 0.0350879239 1.0000000    0.993     1.000
## 5     C - E 1.0000000000 1.0000000    1.000     1.000
## 6     C - F 1.0000000000 1.0000000    1.000     1.000
## 7    C - G1 1.0000000000 1.0000000    1.000     1.000
## 8     E - F 0.0343053961 0.1206625    1.000     0.352
## 9    E - G1 1.0000000000 1.0000000    1.000     1.000
## 10   F - G1 1.0000000000 1.0000000    1.000     1.000
```

St. Pierre LIC Simulation_LICErrOnly

Kapre et al.

December 15, 2019

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Set R2/LIC Ground Truth

This ground truth is generated based on the initial St. Pierre et al paper where the equation relating LIC and R2 was given as

$R_2 = 6.88 + 26.06\ LIC^{0.701} - 0.438\ LIC^{2*0.701}$

https://ashpublications.org/blood/article/105/2/855/20069/Noninvasive-measurement-and-imaging-of-liver-iron

DOI: 10.1182/blood-2004-01-0177

For this simulation, we will set a true R_2 ranging from $10 - 275\ s^{-1}$ spaced by $0.5\ s^{-1}$. This true R_2 will be used to generate the true LIC using the following equation (inverse of above):

$LIC = (29.75 - \sqrt{900.7 - 2.283R_2})^{1.424}$

https://onlinelibrary.wiley.com/doi/abs/10.1002/jmri.26313

DOI: 10.1002/jmri.26313

```
#Set Ground truth for R2true and LICtrue

R2true=seq(10,275,0.5)
n=length(R2true)

LICtrue=(29.75-sqrt(900.7-2.283*R2true))^(1.424)

TrueData=data.frame(R2true,LICtrue)

head(TrueData)
```

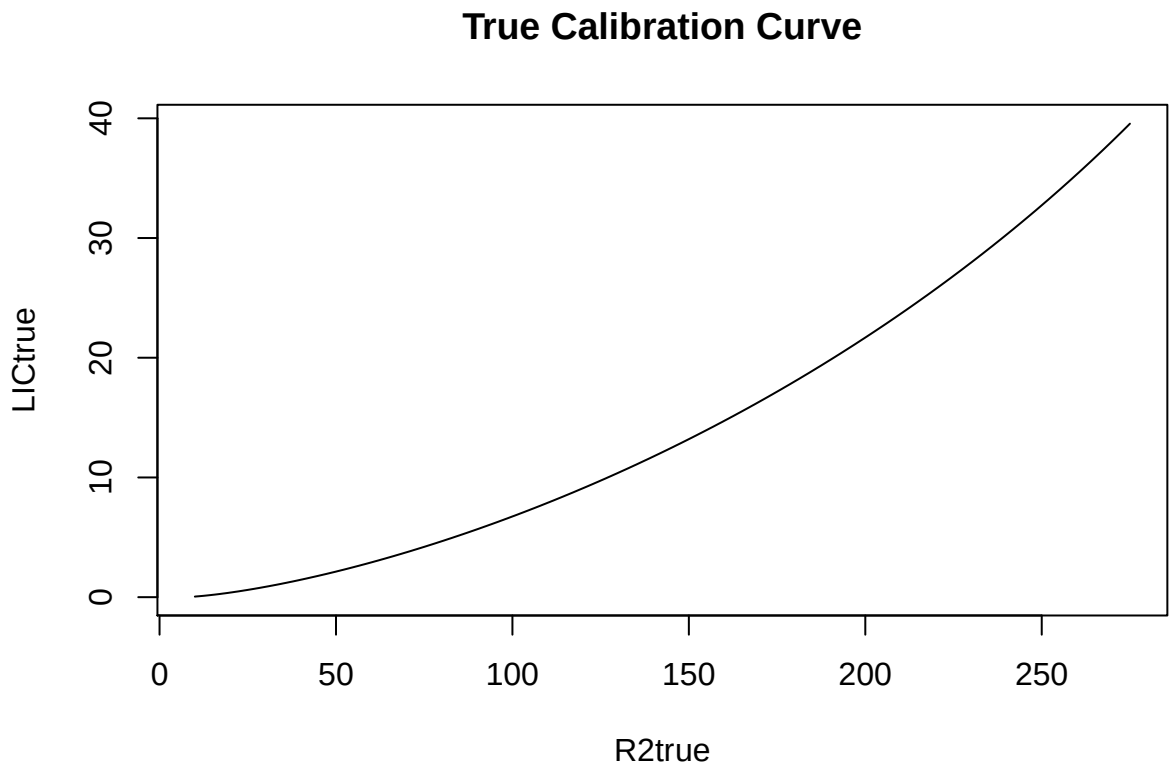
```
##   R2true    LICtrue
## 1   10.0 0.04949315
## 2   10.5 0.06107229
## 3   11.0 0.07335525
## 4   11.5 0.08629338
## 5   12.0 0.09984661
## 6   12.5 0.11398126
```

