ABSTRACT
Getting values of test statistics and p-values out of SAS and R is quite easy in each of the software packages but also quite different from each other. This paper intends to compare the SAS and R methods for obtaining these values from tests involving Chi-Square and Analysis of Variance such that they can be leveraged in tables, listings, and figures.

This paper will include but not be limited to the following topics:

- SAS ODS trace
- SAS PROC FREQ
- PROC GLM
- R stats::chisq.test() function
- R stats::aov() function
- R broom package functions

The audience for this paper is intended to be programmers familiar with SAS and R but not at an advanced level.

INTRODUCTION
This paper intends to provide the reader with details on how to extract test statistics, commonly used in clinical trials, from SAS and R. We will look at specific examples in each software package but will not cover every possible scenario. First, we will discuss the high-level methodology for making these values available for the final report in SAS and R. The aspect of the test statistics matching between SAS and R is outside the scope of this paper. Then, we will discuss examples along with the output.

SAS METHODOLOGY
SAS uses procedures to produce statistical test results. Extracting those results into a usable form for reporting can be accomplished using a few different methods.

SAS PROC FREQ Example Using the Output Statement

```sas
proc freq data=heart;
        table sex*diabetes / list chisq fisher cmh;
        output out=freqstats chisq fisher cmh;
quit;
```

ODS TRACE DATA SETS
The output delivery system has a feature that allows a user to pick off different pieces of the output SAS sends to the results window and save it as a SAS data set. All the user must do is put the following statement into their program prior to using the statistical procedure of interest.
ODS trace on;

Once ODS trace is turned on, the available data sets are listed in the SAS log. Once you identify the piece of output that you would like as a SAS data set, you can use the ODS output statement to access it. See the excerpt below from a SAS log which utilizes ods trace output.

**SAS PROC FREQ Code Example & Log Excerpt Showing ODS Trace Information**

```sas
ods trace on;
proc freq data=heart;
    table sex*diabetes / list chisq fisher cmh;
quit;
```

**ODS Trace Log Output**

Output Added:

```
Name:      CrossTabFreqs
Label:     Cross-Tabular Freq Table
Template:  Base.Freq.CrossTabFreqs
Path:      Freq.Table1.CrossTabFreqs
```

Output Added:

```
Name:      ChiSq
Label:     Chi-Square Tests
Template:  Base.Freq.ChiSq
Path:      Freq.Table1.ChiSq
```

Output Added:

```
Name:      FishersExact
Label:     Fisher's Exact Test
Template:  Base.Freq.ChiSqExactFactoid
Path:      Freq.Table1.FishersExact
```
Output Added:
--------------
Name: CMH
Label: Cochran-Mantel-Haenszel Statistics
Template: Base.Freq.CMH
Path: Freq.Table1.CMH

--------------

Output Added:
--------------
Name: CommonRelRisks
Label: Common Relative Risk Estimates
Template: Base.Freq.CommonRelRisks
Path: Freq.Table1.CommonRelRisks

--------------

Below are the ods output statements to capture the output in SAS Data Sets.

ods output chisq=chisqout;
ods output fishersexact=fishersexactout;
ods output cmh=cmhout;

proc freq data=heart;
   table sex*diabetes / list chisq fisher cmh;
quit;

After running the above code, output data sets containing the Chi-Square, Fisher’s Exact test, and Cochran-Mantel-Haenszel statistics would be output to work.chisqout, work.fisherexatout, and work.cmhout respectively.

SAS PROC GLM Code Example Using the Output Statement/Option Strategy

proc glm data=heart outstat=glmoutstat;
   class sex;
   model platelets = sex anaemia sex*anaemia;
quit;
### ODS Trace Log Output

Output Added:

-------------
Name: ClassLevels  
Label: Class Levels  
Template: STAT.GLM.ClassLevels  
Path: GLM.Data.ClassLevels
-------------

Output Added:

-------------
Name: NObs  
Label: Number of Observations  
Template: STAT.GLM.NObsNotitle  
Path: GLM.Data.NObs
-------------

Output Added:

-------------
Name: OverallANOVA  
Label: Overall ANOVA  
Template: stat.GLM.OverallANOVA  
Path: GLM.ANOVA.platelets.OverallANOVA
-------------

Output Added:

-------------
Name:          FitStatistics
Label:         Fit Statistics
Template:      stat.GLM.FitStatistics
Path:          GLM.ANOVA.platelets.FitStatistics
-----------------

Output Added:
-----------------
Name:          ModelANOVA
Label:         Type I Model ANOVA
Template:      stat.GLM.Tests
Path:          GLM.ANOVA.platelets.ModelANOVA
-----------------

Output Added:
-----------------
Name:          ModelANOVA
Label:         Type III Model ANOVA
Template:      stat.GLM.Tests
Path:          GLM.ANOVA.platelets.ModelANOVA
-----------------

Output Added:
-----------------
Name:          ANCOVAPlot
Label:         ANCOVA Plot
Template:      Stat.GLM.Graphics.ANCOVAPlot
Path:          GLM.ANOVA.platelets.ANCOVAPlot
-----------------

Below are the ods output statements to capture the output in SAS Data Sets.

```sas
ods output overallanova=overallanovaout;
proc glm data=heart;
   class sex;
   model platelets = sex anaemia sex*anaemia;
quit;
```
After running the above code, output data sets containing the overall ANOVA statistics would be output to work.overallanovaout.

