PharmaSUG 2026 - Paper AP155-2026

TFL Generation Using R: Negative Binomial Regression Implementation and Comparison with SAS

Ye Sun, Jianchang Hu, Boehringer Ingelheim

ABSTRACT

Tables, Figures, and Listings (TFLs) are fundamental for reporting and interpreting clinical trial data in regulatory submissions and are traditionally produced using SAS. With increasing interest in R as an open-source alternative for clinical trial analyses, this paper provides a comprehensive comparison of key features between SAS and R implementations for negative binomial regression.

The paper demonstrates that, by manually adjusting the estimated variance-covariance matrix in R, exact matches (to the 0.001 level) for parameter estimates, least-squares means, confidence intervals, and p-values can be achieved using R's glm.nb function compared to SAS's PROC GENMOD procedure. Special attention is required for the dispersion parameter due to differing parameterizations between the two languages. Additionally, the use of the gtsummary package in R for generating formatted output is illustrated.

INTRODUCTION

SAS has been the pre-dominant programming language for statistical outputs production for internal and external communications across the pharmaceutical industry. In recent years, there has been an increasing interest in transition from SAS to open-sourced programming language R. Although many efforts would still be needed to formalize and standardize the validation process for outputs creation with R, many companies have already been using R for internal discussions even for decision making. However, discrepancy between SAS and R outputs exists due to differences in their underlying numerical implementation which may cause confusion when comparing the same set of outputs from both languages and would thus require extra communication efforts.

In this work, we consider negative binomial regression for count endpoint which is widely encountered in clinical trials. Specifically, we assume that

$$Y_i \sim NB(\mu_i, \theta),\ i = 1, \ldots, n$$

with

$$\mathrm{E}(Y_i) = \mu_i = \exp(x_i'\beta), \text{ and } \mathrm{Var}(Y_i) = \mu_i + \mu_i^2/\theta,$$

where Y_i is the count endpoint with its mean being μ_i, θ is the size parameter, x_i denotes the covariates, and β represents the effect sizes for covariates. The reciprocal of the size parameter is often referred as dispersion parameter. We would like to demonstrate the discrepancy in outputs for the negative binomial regression from SAS and R, analyze the root cause of such discrepancy, and show how to reproduce the results by SAS in R. This could act as a reference example for other statistical models on points to check when using R for output creation.

We would also like to demonstrate an R package gtsummary for formatted output generation as such formatted outputs are beneficial for results interpretation and easier for communication.

DATA SIMULATION

A dummy dataset is simulated, including:

- 100 subjects
- grp: a dummy variable with 1:1 subject assignment to treatment (grp = 1) vs placebo (grp = 0); note, variable grpc is a character version of , which takes the value of "Trt" or "Plb".

- $x1$: a continuous variable which follows a normal distribution of mean of 0 and sd of 1;
- $x2$: a categorical variable which take the value of “A” or “B” or “C” with a probability of 0.3, 0.2, 0.5, respectively.
- logtime: An offset for the calculation of rates (e.g time in years) on the log-scale
- y: a negative binomial outcome giving the event counts;

```
library(tidyverse)
library(MASS)
N = 100
# set seed for replication
set.seed(123)

# Treatment Group; 1:1 ratio
grp <- rep(c(0, 1), each = N / 2)

# Covariates (one continuous; one categorical)
x1 <- rnorm(N)
x2 <- factor(sample(LETTERS[1:3], N, replace = TRUE, prob = c(0.3, 0.2, 0.5)))

# Offset
logtime <- log(runif(N, 1, 2))

# Model parameter assumption
beta0 = 0.6
betaTrt = -0.5
beta1 = 0.25
beta2 = c(-0.1, 0.2)
theta = 1 / 2

# Dummy dataset
df <- data.frame(grp, x1, x2, logtime) %>%
  mutate(
    log_rate = case_when(
      x2 == "A" ~ beta0 + betaTrt * grp + beta1 * x1 + logtime,
      x2 == "B" ~ beta0 + betaTrt * grp + beta1 * x1 + beta2[1] + logtime,
      x2 == "C" ~ beta0 + betaTrt * grp + beta1 * x1 + beta2[2] + logtime
    ),
    y = rnegbin(N, mu = exp(log_rate), theta = theta),
    grpc = factor(case_when(grp == 0 ~ "Plb", grp == 1 ~ "Trt"))
  )
```

Program 1 data simulation

EVALUATING MODEL FIT

Assessing the performance and suitability of the adopted Negative Binomial Regression model is a key step in the modeling process. However, since this is not the main topic of this work, here we only list some commonly used diagnostic tests, goodness-of-fit measures, and residual analysis.

