Bugs Affecting SUDAAN 11.0.1

This list was updated on July 15, 2013. The most recent list can be found at http://www.rti.org/sudaan/page.cfm/Known_Bugs.

326. SURVIVAL Procedure: Computing variances for conditional and predicted marginals may require a very long run time.

332. ALL Procedures: Formats are not applied to PSUDATA variables when the NOTSORTED option is used

333. DESCRIPT, LOGISTIC (RLOGIST), WTADJUST and WTADJX Procedures: SEMANTIC ERROR when DDF is on an OUTPUT statement with certain other keywords

335. MULTILOG Procedure: Extraneous row in ITBETAS output table with exchangeable working correlations

361. LOGISTIC (RLOGIST) and LOGLINK Procedures: Additional iteration in convergence algorithm for exchangeable working correlations

326. SURVIVAL Procedure: Computing variances for conditional and predicted marginals may require a very long run time.

Description:

The SURVIVAL procedure may run for a very long time when computing variances for conditional and predicted marginals. In addition to cases when variance and/or standard error keywords are explicitly requested on the PRINT or OUTPUT statement, computing variances is also required when confidence intervals or test statistics are requested.

The actual run time is dependent on numerous factors, including the number of observations in the analysis population, the complexity of the model statement, the choice of sampling design, and the specifications of the computer running the analysis. Therefore, run times may vary widely from one analysis to another. However, we've included the following to illustrate just how long "a very long time" might be: We set up a SURVIVAL analysis with about 75,000 observations, a with-replacement sampling design, a single variable on the MODEL statement, and a single variable on the CONDMARG statement. The analysis required 5 seconds to run when the conditional marginal variance wasn't requested. After adding SECNDMRG to the PRINT statement, the analysis required approximately 50 hours to run.

Work-around:

Specifying the TIES=BRESLOW option on the MODEL statement may make the procedure run faster than TIES=EFRON (the default), although it still may take a long time. In the example described above, the procedure finished in 5 hours after switching to TIES=BRESLOW.

Example:

The following SURVIVAL example requests the confidence interval for the conditional marginal, which triggers the computation of the conditional marginal variance. Therefore, this analysis may take a long time to run.

```
proc survival data=mydata design=wr;
    nest stratum psu;
    weight wt;
    class spd2;
    event mortstat;
    model interval = spd2;
    condmarg spd2;
    print beta condmrg lowcm upcm;
run;
```

Adding TIES=BRESLOW to the MODEL statement may help it run faster:

```
    model interval = spd2 / ties=breslow;
```

System	Release Reported	Release Fixed
Windows	10.0.1	not fixed
Solaris	10.0.1	not fixed
Linux	10.0.1	not fixed

332. ALL Procedures: Formats are not applied to PSUDATA variables when the NOTSORTED option is used

Description:

If the NOTSORTED option appears on the PROC statement, and a variable from the PSUDATA file appears on the RFORMAT statement, then SUDAAN displays the unformatted values of that variable in the printed results instead of the formatted values.

Work-around:

Working around this bug is possible using two different approaches.

The first approach is to omit the NOTSORTED option. Choosing this method requires you to sort both the DATA file and the PSUDATA file by the NEST variables prior to running the procedure.

The second approach is to make sure any variable that should be formatted appears on the DATA file. Prior to running the procedure, you should move such variables from the PSUDATA file to the DATA file.

Example:

Suppose the variable named REGION appears on your PSUDATA file, and you want to format its values in the printed results. Also suppose that your DATA file isn't already sorted by the NEST variables. You might do something like this:

```
proc format;
    value regf
        1 = "east"
        2 = "west"
        3 = "south"
        ;
    run;

proc crosstab data=mydata psudata=mypsudata notsorted design=wr;
    nest stratum psu;
    weight wt;
    class region gender;
    table region * gender;
    rformat region regf.;
    print nsum rowper serow;
    run;
```

In this case, SUDAAN will display the values of the REGION variable as "1", "2", and "3" in the CROSSTAB table. If you sort the DATA file ahead of time and omit the NOTSORTED option, then SUDAAN will correctly display the values of REGION as "east", "west", and "south".