### R METHODOLOGY

R uses function calls to produce statistical test results. R typically stores its results in an R object called a list. To make these results easier to work with, a package called broom can be leveraged. The broom package contains functions such as tidy(), augment(), and glance() which turn results stored in lists to well-organized tibbles.

### R FUNCTION OUTPUT TO LIST

**chisq.test() Function**

```r
stats_chisq <- stats::chisq.test(heart$sex, heart$diabetes)
```
Resulting list named stats_chisq

```
$ statistic: Named num 6.78
 .- attr(*, "names")= chr "X-squared"
$ parameter: Named int 1
 .- attr(*, "names")= chr "df"
$ p.value : num 0.0092
$ method : chr "Pearson's chi-squared test with Yates' continuity correction"
$ data.name: chr "data_heart$sex and data_heart$diabetes"
$ observed : 'table' int [1:2, 1:2] 124 50 70 55
 .- attr(*, "dimnames")=List of 2
 . . ..$ data_heart$sex : chr [1:2] "1" "0"
 . . ..$ data_heart$diabetes: chr [1:2] "0" "1"
$ expected : num [1:2, 1:2] 112.9 61.1 81.1 43.9
 .- attr(*, "dimnames")=List of 2
 . . ..$ data_heart$sex : chr [1:2] "1" "0"
 . . ..$ data_heart$diabetes: chr [1:2] "0" "1"
$ residuals: 'table' num [1:2, 1:2] 1.05 -1.42 -1.23 1.68
 .- attr(*, "dimnames")=List of 2
 . . ..$ data_heart$sex : chr [1:2] "1" "0"
 . . ..$ data_heart$diabetes: chr [1:2] "0" "1"
$ stdres : 'table' num [1:2, 1:2] 2.73 -2.73 -2.73 2.73
 .- attr(*, "dimnames")=List of 2
 . . ..$ data_heart$sex : chr [1:2] "1" "0"
 . . ..$ data_heart$diabetes: chr [1:2] "0" "1"
```

USING BROOM PACKAGE FUNCTIONS

The broom package functions facilitate retrieving statistical results from the list object created by the statistical function and returning them in a tibble.

* tidy()*

The tidy() function summarizes a models statistical findings.

* augment()*

The augment() function adds columns containing statistical results to the original data.

* glance()*

The glance function provides a one row summary of model level statistics.

For the sake of this paper, we will just look at the tidy() function as we just want to save the test statistic and p-value into a tibble; which is easier to work with versus a list.
tidy_stats_chisq <- tidy(stats_chisq)

Resulting tibble named tidy_stats_chisq

<table>
<thead>
<tr>
<th>statistic</th>
<th>p.value</th>
<th>parameter</th>
<th>method</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.783853</td>
<td>0.039198613</td>
<td>1</td>
<td>Pearson's Chi-squared test with Yates' continuity correction</td>
</tr>
</tbody>
</table>

R FUNCTION OUTPUT TO LIST

aov() Function

aov1 <- aov(platelets ~ sex + anaemia + sex:anaemia, data = data_heart)

Resulting list named aov1
aov1_tidy <-
aov1 %>%
tidy()

Resulting aov1_tidy tibble

<table>
<thead>
<tr>
<th>term</th>
<th>df</th>
<th>sumsq</th>
<th>meansq</th>
<th>statistic</th>
<th>p.value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 sex</td>
<td>1</td>
<td>4.462605e+10</td>
<td>44626047643</td>
<td>4.7159809</td>
<td>0.03068018</td>
</tr>
<tr>
<td>2 anaemia</td>
<td>1</td>
<td>8.905784e+09</td>
<td>8905783534</td>
<td>0.9411433</td>
<td>0.33277871</td>
</tr>
<tr>
<td>3 sexanaemia</td>
<td>1</td>
<td>5.532510e+09</td>
<td>5532509893</td>
<td>0.5848633</td>
<td>0.44510146</td>
</tr>
<tr>
<td>4 Residuals</td>
<td>295</td>
<td>2.791505e+12</td>
<td>9462728631</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

CONCLUSION

Obtaining test statistics and p-values from SAS and R is not difficult but is quite different. In SAS you use output SAS data set options and/or ODS TRACE statements. In R you can use the list output from the statistical function or use the broom package functions to get your results. It takes a little time to learn how to work in each software package, but the results are there for the taking!

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Brian Varney
Experis
269-365-1755
brian.varney@experis.com
https://www.experis.com

Any brand and product names are trademarks of their respective companies.