PharmaSUG 2017 - Paper QT12

Same statistical method different results? Don't panic the reason might be obvious.

Iryna Kotenko, Intego Group / Experis Clinical A Manpower Group Company, Kharkiv, Ukraine

ABSTRACT

Modern statisticians and statistical programmers are able to use a variety of analytics tools to perform a statistical analysis. Performing quality control checks or verification analysis using different statistical packages for the same statistical method one may receive unexpectedly unequal results. So the question comes: what is wrong with the calculations? The answer might disconcert: the calculations are valid and the root of discrepancies is the difference in computational methods and default settings that are implemented in each statistical package. The aim of this article is to bring an awareness to the auditory about known inconsistency in computational methods in commonly used analytics tools: SAS®, Python®, R®, and SPSS®.

INTRODUCTION

Many statistical programmer and biostatisticians have their preferences in statistical packages. They are experts in writing a code in given software and are confident in the results this code produces. But sometimes it happens that the person that used to use let's say R has to switch to SAS or vice versa, or the independent qc-er has different taste and uses Python. Quite often the results that different statistical packages produce do not agree and that might rise the questions: do I do anything wrong? Is this something wrong with alternative statistical package? And the answer usually simple: the results are correct to the point of used settings.

This article is aimed to show number of frequently notified differences in most common statistical calls. For the purpose of illustration four statistical packages were used: SAS and SPSS as the most commonly used in analysis of clinical trials, and R and Python the most popular open-sourced statistical packages.

ARE YOU AFRAID OF DISCREPANCIES?

EXAMPLE #1 KOLMAHOROV-SMIRNOFF TEST FOR NORMALITY

In statistics, normality tests are used to determine if a data set is well-modeled by a normal distribution and to compute how likely it is for a random variable underlying the data set to be normally distributed.

The Kolmogorov–Smirnov test (K–S test or KS test) is a nonparametric test of the equality of continuous, one-dimensional probability distributions that can be used to compare a sample with a reference probability distribution (one-sample K–S test), or to compare two samples (two-sample K–S test). It is named after Andrey Kolmogorov and Nikolai Smirnov. In our case it is going to be one-sample K–S test and the reference probability distribution is normal with estimate the population mean and population variance based on the data.

The Kolmogorov–Smirnov statistic quantifies a distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution and the null distribution of this statistic is calculated under the null hypothesis that the sample is drawn from the reference distribution.

The K-S test has been run in all 4 packages on the data that contains 49 records (one per patient) with some overall efficacy score.

SAS

A UNIVARIATE procedure has been used in SAS for the purpose to obtain KS-score:

```
proc univariate data=results normaltest;
  var EFSCORE;
run;
```

and the results are:

Tests for Normality				
Test	**Statistic**		**p Value**	
Shapiro-Wilk	**W**	0.921376	**Pr < W**	0.0030
Kolmogorov-Smirnov	**D**	0.117807	**Pr > D**	0.0876
Cramer-von Mises	**W-Sq**	0.162415	**Pr > W-Sq**	0.0170
Anderson-Darling	**A-Sq**	1.165686	**Pr > A-Sq**	<0.0050

Output 1. Output from a PROC UNIVARIATE

SPSS

K-S test produced by SPSS will have the following code and results:

```
*Nonparametric Tests: One Sample.
NPTESTS
  /ONESAMPLE TEST (EFSCORE)
  /MISSING SCOPE=ANALYSIS USERMISSING=EXCLUDE
  /CRITERIA ALPHA=0.05 CILEVEL=95.
```

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The distribution of EFSCORE is normal with mean 2.003 and standard deviation 0.714.	One-Sample Kolmogorov-Smirnov Test	.086[1]	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

[1] Lilliefors Corrected

Output 2. Output from a NPTESTS SPSS run

As you can see the p-value for K-S test obtained in SPSS corresponds to SAS result to the certain degree.