	Explanation
Diagnostic tests for overdispersion	To consider a negative binomial model as appropriate, it is critical to diagnose whether overdispersion is present. A common test is to compare the variance and the mean. If the variance is significantly larger than the mean, overdispersion is likely present, and a negative binomial model is more appropriate than a Poisson model.

	Explanation
Goodness-of-fit Measures	Goodness-of-fit measures help determine whether the model can effectively fit to the pattern of the data. Commonly used measures include: Deviance: Smaller residual deviance suggests a better fit. Akaike Information Criterion (AIC): Lower AIC values indicate a better overall fit including the consideration on the number of covariates of the model.
Interpreting Residuals and Errors	Residual analysis can be used for detecting patterns that current model may not capture: Pearson Residuals: Compare observed counts versus the expected counts based on the model. Deviance Residuals: A measure that accounts for discrepancies between the observed and predicted values, providing insight into the model's performance. Plotting these residuals can help identify heterogeneous variance or unmodeled nonlinear relationships.

Table 1. Model fit estimation

NEGATIVE BINOMIAL REGRESSION IN SAS

Negative binomial models can be estimated in SAS using **proc genmod**. On the **class** statement we list the variable **GRPC** and **X2**. The parameter options are given in parentheses. The **param=ref** option changes the coding of **GRPC** and **X2** from effect coding, which is the default, to reference coding. The **ref=first** option changes the reference group to the first level of prog. We have used two options on the model statement. The **type3** option is used to get the multi-degree-of-freedom test of the categorical variables listed on the class statement, and the **dist = negbin** option is used to indicate that a negative binomial distribution should be used. The **link=log** means the model is on the log scale. The **offset=logtime** means logtime is included with coefficient fixed at 1. This makes the model interpret effects on a rate rather than a raw count.

```
ods output ParameterEstimates=ORs lsmeans=lsm;

proc genmod data=df;

    class GRPC (ref='Plb' ref=first) X2 (ref='A' ref=first);

    model y = GRPC x1 x2 / type3 dist=negbin link=log offset=logtime;

    lsmeans GRPC /cl;

    lsmeans GRPC /cl OM;

run;

ods output close;
```

Program 2 SAS Program for Negative binomial regression

Negative binomial regression SAS Outputs

For goodness of fit section, a deviance per degree of freedom close to 1 suggests the model fits reasonably well. In table 2, SAS displays the estimated effects from the negative binomial regression. Take the estimate of grpc for example, if the coefficient for a treatment is −0.6140, the rate ratio

is e^(−0.6140)≈0.54, meaning the treatment group has about 54% the count rate of the reference group. So the treatment effect is about 45% reduction.

Criteria For Assessing Goodness Of Fit			
Criterion	DF	Value	Value/DF
Deviance	95	95.0634	1.0007
Scaled Deviance	95	95.0634	1.0007
Pearson Chi-Square	95	79.8102	0.8401
Scaled Pearson X2	95	79.8102	0.8401
Log Likelihood		72.4815	
Full Log Likelihood		-187.0839	
AIC (smaller is better)		386.1679	
AICC (smaller is better)		387.0711	
BIC (smaller is better)		401.7989	

Table 1 Criteria for Assessing Goodness of Fit

Analysis Of Maximum Likelihood Parameter Estimates							
Parameter		DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept		1	0.7265	0.3518	0.0370	1.4160	4.27	0.0389
grpc	Trt	1	-0.6140	0.3480	-1.2960	0.0680	3.11	0.0776
grpc	Plb	0	0.0000	0.0000	0.0000	0.0000	.	.
x1		1	0.2566	0.2066	-0.1484	0.6617	1.54	0.2143
x2	B	1	-0.3741	0.5074	-1.3685	0.6204	0.54	0.4610
x2	C	1	-0.0500	0.4013	-0.8366	0.7366	0.02	0.9009
x2	A	0	0.0000	0.0000	0.0000	0.0000	.	.
Dispersion		1	2.3705	0.5051	1.5613	3.5992		