```
tail(TrueData)
```

```
##     R2true  LICtrue
## 526  272.5 38.82090
```

```
## 527  273.0 38.96589
## 528  273.5 39.11133
## 529  274.0 39.25723
## 530  274.5 39.40360
## 531  275.0 39.55044
```

```
plot(R2true,LICtrue,pch=NA,main="True Calibration Curve")
lines(R2true,LICtrue)
```



Set Error Parameters

In this section, we set the coefficient of variation (CV) for both R2 and LIC. The first simulation, we will only be considering error in LIC, but no error in R2. The second simulation will consider errors in both axes, though technically, all fits will still be conditional on no error in R2. Neither OLS nor GGLM are errors-in-variables models which would further improve calibration.

For R2, CV=8% while for LIC, the CV is piecewise. Based on the literature, it will be set to 19% for true LIC< 4 mg/g and 9% for true LIC>9 mg/g. In between 4 mg/g and 9 mg/g, we will simulate the CV later from a uniform distribution ranging from 9% to 19% as the literature does not provide a value for this range.

```
CVR2=0.08

CVLIC=rep(NaN,n)
CVLIC[LICtrue<= 4] = 0.19
CVLIC[LICtrue>=9] = 0.09
```

```
CVLIC[LICtrue>4 & LICtrue < 9] = 0.14 #This averaging is just for now-will change to uniform in loop la

W=1/(CVLIC*LICtrue)^2 #Set weight for loss functions later
```

Simulation LIC Error Only

In this section, we will simulate from a lognormal distribution instead of normal, since a lognormal avoids complications near R2 or LIC=0. The lognormal is parametrized by the mean and SD on the log scale which to the first order approximately backtransforms to the log mean and CV on the original scale.

Note that sometimes the GLM algorithm may have trouble converging in the simulation. This does not mean it is bad, as simulation is artificial data and this is unlikely to occur on a real dataset. We have supplied starting values to mitigate this phenomenon and are going to use glm2 package for the quadratic fit. At the end of the day, even if the algorithm does not fully converge, if the prediction error is low, it is practically irrelevant.

This simulation will be considering the impact of ONLY LIC Error

```
library(glm2)

set.seed(6823)
runs=10000
a=rep(NaN,runs)
PE=data.frame("QOLS"=a,"QGGLM"=a,"LOLS"=a,"LGGLM"=a)

for(i in 1:runs){

  midlen=length(CVLIC[LICtrue>4 & LICtrue < 9])
  CVLIC[LICtrue>4 & LICtrue < 9]=runif(midlen,0.09,0.19)

  R2obs=R2true
  #R2obs=rlnorm(n,log(R2true),CVR2)
  LICobs=rlnorm(n,log(LICtrue),CVLIC)

  SimData<-data.frame("R2"=R2obs,"LIC"=LICobs)

  #NLS<-nlsLM(LIC~(b0-sqrt(b1-b2*R2))^b3,start=list(b0=30,b1=900,b2=2,b3=1),control=list(warnOnly=T), d

  QOLS<-lm(LIC~R2+I(R2^2),data=SimData)

  QGGLM<-glm2(LIC~R2+I(R2^2),family=Gamma(link=identity),start=c(0.05,0.019,0.0004),data=SimData)

  LOLS<-lm(log(LIC)~log(R2),data=SimData)

  LGGLM<-glm(LIC~log(R2),family=Gamma(link=log),data=SimData)

  #PE$NLS[i]=mean((predict(NLS)-TrueData$LICtrue)^2*W)