R

```
ks.test(df$X, "pnorm", mean = 2.003, sd = 0.714)

## Warning in ks.test(df$X, "pnorm", mean = 2.003, sd = 0.714): ties should
## not be present for the Kolmogorov-Smirnov test
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  df$X
## D = 0.11758, p-value = 0.5072
## alternative hypothesis: two-sided
```

The default R run gives the result that differs from the rest. The reason for that is hidden in the note produced by SPSS run. It says “Lilliefors Corrected”.

The Lilliefors test, named after Hubert Lilliefors, professor of statistics at George Washington University, is a normality test based on K-S test and is implemented by default in SAS, SPSS, and Python. It should be mentioned that apart from SPSS none of the packages indicates that Liliefors correction is used.

In order to bring R results in agreement with the rest of the packages one could use the following code:

```
library(nortest)
lillie.test(df$X)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  df$X
## D = 0.11781, p-value = 0.08643
```

PYTHON

Before running K-S test on Python one should prepare the data and calculate z-score because kstest() from scipy.stats library deals with standard normal distribution $\mu = 0$ and $\sigma = 1$.

```
import numpy as np
import scipy.stats as stats
>>> ar_n=stats.zscore(ar)

and after this call K-S test:

>>> stats.kstest(ar_n,'norm')
KstestResult(statistic=0.12029482392816848,pvalue=0.44776961593152764)
```

The results we receive are closer to K-S test results obtained with ks.test in R on the default settings, the reason is the same: Phyton does not produce Lilliefors test instead of K-S for normality testing. If you want to obtain Lilliefors Corrected p-value you should call the following function:

```
import statsmodels.api as sm
sm.stats.lillifors(ar)
```

and the results you receive is in harmony with the others:

```
>>> sm.stats.lillifors(ar)
(0.11780700082211182, 0.086433013548767562)
```

EXAMPLE #2 PEARSON'S CHI SQUARE TEST FOR 2 X 2 TABLE

Let's assume you are asked to calculate chi square test for the 2x2 table:

		Cold	
		Cold	No Cold
Treatment	Placebo	31	109
	VitaminC	17	122

Table 1. Data of incidences of cold depending on preventive Vitamin C treatment

Pearson's chi-squared test ($\chi 2$) is a statistical test applied to sets of categorical data to evaluate how likely it is that any observed difference between the sets arose by chance.

SAS

To perform chi-square test for the given tale you should run the following code in SAS:

```
data ski;
input treatment $ response $ count;
datalines;
placebo cold 31
placebo nocold 109
ascorbic cold 17
ascorbic nocold 122;
run;

proc freq;
weight count;
tables treatment*response/ chisq;
run;
```

The results are:

The FREQ Procedure

Statistics for Table of treatment by response

Statistic	DF	Value	Prob
Chi-Square	1	4.8114	0.0283
Likelihood Ratio Chi-Square	1	4.8717	0.0273
Continuity Adj. Chi-Square	1	4.1407	0.0419
Mantel-Haenszel Chi-Square	1	4.7942	0.0286
Phi Coefficient		-0.1313	
Contingency Coefficient		0.1302	
Cramer's V		-0.1313	

Output 3. Output from a PROC FREQ

R

In R you submit the following code:

```
> ski<-matrix(c(31, 17, 109, 122), ncol=2,
dimnames=list(Treatment=c("Placebo", "VitaminC"), Cold=c("Cold", "NoCold"))
> result<-chisq.test(ski)
> result

data:  ski
X-squared = 4.1407, df = 1, p-value = 0.04186
```

The results do not agree with Chi-Square produced by SAS, however are comparably equal to SAS Continuity Adjusted Chi-Square. These discrepancy is caused by Yates correction that is implemented in with Continuity Adjusted Chi-Square and is used in R by default. The effect of Yates' correction is to prevent overestimation of statistical significance for small data. To obtain uncorrected chi-square value the following code should be run:

```
> ###Pearson's Chi-squared test withOUT Yates' continuity correction
> result<-chisq.test(ski, correct=FALSE)
> result

Pearson's Chi-squared test
data:  ski
X-squared = 4.8114, df = 1, p-value = 0.02827
```