Table 2 Analysis of Maximum Likelihood Parameter Estimates

grpc Least Squares Means							
grpc	Estimate	Standard Error	z Value	Pr > |z|	Alpha	Lower	Upper
Trt	-0.00565	0.2679	-0.02	0.9832	0.05	-0.5307	0.5194
Plb	0.6084	0.2449	2.48	0.0130	0.05	0.1284	1.0883

grpc Least Squares Means								
grpc	Margins	Estimate	Standard Error	z Value	Pr > |z|	Alpha	Lower	Upper
Trt	WORK.DF	0.03864	0.2497	0.15	0.8770	0.05	-0.4508	0.5281
Plb	WORK.DF	0.6527	0.2371	2.75	0.0059	0.05	0.1880	1.1173

Table 3 LS-means with observed margins used or not

NEGATIVE BINOMIAL REGRESSION IN R

Negative binomial regression vanilla R outputs

In program 3, we use the glm.nb function from the MASS package to estimate a negative binomial regression. In output 1, R first displays the call. Next, we see the regression coefficients for each of the variables, along with standard errors, z-scores, and p-values.

1. Intercept = 0.72652 means the log expected rate for the reference group: grpc=plb, x2=A and x1=0. By exponentiating, we get the baseline rate as 2.07 events per unit of exposure time.
2. The variable **grpcTrt** has a coefficient of -0.61402. The treatment group has a lower rate than the reference group. We can get the rate ratio of 0.54 by exponentiating it. This suggests the treatment rate is about 46% lower than the reference group, adjusting for x1, x2, and exposure time.
3. For x1=0.25663, it means a one-unit increase in x1, the rate changes by 1.29. So the estimated rate is about 29% higher per 1-unit increase in x1, holding other variables fixed.
4. For x2B=-0.37406, level B has estimated rate ratio of 0.69 by comparison with the reference level of x2. This suggests that level B is about 31% lower than level A.
5. For x2C=-0.04999, level C has estimated rate ratio of 0.95 by comparison with the reference level of x2. This suggests that level C is about 5% lower than level A.

```
fit <- glm.nb(y ~ grpc + x1 + x2 + offset(logtime), data = df, x = TRUE)
summary(fit)
```

Program 3 R Program for Negative binomial regression

```
Call:
glm.nb(formula = y ~ grpc + x1 + x2 + offset(logtime), data = df,
    x = TRUE, init.theta = 0.4218475681, link = log)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.72652    0.35071   2.072   0.0383 *
grpcTrt     -0.61402    0.34148  -1.798   0.0722 .
x1           0.25663    0.18905   1.358   0.1746
x2B         -0.37406    0.50695  -0.738   0.4606
x2C         -0.04999    0.39167  -0.128   0.8984
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(0.4218) family taken to be 1)

    Null deviance: 99.521  on 99  degrees of freedom
Residual deviance: 95.063  on 95  degrees of freedom
AIC: 386.17

Number of Fisher Scoring iterations: 1

              Theta:  0.4218
          Std. Err.:  0.0899
 2 x log-likelihood:  -374.1680
```

Output 1 R Result for negative binomial regression

Comparison and alignment with SAS results

By comparing output 1 with table 2, the regression coefficient estimates are exactly matching those in SAS, but the standard errors are slightly off. The differences in SEs of coefficient estimates are due to the difference in covariance estimation. By default, SAS uses observed Fisher Information matrix in SE calculation while glm.nb() gives standard errors for the regression coefficients and for theta based on the expected Fisher Information. Both are asymptotically correct but would have differences in finite sample performance. It can also be noticed from the comparison that the estimate and SE of dispersion parameter are different. This difference is largely due to different parameterization adopted by SAS and glm.nb(). SAS uses the so-called NB2 parameterization where for $Y_i \sim NB(\mu_i^2, \alpha)$ its mean is μ_i and variance is $\mu_i + \alpha\mu_i^2$. glm.nb() on the other hand uses the size parameterization; that is for $Y_i \sim NB(\mu_i, \theta)$ its mean is μ_i and variance is $\mu_i + \mu_i^2/\theta$. Therefore, the size parameter (theta) estimated from glm.nb() is the reciprocal of the dispersion parameter in SAS.

With these in mind, we can then manually adjust the R outputs to align with SAS. Specifically, we need to calculate the observed Fisher Information matrix and its inverse for SEs. We also need to apply delta method to the size parameter to obtain results for dispersion parameter. Mathematically, assume that

$$Y_i \sim NB(\mu_i, \theta),\ i = 1, \ldots, n$$

with

$\mathrm{E}(Y_i) = \mu_i = \exp(x_i'\beta)$, and $\mathrm{Var}(Y_i) = \mu_i + \mu_i^2/\theta$.