  PE$QOLS[i]=mean((predict(QOLS)-TrueData$LICtrue)^2/(TrueData$LICtrue)^2)

  PE$QGGLM[i]=mean((predict(QGGLM)-TrueData$LICtrue)^2/(TrueData$LICtrue)^2)
```

```
  PE$LOLS[i]=mean((exp(predict(LOLS))-TrueData$LICtrue)^2/(TrueData$LICtrue)^2)
  PE$LGGLM[i]=mean((exp(predict(LGGLM))-TrueData$LICtrue)^2/(TrueData$LICtrue)^2)

}

head(PE)
```

```
##          QOLS        QGGLM       LOLS      LGGLM
## 1 0.009825465 0.0015433428 0.01218277 0.01410402
## 2 0.001583699 0.0011553449 0.01089737 0.01272007
## 3 0.065762788 0.0009749041 0.01054527 0.01200140
## 4 0.130344637 0.0013492468 0.01166390 0.01404554
## 5 0.045130920 0.0011799924 0.01139678 0.01317200
## 6 0.210483434 0.0017129366 0.01220331 0.01409966
```

Calculate MRMPPE

In this section, we will calculate the median root mean percent prediction error (MRMPPE)

```
MRMPPE=sqrt(apply(PE,2,median))

Quant025<-function(v){

  return(quantile(v,0.025))
}

Quant975<-function(v){

  return(quantile(v,0.975))
}


Q025=sqrt(apply(PE,2,Quant025))

Q975=sqrt(apply(PE,2,Quant975))

MRMPPE
```

```
##       QOLS      QGGLM       LOLS      LGGLM
## 0.24910472 0.03497813 0.10570216 0.11389864
```

```
Q025
```

```
##       QOLS      QGGLM       LOLS      LGGLM
## 0.04754239 0.03134719 0.10104189 0.10616658
```

```
Q975
```

```
##       QOLS      QGGLM       LOLS      LGGLM
## 0.61573050 0.04448773 0.11346663 0.12471859
```

St. Pierre LIC Simulation_LICR2ErrBoth

Kapre et al.

December 15, 2019

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Set R2/LIC Ground Truth

This ground truth is generated based on the initial St. Pierre et al paper where the equation relating LIC and R2 was given as

$R_2 = 6.88 + 26.06\ LIC^{0.701} - 0.438\ LIC^{2*0.701}$

https://ashpublications.org/blood/article/105/2/855/20069/Noninvasive-measurement-and-imaging-of-liver-iron

DOI: 10.1182/blood-2004-01-0177

For this simulation, we will set a true R_2 ranging from $10 - 275\ s^{-1}$ spaced by $0.5\ s^{-1}$. This true R_2 will be used to generate the true LIC using the following equation (inverse of above):

$LIC = (29.75 - \sqrt{900.7 - 2.283R_2})^{1.424}$

https://onlinelibrary.wiley.com/doi/abs/10.1002/jmri.26313

DOI: 10.1002/jmri.26313

```
#Set Ground truth for R2true and LICtrue

R2true=seq(10,275,0.5)
n=length(R2true)

LICtrue=(29.75-sqrt(900.7-2.283*R2true))^(1.424)

TrueData=data.frame(R2true,LICtrue)