PYTHON

Those who use Python to test 2x2 table with good chance would write the following simple code:

```
stats.chisquare([[17,122],[31,109]])
```

and would receive the following results:

```
>>> stats.chisquare([17,122],[31,109])
Power_divergenceResult(statistic=7.8730393607576206,
pvalue=0.0050177245029165646)
```

As we can see p-value 0.0050 does not correspond to any of p-values obtained by different statistical packages. The reason is that actually *stats.chisquare* was built to test one-way chi-square test. This chi-square test tests the null hypothesis that the categorical data has the given frequencies and by default the categories are assumed to be equally likely.

In contrast to this our example is aimed to test of independence of variables in a contingency table and thus a special application of chi-square should be used:

```
>>> stats.chi2_contingency([[17,122],[31,109]])(4.1406789213805189,
0.041864375625494867, 1, array([[ 23.91397849, 115.08602151], [24.08602151,
115.91397849]]))
```

As we can see we obtain Yates corrected result by default (p-value = 0.0419). In order to eliminate this correction you should set correction parameter in this function to FALSE:

```
>>> stats.chi2_contingency([[17,122],[31,109]],correction=False)
(4.8114126463207896, 0.028271860246822603, 1, array([[ 23.91397849,
115.08602151], [ 24.08602151, 115.91397849]]))
```

EXAMPLE #3 ANOVA F-VALUE

Let's assume you are asked to perform analysis of variance and calculate F values for the efficacy outcome EFOVR (from example #1) with gender and treatment arm as fixed factors.

ANOVA is used to simultaneously compare two or more group means based on independent samples from each group. To use the F-test to determine whether group means are equal, it's just a matter of including the correct variances in the ratio.

SAS

In SAS the ANOVA call looks like this:

```
proc glm data=result alpha=0.05;
   class ARMCD DMSEX;
   model EFOVR = ARMCD DMSEX ARMCD*DMSEX /intercept;
run;
quit;
```

and the results are:

The GLM Procedure

Dependent Variable: EFOVR

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	4	198.8449254	49.7112313	101.18	<.0001
Error	45	22.1081996	0.4912933		
Uncorrected Total	49	220.9531250			

R-Square	Coeff Var	Root MSE	EFOVR Mean
0.095883	35.00151	0.700923	2.002551

Source	DF	Type I SS	Mean Square	F Value	Pr > F
Intercept	1	196.5003189	196.5003189	399.97	<.0001
ARMCD	1	0.8954103	0.8954103	1.82	0.1838
DMSEX	1	1.4356954	1.4356954	2.92	0.0943
ARMCD*DMSEX	1	0.0135009	0.0135009	0.03	0.8691

Source	DF	Type III SS	Mean Square	F Value	Pr > F
Intercept	1	44.40527635	44.40527635	90.38	<.0001
ARMCD	1	0.35630441	0.35630441	0.73	0.3989
DMSEX	1	0.63832719	0.63832719	1.30	0.2604
ARMCD*DMSEX	1	0.01350085	0.01350085	0.03	0.8691

Output 4. Output from a PROC GLM procedure

By default SAS produces a series of tables with information about general fit of the model, R-square value, and Type I Sum of Squares (SS) and Type III SS statistics.

What is a difference between Type I SS and Type III SS? Consider a model that includes two factors A and B; there are therefore two main effects, and an interaction, AB. The full model is represented by SS(A, B, AB). Type I SS also called "sequential" sum of squares and calculates SS(A) for factor A, SS(A, B) - SS(A) for factor B and SS(A, B, AB) - SS(A, B) for interaction. Type I SS is sensitive to the order of factors in your model. Type III SS calculates SS(A, B, AB) - SS(B, AB) for factor A and SS(A, B, AB) - SS(A, AB) for factor B and does not care the order of factors in the model.