System	Release Reported	Release Fixed
Windows	11.0.0	not fixed
Solaris	11.0.0	not fixed
Linux	11.0.0	not fixed

333. DESCRIPT, LOGISTIC (RLOGIST), WTADJUST and WTADJX Procedures: SEMANTIC ERROR when DDF is on an OUTPUT statement with certain other keywords

Description:

When the DDF keyword and another specific keyword are listed together on a single OUTPUT statement in PROC DESCRIPT, RLOGIST, WTADJUST and WTADJX, SUDAAN produces the following semantic error: "On OUTPUT statement #1 please specify statistics from exactly one output group using the option group_name=DEFAULT or the option group_name=ALL". SUDAAN produces this semantic error even when DDF and the other keyword belong to the same output group. The affected keyword combinations are:

In DESCRIPT:
OUTPUT DDF TOTAL

In RLOGIST, WTADJUST, and WTADJX:
OUTPUT DDF MEAN
OUTPUT DDF PERCENT
OUTPUT DDF RHAT (but only when the VAR statement is also present)

Work-arounds:

One method for working around this bug is to specify the keywords on separate OUTPUT statements instead of on a single output statement. Using this method, the two keywords will be saved to separate output files.

Another method is to remove the explicit references to the keywords from the OUTPUT statement and instead request them using the group_name=ALL syntax. Using this method, the two keywords will be saved to a single output file, but additional keywords will be also computed and saved to the output file.

Example:

The following statements will produce the semantic error even though the DDF and MEAN keywords both belong to the MEANCOV output group in RLOGIST:

```
proc rlogist data=mydata design=wr;
    nest stratum psu;
    weight wt;
    class region gender;
    model resp = region gender;
    var cig;
    output ddf mean / filename=out_meancov replace;
    run;
```

By placing the DDF and MEAN keywords on separate OUTPUT statements, the semantic error is avoided:

```
proc rlogist data=mydata design=wr;
    nest stratum psu;
    weight wt;
    class region gender;
```

```
    model resp = region gender;
    var cig;
    output ddf / filename=out_ddf replace;
    output mean / filename=out_mean replace;
    run;
```

The semantic error is also avoided when requesting the entire MEANCOV group instead of requesting DDF and MEAN explicitly:

```
proc rlogist data=mydata design=wr;
    nest stratum psu;
    weight wt;
    class region gender;
    model resp = region gender;
    var cig;
    output / meancov=all filename=out_meancov replace;
    run;
```

System	Release Reported	Release Fixed
Windows	11.0.0	not fixed
Solaris	11.0.0	not fixed
Linux	11.0.0	not fixed

335. MULTILOG Procedure: Extraneous row in ITBETAS output table with exchangeable working correlations

Description:

When assuming exchangeable working correlations in the MULTILOG procedure (that is, when either the R=EXCHANGEABLE or RSTEPS=*n*>0 parameter is specified), SUDAAN sometimes inserts an extraneous set of betas (regression coefficient estimates) into the next-to-last row of the exchangeable portion of the ITBETAS output table. This set of betas also contributes to the reported number of iterations required for the exchangeable step to converge, making that number one greater than it should be. The extraneous row represents a set of betas that were rejected during the convergence algorithm for failing to improve the likelihood when compared to the previous iteration.

Work-around:

No work-around is available for this bug. The extraneous row in the table may be safely ignored.

