One factor Repeated Measures ANOVA with R

A template document for the many and varied approaches.

Bruce Dudek

2020-05-10

Contents

Preface

This document can be a standalone "how-to" document for R users. However, it is primarily intended for students in the APSY510/511 statistics sequence at the University at Albany. It is a fairly thorough treatment of graphical and inferential evaluation of one-factor designs. It presumes prior background coverage of the repeated measures ANOVA logic from standard textbooks such as Howell (2013), Keppel (2004), or Maxwell, Delaney and Kelley (2017). The analyses are intended to parallel and exhaust the methods already covered with SPSS, and to extend them to many additional topics.

This book/monograph uses the **bookdown** package (Xie, 2020a) for R (R Core Team, 2020), which was built on top of **rmarkdown** (Allaire et al., 2020) and **knitr** (Xie, 2015). RStudio (RStudio Team, 2015) was used for all writing and R programming.

Chapter 1

Background and R Setup

The goal of this document is provision of a template for using R to evaluate data from a 1-factor repeated measures design that is often called a within-subjects problem. Rather than providing one data point for a DV measurement as was the case for the "between-groups" design, each case provides more than one measurement since they are measured "repeatedly".

The standard R axiom that there are always multiple ways of performing any task is never more accurate than with the ANOVA models. Beginning with graphical depiction and extending to standard NHST inferences, contrast analysis and post hoc tests, and evaluation of assumptions, etc., we can add to that list major divisions in approaches to repeated measures analysis, and this document could become very very long.

This document

- Is intended for use by APSY511 course at the University at Albany, but can be more broadly used by data analysts.
- Is a fairly full one-factor repeated measures anova exposition for a five category design.
- Implements graphical summaries and numerical descriptions in an EDA section.
- Approaches ANOVA as linear modeling and is supplemented with analytical contrasts, and multiple comparison tests.
- Includes a section on the Multivariate approach to the repeated measures problem.
- Provides templates for both the traditional Univariate/GLM approach as well as linear mixed models approaches.
- Includes graphical and inferential evaluation of assumptions.
- Provides brief illustrations of Bayes Factor, resampling, and robust methods, as well as a non-parametric approach.

The document is always under development.

One of the primary goals is to reproduce all the work we have accomplished with the SPSS GLM, and MANOVA procedures (and then some).

Several R packages are required:

```
#if (!requireNamespace("BiocManager", quietly = TRUE))
#    install.packages("BiocManager")
#BiocManager::install("Biobase", version = "3.8")
```

```
# load packages and import data
library(afex)
library(BayesFactor)
library(car)
library(emmeans)
library(ez)
library(foreign)
library(ggplot2)
library(ggthemes)
library(granova)
library(gt)
library(kableExtra)
library(knitr)
library(lme4)
library(multcomp)
library(nlme)
library(nortest)
library(permuco)
library(phia)
library(plyr)
library(psych)
library(rmarkdown)
library(sciplot)
library(sjstats)
library(WRS2)
```

Package citations for packages loaded here (in the above order): **afex** (Singmann et al., 2020), **BayesFactor** (Morey and Rouder, 2018),
car (Fox et al., 2020), **emmeans** (Lenth, 2020), **ez** (Lawrence, 2016), **ggplot2** (Wickham et al., 2020), **ggthemes** (Pruzek and Helmreich, 2014), **granova** (Pruzek and Helmreich, 2014), **gt** (Iannone et al., 2019), **kableExtra** (Zhu, 2019), **knitr** (Xie, 2020b), **lme4** (Bates et al., 2019), **multcomp** [@-multcomp], **nlme** [@-nlme], **nortest** (Gross and Ligges, 2015), **permuco** (Frossard and Renaud, 2019), **phia** (De Rosario-Martinez, 2015), **psych** (Revelle, 2020), **rmarkdown**(Allaire et al., 2020), **sciplot** (Morales et al., 2020), **sjstats** (Lüdecke, 2020), **WRS2** [@-WRS22019]

1.1 Resources

The following list will provide a good start for those needing a broader background in ANOVA techniques and more detailed sources for the primary packages employed in this document.

In addition, the following internet resources can be helpful.

- Salvatore S. Mangiafico's R Companion: https://rcompanion.org/handbook/I_09.html
- http://dwoll.de/rexrepos/posts/anovaRBp.html
- http://www.jason-french.com/tutorials/repeatedmeasures.html
- https://www.datanovia.com/en/lessons/repeated-measures-anova-in-r/#one-way-repeated-measures-anova

Chapter 2

Import the primary data set and do EDA

Many applications of repeated measures designs involve simply tracking participant across time and measuring the DV at fixed time points. It is also possible to employ such a design when the IV is a manipulated variable. The levels of the repeated factor thus represent different "treatment" conditions. The primary example data set used here is of this latter type. In such experiments, the order of treatments is often randomized across participants/subjects. The data set used here is a textbook example, taken from the Keppel textbook (Keppel and Wickens, 2004, exercise #1, Ch. 16, pp 366-367).

The study oulined in the exercise presumed to evaluate an appetitive behavior among rats, tongue protrusion (called DV in the data set). Tongue protrusions are also used in social situations, presumably as part of a chemical senses system for evaluating airborne molecules that may carry social significance. Accordingly, a study was designed where rats were exposed to bedding types. Two control types of conditions were employed, clean bedding and bedding from the rat's own home cage. Three other conditions were bedding from other species, iguana, whiptail lizard, and kangaroo rat. The IV (called "type" in the data frame) thus had five levels, with each of ten subjects being measured under each level, making the study a simple one-factor repeated measures design.

2.1 Data Import

Many of the methods for repeated measures analysis require the "long" format data file. This is available as a .csv file and is imported with the following code. In a separate document, the conversion from wide to long format (or vice versa) is illustrated with the **reshape2** package. There are several ways of doing this convesion in R, but only one is illustrated in that document.

This code chunk also desgnates the "snum" case number variable as a factor since that is required in most of the analyses to follow in later chapters. It is imported as a numeric variable. In addition, the ordering of the "type" levels is changed from the default alphabetical to match prior graphical work with the data set prior analyses and analyses using contrasts.

```
# read data set - long form exported from SPSS
rpt1.df <- read.csv("data/1facrpt_long.csv")
# change the snum variable to a factor variable (was numeric)
rpt1.df$snum <- as.factor(rpt1.df$snum)
# change the order of the factor levels of type to match the
# original order and match our prior SPSS work,
# including setting  up contrasts
rpt1.df$type <- ordered(rpt1.df$type,
                        levels=c("clean","homecage","iguana",
                                 "lizard", "krat"))
# look at a few lines from the data frame
headTail(rpt1.df)
```

```
##      snum     type  DV
## 1       1    clean  24
## 2       1 homecage  15
## 3       1   iguana  41
## 4       1   lizard  30
## ... <NA>     <NA> ...
## 47     10 homecage   7
## 48     10   iguana   4
## 49     10   lizard   7
## 50     10     krat  23
```

2.2 Numerical Exploratory Data Analysis

Using `describeBy` from the **psych** package, we can now examine a few descriptive statistics for the five conditions. Note that only a subset of the statistics produced by `describeBy` are requested, and the resultant table is more nicely formatted using `gt`. The summaries are also split into two tables to control the width of the tables.

```
d1 <- describeBy(rpt1.df$DV, group=rpt1.df$type,
                 type=2, mat=T)[,c(2,4:9)]
gt(d1)
```

group1	n	mean	sd	median	trimmed	mad
clean	10	6.8	7.177124	6.5	5.500	5.1891
homecage	10	6.0	5.676462	5.5	5.625	6.6717
iguana	10	7.3	12.445883	3.0	4.000	4.4478

lizard	10	9.4	8.796464	6.5	8.000	7.4130
krat	10	23.1	18.174769	16.5	22.125	10.3782

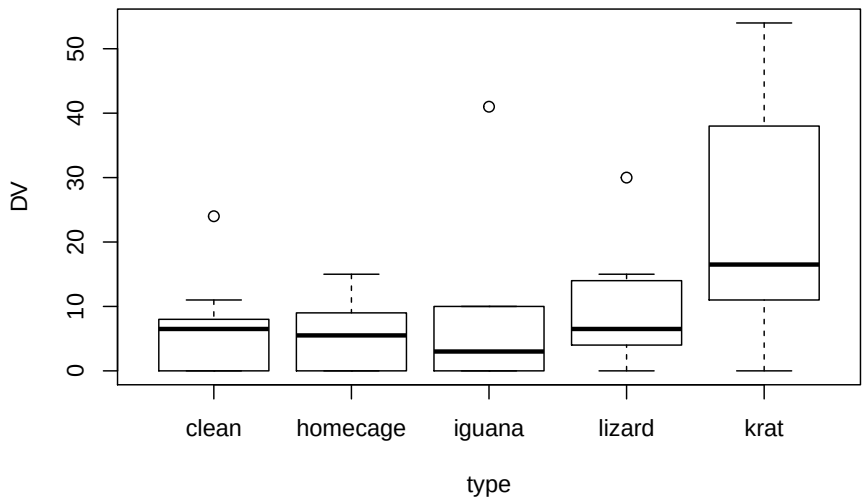
```
d2 <- describeBy(rpt1.df$DV, group=rpt1.df$type,
                 type=2, mat=T)[,c(2,10:15)]
gt(d2)
```

group1	min	max	range	skew	kurtosis	se
clean	0	24	24	1.5779391	3.4183872	2.269606
homecage	0	15	15	0.6150626	-0.7291575	1.795055
iguana	0	41	41	2.6438283	7.4942010	3.935734
lizard	0	30	30	1.4989085	2.7856163	2.781686
krat	0	54	54	0.7791971	-0.6475045	5.747367

2.3 Graphical EDA

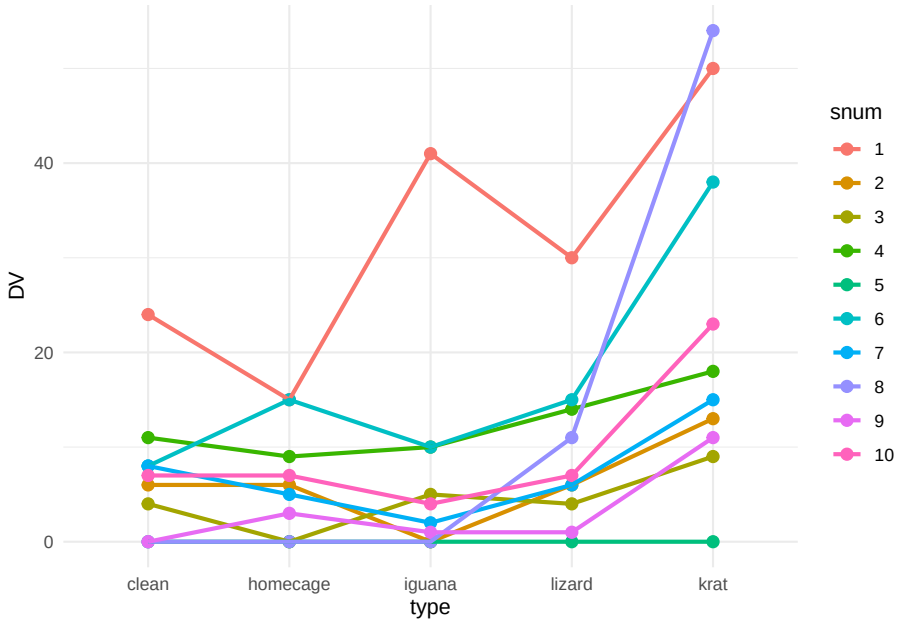
Simple boxplots are readily obtained wtih this long-format data set. The `DV~type` model specification is possble because of this long-format data structure, even though the "type" variable is not variable based on different groups of participants.

```
boxplot(DV~type, data=rpt1.df)
```



One EDA plot that is often used for repeated measures designs is called a profile plot. It is a simple line graph where each case is represented by one line. This type of plot is argued to enable a visual comparision of the patterns across conditions and the assesment of how consistently the cases show the overall pattern. The negative aspect of the plot is that for a study like this where the IV is categorical, the shape of the lines is actually meaningless. If the repeated measure factor were Time, then such a plot would have greater value. Recall that the placement of the categories along the X axis is actually arbitrary here, so the profiles have no intrinsic meaning - they just permit a comparison of cases to one another. Also, apologies for the non-colorblind friendly palette of colors used for the lines…….. (see a later section of this chapter for a method of control of color palettes in `ggplot`).