SPSS

Interactive run of the following code gives us the table below:

```
UNIANOVA EFOVR BY ARMCD DMSEX
  /METHOD=SSTYPE(3)
  /INTERCEPT=INCLUDE
  /CRITERIA=ALPHA(0.05)
  /DESIGN=V1 V2 V1*V2.
```

Tests of Between-Subjects Effects

Dependent Variable: EFOVR

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	2.345[a]	3	.782	1.591	.205
Intercept	44.405	1	44.405	90.384	.000
DMSEX	.638	1	.638	1.299	.260
ARMCD	.356	1	.356	.725	.399
DMSEX *ARMCD	.014	1	.014	.027	.869
Error	22.108	45	.491		
Total	220.953	49			
Corrected Total	24.453	48			

a. R Squared = .096 (Adjusted R Squared = .036)

Output 5. Output from a UNIANOVA SPSS run

The F values for fixed factors its interaction and intercept agree in SAS and SPSS, although F value for model is different in these two packages. The explanation of this effect is in number of variables (degrees of freedom) considered for building the F-score. SAS uses 4 (gender, arm, its interaction and intercept) when SPSS uses 3 (gender, arm and intercept).

R

If you run the standard crf.lm function in R you will definitely receive results that differ from both SAS and SPSS. R doesn't do anything wrong, but as we see could already notice the example above R simply has a different default configurations. So here are 3 thing you have to check before you run you R script:

- Set each independent variable as a factor
- Set the default contrast to helmert
- Conduct analysis using Type III Sums of Squares

The code that repeats SAS and SPSS results should be as follows:

```
# Set the variables to factors
> my.data$dmsex <- as.factor(my.data$dmsex)
> my.data$armcd <- as.factor(my.data$armcd)
> options(contrasts = c("contr.helmert", "contr.poly"))
> install.packages("car",dependencies = TRUE)
> crf.lm <- lm(efovr~dmsex*armcd,data=my.data)
> library(car)
> Anova(crf.lm,type=3)
```

The results will be:

```
## Anova Table (Type III tests)
##
## Response: efovr
##             Sum Sq Df F value    Pr(>F)
## (Intercept) 44.405  1 90.3844 2.473e-12 ***
## dmsex        0.638  1  1.2993    0.2604
## armcd        0.356  1  0.7252    0.3989
## dmsex:armcd  0.014  1  0.0275    0.8691
## Residuals   22.108 45
```

CONCLUSION

All four tested packages gave algorithmically correct results. It is user's responsibility to understand what is implemented in certain methods and what statistics is behind it. In all of the described cases reading documentation and statistical literature was really helpful for getting understanding what the source of discrepancies was.

It is always nice to try something new as it usually challenges you to gain more knowledge on the matter you considered yourself familiar with.

REFERENCES

Walker, Glenn A., and Shostak, Jack. 2010. Common Statistical Methods for Clinical Research with SAS® Examples, Third Edition. Cary, NC: SAS Institute Inc.

Venables, W.N., Smith, D.M. and the R Development Core Team. 2005. An Introduction to R, Version 2.2.0.

"ANOVA (and R)" http://goanna.cs.rmit.edu.au/~fscholer/anova.php.

David Stanley. "Ensuring R Generates the Same ANOVA F-values as SPSS". R-bloggers. 2015. https://www.r-bloggers.com/ensuring-r-generates-the-same-anova-f-values-as-spss/

ACKNOWLEDGMENTS

Thanks to Iaroslav Domin for providing R support, Peter Lord and Chad Melson for reviewing the article and Andrii Rekalo for inspiring to explore this topic.

RECOMMENDED READING

- *SAS/STAT(R) 9.2 User's Guide, Second Edition*
- *Stack Overflow http://stackoverflow.com*
- *Stack Exchange http://stats.stackexchange.com*
- *Numpy and Scipy Documentation*
- *Getting Help with R https://www.r-project.org/help.html*

CONTACT INFORMATION

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

Iryna Kotenko
Irina.Kotenko@intego-group.com