head(TrueData)
```

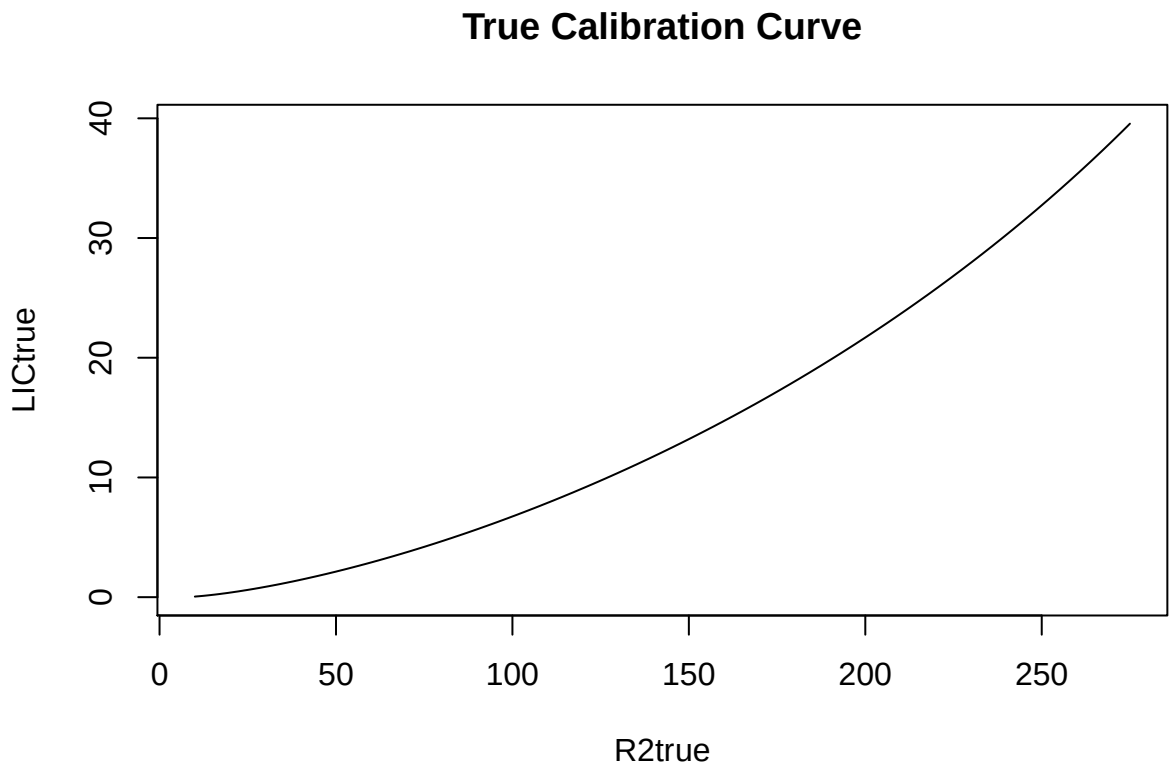
```
##   R2true    LICtrue
## 1   10.0 0.04949315
## 2   10.5 0.06107229
## 3   11.0 0.07335525
## 4   11.5 0.08629338
## 5   12.0 0.09984661
## 6   12.5 0.11398126
```

```
tail(TrueData)
```

```
##     R2true  LICtrue
## 526  272.5 38.82090
```

```
## 527  273.0 38.96589
## 528  273.5 39.11133
## 529  274.0 39.25723
## 530  274.5 39.40360
## 531  275.0 39.55044
```

```
plot(R2true,LICtrue,pch=NA,main="True Calibration Curve")
lines(R2true,LICtrue)
```



Set Error Parameters

In this section, we set the coefficient of variation (CV) for both R2 and LIC. The first simulation, we will only be considering error in LIC, but no error in R2. The second simulation will consider errors in both axes, though technically, all fits will still be conditional on no error in R2. Neither OLS nor GGLM are errors-in-variables models which would further improve calibration.

For R2, CV=8% while for LIC, the CV is piecewise. Based on the literature, it will be set to 19% for true LIC< 4 mg/g and 9% for true LIC>9 mg/g. In between 4 mg/g and 9 mg/g, we will simulate the CV later from a uniform distribution ranging from 9% to 19% as the literature does not provide a value for this range.

```
CVR2=0.08

CVLIC=rep(NaN,n)
CVLIC[LICtrue<= 4] = 0.19
CVLIC[LICtrue>=9] = 0.09
```

```
CVLIC[LICtrue>4 & LICtrue < 9] = 0.14 #This averaging is just for now-will change to uniform in loop la

W=1/(CVLIC*LICtrue)^2 #Set weight for loss functions later
```

Simulation Both LIC and R2 Error

In this section, we will simulate from a lognormal distribution instead of normal, since a lognormal avoids complications near R2 or LIC=0. The lognormal is parametrized by the mean and SD on the log scale which to the first order approximately backtransforms to the log mean and CV on the original scale.

Note that sometimes the GLM algorithm may have trouble converging in the simulation. This does not mean it is bad, as simulation is artificial data and this is unlikely to occur on a real dataset. We have supplied starting values to mitigate this phenomenon and are going to use glm2 package for the quadratic fit. At the end of the day, even if the algorithm does not fully converge, if the prediction error is low, it is practically irrelevant.

This simulation will be considering the impact of BOTH LIC and R2 Errors