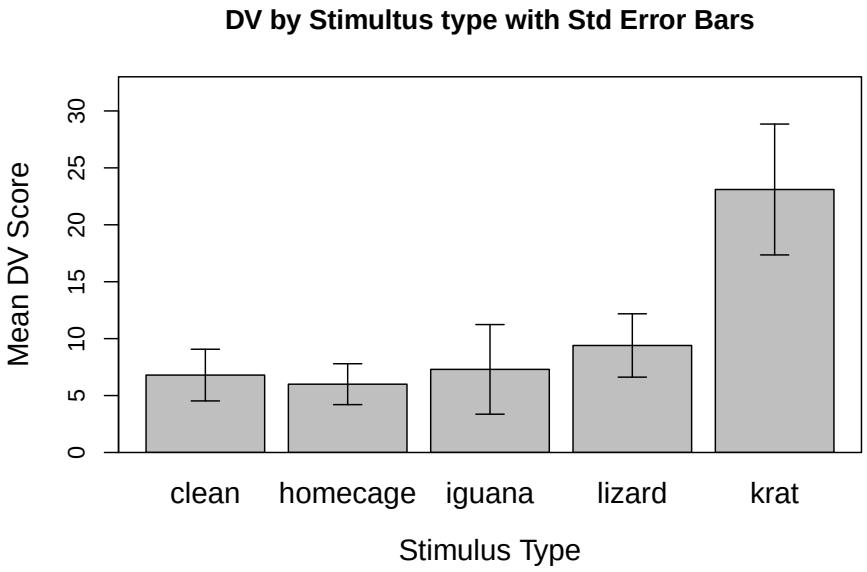
```
library(ggplot2)
ggplot(rpt1.df, aes(type, DV, colour=snum)) +
  geom_point(size = 2.5) +
  geom_line(aes(group = snum), size = 1) +
  theme_minimal()
```



When the repeated factor is categorical rather than quantitative (like time), it is almost expected that the data be summarized graphically with bar graphs plotting the means with error bars added. This plot, also called a dynamite plot, has received considerable criticism as we have discussed. The `bargraph.CI` function can readily draw such a graph, as easily as we saw it done for between-groups designs - this is enabled because of the long-format structure of the data file.

Note that those std errors of the mean are based on between subject variation and have no specific usage in regard to the omnibus inferential tests performed in chapter 3 and later.

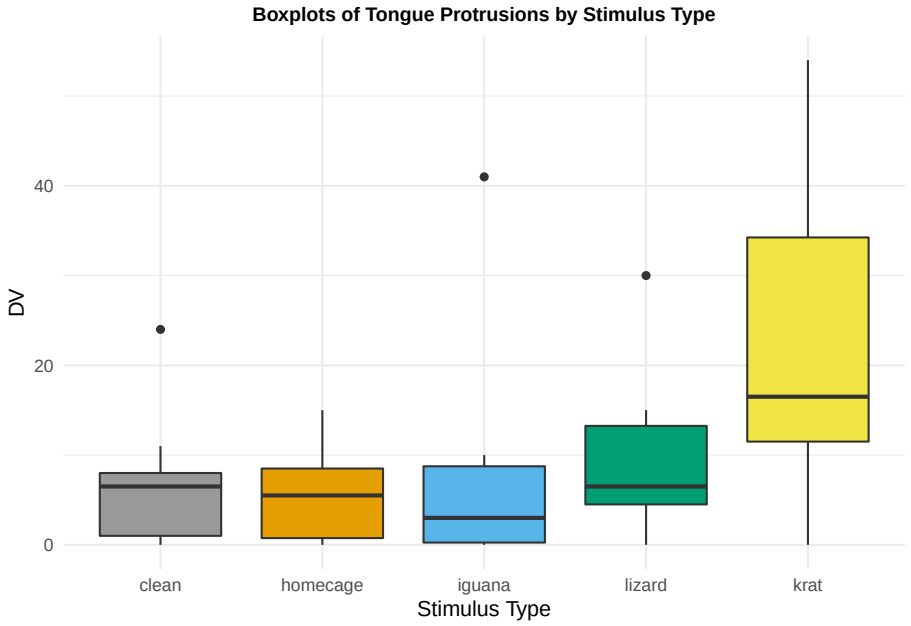
```
#require(sciplot)
bargraph.CI(rpt1.df$type,rpt1.df$DV,lc=TRUE, uc=TRUE,legend=T,
            cex.leg=1,bty="n",col="gray75",
            ylim=c(0,33),
            xlab="Stimulus Type",
            ylab="Mean DV Score",main="DV by Stimultus type with Std Error Bars",
            cex.names=1.25,cex.lab=1.25)
box()
```



Next I show a `ggplot` version of the boxplot to reinforce the idea that the **ggplot2** package is a useful tool. The core of the plot is drawn with the first two lines of the `ggplot` function code, and the remainder of the lines of code control stylistic attributes of the plot. A vector of colorblind-friendly color codes is established first and then used for the "fills". Other commented lines of code show other ways of controlling the color scheme.

```
# first, establish a colorblind friendly palette
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73",
               "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
p2<-ggplot(data = rpt1.df, aes(x = type, y = DV, fill=type)) +
  geom_boxplot() +xlab("Stimulus Type") + ylab("DV") +
  scale_fill_manual(values=cbPalette) +
```

```
  #scale_fill_brewer(palette="Paired") +
  #scale_colour_grey() + scale_fill_grey() +
  ggtitle("Boxplots of Tongue Protrusions by Stimulus Type") +
  guides(fill=FALSE) + # removes legend
  theme_minimal() +
  theme(plot.title = element_text(size=10,
                                  face = "bold", hjust = .5))
p2
```



With the EDA, it becomes clear that the rate of tongue protrusion among the ten rats was fairly consistently low, except for the "krat" condition where it was higher and more dispersed. Inferential methods for evaluating this pattern are found in the next several chapters.

Chapter 3

Traditional Approaches to 1 Factor Repeated Measures Designs

The primary goal of this chapter is the elaboration of the traditional "Univariate" approach to the 1-factor Repeated Measures design, evaluation of it's sphericity assumption, and application of contrast and pairwise condition comparisons. These methods have been the topic of considerable discussion in light of the development of alternatives such as linear mixed effects modeling. Extreme perspectives even argue that the SS partitioning approach of the traditional method is "dead" (McCulloch, 2005). We emphasize that well executed traditional methods, especially those centered on contrast based hypothesis testing, are sound. Later sections delve into alternative methods. A secondary goal of the chapter is provision of code to implement the multivariate approach to evaluation of the omnibus null hypothesis that all condition means are equal. The multivariate approach is not burdened with the sphericity assumption that has brought the univariate approach under strong criticism. The discuusion of which of these to use had been centered around relative power given varying sample sizes and degree of non-sphericity. That conversation has largely been replaced by strong recommendations to use the linear mixed models approach outline in the next chapter. The third goal is to explore methods for examining analytical/orthogonal contrasts and pairwise comparison post hoc tests.

3.1 The traditional "univariate" GLM approach to the repeated measures problem and the Multivariate approach

The sections on the univariate approach emphasize the implementation of the repeated measures design as the theoretical sibling of randomized block designs. It approaches the question in the style of the long litany of experimental design textbooks such as those by Winer, Kirk, Myers, Keppel, and Hays, as well as the current comprehensive textbook by Maxwell, Delaney and Kelley (Howell, 2013; Hays, 1994; Keppel and Wickens, 2004; Kirk, 2013; Myers et al., 2010; Winer et al., 1991). In this variance partitioning approach, three sources of variance are identified:

1. "treatment", the repeated measures factor, called factor A in the standard textbook approach
2. "subject" or case.
3. The Axs interaction term which will serve as the error term for factor A in this approach.

The Axs interaction is usually listed in software ouput from R as the "residual". It's characterization here as an interaction stems from the perspective that a 1 factor repeated measures design is a special case of a randomized block design where the treatment and subject factors are marginal effects and the interaction term is possible because each subject is tested under all treatment conditions. In many repeated measures studies, the repeated factor (IV) is characterized simply as a time factor where the same DV is measured at varying time points. In our first example data set, oulined in the previous chapter, the IV was actually a collection of five treatment conditions and each participant was measured under each condition.

The multivariate evaluation approaches the test of the omnibus null hypothesis in a different way, and it is also illustrated in two of the following sections.

The major challenge in replicating the information set we learned to derive from the SPSS MANOVA or GLM procedure is that there is not simply one function or a small set of functions that can give us everything. That list includes:

1. The omnibus F test of the repeated factor
2. Evalaution of the sphericity assumption
3. GG and HF correction factors and adjusted p values
4. Easy implmentation of orthogonal/analytical contrasts
5. Error terms specific to the contrast
6. Post hoc evaluation of pairwise comparisons among levels
7. Implementation the multivariate appraoach to evaluation of the omnibus null hypothesis

We will piece it together with several different approaches, and will find that finding correct error terms for contrasts (specific error terms that permit avoid-

ance of the sphericity assumption) is not easily accomplished.

3.1.1 Method I: Use of the aov function

Structuring the data set in the "long" format as was done in Chapter 2 permits the standard aov function to partition the variation into the two marginal (main) effects and the interaction term. This requires only a slight change in the code structure compared to the use of aov in "between-subjects" designs. The "Error" argument tells aov that the subject factor is repeated across levels of type. The notation appears to be similar to what we have learned as "nesting". However, subject is not nested within type as the code might imply with the use of the forward slash, even though some in the R world speak of it that way. It is better to read the notation as implying that all levels of type are found under each subject. It strikes me as an unwieldy way of specifying that the Axs interaction term is the Error term to test the main effect of type. But the whole perspective regarding 1-factor repeated measures designs in R is not conceptually aligned with the randomized block idea - I would wager that most users and many programmers are not aware of the analogy.

It is important to understand that use of the aov function this way has a requirement that the "case" variable (snum here) is a factor. We already specified that factor characteristic in the section where the data frame was imported in chapter 2.

```
# perform the 1-factor repeated measures anova
# the notation for specification of the error is not intrinsically
# obvious.  some reading in R and S model specification is required.
# always best to use "sum to zero" contrasts for ANOVA
contrasts(rpt1.df$type) <- contr.sum
fit.1 <- aov(DV~type + Error(snum/type),rpt1.df)
summary(fit.1)
```

```
##
## Error: snum
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  9   3740   415.5
##
## Error: snum:type
##           Df Sum Sq Mean Sq F value   Pr(>F)
## type       4   2042   510.4   8.845 4.42e-05 ***
## Residuals 36   2077    57.7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Checking the df for the terms summarized to doublecheck that the parallel to the two main effects and the interaction term of the randomized blocks perspective is correct. N was ten, so 9 df for the main effect of snum is correct. The "type"

IV had five levels, so we expected the 4 df there. And the Axs interaction term should, and does have 9x4=36 df. The F and p values match our work with SPSS for this same data set.

One way to quickly obtain additional information is with use of the `model.tables` function. I obtained the five condition means and the set of deviation effects for each level of type (deviation effects as would be found using effect coding)

```
#report the means of each level
print(model.tables(fit.1,"means"),digits=3)
```

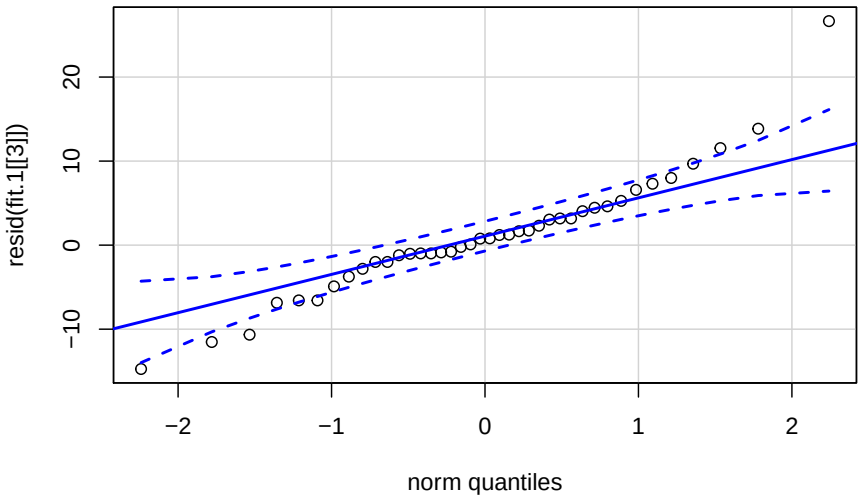
```
## Tables of means
## Grand mean
##
## 10.52
##
##  type
## type
##    clean homecage   iguana   lizard     krat
##      6.8      6.0      7.3      9.4     23.1
```

```
# report deviation effects for each level
print(model.tables(fit.1,"effects", n=TRUE),digits=3)
```

```
## Tables of effects
##
##  type
## type
##    clean homecage   iguana   lizard     krat
##    -3.72    -4.52    -3.22    -1.12    12.58
```

We can evaluate the normality assumption for the residuals with a qqplot. Finding those residuals requires looking at the structure of the fit.1 object and locating the residuals for the Axs interaction. It is in the third section of the list of three components in the fit object (called snum:type here if the 'str(fit.1) function is executed). One major outlier is present and there is a hint of positive skew to the distribution but on balance, most of the data points seem to indicate a reasonable fit to the normality assumption. Note that there are only 40 residuals here, but there are 50 data points. The discrepancy is that the original five variables are transformed to contrast vectors and only four vectors are needed for the five-level factor. Thus four new transformed variables and n=10 yields 40 residuals.