The log-likelihood is given as

$$\ell(\beta, \theta) = \sum_{i=1}^{n} \log\Gamma(y_i + \theta) - \log\Gamma(\theta) - \log(y_i!) + \theta\log\left(\frac{\theta}{\theta + \mu_i}\right) + y_i\log\left(\frac{\mu_i}{\theta + \mu_i}\right).$$

Therefore, the observed Fisher Information can be calculated as follows.

$$\mathbf{I}_O = -\begin{pmatrix} \frac{\partial^2\ell}{\partial\beta\partial\beta'} & \frac{\partial^2\ell}{\partial\beta\partial\theta} \\ \frac{\partial^2\ell}{\partial\theta\partial\beta'} & \frac{\partial^2\ell}{\partial\theta^2} \end{pmatrix},$$

where

$$\begin{aligned} \frac{\partial^2\ell}{\partial\beta\partial\beta'} &= -\sum_{i=1}^{n} \frac{\theta\mu_i(y_i + \theta)}{(\theta + \mu_i)^2} x_i x_i', \\ \frac{\partial^2\ell}{\partial\theta^2} &= \sum_{i=1}^{n} \psi'(y_i + \theta) - \psi'(\theta) + \frac{1}{\theta} - \frac{1}{\theta + \mu_i} + \frac{y_i - \mu_i}{(\theta + \mu_i)^2}, \\ \frac{\partial^2\ell}{\partial\beta\partial\theta} &= \sum_{i=1}^{n} \frac{\mu_i(y_i - \mu_i)}{(\theta + \mu_i)^2} x_i, \\ \frac{\partial^2\ell}{\partial\theta\partial\beta'} &= \left(\frac{\partial^2\ell}{\partial\beta\partial\theta}\right)'. \end{aligned}$$

Here $\psi'(\cdot)$ is the trigamma function. In the implementation, use fitted values $\hat{\mu}_i$ and estimated $\hat{\theta}$ for the calculation of the matrix. The diagonal elements of the inverse of above observed Fisher Information matrix are then the adjusted variance. Thus the adjusted SEs are given by

$$\left(\mathrm{SE}(\hat{\beta}), \mathrm{SE}(\hat{\theta})\right) = \sqrt{\mathrm{diag}\left(\mathbf{I}_O^{-1}\right)}.$$

For the estimation of dispersion parameter, once we obtain the adjusted SE of $\hat{\theta}$ in the previous calculation, then the delta method leads to

$$\text{Var}(\hat{\alpha}) = \text{Var}(1/\hat{\theta}) \approx \left(\frac{1}{(\hat{\theta})^2}\right)^2 \text{Var}(\hat{\theta}) = \frac{\text{Var}(\hat{\theta})}{(\hat{\theta})^4}.$$

Hence, we can obtain

$$\hat{\alpha} = 1/\hat{\theta} \text{ and } \text{SE}(\hat{\alpha}) = \frac{\text{SE}(\hat{\theta})}{(\hat{\theta})^2}.$$

In terms of confidence intervals, SAS provides $(1-\alpha)$ level Wald confidence interval for regression coefficients; that is

$$\text{CI}(\beta_i) = \hat{\beta}_i \pm z_{1-\alpha/2}\text{SE}(\hat{\beta}_i), i = 1, \ldots, p,$$

where $z_{1-\alpha/2}$ is the $(1-\alpha/2)$-th percentile of standard normal distribution.

For the confidence interval of dispersion parameter, by default, SAS constructs a $(1-\alpha)$ level Wald confidence interval at logarithm scale and then back transforms to the original scale to avoid negativity. To align with this result in R, delta method can again be applied to derive the SE of dispersion parameter at log-scale and it yields

$$\text{SE}(\log(\hat{\alpha})) = \text{SE}\left(-\log(\hat{\theta})\right) \approx \text{SE}(\hat{\theta})/|\hat{\theta}|.$$

Then a $(1-\alpha)$ level Wald confidence interval for dispersion parameter is given as

$$\exp(-\log(\hat{\theta}) \pm z_{1-\alpha/2}\text{SE}(\hat{\theta})/|\hat{\theta}|).$$