```
library(glm2)

set.seed(6823)
runs=10000
a=rep(NaN,runs)
PE=data.frame("QOLS"=a,"QGGLM"=a,"LOLS"=a,"LGGLM"=a)

for(i in 1:runs){

  midlen=length(CVLIC[LICtrue>4 & LICtrue < 9])
  CVLIC[LICtrue>4 & LICtrue < 9]=runif(midlen,0.09,0.19)

  #R2obs=R2true
  R2obs=rlnorm(n,log(R2true),CVR2)
  LICobs=rlnorm(n,log(LICtrue),CVLIC)

  SimData<-data.frame("R2"=R2obs,"LIC"=LICobs)

  #NLS<-nlsLM(LIC~(b0-sqrt(b1-b2*R2))^b3,start=list(b0=30,b1=900,b2=2,b3=1),control=list(warnOnly=T), d

  QOLS<-lm(LIC~R2+I(R2^2),data=SimData)

  QGGLM<-glm2(LIC~R2+I(R2^2),family=Gamma(link=identity),start=c(0.05,0.019,0.0004),data=SimData)

  LOLS<-lm(log(LIC)~log(R2),data=SimData)

  LGGLM<-glm(LIC~log(R2),family=Gamma(link=log),data=SimData)

  #PE$NLS[i]=mean((predict(NLS)-TrueData$LICtrue)^2*W)

  PE$QOLS[i]=mean((predict(QOLS)-TrueData$LICtrue)^2/(TrueData$LICtrue)^2)

  PE$QGGLM[i]=mean((predict(QGGLM)-TrueData$LICtrue)^2/(TrueData$LICtrue)^2)
```

```
  PE$LOLS[i]=mean((exp(predict(LOLS))-TrueData$LICtrue)^2/(TrueData$LICtrue)^2)
  PE$LGGLM[i]=mean((exp(predict(LGGLM))-TrueData$LICtrue)^2/(TrueData$LICtrue)^2)

}

head(PE)
```

```
##       QOLS      QGGLM       LOLS      LGGLM
## 1 1.648375 0.02575817 0.03464317 0.03868867
## 2 1.623695 0.02374407 0.03196180 0.03570812
## 3 2.346517 0.02201613 0.03197614 0.03553275
## 4 2.324628 0.02537656 0.03074590 0.03476602
## 5 4.541297 0.02597984 0.03212812 0.03537675
## 6 2.981063 0.02736144 0.03279476 0.03657525
```

Calculate MRMPPE

In this section, we will calculate the median root mean percent prediction error (MRMPPE)

```
MRMPPE=sqrt(apply(PE,2,median))

Quant025<-function(v){

  return(quantile(v,0.025))
}

Quant975<-function(v){

  return(quantile(v,0.975))
}


Q025=sqrt(apply(PE,2,Quant025))

Q975=sqrt(apply(PE,2,Quant975))

MRMPPE
```

```
##      QOLS     QGGLM      LOLS     LGGLM
## 1.6528639 0.1615222 0.1826222 0.1940056
```

```
Q025
```

```
##      QOLS     QGGLM      LOLS     LGGLM
## 0.9433909 0.1474357 0.1667543 0.1751523
```

```
Q975
```

```
##      QOLS     QGGLM      LOLS     LGGLM
## 2.4023205 0.1821061 0.2023964 0.2172775
```

LIC Simulated 95% Prediction Intervals

Kapre et al.

December 7, 2019

Load in Data

This data has been approximated from the original St.Pierre at al paper using PlotDigitizer: http://plotdigitizer.sourceforge.net/

Original St. Pierre at al. paper:

https://ashpublications.org/blood/article/105/2/855/20069/Noninvasive-measurement-and-imaging-of-liver-iron

DOI: 10.1182/blood-2004-01-0177

```
setwd("C:/Users/rokap/Desktop/Local Docs")
Liver=read.table("LIC_R2_StPierre_ApproxData.txt",header=T)
order=order(Liver$LIC)
Liver=Liver[order,]

n=length(Liver$LIC)

str(Liver)
```

```
## 'data.frame':    93 obs. of  2 variables:
##  $ LIC: num  0.331 0.339 0.505 0.567 0.614 ...
##  $ R2 : num  18.9 24.4 26.1 27.6 27.8 ...
```

Fit OLS and GGLM-ID on Full Data

First, we will fit the OLS model and GGLM-ID models to the full dataset. For convenience to see AIC explicitly in the summary below, we have fit the OLS model as a NGLM-ID (Normal GLM, identity link). NGLM-ID is identical to OLS.