```
#str(fit.1)
qqPlot(resid(fit.1[[3]]),id=FALSE)
```




```
#hist(fit.1[[3]]$residuals)
```

3.1.2 Method II, a multivariate linear model

Method II uses the multivariate linear model. It implements both the multivariate test of the omnibus null hypothesis as well as the averaged F test approach with GG and HF corrections. It also permits the user to obtain the Mauchley Sphericity test and GG/HF corrections.

This approach requires the data frame to be "wide", AND, only contain the variables that are the levels of the repeated factor. No additional variables such as a subject code can exist in the matrix of variables that is created from the data frame.

This appproach is modeled after an article by Dalgaard:

http://www.r-project.org/doc/Rnews/Rnews_2007-2.pdf

This approach requires the "wide" version of the data file, and that file is available as the .csv file loaded here. The functions to be used require that the data be in matrix form, not as a data frame. So, that conversion is also accomplished here, following data import.

```
# read data file
rpt1w.df <- read.table("data/1facrpt_wide.csv",header=T,sep=",")
# change the data from a data frame to a matrix,
# leaving out the snum variable.
```

```
# this method requires that the matrix contains
# only the data values
rpt1w.mat <- as.matrix(rpt1w.df)[,2:6]
# view the data
rpt1w.mat
```

```
##       clean homecage iguana lizard krat
##  [1,]    24       15     41     30   50
##  [2,]     6        6      0      6   13
##  [3,]     4        0      5      4    9
##  [4,]    11        9     10     14   18
##  [5,]     0        0      0      0    0
##  [6,]     8       15     10     15   38
##  [7,]     8        5      2      6   15
##  [8,]     0        0      0     11   54
##  [9,]     0        3      1      1   11
## [10,]     7        7      4      7   23
```

The multivariate linear model is based on use of the standard `lm` function. The model specification argument uses the "~1" syntax to indicate that all variable in the matrix are to be included. Then the `estVar` function can be used to visualize the variance-covariance matrix of the five variables.

```
# first fit the five variate model to get the var/cov matrix
mlmfit.1 <- lm(rpt1w.mat~1)
estVar(mlmfit.1)
```

```
##             clean homecage    iguana    lizard      krat
## clean    51.51111 32.88889  82.40000  55.97778  58.46667
## homecage 32.88889 32.22222  50.88889  39.44444  49.22222
## iguana   82.40000 50.88889 154.90000  99.42222 122.41111
## lizard   55.97778 39.44444  99.42222  77.37778 124.51111
## krat     58.46667 49.22222 122.41111 124.51111 330.32222
```

```
# doublecheck with the `cov` function
#cov(rpt1w.mat)
```

Next, an intercept only model is fit by removing the variates from the full model using the `update` function.

```
# now fit an intercept only model and use the
# anova function in the next section to compare models,
mlmfit.0 <- update(mlmfit.1, ~0)
#estVar(mlmfit.0)
```

A test based on multivariate normal theory evaluates an hypothesis that all variates (repeated measures levels) have equal means is possible with this syntax. A model comparison approach is used by passing the two models created above

to the anova function. The Pillai test statistic, approximate F value, df and p value match what we produced in SPSS. This test does not have the sphericity assumption. The multivariate test is evaluated as the degree of improvement in fiit (or reduction in residuals) in the full model (mlmfit.1) relative to the intercept-only model (mlmfit.0).

```
anova(mlmfit.1, mlmfit.0, X=~1)
```

```
## Analysis of Variance Table
##
## Model 1: rpt1w.mat ~ 1
## Model 2: rpt1w.mat ~ 1 - 1
##
## Contrasts orthogonal to
## ~1
##
##   Res.Df Df Gen.var.  Pillai approx F num Df den Df Pr(>F)
## 1      9      9.6149
## 2     10  1  11.2467 0.64954   2.7801      4      6 0.1269
```

The Mauchly test can be performed on the mlmfit.1 object. The W test statistic and p value match our SPSS MANOVA results.

```
mauchly.test(mlmfit.1, X=~1)
```

```
##
##  Mauchly's test of sphericity
##  Contrasts orthogonal to
##  ~1
##
##
## data:  SSD matrix from lm(formula = rpt1w.mat ~ 1)
## W = 0.0038542, p-value = 7.391e-06
```

The univariate averaged F test can also be produced by adding the "test" argument. This produces the same F value as we derived with the standard ANOVA approach in SPSS MANOVA, the approach that assumes sphericity. Note that the GG and HF epsilons are produced and that the adjustment produces the same p values as we obtained in MANOVA.

```
anova(mlmfit.1, mlmfit.0, X=~1, test="Spherical")
```

```
## Analysis of Variance Table
##
## Model 1: rpt1w.mat ~ 1
## Model 2: rpt1w.mat ~ 1 - 1
##
## Contrasts orthogonal to
## ~1
```

```
##
## Greenhouse-Geisser epsilon: 0.3793
## Huynh-Feldt epsilon:        0.4401
##
##   Res.Df Df Gen.var.      F num Df den Df     Pr(>F)    G-G Pr   H-F Pr
## 1      9      9.6149
## 2     10  1  11.2467 8.8447      4     36 4.4236e-05 0.0055009 0.003391
```

While Method II gives the multivariate test and provides another way to obtain the traditional univariate F test along with GG and HF corrections (as well as the Mauchley test) it has a downside. The model specification formulae are difficult arguments for the relative novice at R programming, especially the ~1 and ~0 types of structures. There is also not a simple way to extract tests of contrasts with this approach, that I have found. Method III uses a similar underlying logic, but is simpler to implement.

3.1.3 Method III, also a Multivariate Linear Model but using `Anova` from car

Method III is similar to Method II, but uses an `Anova` approach in John Fox's car package that permits the Mauchly test and the GG and HF corrections as well as permitting Type III sums of squares (which lm and anova does not. However, Type III SS is not relevant here - since there are no missing data points, the data set is balanced. But this approach can be useful for advanced designs that contain both repeated measures factors and between-groups factors as well as unequal N.

```
# use the same "wide" data frame as from Method II
# change the data from a data frame to a matrix
# and leave out the snum variable
# this method requires that the matrix only contains the data values
rpt1w.mat <- as.matrix(rpt1w.df[,-1])
# view the data
rpt1w.mat
```

```
##       clean homecage iguana lizard krat
##  [1,]    24       15     41     30   50
##  [2,]     6        6      0      6   13
##  [3,]     4        0      5      4    9
##  [4,]    11        9     10     14   18
##  [5,]     0        0      0      0    0
##  [6,]     8       15     10     15   38
##  [7,]     8        5      2      6   15
##  [8,]     0        0      0     11   54
##  [9,]     0        3      1      1   11
## [10,]     7        7      4      7   23
```

```
# start by defining the multivariate linear model
# this code returns the level means.
mlmfit.2 <- lm(rpt1w.mat ~1)
mlmfit.2
```

```
##
## Call:
## lm(formula = rpt1w.mat ~ 1)
##
## Coefficients:
##             clean  homecage  iguana  lizard  krat
## (Intercept)   6.8    6.0       7.3     9.4    23.1
```

The completion of the analysis requires the creation of the repeated measures factor. The data set, in the wide format, is simply a collection of five variables. The "type" variable used in Method I does not exist. So we need to create an object that is that five-level factor. The factor is also "ordered" to keep the levels in the same order as was done above for graphing purposes, but also to have the upcoming contrast analysis match what we did in SPSS MANOVA. The creation of this "stimulus" factor (really the same thing as "type" in the long format data set) is analogous to using the "wsfactors" subcommand in SPSS MANOVA and GLM. My choice of the word "stimulus" is shorthand for "stimulus type" and this reinforces the identity to the "type" variable as used in Method I.

```
# now define a variable that gives the design of the study,
# the levels of the repeated factor, and order them the
# same way that we did in the original SPSS analyses
stimulus <- factor(c("clean", "homecage", "iguana" ,
                     "lizard", "krat"))
stimulus  <- ordered(stimulus,
                     levels=c("clean","homecage","iguana",
                              "lizard", "krat"))
stimulus
```

```
## [1] clean    homecage iguana   lizard   krat
## Levels: clean < homecage < iguana < lizard < krat
```

The implementation of the analysis with this approach focuses on use of the `Anova` function from the **car** package. Two crucial arguments are required to specify the repeated measures nature of the analysis of the design. These are the "idata" and "idesign" arguments. `Anova` uses the multivariate object defined above and then applies the idesign specification. Applying `summary` to that `Anova` object produces the multivariate and univariate tests of the omnibus hypothesis. It also provides the Mauchley test of sphericity, the GG and HF epsilons, and corrected univariate test p-values, all matching the Method II and SPSS results. The terms "idesign", "idata", and "icontrasts" derive from the

phrase "Intra-subject" which implies repeated measures.

When using the idesign argument, `Anova` transforms the levels of the repeated measures factors into orthogonal contrasts. Look carefully at the product of the `summary` function here and see if you can guess what this default set of contrasts is. Hint: it is the default because the most common type of repeated measure factor is a collection of measurements at various time points, and is thus a quantitative IV.

```
#require(car)
mlmfit2.anova <- Anova(mlmfit.2,
                       idata=data.frame(stimulus),
                       idesign=~stimulus, type="III")
# with the summary function, when multivariate=TRUE,
# and univariate=TRUE,
# both multivariate and averaged F tests of the
# repeated effect are printed.
summary(mlmfit2.anova, multivariate=T, univariate=T)
```

```
##
## Type III Repeated Measures MANOVA Tests:
##
## ----------------------------------------
##
## Term: (Intercept)
##
##  Response transformation matrix:
##          (Intercept)
## clean              1
## homecage           1
## iguana             1
## lizard             1
## krat               1
##
## Sum of squares and products for the hypothesis:
##             (Intercept)
## (Intercept)     27667.6
##
## Multivariate Tests: (Intercept)
##                   Df test stat approx F num Df den Df    Pr(>F)
## Pillai             1 0.5967217  13.3171      1      9 0.0053237 **
## Wilks              1 0.4032783  13.3171      1      9 0.0053237 **
## Hotelling-Lawley   1 1.4796774  13.3171      1      9 0.0053237 **
## Roy                1 1.4796774  13.3171      1      9 0.0053237 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## ----------------------------------------
##
## Term: stimulus
##
##  Response transformation matrix:
##          stimulus.L stimulus.Q    stimulus.C stimulus^4
## clean    -0.6324555  0.5345225 -3.162278e-01  0.1195229
## homecage -0.3162278 -0.2672612  6.324555e-01 -0.4780914
## iguana    0.0000000 -0.5345225 -4.095972e-16  0.7171372
## lizard    0.3162278 -0.2672612 -6.324555e-01 -0.4780914
## krat      0.6324555  0.5345225  3.162278e-01  0.1195229
##
## Sum of squares and products for the hypothesis:
##            stimulus.L stimulus.Q stimulus.C stimulus^4
## stimulus.L  1296.0000   906.6815  342.00000  164.64132
## stimulus.Q   906.6815   634.3143  239.26317  115.18306
## stimulus.C   342.0000   239.2632   90.25000   43.44702
## stimulus^4   164.6413   115.1831   43.44702   20.91571
##
## Multivariate Tests: stimulus
##                  Df test stat approx F num Df den Df  Pr(>F)
## Pillai            1 0.6495405 2.780095      4      6 0.12692
## Wilks             1 0.3504595 2.780095      4      6 0.12692
## Hotelling-Lawley  1 1.8533965 2.780095      4      6 0.12692
## Roy               1 1.8533965 2.780095      4      6 0.12692
##
## Univariate Type III Repeated-Measures ANOVA Assuming Sphericity
##
##             Sum Sq num Df Error SS den Df F value    Pr(>F)
## (Intercept) 5533.5      1   3739.7      9 13.3171  0.005324 **
## stimulus    2041.5      4   2077.3     36  8.8447 4.424e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Mauchly Tests for Sphericity
##
##          Test statistic    p-value
## stimulus      0.0038542 7.3907e-06
##
##
## Greenhouse-Geisser and Huynh-Feldt Corrections
##  for Departure from Sphericity
##
##          GG eps Pr(>F[GG])
## stimulus 0.3793   0.005501 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##             HF eps  Pr(>F[HF])
## stimulus 0.4400688 0.003391046
```

3.1.4 Partitioning the omnibus effect into contrasts

Since orthogonal polynomial trend analysis, as done in the intial Method III analysis, is not appropriate for our "stimulus/type" factor, next we can add in the ability to specify our own contrasts. Then we will rerun the `Anova` function from car, using these contrasts.

