SAS Code Skeletons – Units 1-6

Most of these were used in examples in class or the asynchronous material. If you need more help or want a filled in example, I suggest starting there.

I have added in some starting points for R functions as well, but did not elaborate on them nearly as much since we are primarily using SAS. You should expect to do some extra research on the functions if you are working in R; the information here will generally not be enough to jump right in. A good start for working iwht any function is to go to the R help page for that function, which can be accessed by, for instance, running "?t.test" (no quotes and, of course, the function you want) in R.

t-tests

1 Sample t-test

```
PROC TTEST SIDES= ___ DATA= ___  H0= ___ ;
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

R function: t.test()

- use alternative = "two.sided", "less" or "greater" for sides of test. 2-sided is the default
- use mu = ___ to set the number in your null hypothesis (0 by default)
- if you need a log transformation, you would need to do the transformation (using log()) yourself and transform the confidence interval endpoints back (using exp())

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”

R function: t.test()

- Default is unequal variances (Welch’s). For equal variances (Pooled) use argument “var.equal=F”
- use alternative = "two.sided", "less" or "greater" for sides of test. 2-sided is the default
- if you need a log transformation, you would need to do the transformation (using log()) yourself and transform the confidence interval endpoints back (using exp())

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

R function: t.test() with argument “paired=T”

- use alternative = "two.sided", "less" or "greater" for sides of test. 2-sided is the default
- if you need a log transformation, you would need to do the transformation (using log()) yourself and transform the confidence interval endpoints back (using exp())

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).

R function: anova(lm()) or anova(aov())

- the "formula" (first argument) in lm() or aov() should be entered as "[response]~[grouping]". If the grouping variable is entered as numbers, use "[response]~as.factor([grouping])", so R knows to treat it as categories, not a numeric variable.
- You could also first run lm() or aov() and save the results to an object and then run anova() with that saved object as the first argument

Multiple Comparisons

```
PROC GLM DATA = ____ ;
CLASS ____;
MODEL ____ = ____;
MEANS ____ / TUKEY;
LSMEANS ____ / ADJUST=TUKEY;
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 https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_glm_sect018.htm under "Perform multiple comparison tests" in the table.
- LSMEANS: grouping variable. `TUKEY` can also be exchanged for different adjustments here.
- (Other options the same as above. You can also include the plots here if you want.)

R functions (disclaimer: I do not have any experience using R for multiple comparisons. Google turned up a few options, which I tested out, but I'm definitely not an expert here! You might want to do a little digging around yourself.)

- Tukey: TukeyHSD(aov()) with aov() as above. You could also save your aov() results when you do ANOVA and then put that object into TukeyHSD()
- Bonferroni: pairwise.t.test() with argument p.adj="bonf"
- I unfortunately couldn't quickly turn up a quick way to do Dunnett's test, but I bet it is in the package "multcomp" that I didn't have time to go through thoroughly.

Contrasts

```
PROC GLM DATA=___ ALPHA = ___ ;
CLASS ___ ;
MODEL ___ = ___ / CLPARM;
ESTIMATE "___" ___ ___ / divisor = ___;
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 $\left(\frac{0.05}{2}\right)$, for three 0.01667 $\left(\frac{0.05}{3}\right)$, 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.

I don't know any R functions for contrasts and wasn't able to find any simple function to do so. You would figure them out step by step by finding the estimate and its standard error and then calculating the p-value and/or confidence interval from that (formulas in the text/asynchronous material).

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.

R function: oneway.test()

- formula argument same as in lm() or aov()

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.

R functions

- Signed Rank Test: wilcox.test()
- Sign Test: use binom.test(). The first argument is the number of positives, the second is the number of non-zero observations. If you already have your data as a numeric vector (say it was called "x"), you can use binom.test(sum(x>0),sum(abs(x)>0)). The output will refer to the proportion of "successes" (positives), but the p-value is the same as for a Sign Test since the Binomial/Proportion Test is and equivalent test.

Rank Sum or Kruskal-Wallis Test

```
PROC NPAR1WAY WILCOXON DATA= ___ ;
VAR ___ ;
CLASS ___ ;
EXACT WILCOXON / 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.
- 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)

R functions

- Rank Sum Test: wilcox.test()
- Kruskal-Wallis Test: Kruskal.test() [you could actually use this for Rank Sum as well]

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).

R: I don't know of a function that will do this in R. You would need to code the permutation (probably using sample()), calculation of the test statistic from each permutation, and, finally, calculation of the p-value from all your test statistics. You will obviously need to do the permutation and test statistic calculation repeatedly; a good place to start here would be to look into for() loops or using replicate(). This is all very doable, but be warned that figuring it all out the first time will probably be fairly time consuming, especially if you are new to R.