SAS Code Skeletons – Units 1-6

# t-tests

## *1 Sample t-test*

**PROC** **TTEST** SIDES= \_\_\_ DATA= \_\_\_ MU0= \_\_\_ ;

VAR \_\_\_ ;

**RUN**;

Notes:

* SIDES: u, l, or 2 for upper, lower, or 2-sided, respectively. 2-sided is the default if you do not specify.
* MU0: number in null hypothesis (0 by default)
* VAR: response variable
* use “DIST = LOGNORMAL” in first line to do a t-test on log transformed data

## *2 (independent) Sample t-test or Welch’s t-test*

**PROC** **TTEST** SIDES= \_\_\_ DATA= \_\_\_ ;

VAR \_\_\_ ;

CLASS \_\_\_ ;

**RUN**;

Notes:

* SIDES: u, l, or 2 for upper, lower, or 2-sided, respectively. 2-sided is the default if you do not specify.
* VAR: response variable
* CLASS: grouping variable
* use “DIST = LOGNORMAL” in first line to do a t-test on log transformed data
* Welch’s t-test is labelled “Satterthwaite”

## Paired t-test

**PROC** **TTEST** DATA= \_\_\_ SIDES= \_\_\_ ;

PAIRED \_\_\_ \* \_\_\_;

**RUN**;

Notes:

* SIDES: u, l, or 2 for upper, lower, or 2-sided, respectively. 2-sided is the default if you do not specify.
* PAIRED: response variables (don’t forget the \* between them)
* use “DIST = LOGNORMAL” in first line to do a t-test on log transformed data

# ANOVA & associated tests

## ANOVA

**PROC** **GLM** DATA = \_\_\_ PLOTS = (DIAGNOSTICS RESIDUALS);

CLASS \_\_\_;

MODEL \_\_\_ = \_\_\_;

**RUN**; **QUIT**;

Notes:

* The PLOTS option is optional. Here I have it displaying all the diagnostic and residual plots, but some of them are “packed” together. To “unpack” them, add “(UNPACK)” (with the parentheses) after PLOTS but before the equal sign.
* CLASS: grouping variable
* MODEL: response variable = grouping variable
* QUIT is usually not necessary, but there are occasional problems with PROC REG and PROC GLM (among others), continuing to run even after they should stop. This is to make them stop when they are done, preventing those occasional problems in SAS. It is good practice to use it every time on these procs to be safe (but I’m sure you’ll see a time or two this semester that I forget).

## Multiple Comparisons

**PROC** **GLM** DATA = \_\_\_ ;

CLASS \_\_\_;

MODEL \_\_\_ = \_\_\_;

MEANS \_\_\_ / TUKEY;

LSMEANS \_\_\_ / ADJUST=TUKEY;

**RUN**; **QUIT**;

**Compare all groups to eachother - Tukey**

**PROC** **GLM** DATA=SPOCK;

CLASS JUDGE;

MODEL PERCENT=JUDGE;

MEANS JUDGE / TUKEY LINES CLDIFF;

**RUN**; **QUIT**;

Notes:

* MEANS gives you confidence intervals or the table telling you which are different (these options can be forced with CLDIFF or LINES, respectively). LSMEANS gives the estimated (sample) means and two-way table of p-values
* MEANS: grouping variable
* Instead of TUKEY, you could do BON or REGWQ or, if applicable, DUNNETT ("\_\_\_") where the group you are comparing to goes in the blank, or one of the other options. There are lots of options, all listed at
* LSMEANS: grouping variable. TUKEY can also be exchanged for different adjustments here.
* While this is enough to get confidence intervals and determine which are significantly different, it does not give us p-values.

For p-values, add the line

LSMEANS JUDGE / ADJUST=TUKEY; into your PROC GLM.

**Compare one control group to everything else – DUNNETT**

* Though we wouldn’t necessarily call Spock’s judge a control, we are specifically interested in comparing the other judges to him, so we could argue that Dunnett’s procedure is appropriate.
  + *Note: this will not allow us to answer the second question about how the other judges compare to each other.*

PROC GLM DATA=SPOCK;

CLASS JUDGE;

MODEL PERCENT=JUDGE;

MEANS JUDGE /DUNNETT ("Spock's");

RUN; QUIT;

## Contrasts

**PROC** **GLM** DATA=\_\_\_ ALPHA = \_\_\_ ;

CLASS \_\_\_ ;

MODEL \_\_\_ = \_\_\_ / CLPARM;

ESTIMATE "\_\_\_" \_\_\_ **\_\_\_** / divisor = **\_\_\_**;

**RUN**;

**PROC GLM DATA=SPOCK;**

**CLASS JUDGE;**

**MODEL PERCENT=JUDGE / CLPARM;**

**ESTIMATE "Spock's vs. Others" JUDGE 1 1 1 1 1 1 -6 / divisor = 6;**

**RUN; QUIT;**

Notes:

* ALPHA: set the alpha level for your confidence intervals. Default is 0.05 (95% confidence), but if you are doing multiple contrasts, you should adjust this. I suggest using the Bonferroni adjustment; divide 0.05 by the number of contrasts you are going to do. So for two contrasts use 0.025 , for three 0.01667 , for four 0.0125, etc.
* CLASS and MODEL same as above (CLPARM is to get a confidence interval for the contrast)
* ESTIMATE: in order, the blanks will be (1) a title for your contrast, (2) the grouping variable, (3) the weights for your groups (which should add to 0), and (4) the divisor if you want to scale down the weights. If you do not need the divisor you can stop this statement after the weights.
* You can also use the CONTRAST statement, but it will only give you the ANOVA row and p-value, not a point estimate or confidence interval. The structure is the same as the ESTIMATE statement, just without the divisor option.

## Welch’s ANOVA

**PROC** **GLM** DATA = \_\_\_ ;

CLASS \_\_\_;

MODEL \_\_\_ = \_\_\_;

MEANS \_\_\_ / WELCH HOVTEST;

**RUN**; **QUIT**;

Notes:

* MEANS: grouping variable
* (Other options the same as above. You can also include the plots here if you want.)
* WELCH is all that is necessary after the “/” to get Welch’s ANOVA (there will be a table below our usual ANOVA tables labelled “Welch’s ANOVA for [response variable]”).
* HOVTEST is to run a test for homogeneity of variance. I generally don’t like these tests (see Unit 3 slides if you don’t remember why), but this is how you can get them if you want to use them as secondary evidence. The default is Levene’s test, but there are other options you can use instead. See <https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/statug_glm_sect018.htm#statug.glm.means_opt_hov> for other possibilities for HOVTEST.
* to do multiple comparisons or planned contrasts after Welch’s ANOVA, do multiple t-tests with an adjusted significance level. I suggest a Bonferroni adjustment because it is easiest to adjust on your own.

# Nonparametric Tests

## Signed Rank or Sign Test

**PROC** **UNIVARIATE** DATA= \_\_\_ MU0= \_\_\_;

VAR \_\_\_;

**RUN**;

Notes:

* MU0: number in null hypothesis (0 by default)
* VAR: response variable
* There is no way to specify a one-sided test, but if your median (also displayed) is on the “correct” side (i.e. if it falls within the range of the alternative hypothesis), you can divide the reported p-value in half to get the correct p-value. For example, if your alternative is that the median is bigger than 0 and your data’s median is bigger than 0, you can divide the reported p-value in half to get the correct p-value.

## Rank Sum or Kruskal-Wallis Test

**PROC** **NPAR1WAY** WILCOXON DATA= \_\_\_ ;

VAR \_\_\_ ;

CLASS \_\_\_ ;

EXACT WILCOXON / MC SEED= \_\_\_ ;

**RUN**;

**PROC** **NPAR1WAY** WILCOXON DATA=SPOCK;

VAR PERCENT;

CLASS JUDGE;

EXACT WILCOXON / MC SEED = **9011957**;

TITLE 'Kruskal-Wallis Test (all judges have separate medians)';

**RUN**;

Notes:

* VAR: response variable
* CLASS: grouping variable
* SEED: an option to set a number (any number) to start your permutations so you can replicate them should you need to. In other words, you will get the same results every time even though the groups are randomly permuted.
* The EXACT statement is optional, but if you use it **do not forget to include “MC”**. MC tells it to not do every permutation of ranks, but only a set amount (default is 10000)

## Permutation Test

In addition to the code we have used previously (the PROC IML code, not included here), you can use PROC NPAR1WAY to do a permutation test. This will not give you the more detailed information that we got before, such as the histogram and information on the individual permutations, but it will give you the p-value (labeled as “Estimate” in the last table).

**PROC** **NPAR1WAY** SCORES=DATA DATA=\_\_\_;

VAR \_\_\_ ;

CLASS \_\_\_ ;

EXACT SCORES=DATA / MC seed = \_\_\_ ;

**RUN**;

Notes:

* VAR: response variable
* CLASS: grouping variable
* SEED: an option to set a number – any number – to start your permutations so you can replicate them should you need to. In other words, you will get the same results every time even though the groups are randomly permuted.
* **Do not forget to include “MC”**. MC tells it to not do every permutation, but only a pre-set amount (default is 10000).

**Beer Data Code**

Libname home “/home/bkolovich1/my\_courses”;

data home.beer\_modified;

infile "/home/bkolovich1/my\_courses/Beer Price Data.csv" dlm=',' firstobs=2;

input group Price\_per\_pack Price\_per\_pack\_ln Cases Cases\_ln;

run;

proc contents data=home.beer\_modified;

proc print data=home.beer\_modified;

proc means data=home.beer\_modified;

class group;

var Cases;

run;

proc univariate data=home.beer\_modified plots;

class group;

var price\_per\_pack;

histogram;

qqplot;

run;

proc npar1way data=home.beer\_modified wilcoxon;

class group;

var price\_per\_pack;

exact wilcoxon / mc;

run;