```
contrasts.stimulus <- matrix(c(4, -1, -1, -1, -1,
                               0,  3, -1, -1, -1,
                               0,  0, -1, -1,  2,
                               0,  0, -1,  1,  0),
                             ncol=4)
contrasts(stimulus) <- contrasts.stimulus
contrasts(stimulus)
```

```
##          [,1] [,2] [,3] [,4]
## clean       4    0    0    0
## homecage   -1    3    0    0
## iguana     -1   -1   -1   -1
## lizard     -1   -1   -1    1
## krat       -1   -1    2    0
```

```
icontr <- contrasts(stimulus)
```

The new "icontr" object is specified with one additional argument in `Anova`, called "icontrasts".

Also notice:

1. Hypothesis SS and for each contrast are produced, but only found in the SSP matrix and t/F tests not performed. We will have to compute the F's manually after computing the respective MS.
2. The error SS are computed for the contrasts but not printed by use of the `summary` function. We will have to manually extract them.
3. The hypothesis SS and error SS are not corrected for the size of the coefficients chosen. I.e., the coefficients are not orthonormalized in this illustration, so the SS won't match your SPSS-generated SS unless you divide each by the respective sum of the squared contrast coefficients.
4. Despite using the contrasts that we specified, `Anova` does not produce tests of those individual contrasts with the `summary` function. I have not found

a direct way to obtain those tests. In the next sections, I show ways to obtain them more "manually"

```
mlmfit3.anova <- Anova(mlmfit.2,
                       idata=data.frame(stimulus),
                       idesign=~stimulus,
                       icontrasts=icontr,type="III")
#str(mlmfit3.anova)
summary(mlmfit3.anova)
```

```
##
## Type III Repeated Measures MANOVA Tests:
##
## ----------------------------------------
##
## Term: (Intercept)
##
##  Response transformation matrix:
##          (Intercept)
## clean              1
## homecage           1
## iguana             1
## lizard             1
## krat               1
##
## Sum of squares and products for the hypothesis:
##             (Intercept)
## (Intercept)     27667.6
##
## Multivariate Tests: (Intercept)
##                  Df test stat approx F num Df den Df    Pr(>F)
## Pillai            1 0.5967217  13.3171      1      9 0.0053237 **
## Wilks             1 0.4032783  13.3171      1      9 0.0053237 **
## Hotelling-Lawley  1 1.4796774  13.3171      1      9 0.0053237 **
## Roy               1 1.4796774  13.3171      1      9 0.0053237 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ----------------------------------------
##
## Term: stimulus
##
##  Response transformation matrix:
##          stimulus1 stimulus2 stimulus3 stimulus4
## clean            4         0         0         0
## homecage        -1         3         0         0
```

```
## iguana          -1        -1        -1        -1
## lizard          -1        -1        -1         1
## krat            -1        -1         2         0
##
## Sum of squares and products for the hypothesis:
##           stimulus1 stimulus2 stimulus3 stimulus4
## stimulus1    3459.6    4054.8   -5487.0    -390.6
## stimulus2    4054.8    4752.4   -6431.0    -457.8
## stimulus3   -5487.0   -6431.0    8702.5     619.5
## stimulus4    -390.6    -457.8     619.5      44.1
##
## Multivariate Tests: stimulus
##                  Df test stat approx F num Df den Df  Pr(>F)
## Pillai            1 0.6495405 2.780095      4      6 0.12692
## Wilks             1 0.3504595 2.780095      4      6 0.12692
## Hotelling-Lawley  1 1.8533965 2.780095      4      6 0.12692
## Roy               1 1.8533965 2.780095      4      6 0.12692
##
## Univariate Type III Repeated-Measures ANOVA Assuming Sphericity
##
##             Sum Sq num Df Error SS den Df F value    Pr(>F)
## (Intercept) 5533.5      1   3739.7      9 13.3171  0.005324 **
## stimulus    2041.5      4   2077.3     36  8.8447 4.424e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Mauchly Tests for Sphericity
##
##          Test statistic    p-value
## stimulus      0.0038542 7.3907e-06
##
##
## Greenhouse-Geisser and Huynh-Feldt Corrections
##  for Departure from Sphericity
##
##          GG eps Pr(>F[GG])
## stimulus 0.3793   0.005501 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##             HF eps  Pr(>F[HF])
## stimulus 0.4400688 0.003391046
```

The omnibus univariate and multivariate tests are identical the the first illustration above. Unfortunately, there does not appear to be a way of requesting

that `Anova` test these newly created contrasts.

The output just above did give the SS for each of the contrasts, seen on the leading diagonal of the SSP (hypothesis matrix). The `Anova` function did create the error SS as well, but they were not printed with the results. In order to find the error SS for each contrast (the Acontrast x s) terms for specific error terms, we need to examine the structure of the mlmfit3.anova object. It is called the SSPE matrix. Note that in addition to the SShypothesis and SSerror values, we can look for the error df as well (it is called error.df)

```
str(mlmfit3.anova)
```

```
## List of 14
##  $ SSP       :List of 2
##   ..$ (Intercept): num [1, 1] 27668
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr "(Intercept)"
##   .. .. ..$ : chr "(Intercept)"
##   ..$ stimulus   : num [1:4, 1:4] 3460 4055 -5487 -391 4055 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:4] "stimulus1" "stimulus2" "stimulus3" "stimulus4"
##   .. .. ..$ : chr [1:4] "stimulus1" "stimulus2" "stimulus3" "stimulus4"
##  $ SSPE      :List of 2
##   ..$ (Intercept): num [1, 1] 18698
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr "(Intercept)"
##   .. .. ..$ : chr "(Intercept)"
##   ..$ stimulus   : num [1:4, 1:4] 4976 4381 -5133 -169 4381 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:4] "stimulus1" "stimulus2" "stimulus3" "stimulus4"
##   .. .. ..$ : chr [1:4] "stimulus1" "stimulus2" "stimulus3" "stimulus4"
##  $ P         :List of 2
##   ..$ (Intercept): num [1:5, 1] 1 1 1 1 1
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:5] "clean" "homecage" "iguana" "lizard" ...
##   .. .. ..$ : chr "(Intercept)"
##   ..$ stimulus   : num [1:5, 1:4] 4 -1 -1 -1 -1 0 3 -1 -1 -1 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:5] "clean" "homecage" "iguana" "lizard" ...
##   .. .. ..$ : chr [1:4] "stimulus1" "stimulus2" "stimulus3" "stimulus4"
##  $ df        : Named num [1:2] 1 1
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "stimulus"
##  $ error.df  : int 9
##  $ terms     : chr [1:2] "(Intercept)" "stimulus"
##  $ repeated  : logi TRUE
##  $ type      : chr "III"
##  $ test      : chr "Pillai"
```

```
##  $ idata     :'data.frame':  5 obs. of  1 variable:
##   ..$ stimulus: Ord.factor w/ 5 levels "clean"<"homecage"<..: 1 2 3 4 5
##   .. ..- attr(*, "contrasts")= num [1:5, 1:4] 4 -1 -1 -1 -1 0 3 -1 -1 -1 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:5] "clean" "homecage" "iguana" "lizard" ...
##   .. .. .. ..$ : NULL
##  $ idesign   :Class 'formula'  language ~stimulus
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##  $ icontrasts: num [1:5, 1:4] 4 -1 -1 -1 -1 0 3 -1 -1 -1 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:5] "clean" "homecage" "iguana" "lizard" ...
##   .. ..$ : NULL
##  $ imatrix   : NULL
##  $ singular  : logi FALSE
##  - attr(*, "class")= chr "Anova.mlm"
```

This SSP matrix duplicates the matrix that was seen in the `summary` output and the SS for the contrasts are on the leading diagonal.

```
# view the contrast hypothesis SS_CP matrix
mlmfit3.anova$SSP$stimulus
```

```
##           stimulus1 stimulus2 stimulus3 stimulus4
## stimulus1    3459.6    4054.8   -5487.0    -390.6
## stimulus2    4054.8    4752.4   -6431.0    -457.8
## stimulus3   -5487.0   -6431.0    8702.5     619.5
## stimulus4    -390.6    -457.8     619.5      44.1
```

Similarly, the SS for the specific errors are found on the leading diagonal of this SSPE matrix.

```
# view the contrast error SS_CP matrix
mlmfit3.anova$SSPE$stimulus
```

```
##           stimulus1 stimulus2 stimulus3 stimulus4
## stimulus1    4976.4    4381.2   -5133.0    -169.4
## stimulus2    4381.2    6371.6   -4069.0     369.8
## stimulus3   -5133.0   -4069.0    6882.5     735.5
## stimulus4    -169.4     369.8     735.5     300.9
```

In order to do the F tests, we need to:

1. Extract the relevant SS from the leading diagonals of the matrices examined above.
2. Create the MS error for each contrast by dividing by the correct df for the Axs term (1x(s-1))
3. Realize that each contrast is a 1df source, so each hypothesis SS is already a MS
4. Divide the MSeffect values by the MSerror values to produce the F values

5. find p values for each F

It made the naming easier to "attach" the mlmfit3.anova object. First, obtain the SSeffect values (equivalent to the MSeffect values since they are 1 df terms:

```
attach(mlmfit3.anova)
#show effect/cov values - the SS of the contrasts are on the diagonal
#SSP$stimulus
effect <- diag(SSP$stimulus)
#note that the diagonal from SSP matrix is already the MS since df = 1
effect
```

```
## stimulus1 stimulus2 stimulus3 stimulus4
##    3459.6    4752.4    8702.5      44.1
```

Then obtain the MSerror values:

```
#show error/cov values - the specific error SS are on the diagonal
#SSPE$stimulus
x1 <- diag(SSPE$stimulus)
x1
```

```
## stimulus1 stimulus2 stimulus3 stimulus4
##    4976.4    6371.6    6882.5     300.9
```

```
error <- x1/error.df
```

Now compute the F values and obtain p values for each contrast. Note that they match our SPSS work.

```
# now compute F's; F values are on the diagonal
Fvalues <- effect/error
Fvalues
```

```
## stimulus1 stimulus2 stimulus3 stimulus4
##  6.256812  6.712851 11.379949  1.319043
```

```
# obtain the p values for the four F tests (all have 1,9 df)
pf(Fvalues,1,9,lower.tail=F)
```

```
##   stimulus1   stimulus2   stimulus3   stimulus4
## 0.033786236 0.029169567 0.008212304 0.280372227
```

There is a more efficient way of doing this by writing a function and then passing the `Anova` object to it. F values and p values are returned.