```
RegMod<-glm(LIC~R2+I(R2^2),data=Liver) #GLM default is gaussian identity link (same as OLS)
GMod<-glm(LIC~R2+I(R2^2),family=Gamma(link=identity),data=Liver)

summary(RegMod)
```

```
##
## Call:
## glm(formula = LIC ~ R2 + I(R2^2), data = Liver)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.8691  -1.2797  -0.3111   1.0150   8.6747
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) -3.999e-01  8.985e-01  -0.445   0.6573
## R2           4.100e-02  1.677e-02   2.444   0.0165 *
## I(R2^2)      3.480e-04  6.221e-05   5.594 2.36e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 8.709747)
##
##     Null deviance: 10821.16  on 92  degrees of freedom
## Residual deviance:   783.88  on 90  degrees of freedom
## AIC: 470.17
##
## Number of Fisher Scoring iterations: 2
```

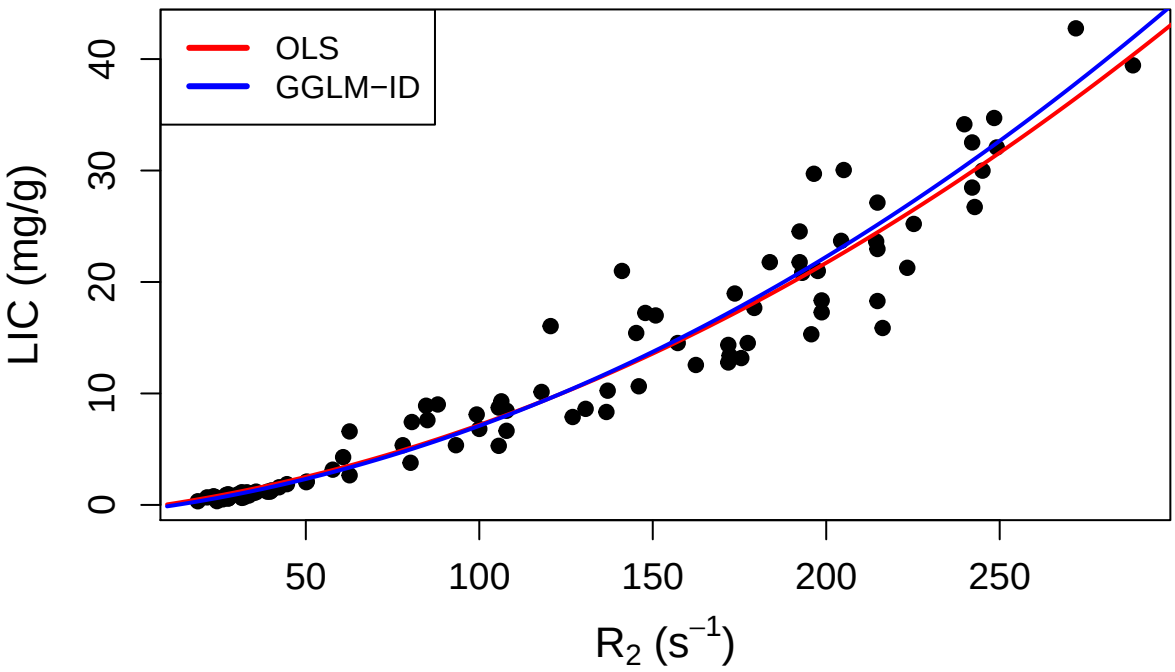
```
summary(GMod)
```

```
##
## Call:
## glm(formula = LIC ~ R2 + I(R2^2), family = Gamma(link = identity),
##     data = Liver)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.54634  -0.21599  -0.04509   0.12213   0.76082
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.534e-01  1.687e-01  -3.281  0.00147 **
## R2           3.890e-02  7.439e-03   5.230 1.10e-06 ***
## I(R2^2)      3.758e-04  4.698e-05   7.998 4.11e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.07256018)
##
##     Null deviance: 125.5624  on 92  degrees of freedom
## Residual deviance:   5.9887  on 90  degrees of freedom
## AIC: 355.09
##
## Number of Fisher Scoring iterations: 7
```

Initial Plot

Lets visualize the data and the fits in an initial plot.