Example:

Consider the following SUDAAN analysis:

```
proc multilog data=mydata design=wr r=exchangeable;
    nest stratum psu;
    weight wt;
    class group age gender;
    model group = age gender;
    print beta sebeta itbeta;
run;
```

If the convergence algorithm in one of the exchangeable steps (step > 0) rejects a set of betas, the ITBETAS table will include an extraneous row. For example, the betas for the first couple of terms of the model may look like this in the printed ITBETAS table:

```
for: Exchangeable Step = 1.

-------------------------------------------------------------------
Iteration             Independent Variables and Effects
   GROUP (log-                                              AGE
     odds)                      Intercept                   1
-------------------------------------------------------------------
0
   1 vs 4                    3.5098996735          1.4615414746
   2 vs 4                    3.2509864612          0.5918951071
   3 vs 4                    1.0088636412         -0.9158914912
1
   1 vs 4                    3.4365307157          1.6320917865
   2 vs 4                    3.1748479704          0.7655687843
   3 vs 4                    0.8561980635         -0.8061795300
2
   1 vs 4                    3.3631994196          1.4965533627
   2 vs 4                    3.0994801423          0.6275223137
   3 vs 4                    0.7631895882         -0.9405012131
3
   1 vs 4                    3.4365306807          1.6320917219
   2 vs 4                    3.1748479345          0.7655687184
```

```
   3 vs 4                       0.8561980192          -0.8061795941
-----------------------------------------------------------------
```

Looking at just the intercept term for the 1 vs 4 response, we see the following progression of betas through the iterations:

Iteration	BETA
0	3.5098996735
1	3.4365307157
2	3.3631994196
3	3.4365306807

The betas for iterations 1 and 3 are very similar to each other, but the beta for iteration 2 isn't similar to either of them, contrary to what one would expect from a convergent process. The explanation is that the beta for iteration 2 was actually ignored when determining convergence and should have been omitted from the table. The other betas in the ITBETAS table show a similar pattern.

System	Release Reported	Release Fixed
Windows	11.0.0	not fixed
Solaris	11.0.0	not fixed
Linux	11.0.0	not fixed

361. LOGISTIC (RLOGIST) and LOGLINK Procedures: Additional iteration in convergence algorithm for exchangeable working correlations

Description:

When assuming exchangeable working correlations in the LOGISTIC (RLOGIST) and LOGLINK procedures (that is, when either the R=EXCHANGEABLE or RSTEPS=*n*>0 parameter is specified), SUDAAN sometimes continues the convergence algorithm in the exchangeable steps for one iteration beyond the number required for convergence. It computes the additional iteration whenever the MAXXITER parameter is set to a value greater than *k*, where *k* is the number of iterations required for convergence in the exchangeable step.

The impact of the additional iteration is a small change in the BETA and SEBETA keywords as well as any keywords derived from them. Because the model is convergent, the additional iteration should produce a slightly more accurate approximation of these keywords.

Work-around:

If *k* is the number of iterations required for convergence of the exchangeable steps, then set MAXXITER=*k* to force SUDAAN to stop iterating after exactly *k* iterations. To find the value of *k*, first run the procedure with a larger value of MAXXITER, and then subtract 1 from the reported number of iterations.

Example:

Consider the following SUDAAN analysis:

```
proc rlogist data=mydata design=wr r=exchangeable maxxiter=10;
    nest stratum psu;
    weight wt;
    class age gender;
    model resp = age gender;
    print beta sebeta;
run;
```

Suppose the exchangeable step converges in 5 iterations. If you run the RLOGIST procedure with MAXXITER=10, SUDAAN reports that the exchangeable step converges in 6 iterations:

```
Step 1 parameters have converged in 6 iterations.
```

If you re-run the procedure with MAXXITER=5, SUDAAN now reports that the exchangeable step converges in 5 iterations:

```
Step 1 parameters have converged in 5 iterations.
```

The BETAs and SEBETAs produced by these analyses will be different, but the difference should be very small. The following table demonstrates the magnitude of the difference for one particular example:

Iterations	BETA	SEBETA
5	2.8490728214628680	0.0955830138706311

Iterations	BETA	SEBETA
6	2.8490728210778776	0.0955830137995846

System	Release Reported	Release Fixed
Windows	11.0.0	not fixed
Solaris	11.0.0	not fixed
Linux	11.0.0	not fixed