```
model <- mlmfit3.anova
rptcontr <- function(model){
  SSCPeffect <- as.matrix(model$SSP$stimulus,ncol=4)
  effect <- diag(SSCPeffect)
  #effect
  #return(effect)
```

```
  SSCPerror <- as.matrix(model$SSPE$stimulus,ncol=4)
  error <- diag(SSCPerror)
  dferror <- mlmfit3.anova$error.df
  mserror <- error/dferror
  Fval <- round(effect/mserror,3)
  pval <- round(pf(Fval, 1,dferror, lower.tail=F),4)
  vals <- as.data.frame(cbind(Fval, pval, dferror))
  vals
  return(gt(vals,rownames_to_stub=T))
}
rptcontr(model=mlmfit3.anova)
```

	Fval	pval	dferror
stimulus1	6.257	0.0338	9
stimulus2	6.713	0.0292	9
stimulus3	11.380	0.0082	9
stimulus4	1.319	0.2804	9

3.1.5 Recall how to "manually" implement contrasts for repeated factors.

The method for obtaining contrasts just reviewed is very laborious and does not easily extend to multiple factor repeated measures designs. At this point, it would be useful to recall another class demonstration of how to conceptualize contrasts for repeated measures.

Since each case provides a data point for each level of the repeated measure factor, we can create contrast vectors manually as new variables - essentially transformations. I have used the `mutate` function from **dplyr** to do this. For a review of `mutate`, see the separate document on subsetting and variable creation.

```
rpt1w.df <- mutate(rpt1w.df,
            ac1=(4*clean+(-1)*homecage+(-1)*iguana+(-1)*lizard+(-1)*krat))
rpt1w.df <- mutate(rpt1w.df,
            ac2=(0*clean+(3)*homecage+(-1)*iguana+(-1)*lizard+(-1)*krat))
rpt1w.df <- mutate(rpt1w.df,
            ac3=(0*clean+(0)*homecage+(-1)*iguana+(-1)*lizard+(2)*krat))
rpt1w.df <- mutate(rpt1w.df,
            ac4=(0*clean +(0)*homecage+(-1)*iguana+(1)*lizard+(0)*krat))
gt(rpt1w.df)
```

snum	clean	homecage	iguana	lizard	krat	ac1	ac2	ac3	ac4
1	24	15	41	30	50	-40	-76	29	-11
2	6	6	0	6	13	-1	-1	20	6

3	4	0	5	4	9	-2	-18	9	-1
4	11	9	10	14	18	-7	-15	12	4
5	0	0	0	0	0	0	0	0	0
6	8	15	10	15	38	-46	-18	51	5
7	8	5	2	6	15	4	-8	22	4
8	0	0	0	11	54	-65	-65	97	11
9	0	3	1	1	11	-16	-4	20	0
10	7	7	4	7	23	-13	-13	35	3

Once created, we can test a hypothesis that each contrast has a value of zero (the null). Each contrast is now a single vector of ten values, and the sd of each vector is the square root of the specific error term MS values calculated above. So, when we perform a one sample t-test with a null value of zero, the t's that are produced are the square roots of the F values computed above. One might compare the mean of the first contrast vector to the value we examined when we manually did this exercise in spreadsheet form at the point in time that the repeated measures theory was first presented in class.

```
t.test(rpt1w.df$ac1)
```

```
##
##  One Sample t-test
##
## data:  rpt1w.df$ac1
## t = -2.5014, df = 9, p-value = 0.03379
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -35.421285  -1.778715
## sample estimates:
## mean of x
##     -18.6
```

```
t.test(rpt1w.df$ac2)
```

```
##
##  One Sample t-test
##
## data:  rpt1w.df$ac2
## t = -2.5909, df = 9, p-value = 0.02917
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -40.833812  -2.766188
## sample estimates:
## mean of x
##     -21.8
```

```
t.test(rpt1w.df$ac3)
```

```
##
##  One Sample t-test
##
## data:  rpt1w.df$ac3
## t = 3.3734, df = 9, p-value = 0.008212
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##   9.717798 49.282202
## sample estimates:
## mean of x
##      29.5
```

```
t.test(rpt1w.df$ac4)
```

```
##
##  One Sample t-test
##
## data:  rpt1w.df$ac4
## t = 1.1485, df = 9, p-value = 0.2804
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -2.036306  6.236306
## sample estimates:
## mean of x
##       2.1
```

This manual approach works well, is fairly efficient, and is important because of its use of the error terms specific to the contrast rather than the omnibus Axs error term which is burdened with the sphericity assumption. There is a literature that argues for the use of specific error terms to avoid this non-sphericity problem. Using the omnibus Axs error term may inflate type I error rates more for contrasts than for the omnibus F test in repeated measures designs when the sphericity assumption is violated (see section in the tookit bibliography, in particular a paper by Boik (Boik, 1981).

3.1.6 Method IV: Implement the univariate analysis with `ezanova` from the ez package

The `ezanova` function is a wrapper for the `aov` function and may be simpler to use than the `aov` function for some designs. It has the added value of performing the Mauchly sphericity test and providing GG and HF epsilons and corrected p values as well as an effect size indicator. The structure of the code requires an argument for the data frame (needs the long format data frame), the dependent variable, the ID factor for the case variable (subject, or snum in our data set), and the repeated measures factor (the within argument).

This might be the simplest of all approaches to obtain the omnibus F test plus the sphericity test and associated corrections. The main arguments that create the repeated measures analysis are "dv" which is the dependent variable name, "wid" which is the subject variable ID code (and the period is a odd but necessary part of the code) and "within" which is the repeated measure IV.

```
#library(ez)
fit1.ez <- ezANOVA(data=rpt1.df, detailed=T, return_aov=T,
                   dv=DV,
                   wid=.(snum),
                   within=.(type))
fit1.ez
```

```
## $ANOVA
##        Effect DFn DFd     SSn     SSd         F            p p<.05       ges
## 1 (Intercept)   1   9 5533.52 3739.68 13.317097 5.323652e-03     * 0.4875125
## 2        type   4  36 2041.48 2077.32  8.844723 4.423582e-05     * 0.2597805
##
## $`Mauchly's Test for Sphericity`
##   Effect           W            p p<.05
## 2   type 0.003854226 7.390741e-06     *
##
## $`Sphericity Corrections`
##   Effect       GGe       p[GG] p[GG]<.05       HFe       p[HF] p[HF]<.05
## 2   type 0.3792955 0.005500897         * 0.4400688 0.003391046         *
##
## $aov
##
## Call:
## aov(formula = formula(aov_formula), data = data)
##
## Grand Mean: 10.52
##
## Stratum 1: snum
##
## Terms:
##                 Residuals
## Sum of Squares    3739.68
## Deg. of Freedom         9
##
## Residual standard error: 20.38431
##
## Stratum 2: snum:type
##
## Terms:
##                     type Residuals
```

```
## Sum of Squares  2041.48   2077.32
## Deg. of Freedom       4        36
##
## Residual standard error: 7.596271
## Estimated effects may be unbalanced
```

3.1.7 Method V: Implement the univariate analysis with the afex package

The **afex** package permits specification of repeated measures models using styles from **ez**, `aov`, `lm`, `Anova` and other modeling functions. Here, the code reflects the basic `aov` syntax for the model specification and uses `Anova` from **car** for calculations. The correction argument in the anova_table list permits "none", "GG" or "HF" for sphericity corrections. Although the `aov_car` function does not produce a print out of contrast tests, their inclusion sets up the usage later.

```
# use rpt1.df, the long data frame
# make sure "sum to zero" contrasts are specified
# using attach makes the naming simple in `aov_car`
attach(rpt1.df)
```

```
## The following object is masked from mlmfit3.anova:
##
##     type
```

```
contrasts.type <- matrix(c(4, -1, -1, -1, -1,
                           0,  3, -1, -1, -1,
                           0,  0, -1, -1,  2,
                           0,  0, -1,  1,  0),ncol=4)
contrasts(rpt1.df$type) <- contrasts.type
contrasts(rpt1.df$type)
```

```
##          [,1] [,2] [,3] [,4]
## clean       4    0    0    0
## homecage   -1    3    0    0
## iguana     -1   -1   -1   -1
## lizard     -1   -1   -1    1
## krat       -1   -1    2    0
```

```
fit1.afex <- aov_car(DV~type + Error(snum/type), data=rpt1.df,
                     anova_table = list(correction = "HF"))
#fit1.afex
#str(fit1.afex)
kable(nice(fit1.afex))
```

Effect	df	MSE	F	ges	p.value
type	1.76, 15.84	131.12	8.84 **	.260	.003

3.1.8 Working with emmeans for contrasts and pairwise comparison follow ups

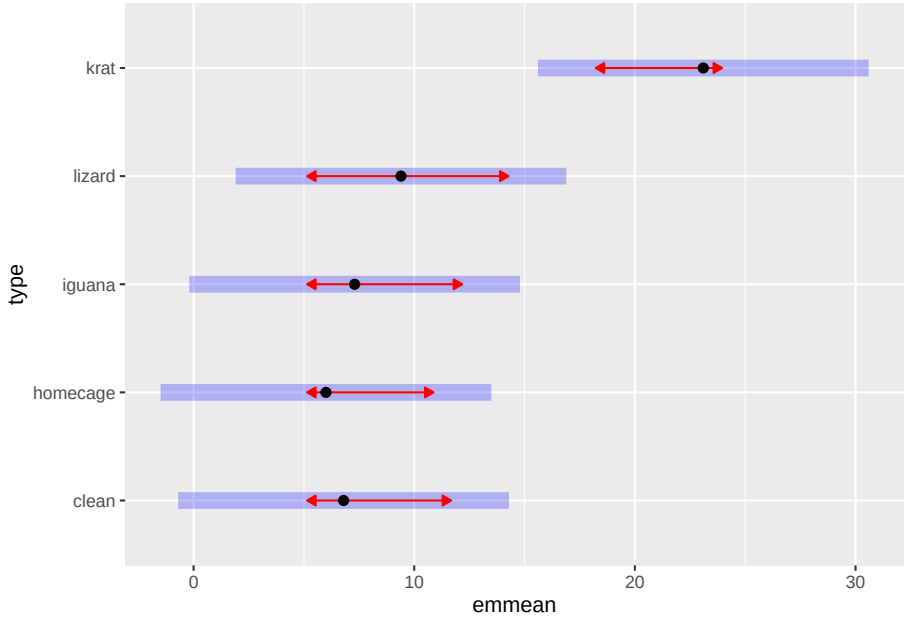
The **emmeans** package provides strong facilities for performing follow up analyses such as pairwise multiple comparisons and contrasts. This section briefly demonstrates both of those. Unfortunately, it does not appear that the tests of contrasts in this repeated measures design employ specific error terms, so the approach cannot be recommended when non-sphericity is suspected.

The `emmeans` function can utilize anova objects from a variety of sources. It works well with objects from `afex` so that is the approach taken here. The first line of code here simply extracts the information from the afex object to perform pairwise tests of the means of the five conditions. Since this would likely be a post hoc approach, the "adjust" argument in the "pairs" function permits access to the large family of correction types for error rate inflation. I illustrated the use with the Tukey method, but holm, by, bonf, fdr, and others are available. Unfortunately, the df for the error in each of these tests is listed as 36 and the SE is the same for all. This implies that the same error is used for all of the pairwise tests and that error is based on the potentially flawed MSaxs error term.

```
fit1afex.emm <- emmeans(fit1.afex, "type", data=rpt1.df)
pairs(fit1afex.emm, adjust="tukey")
```

```
##  contrast          estimate  SE df t.ratio p.value
##  clean - homecage       0.8 3.4 36  0.235  0.9993
##  clean - iguana        -0.5 3.4 36 -0.147  0.9999
##  clean - lizard        -2.6 3.4 36 -0.765  0.9389
##  clean - krat         -16.3 3.4 36 -4.798  0.0003
##  homecage - iguana     -1.3 3.4 36 -0.383  0.9952
##  homecage - lizard     -3.4 3.4 36 -1.001  0.8532
##  homecage - krat      -17.1 3.4 36 -5.034  0.0001
##  iguana - lizard       -2.1 3.4 36 -0.618  0.9712
##  iguana - krat        -15.8 3.4 36 -4.651  0.0004
##  lizard - krat        -13.7 3.4 36 -4.033  0.0024
##
## P value adjustment: tukey method for comparing a family of 5 estimates
```

```
#pairs(fit1afex.emm, adjust="none")
plot(fit1afex.emm, comparisons = TRUE)
```



The implementation of a pairwise comparison method that uses a specific error term, in this 1 factor repeated measure design, could easily be done as a dependent samples "t-test". For example, the comparision of the iguana and lizard conditions is illustrated. Note that the mean difference is the same as the pairwise comparison just above from `emmeans`, but the t value is different because the error is limited to the error from just the two conditions being compared in the dependent samples t-test here. This t value, and the p value, do match the values from the "manual" approach taken above for contrast4 which is the same hypothesis test (comparing iguana and lizard conditions). By using the dependent samples t-test for all possible pairs in a post hoc type of approach, the p values could be submitted to the `p.adjust` function for error rate inflation correction.