```
par(mfrow=c(1,1))
df<-data.frame(R2=seq(10,300,1))
plot(Liver$R2,Liver$LIC,cex.main=1.5,cex.axis=1.2,cex.lab=1.3,xlab=expression(paste(R[2]," ","(",s^-1,"
lines(df$R2,predict(RegMod,df),col="red",lwd=2)
lines(df$R2,predict(GMod,df),col="blue",lwd=2)
legend(x="topleft",legend=c("OLS","GGLM-ID"),col=c("red","blue"),lwd=3)
```



The fits appear close visually, however this cursory examination can be misleading. In order to get an idea of the prediction uncertainty, we need to fit the models to different realizations of the same data. To do this, we will use the same lognormal error structure as for the parametric simulations done before:

Set Error Parameters

For R2, CV=8% while for LIC, the CV is piecewise. Based on the literature, it will be set to 19% for true LIC< 4 mg/g and 9% for true LIC>9 mg/g. In between 4 mg/g and 9 mg/g, we will simulate the CV later from a uniform distribution ranging from 9% to 19% as the literature does not provide a value for this range.

We will also be using matrices to store the predictions. The rows are the predictions for each run while the columns are different runs. 1000 runs will be performed.

```
TestData<-data.frame("R2obs"=seq(10,300,1))
npred=length(TestData$R2obs)

runs=1000

OLSPred=matrix(rep(NaN,npred*runs),nrow=npred,ncol=runs)
GGLMPred=OLSPred

LIC=Liver$LIC
R2=Liver$R2


CVLIC=rep(0,n)
```

```
CVLIC[which(LIC<4)]=0.19
CVLIC[which(LIC>9)]=0.09
m=length(CVLIC[which(LIC>=4 & LIC<=9)])
CVLIC[which(LIC>=4 & LIC<=9)]=runif(m,0.09,0.19)

CVR2=0.08
```

Perform Simulation

In this section, we simulate R2 and LIC from a lognormal distribution with approximate mean (on log scale) given by log(R2) and log(LIC) and SD (on log scale) given by the CV on the original scale. The beginning (head) of the prediction matrices are also shown. Note that sometimes in the simulation due to the exact data points generated, the GLM algorithm may not converge though this is not a major issue in a practical sense if it does not affect the predictions given by GLM. This simply means that the maximum likelihood (MLE) was not perfectly achieved. We opted to use the glm2 library here which is somewhat more stable than the base glm when a simulation is being performed.We can confirm that there are no NaN values in the predictions.

```
library(glm2)
set.seed(4513)
for(i in 1:runs){

R2obs=rlnorm(n,log(R2),CVR2)
LICnew=rlnorm(n,log(LIC),CVLIC)

OLS<-lm(LICnew~R2obs+I(R2obs^2))
GGLM<-glm2(LICnew~R2obs+I(R2obs^2),family=Gamma(link=identity))

OLSPred[,i]=predict(OLS,TestData)
GGLMPred[,i]=predict(GGLM,TestData)
}
```

```
## Warning: glm.fit2: algorithm did not converge. Try increasing the maximum
## iterations