The following function builds the observed Fisher Information matrix manually in R.

```
glm_nb_cov <- function(mod) {

  # lintr: off
  # please rm -- variable not used!
  # k <- mod$theta
  # lintr: on
  # p is number of regression coefficients
  p <- dim(vcov(mod))[1]

  # construct observed information matrix
  obsInfo <- array(0, dim = c(p + 1, p + 1))

  # first calculate top left part for regression coefficients
  for (i in 1:p) {
    for (j in 1:p) {
      obsInfo[i, j] <- sum(
        (1 + mod$y / mod$theta) *
          mod$fitted.values *
          mod$x[, i] *
          mod$x[, j] /
          (1 + mod$fitted.values / mod$theta)^2
      )
    }
  }

  # information for dispersion parameter
  obsInfo[(p + 1), (p + 1)] <- -sum(
    trigamma(mod$theta + mod$y) -
      trigamma(mod$theta) -
```

```
      1 / (mod$fitted.values + mod$theta) +
      (mod$theta + mod$y) / (mod$theta + mod$fitted.values)^2 -
      1 / (mod$fitted.values + mod$theta) +
      1 / mod$theta
  )

  # covariance between regression coefficients and dispersion
  for (i in 1:p) {
    obsInfo[(p + 1), i] <- -sum(
      ((mod$y - mod$fitted.values) *
        mod$fitted.values /
        ((mod$theta + mod$fitted.values)^2)) *
        mod$x[, i]
    )
    obsInfo[i, (p + 1)] <- obsInfo[(p + 1), i]
  }

  # return variance covariance matrix
  solve(obsInfo, tol = 1e-20)
}
```

Program 4 glm_nb_cov function

Match R with SAS results

To obtain exactly the same results as in SAS we need to re-estimate the covariance matrix using the glm_nb_cov function we defined earlier. Note that to use this function with the fitted results we needed to specify x = TRUE in the glm.nb function so that the design matrix is available.

```
sigma_hat <- glm_nb_cov(fit)

## recalculate confidence intervals, and p-values
coef_est <- coef(fit)
coef_se <- sqrt(diag(sigma_hat)[1:5])

coef_lower <- coef_est - qnorm(0.975) * coef_se
coef_upper <- coef_est + qnorm(0.975) * coef_se

zstat <- coef_est / coef_se
pval <- 2 * (1 - pnorm(abs(zstat)))

new_summary <- cbind(coef_est, coef_se, coef_lower, coef_upper, zstat, pval)

colnames(new_summary) <- c(
  "Estimate",
  "Std. Error",
  "CI_lower",
  "CI_upper",
  "z value",
  "Pr(>|z|)"
)
rownames(new_summary) <- rownames(summary(fit)$coefficients)
new_summary
```

Program 5.1 Re-estimate covariance matrix

```
  sigma_hat <- glm_nb_cov(fit)
  n_coef    <- length(coef(fit))

  ####dispersion match with SAS START##############
  theta.r    <- fit$theta
  se.theta.r <- sqrt(sigma_hat[n_coef + 1, n_coef + 1])

  disp.r    <- 1 / theta.r
  se.disp.r <- se.theta.r / theta.r^2

  se.logdisp.r <- se.theta.r / abs(theta.r)  #delta-method calculation

  ci.logdisp.r <- log(disp.r) + qnorm(0.975) * c(-se.logdisp.r,
  se.logdisp.r)
  ci.disp.r    <- exp(ci.logdisp.r)

  tbl_dispersion <- data.frame(
    Parameter = "Dispersion",
    Estimate  = round(disp.r, 4),
    Std_Error = round(se.disp.r, 4),
    CI_Lower  = round(ci.disp.r[1], 4),
    CI_Upper  = round(ci.disp.r[2], 4),
    Z_Value   = NA,
    P_Value   = NA
  )
```

Program 5.2 Re-estimate dispersion parameter

Parameter	Estimate	Std_Error	CI_Lower	CI_Upper	Z_Value	P_Value
(Intercept)	0.7265	0.3518	0.0370	1.4160	2.0652	0.0389
grpcTrt	-0.6140	0.3480	-1.2960	0.0680	-1.7646	0.0776
x1	0.2566	0.2066	-0.1484	0.6617	1.2419	0.2143
x2B	-0.3741	0.5074	-1.3685	0.6204	-0.7373	0.4610
x2C	-0.0500	0.4013	-0.8366	0.7366	-0.1246	0.9009
Dispersion	2.3705	0.5051	1.5613	3.5992	NA	NA

Output 2 R results matching with SAS

Now the estimates, standard errors, 95% confidence interval limits and p-values are exactly matching those in SAS up to the 4th digit. We can also provide an estimate and CI for weights = proportional, equivalent to SAS OM option.

```
library(emmeans)
# lsmeans with weights = equal, equivalent to SAS default
lsmean1 <- emmeans(
  fit,
  ~grpc,
  data = df,
  vcov. = sigma_hat[1:5, 1:5],
  weights = "equal",
  offset = 0
)
lsmean1
test(lsmean1)
```