```
t.test(rpt1w.df$iguana, rpt1w.df$lizard, paired=TRUE)
```

```
##
##  Paired t-test
##
## data:  rpt1w.df$iguana and rpt1w.df$lizard
## t = -1.1485, df = 9, p-value = 0.2804
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -6.236306  2.036306
## sample estimates:
## mean of the differences
##                    -2.1
```

Linear Combinations (contrasts) can also be evaluated in the **emmeans** approach using the `contrast` function to produce the matrix of coefficients, and the `test` function to test them. Once again though, the omnibus rather than specific error term is employed in all the tests.

```
lincombs <- contrast(fit1afex.emm,
               list(type1=c(4, -1,-1,-1,-1),
                    type2=c(0,  3, -1, -1, -1),
                    type3=c(0,  0, -1, -1,  2),
                    type4=c(0,  0, -1,  1,  0)
                    ))
test(lincombs, adjust="none")
```

```
##  contrast estimate    SE df t.ratio p.value
##  type1       -18.6 10.74 36 -1.731  0.0919
##  type2       -21.8  8.32 36 -2.620  0.0128
##  type3        29.5  5.88 36  5.014  <.0001
##  type4         2.1  3.40 36  0.618  0.5404
```

A plot and table of confidence intervals for the contrasts can also be obtained, even employing p value adjustments as illustrated here with the bonferroni-sidak adjustment. Once again, the omnibus MSaxs error term is used rather than specific error terms.

```
confint(lincombs, adjust="sidak")
```

```
##  contrast estimate    SE df lower.CL upper.CL
##  type1       -18.6 10.74 36   -46.76   9.5639
##  type2       -21.8  8.32 36   -43.62   0.0156
##  type3        29.5  5.88 36    14.07  44.9260
##  type4         2.1  3.40 36    -6.81  11.0062
##
## Confidence level used: 0.95
## Conf-level adjustment: sidak method for 4 estimates
```

3.2 Commentary on the Univariate/Multivariate methods

The user has two kinds of decisions to make. The first is whether the multivariate GLM approach is desired, or the univariate one (the averaged F test). Method II and Method III produce both types of test and the choice depends, in part on examination of the sphericity assumption. The real question is whether to use the multivariate test or the GG/HF corrected univariate approach. Typically researchers have not frequently used the multivariate approach because of its relative low power for the typical small sample sizes of many repeated measures experiments. A literature on the relative merits of the two approaches can be

found in the stat toolkit bibliography.

Of all the methods, the simplest to implement might be method IV, using `ezanova`. This provides the sphericity test and GG/HF corrections to the univariate p value for the omnibus test. If the user does not prefer the corrected univariate test in the face non-sphericity, then Method II or Method III can be employed once the sphericity assumption is examined.

Even if the first decision is made regarding univariate vs multivariate tests, the user must still decide how to proceed in following up these omnibus tests with pairwise comparisons or analytical/orthogonal contrasts. Unfortunately, all of the built-in methods I have found employ the omnibus error term in creating standard errors or F tests for such comparisions. The only ways to obtain specific error terms are with the manual approaches outlined above. The manual approaches are not terribly burdensome for a simple 1 factor repeated measures design, but will become very much more challenging for larger designs - e.g., two repeated factors or mixed designs with grouping factors and repeated measures factors.

There may be another way of approaching contrasts using the Linear Mixed Models methodology seen in the next chapter.

The conclusion I still hold is that the SPSS MANOVA procedure is effecient and helpful in pursuing these followup contrast types of questions. Its default setting of specific error terms is of considerable value in the MANOVA procedure. It is not clear to me why this has not found its way into the R ecosystem such that it is simple to implement. (perhaps there is still something I havn't yet learned about the suite of R functions employed here, especially perhaps `emmeans`).

Chapter 4

Linear Mixed Models

As an alternative to the traditional methods found in Chapter 3, this chapter briefly introduces Linear Mixed Effects Modeling. Although at this point in the course we have not covered any of the theory of LMM, we can examine the basics of implementation for this simple one-factor repeated measures design. LMM is a class of techniques that handle nested/hierarchical designs, and longitudinal growth curve modeling of which repeated measures designs can be seen as a subset. This suite of methods is designed to handle random factors such as "subjects" in a repeated measures design. The major advantages of LMM approaches to repeated measures are:

- Can handle missing data. The traditional approaches usually employ casewise deletion if any data points are missing for a case.
- Can handle repeated factors such as time where not all participants are measured at exactly the same time points or that vary in total number of time points measured.
- Can specify alternative covariance strutures to the compound symmetry and sphericity patterns that the traditional methods assume.

Of these, the only one that is important for the one-factor design that is illustrated here is the last point.

The illustrations here use two different packages, **nlme** and **lme4**. Detailed training in these methods is advised before routine usage.

4.1 Basic LMM Analysis using `lme`

Here, I illustrate the `lme` function from **nlme**. It requires the same long-format data frame as the `aov` function we used for the traditional analysis. That data

set is imported again here, along with the changes of "snum" to a factor and reordering the "type" variable levels.

```
# need "long" form of the data set as in Method I of chapter 3
rpt1.df <- read.csv("data/1facrpt_long.csv")
# change the snum variable to a factor variable (was numeric)
rpt1.df$snum <- as.factor(rpt1.df$snum)
rpt1.df$type <- ordered(rpt1.df$type,
                        levels=c("clean","homecage","iguana",
                                 "lizard", "krat"))
# look at beginning and ending few lines of the  data frame
gt(headTail(rpt1.df))
```

snum	type	DV
1	clean	24
1	homecage	15
1	iguana	41
1	lizard	30
NA	NA	...
10	homecage	7
10	iguana	4
10	lizard	7
10	krat	23

The syntax of the `lme` function is somewhat similar to the `aov`/`lm` syntax. The most important specification is the way that the random factor is specified. The ~1|snum/type specification is not intuitively obvious. Understanding it requires a background in the theory of LMM and a careful reading of the R help page on the function, neither of which will be reviewed here.

Initially, we fit a model that contains all of the specifics required for the overall/omnibus model evaluation. Note that we examine the model with the standard `anova` and `summary` functions that we have become familiar with in other ANOVA/regression analyses.

In interpreting this initial output, several things become apparent:

1. The `anova` output yields the same F and p value as the traditional Univariate approach. This is because the traditional GLM approach can be viewed as a subset of types of LMM analyses and the default covariance structure is the one of compound symmetry, and that would be a problematic assumption here too.
2. `summary` produces information criteria (AIC/BIC) as well as info on the overall intercept (related to the overall mean) and a term that indicates that the intercept is permitted to vary across subjects (related to the Axs interaction).

3. Information on coefficients reflects the fact that individual contrast vectors were created and by default, they are those of trend analysis. This is because the defaults presume that the repeated measure is a time-related factor and that trend would thus be desireable. Since this is not the case for our illustration the succeeding code chunks changes the contrasts and we re-run the analysis.

```
#require(nlme)
# first, default, LMM model assumes sphericity and reproduces  the
# omnibus F test produced as the "average F" in the SPSS MANOVA approach
fit1.lme <- lme(DV ~ type, random = ~1|snum/type, data=rpt1.df)
anova(fit1.lme)
```

```
##             numDF denDF   F-value p-value
## (Intercept)     1    36 13.317097   8e-04
## type            4    36  8.844723  <.0001
```

```
summary(fit1.lme)
```

```
## Linear mixed-effects model fit by REML
##  Data: rpt1.df
##        AIC      BIC   logLik
##   357.0839 371.5372 -170.542
##
## Random effects:
##  Formula: ~1 | snum
##         (Intercept)
## StdDev:    8.459511
##
##  Formula: ~1 | type %in% snum
##         (Intercept) Residual
## StdDev:    6.930132 3.110725
##
## Fixed effects: DV ~ type
##                 Value Std.Error DF  t-value p-value
## (Intercept) 10.520000  2.882776 36 3.649260  0.0008
## type.L      11.384200  2.402152 36 4.739167  0.0000
## type.Q       7.964385  2.402152 36 3.315521  0.0021
## type.C       3.004164  2.402152 36 1.250614  0.2191
## type^4       1.446227  2.402152 36 0.602055  0.5509
##  Correlation:
##        (Intr) type.L type.Q type.C
## type.L 0
## type.Q 0      0
## type.C 0      0      0
## type^4 0      0      0      0
##
```

```
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -0.756930607 -0.157209242 -0.003407451  0.137647579  1.550657810
##
## Number of Observations: 50
## Number of Groups:
##           snum type %in% snum
##             10             50
```

In this next section, we recreate the same set of orthogonal contrasts used for analyses in the traditional approaches found in chapter 3.

```
# now create contrasts for the type factor
contrasts.type <- matrix(c(4, -1, -1, -1, -1,
                           0,  3, -1, -1, -1,
                           0,  0, -1, -1,  2,
                           0,  0, -1,  1,  0),
                         ncol=4)
contrasts(rpt1.df$type) <- contrasts.type
contrasts(rpt1.df$type)
```

```
##          [,1] [,2] [,3] [,4]
## clean       4    0    0    0
## homecage   -1    3    0    0
## iguana     -1   -1   -1   -1
## lizard     -1   -1   -1    1
## krat       -1   -1    2    0
```

Fitting the model again now employs those orthogonal contrasts. Note that the t values in the summary table below for these contrasts are not the same as those we found in chapter 2 using the "manual" method or from our SPSS work. They do match the t and p values found in using **emmeans** in chapter 3. The "estimates" shown in the table are correct, reflecting the fact that the contrast coefficients have been orthonormalized. For example, multiple the first "estimate" (for type1) by 20 (the sum of the squared coefficients) and the value is -18.6, the mean of the manually created vector for contrast 1 that was done in chapter 3. The std errors and the df for the error reveal why there is a mismatch. In this analysis that is essentially a reproduction of the GLM approach, the 36 df for the error reveals the fact that errors are not specific to the contrast - the omnibus Axs MS is used in the error. This recreates the same problem with the sphericity assumption that the GLM approach has if the omnibus Axs error term is used for contrasts.

```
fit2.lme <- lme(DV ~ type, random = ~1|snum/type,   data=rpt1.df)
anova(fit2.lme)
```

```
##             numDF denDF   F-value p-value
## (Intercept)     1    36 13.317097   8e-04
```

```
## type            4    36  8.844723  <.0001
```

```
summary(fit2.lme)
```

```
## Linear mixed-effects model fit by REML
##  Data: rpt1.df
##        AIC      BIC    logLik
##   365.0495 379.5028 -174.5247
##
## Random effects:
##  Formula: ~1 | snum
##         (Intercept)
## StdDev:    8.459511
##
##  Formula: ~1 | type %in% snum
##         (Intercept) Residual
## StdDev:    6.930132 3.110725
##
## Fixed effects: DV ~ type
##                 Value Std.Error DF   t-value p-value
## (Intercept) 10.520000 2.8827764 36  3.649260  0.0008
## type1       -0.930000 0.5371375 36 -1.731400  0.0919
## type2       -1.816667 0.6934415 36 -2.619784  0.0128
## type3        4.916667 0.9806744 36  5.013557  0.0000
## type4        1.050000 1.6985778 36  0.618164  0.5404
##  Correlation:
##       (Intr) type1 type2 type3
## type1 0
## type2 0      0
## type3 0      0     0
## type4 0      0     0     0
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -0.756930607 -0.157209242 -0.003407451  0.137647579  1.550657810
##
## Number of Observations: 50
## Number of Groups:
##           snum type %in% snum
##             10             50
```

The **phia** package has a function called `testInteractions` that can work on a lme model object. This function is built to handle factorial designs, and thus the name `testInteractions`. But it can handle main effect contrasts which is what we have in this one-factor design. In testing it for the first contrast, the "estimate" is the expected -18.6 value seen before. I have not yet sorted out how the chi-square test statistic is created, but obviously this function approaches

the question in a different way than the standard methods.