## Warning: glm.fit2: algorithm did not converge. Try increasing the maximum
## iterations

## Warning: glm.fit2: algorithm did not converge. Try increasing the maximum
## iterations

## Warning: glm.fit2: algorithm did not converge. Try increasing the maximum
## iterations

## Warning: glm.fit2: algorithm did not converge. Try increasing the maximum
## iterations

## Warning: step size truncated due to increasing deviance

## Warning: glm.fit2: algorithm did not converge. Try increasing the maximum
## iterations
```

```r
sum(is.na(OLSPred))
```

```
## [1] 0
```

```r
sum(is.na(GGLMPred))
```

```
## [1] 0
```

Quantile 95% Prediction Intervals

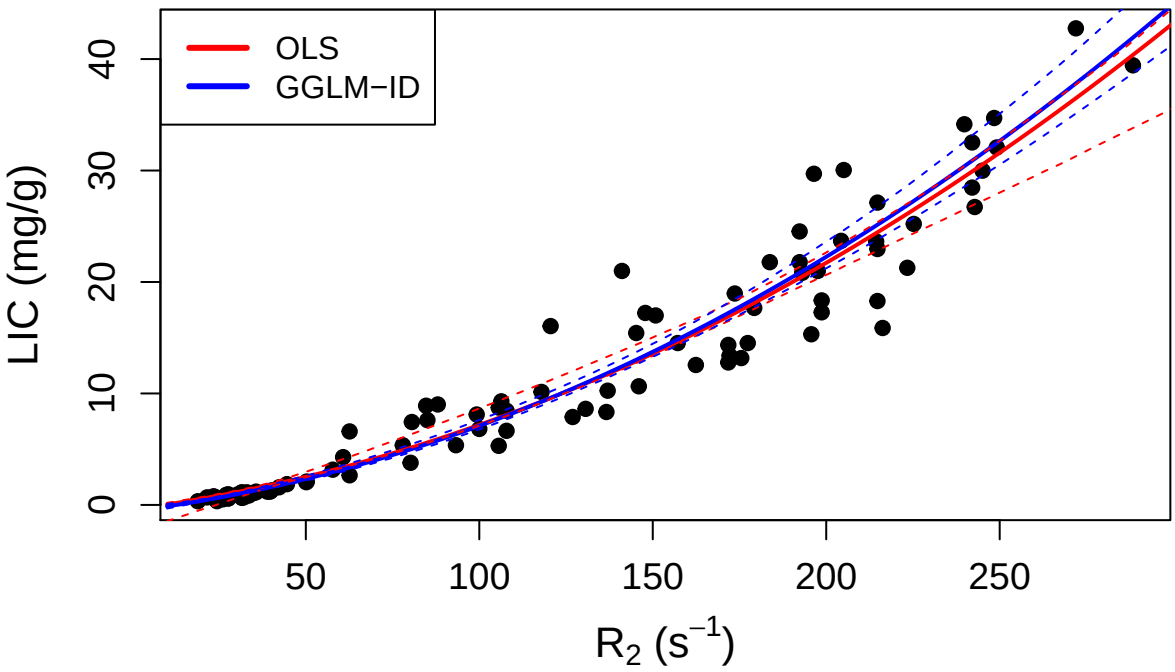
Rather than assuming any parametric distribution, we opted to use the 2.5% and 97.5% quantiles to construct the prediction intervals. First, a function has to be made to feed to the apply() function to make this easier.

```r
Q97.5=function(x){
  return(quantile(x,0.975))
}

Q2.5=function(x){
  return(quantile(x,0.025))
}

OLS97.5=apply(OLSPred,1,Q97.5)
OLS2.5=apply(OLSPred,1,Q2.5)

GGLM97.5=apply(GGLMPred,1,Q97.5)
GGLM2.5=apply(GGLMPred,1,Q2.5)
```

Final Plot

Now we can plot the original fits along with the 95% percentile prediction intervals.

```r
par(mfrow=c(1,1))
plot(Liver$R2,Liver$LIC,cex.main=1.5,cex.axis=1.2,cex.lab=1.3,xlab=expression(paste(R[2]," ","(",s^-1,"
lines(df$R2,predict(RegMod,df),col="red",lwd=2)
lines(df$R2,predict(GMod,df),col="blue",lwd=2)
legend(x="topleft",legend=c("OLS","GGLM-ID"),col=c("red","blue"),lwd=3)
lines(TestData$R2obs,OLS97.5,col="red",lty="dashed")
lines(TestData$R2obs,GGLM97.5,col="blue",lty="dashed")
lines(TestData$R2obs,OLS2.5,col="red",lty="dashed")
lines(TestData$R2obs,GGLM2.5,col="blue",lty="dashed")
```



From the above figure, the OLS fit does not appear to predict well at high R2/LIC. The upper end of the prediction interval for OLS overlaps with the main GGLM-ID curve. The GGLM-ID fit has a more symmetric prediction-interval.