Program 6 Lsmean calculation

```
grpc   emmean    SE  df asymp.LCL asymp.UCL
 Plb   0.60837 0.245 Inf     0.128     1.088
 Trt  -0.00565 0.268 Inf    -0.531     0.519

Results are averaged over the levels of: x2
Results are given on the log (not the response) scale.
Confidence level used: 0.95
 grpc   emmean    SE  df z.ratio p.value
 Plb   0.60837 0.245 Inf   2.484  0.0130
 Trt  -0.00565 0.268 Inf  -0.021  0.9832

Results are averaged over the levels of: x2
Results are given on the log (not the response) scale.
```

Output 3 R results matching with SAS

Table generation with gtsummary

A summary table can be created using the gtsummary package, specially tbl_ard_summary, which works well with ARD – Analysis Results Data – objects.

With program 7 attached, a complete efficacy results table comparing event rates between treatment groups – typical of negative binomial regression outputs in clinical trials. Stat_map is the clinch pin – it correctly aligns externally-computed results with gtsummary's internal column structure. Descriptive stats (cards) and inferential stats (tbl_adj_df, tbl_contrast_df) are computed separately, then merged at the table level.

Program 7.txt

Characteristic	Plb N = 50	Trt N = 50
Number of patients in analysis set, N (%)	50 (100.0%)	50 (100.0%)
Primary endpoint up to week 76, N (%)		
No	22 (44.0%)	26 (52.0%)
Yes	28 (56.0%)	24 (48.0%)
Total number of events	138	83
Total observational time (patient years)	74.6	74.6
Unadjusted event rate (events per 1 patient year)	1.85	1.11
Adjusted* event rate (events per 1 patient year)	1.84	0.99
95% confidence interval	(1.14, 2.97)	(0.59, 1.68)
Adjusted event rate ratio		0.541 (0.188)
95% confidence interval		(0.274, 1.070)
p-value (nominal)		0.0776

Output 3 Table generation with gtsummary

CONCLUSION

This paper demonstrates that R, as an open-source alternative to SAS, can effectively reproduce negative binomial regression results traditionally generated in SAS, provided that manual adjustments are made to the estimated variance-covariance matrix in R. By doing so, exact matches (to the 0.001 level) for parameter estimates, least-squares means, confidence intervals, and p-values can be achieved between R's glm.nb function and SAS's PROC GENMOD procedure. However, special attention is required for the dispersion parameter due to differing parameterizations between the two languages, which may result in discrepancies in confidence intervals.

The growing interest in adopting R for clinical trial analyses highlights its potential as a flexible and transparent tool for generating Tables, Figures, and Listings (TFLs) in regulatory submissions. While challenges remain in standardizing validation processes, this work serves as a reference for addressing discrepancies between SAS and R outputs and ensuring alignment in statistical modeling results.

Additionally, the use of the gtsummary package in R is showcased as a valuable tool for generating formatted outputs, enhancing the interpretability of results and facilitating effective communication. This study underscores the feasibility of transitioning to R for clinical trial reporting while maintaining the rigor and precision required for regulatory submissions.

REFERENCES

[1] MASS package, glm.nb reference manual. https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/glm.nb.html

[2] Venables, W. N. & Ripley, B. D. Modern Applied Statistics with S, 4th Edition. Springer, Chapter 7. https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/glm.nb.html.

[3] Negative Binomial Regression in R With glm.nb(). https://r-statistics.co/Negative-Binomial-Regression-in-R.html.

[4] UCLA OARC, Negative Binomial Regression: R Data Analysis Examples. https://stats.oarc.ucla.edu/r/dae/negative-binomial-regression/.

[5] Hilbe, J. M. Negative Binomial Regression, 2nd Edition. Cambridge University Press (2011).

[6] UCLA OARC, Negative Binomial Regression: SAS Data Analysis Examples. https://stats.oarc.ucla.edu/sas/dae/negative-binomial-regression/.

[7] Gtsummary package. https://github.com/ddsjoberg/gtsummary.

[8] Joseph M.Hilbe. Negative Binomial Regression, 2nd Edition. https://api.pageplace.de/preview/DT0400.9781139005968_A23865716/preview-9781139005968_A23865716.pdf.

[9] R vs SAS: Negative Binomial Regression. https://psiaims.github.io/CAMIS/Comp/r-sas_negbin.html.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Ye Sun
yesun2901@gmail.com

Jianchang Hu
jianchang.hu@boehringer-ingelheim.com