```
# phia may be able to work on these lme objects to obtain contrasts
modmeans <- interactionMeans(fit2.lme)
modmeans
```

```
##       type adjusted mean std. error
## 1    clean           6.8   3.595367
## 2 homecage           6.0   3.595367
## 3   iguana           7.3   3.595367
## 4   lizard           9.4   3.595367
## 5     krat          23.1   3.595367
```

```
interactionMeans(fit2.lme, factors="type" )
```

```
##       type adjusted mean std. error
## 1    clean           6.8   3.595367
## 2 homecage           6.0   3.595367
## 3   iguana           7.3   3.595367
## 4   lizard           9.4   3.595367
## 5     krat          23.1   3.595367
```

```
# define first contrast on the rptd factor
ac1 <- list(type=c(4,-1,-1,-1,-1))
testInteractions(fit2.lme, custom=ac1,adjustment="none")
```

```
## Chisq Test:
## P-value adjustment method: none
##       Value Df  Chisq Pr(>Chisq)
## type1 -18.6  1 2.9977    0.08338 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

At this point, we have not done a full modeling process using LMM. We will need to implement expansion of the analysis to alternative covariance strutures and comparisons of different models. We also need to revisit the question of contrasts. Before we expand the approach, lets examine how a second LMM function handles the design.

4.2 Basic LMM Analysis using `lmer`

The **lme4** package and its `lmer` function are used heavily in some cognitive psychology and linguistics research areas. It also uses the same long-format data frame that we have been using above.

```
#require(lme4)
# using the same contrast set as above and from our SPSS MANOVA example
# we can specify our contrasts
```

```r
#attach(rpt1.df)
contrasts.type <- matrix(c(4, -1, -1, -1, -1,
                           0,  3, -1, -1, -1,
                           0,  0, -1, -1,  2,
                           0,  0, -1,  1,  0),
                         ncol=4)
contrasts(rpt1.df$type) <- contrasts.type
contrasts(rpt1.df$type)
```

```
##          [,1] [,2] [,3] [,4]
## clean       4    0    0    0
## homecage   -1    3    0    0
## iguana     -1   -1   -1   -1
## lizard     -1   -1   -1    1
## krat       -1   -1    2    0
```

```r
fit1.lmer <- lmer(DV ~ type + (1|snum), data=rpt1.df)
anova(fit1.lmer)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##      Sum Sq Mean Sq NumDF DenDF F value    Pr(>F)
## type 2041.5  510.37     4    36  8.8447 4.424e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(fit1.lmer)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: DV ~ type + (1 | snum)
##    Data: rpt1.df
##
## REML criterion at convergence: 349
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.8484 -0.3839 -0.0083  0.3361  3.7866
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  snum     (Intercept) 71.56    8.460
##  Residual             57.70    7.596
## Number of obs: 50, groups:  snum, 10
##
## Fixed effects:
##             Estimate Std. Error      df t value Pr(>|t|)
## (Intercept)  10.5200     2.8828  9.0000   3.649  0.00532 **
```

```
## type1        -0.9300     0.5371 36.0000  -1.731  0.09194 .
## type2        -1.8167     0.6934 36.0000  -2.620  0.01280 *
## type3         4.9167     0.9807 36.0000   5.014 1.44e-05 ***
## type4         1.0500     1.6986 36.0000   0.618  0.54036
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##       (Intr) type1 type2 type3
## type1 0.000
## type2 0.000  0.000
## type3 0.000  0.000 0.000
## type4 0.000  0.000 0.000 0.000
```

The results from the `lmer` analysis are identical to those from `lme`. But neither of these approaches utilizes the full set of modeling strategies that typical LMM methods use. The next sections begin to do that. Our conclusion at this point is that we have put in place LMM code that can recreate the GLM outcome for this simple 1-factor repeated measures design, including tests of contrasts that use the potentially flawed omnibus residual term (Axs).

4.3 A modeling approach

The typical approach to LMM methods is to compare models. We have seen the basics of this idea in an earlier "modeling" document. Here, for LMM models, there are several possible comparisons. The most rudimentary is to compare the fully fit model that includes the "type" IV with an intercept-only model. Those two models are created here, using the `lme` function.

```
# Intercept-only Model
basefit1.lme <- lme(DV ~ 1, random = ~1 | snum/type, data=rpt1.df,
                    method = "ML")
# Augmented Model
typemodel1.lme <- lme(DV ~ type, random = ~1 | snum/type,
                      data=rpt1.df, method = "ML")
#summary(basefit1.lme)
```

Now we can use the `anova` function to compare the two. The full model, called typemodel1.lme, has lower AIC/BIC indices and the liklihood ratio test is significant, indicating that the full model is a better fit. Note that this involves evaluation of only the omnibus effect, and the sphericity assumption is in place for typemodel1.lme, so it may not be the best model available. See the following section for an approach that changes the covariance matrix specification.

```
anova(basefit1.lme, typemodel1.lme)
```

```
##                Model df      AIC      BIC    logLik   Test  L.Ratio p-value
```

```
## basefit1.lme       1  4 394.5131 402.1612 -193.2565
## typemodel1.lme     2  8 375.1337 390.4299 -179.5669 1 vs 2 27.37933  <.0001
```

```
summary(typemodel1.lme)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: rpt1.df
##        AIC      BIC    logLik
##   375.1337 390.4299 -179.5669
##
## Random effects:
##  Formula: ~1 | snum
##         (Intercept)
## StdDev:    8.025397
##
##  Formula: ~1 | type %in% snum
##         (Intercept) Residual
## StdDev:    6.570104 2.960868
##
## Fixed effects: DV ~ type
##                 Value Std.Error DF   t-value p-value
## (Intercept) 10.520000 2.8827764 36  3.649260  0.0008
## type1       -0.930000 0.5371375 36 -1.731400  0.0919
## type2       -1.816667 0.6934415 36 -2.619784  0.0128
## type3        4.916667 0.9806744 36  5.013557  0.0000
## type4        1.050000 1.6985778 36  0.618164  0.5404
##  Correlation:
##       (Intr) type1 type2 type3
## type1 0
## type2 0      0
## type3 0      0     0
## type4 0      0     0     0
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -0.800517754 -0.166261990 -0.003603666  0.145573887  1.639951002
##
## Number of Observations: 50
## Number of Groups:
##           snum type %in% snum
##             10             50
```

4.4 Alternate covariance structures

One of the major advantages of LMM approaches is their ability to handle specifications of alternative covariance matrix structures. Choices are seen in the

help for `lme` and its "correlation" argument. That will be implemented below, but first we will repeat what was done above with the first lme illustration. Our chosen contrasts are in place.

```
fit2.lme <- lme(DV ~ type, random = ~1|snum/type,   data=rpt1.df)
anova(fit2.lme)
```

```
##             numDF denDF   F-value p-value
## (Intercept)     1    36 13.317097   8e-04
## type            4    36  8.844723  <.0001
```

```
summary(fit2.lme)
```

```
## Linear mixed-effects model fit by REML
##  Data: rpt1.df
##        AIC      BIC    logLik
##   365.0495 379.5028 -174.5247
##
## Random effects:
##  Formula: ~1 | snum
##         (Intercept)
## StdDev:    8.459511
##
##  Formula: ~1 | type %in% snum
##         (Intercept) Residual
## StdDev:    6.930132 3.110725
##
## Fixed effects: DV ~ type
##                 Value Std.Error DF   t-value p-value
## (Intercept) 10.520000 2.8827764 36  3.649260  0.0008
## type1       -0.930000 0.5371375 36 -1.731400  0.0919
## type2       -1.816667 0.6934415 36 -2.619784  0.0128
## type3        4.916667 0.9806744 36  5.013557  0.0000
## type4        1.050000 1.6985778 36  0.618164  0.5404
##  Correlation:
##       (Intr) type1 type2 type3
## type1 0
## type2 0      0
## type3 0      0     0
## type4 0      0     0     0
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -0.756930607 -0.157209242 -0.003407451  0.137647579  1.550657810
##
## Number of Observations: 50
## Number of Groups:
```

```
##            snum type %in% snum
##              10             50
```

The "correlation" argument permits specification of the covariance matrix structure. For IVs such as a time factor, such covariance structures might reasonably be autoregressive types, but for our categorical/manipulated IV (type) it is not clear what the structure might be. First, the compound symmetry/sphericity structure is specified. This reproduces the results of initial default analysis where the correlation argument was left out.

```
fit3.lme <- lme(DV ~ type, random = ~1|snum,
                correlation=corCompSymm(form=~1|snum),
                #correlation=corSymm(form=~1|snum), # specifies "unstructured"
                method = "ML", data=rpt1.df)
anova(fit3.lme)
```

```
##             numDF denDF   F-value p-value
## (Intercept)     1    36 13.317097   8e-04
## type            4    36  8.844723  <.0001
```

```
summary(fit3.lme)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: rpt1.df
##        AIC      BIC    logLik
##   375.1337 390.4299 -179.5669
##
## Random effects:
##  Formula: ~1 | snum
##         (Intercept) Residual
## StdDev:    8.025397 7.206455
##
## Correlation Structure: Compound symmetry
##  Formula: ~1 | snum
##  Parameter estimate(s):
## Rho
##   0
## Fixed effects: DV ~ type
##                 Value Std.Error DF   t-value p-value
## (Intercept) 10.520000 2.8827764 36  3.649260  0.0008
## type1       -0.930000 0.5371375 36 -1.731400  0.0919
## type2       -1.816667 0.6934415 36 -2.619784  0.0128
## type3        4.916667 0.9806744 36  5.013557  0.0000
## type4        1.050000 1.6985778 36  0.618164  0.5404
##  Correlation:
##       (Intr) type1 type2 type3
## type1 0
## type2 0      0
```

```
## type3 0      0     0
## type4 0      0     0     0
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -1.948380076 -0.404665041 -0.008770962  0.354312269  3.991476568
##
## Number of Observations: 50
## Number of Groups: 10
```

Now we can compare a model with a different covariance structure to this compound symmetry model. Three models are created here for purposes of comparison. The first is the intercept-only model that does not require a "correlation" argument since there is no IV specified - this repeats the baseline model illustrated above. Second is the full model with the compound symmetry specification. The first two recreate what was done above, just with different object names specified. The third is the full model, but with an "unstructured" covariance matrix specified.

```
# Intercept-only Model
basefit3.lme <- lme(DV ~ 1, random = ~1 | snum/type,
                    method = "ML", data=rpt1.df)
# Augmented Model with Compound symmetry cov matrix
typemodel3a.lme <- lme(DV ~ type, random = ~1|snum,
                 correlation=corCompSymm(form=~1|snum),
                 #correlation=corSymm(form=~1|snum),
                 method = "ML", data=rpt1.df)
# Augmented Model with unstructured
typemodel3b.lme <- lme(DV ~ type, random = ~1|snum,
                 #correlation=corCompSymm(form=~1|snum),
                 correlation=corSymm(form=~1|snum),
                 method = "ML", data=rpt1.df)
```

Now we can use the `anova` function to compare models. First is the repitition of the intercept-only model and the full model with compound symmetry specified. This recreates the comparison made above where the conclusion was that the full model was a better fit.

```
anova(basefit3.lme, typemodel3a.lme)
```

```
##                 Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## basefit3.lme        1  4 394.5131 402.1612 -193.2565
## typemodel3a.lme     2  8 375.1337 390.4299 -179.5669 1 vs 2 27.37933  <.0001
```

The second comparison is between the two full models, the one with the compound symmetry specification (3a) and the one with the unstructured covariance matrix specification (3b). The unstructured matrix version is a better fit as evaluated by the AIC/BIC criteria and the liklihood ratio test is significant,

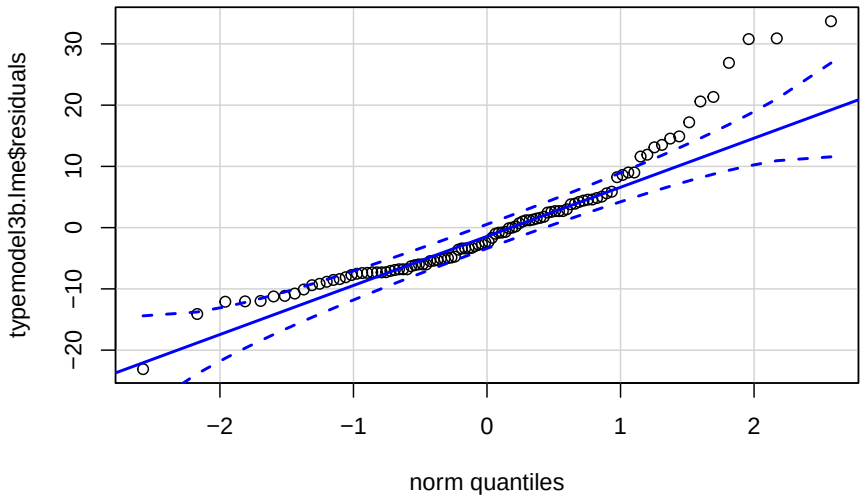
providing another indicator of better fit of the unstructured matrix model.

```
anova(typemodel3a.lme,typemodel3b.lme)
```

```
##                  Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## typemodel3a.lme      1  8 375.1337 390.4299 -179.5669
## typemodel3b.lme      2 17 343.0687 375.5731 -154.5343 1 vs 2 50.06506  <.0001
```

The residuals are more directly accessible with an `lme` object that they were for the `aov` object in chapter 3. We can examine the residuals from this `lme` fit with a qqplot and/or histogram and we find some evidence of non-normality with a positive skewness being present. So even though the modeling suggested the unstructured covariance matrix model was the "best", it may have some issues with the normality assumption. We could also pass those residuals to a normality test such as the Anderson Darling test as we have reviewed in earlier analyses. We might also plot the residuals against the yhats to evaluate homoscedasticity (it is problematic with this data set because of the truncated distributions of the five variables at zero.)

```
#str(typemodel3b.lme)
qqPlot(typemodel3b.lme$residuals,id=FALSE)
```




```
#hist(typemodel3b.lme$residuals)
#library(nortest)
#ad.test(typemodel3b.lme$residuals)
#plot(typemodel3b.lme$fitted,typemodel3b.lme$residuals)
```

4.5 Post hoc pairwise comparisons and planned/orthogonal contrasts

If the analyst wants to perform post hoc pairwise comparison tests, it is also possible to pass the LMM object to the `glht` function from the **multcomp** package. This method employs a strategy not covered in class and is one that produces an approximate standard normal Z test statistic. By evaluating type-model3b.lme, these tests employ standard errors based on the residual from that analysis where the covariance matrix was specified. The error rate inflation problem is addressed by using the Tukey correction method. Since the std errors vary from comparison to comparison, it is clear that one single error term is not employed, thus the errors are specific to the comparision.

```
library(multcomp)
multcomps <- glht(typemodel3b.lme, linfct = mcp(type = "Tukey"))
summary(multcomps)
```

```
## Warning in RET$pfunction("adjusted", ...): Completion with error > abseps
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: lme.formula(fixed = DV ~ type, data = rpt1.df, random = ~1 |
##     snum, correlation = corSymm(form = ~1 | snum), method = "ML")
##
## Linear Hypotheses:
##                          Estimate Std. Error z value Pr(>|z|)
## homecage - clean == 0    -0.800      1.226  -0.653  0.93694
## iguana - clean == 0       0.500      1.906   0.262  0.99775
## lizard - clean == 0       2.600      1.029   2.527  0.05903 .
## krat - clean == 0        16.300      4.734   3.443  0.00366 **
## iguana - homecage == 0    1.300      2.661   0.489  0.97676
## lizard - homecage == 0    3.400      1.455   2.337  0.09371 .
## krat - homecage == 0     17.100      4.777   3.580  0.00240 **
## lizard - iguana == 0      2.100      1.755   1.196  0.64884
## krat - iguana == 0       15.800      4.290   3.683  0.00149 **
## krat - lizard == 0       13.700      3.962   3.458  0.00330 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

The `ghlt` function is also capable of evaluating a set of user-specified contrasts of any kind. Here, I created a matrix of our standard set of orthogonal contrasts employed elsewhere in this document and passed them to the `glht` function

based on the unstructured covariance matrix model. Note that the errors differ for the four contrasts and are specific. This may be the best alternative for evaluating these contrasts with this data set, starting with the best fit LMM model. Note that the "estimates" are the correct values for the means of the four chosen contrasts that we have examined previously.

```
contr <- rbind("clean_vs_all" = c(4,-1,-1,-1,-1),
               "hc_vs_three" = c(0,3,-1,-1,-1),
               "krat_vs_lizards" = c(0, 0,-1, -1, 2),
               "iguana_vs_whiptail" = c(0,0,1,-1,0))
multcontrasts.lmm <- glht(typemodel3b.lme, linfct = mcp(type = contr))
summary(multcontrasts.lmm)
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: User-defined Contrasts
##
##
## Fit: lme.formula(fixed = DV ~ type, data = rpt1.df, random = ~1 |
##     snum, correlation = corSymm(form = ~1 | snum), method = "ML")
##
## Linear Hypotheses:
##                          Estimate Std. Error z value Pr(>|z|)
## clean_vs_all == 0        -18.600      6.726  -2.765   0.0180 *
## hc_vs_three == 0         -21.800      7.672  -2.841   0.0141 *
## krat_vs_lizards == 0      29.500      8.069   3.656   <0.001 ***
## iguana_vs_whiptail == 0   -2.100      1.755  -1.196   0.5053
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

Chapter 5

Robust and Resampling Methods

A few illustrations of robust methods, permutation testing, and bootstrapping are illustrated here. Some authors treat non-parametric tests as members of the class of robust methods. We have already reviewed that topic in a previous chapter.

5.1 Robust Tests

Wilcox has provided a wealth of robust methods for various inferential tests and designs with the **WRS2** package (Mair and Wilcox, 2019). The 1-factor repeated measures design is evaluated with the `rmanova` function. This function requires the long format data and we will use the initial one first used in this document in chapter 2. The robust method evaluates trimmed means and the "tr" argument specifies the degree of trimming. When tr=0, `rmanova` replicates the traditional ANOVA and produces the familiar F value from those earlier analyses. I illustrate here with "tr=.2", but the small number of data points per condition (ten) may mean that this level of trimming is too large. Note that the F statistic from the robust/trimmed approach is smaller than with "tr=0", but the test is still significant even with reduced/adjusted df.

```
library(WRS2)
#rmanova(rpt1.df$DV, groups=rpt1.df$type, blocks=rpt1.df$snum, tr=0)
rmanova(rpt1.df$DV, groups=rpt1.df$type, blocks=rpt1.df$snum, tr=.2)
```

```
## Call:
## rmanova(y = rpt1.df$DV, groups = rpt1.df$type, blocks = rpt1.df$snum,
##     tr = 0.2)
##
```

```
## Test statistic: F = 5.6244
## Degrees of freedom 1: 1.32
## Degrees of freedom 2: 6.61
## p-value: 0.04576
```

Also from **WRS2**, the `rmmcp` function implements tests of pairwise condition comparisons. Wilcox indicates that the method derives critical p-values using the Rom (1990) method where the alpha level is smaller for comparisons with smaller observed p-values. Rom's method is a modification of the Hochberg approach to p value adjustment for multiple comparisons. Note that with ten comparisons and the alpha rate correction, power is not high and only one of the comparisons is significant when tr=.2 (none when tr=0). See Wilcox' writings (Wilcox, 2017) for further explanation. The details are not in the help page for `rmmcp`. Unlike what Wilcox text says, the current version of `rmmcp` from **WRS2** function does not have an argument to change to the method of Hochberg or other methods of adjusting critical p values.

```
allpairs1 <- rmmcp(rpt1.df$DV, groups=rpt1.df$type, blocks=rpt1.df$snum, tr=.2)
#str(allpairs1)
allpairs1
```

```
## Call:
## rmmcp(y = rpt1.df$DV, groups = rpt1.df$type, blocks = rpt1.df$snum,
##     tr = 0.2)
##
##                         psihat  ci.lower ci.upper p.value  p.crit   sig
## clean vs. homecage     0.83333  -2.81237  4.47903 0.32500 0.02500 FALSE
## clean vs. iguana       0.33333  -4.13565  4.80232 0.73634 0.05000 FALSE
## clean vs. lizard      -1.66667  -8.66274  5.32941 0.30702 0.01690 FALSE
## clean vs. krat       -12.33333 -34.36130  9.69464 0.04421 0.00730 FALSE
## homecage vs. iguana    1.16667  -3.49487  5.82820 0.28580 0.01270 FALSE
## homecage vs. lizard   -1.66667  -7.71383  4.38050 0.24540 0.01020 FALSE
## homecage vs. krat    -12.50000 -29.68804  4.68804 0.01782 0.00568 FALSE
## iguana vs. lizard     -2.66667  -8.50075  3.16742 0.08093 0.00851 FALSE
## iguana vs. krat      -12.00000 -24.04772  0.04772 0.00508 0.00511  TRUE
## lizard vs. krat      -11.16667 -27.77203  5.43870 0.02373 0.00639 FALSE
```

One could use the `p.adjust` function to apply other p value adjustment methods. First, we can extract the list of ten p values from the "allpairs1" object.

```
pvals1 <- allpairs1$comp[1:10,6]
pvals1
```

```
##  [1] 0.324998613 0.736344309 0.307018998 0.044212265 0.285798317 0.245404979
##  [7] 0.017822300 0.080927194 0.005084467 0.023730192
```

Then we can apply the `p.adjust` function to that vector. Results not shown to save space.

```
p.adjust(pvals1, method="none")
p.adjust(pvals1, method="bonferroni")
p.adjust(pvals1, method="holm")
p.adjust(pvals1, method="hoch")
p.adjust(pvals1, method="hommel")
p.adjust(pvals1, method="BH")
p.adjust(pvals1, method="BY")
p.adjust(pvals1, method="fdr")
```

The raw, unadjusted, p values find four of the ten pairs to differ at the nominal .05 alpha level. When using any of the `p.adjust` methods, no pairs are found to differ. The Rom method has slightly more power and found that one of the pairs, iguana vs krat, differs.

Looking back at all ten pairs of difference, it is interesting to note that the iguana vs krat comparison is not the pair with the largest mean difference (the psihat value). This seeming discrepancy comes about because each test is done as a paired groups test (dependent samples test) and the standard error of the difference can vary between pairs as well as the mean difference. This assertion can be checked by rerunning the `rmmcp` function without any trimming and then comparing the results from one pair to a dependent samples t-test for the same pair.

```
rmmcp(rpt1.df$DV, groups=rpt1.df$type, blocks=rpt1.df$snum, tr=0)
```

```
## Call:
## rmmcp(y = rpt1.df$DV, groups = rpt1.df$type, blocks = rpt1.df$snum,
##     tr = 0)
##
##                      psihat  ci.lower ci.upper p.value  p.crit   sig
## clean vs. homecage      0.8  -4.14409  5.74409 0.56521 0.01690 FALSE
## clean vs. iguana       -0.5  -8.02647  7.02647 0.81187 0.05000 FALSE
## clean vs. lizard       -2.6  -7.40129  2.20129 0.07680 0.00851 FALSE
## clean vs. krat        -16.3 -35.29012  2.69012 0.01142 0.00730 FALSE
## homecage vs. iguana    -1.3 -12.07890  9.47890 0.66683 0.02500 FALSE
## homecage vs. lizard    -3.4  -9.86598  3.06598 0.08429 0.01020 FALSE
## homecage vs. krat     -17.1 -36.06142  1.86142 0.00883 0.00568 FALSE
## iguana vs. lizard      -2.1  -8.84647  4.64647 0.28037 0.01270 FALSE
## iguana vs. krat       -15.8 -33.89064  2.29064 0.01045 0.00639 FALSE
## lizard vs. krat       -13.7 -28.39754  0.99754 0.00740 0.00511 FALSE
```

```
t.test(rpt1w.df$iguana, rpt1w.df$krat, paired=T)
```

```
##
##  Paired t-test
##
## data:  rpt1w.df$iguana and rpt1w.df$krat
```

```
## t = -3.2225, df = 9, p-value = 0.01045
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -26.891493  -4.708507
## sample estimates:
## mean of the differences
##                   -15.8
```

5.2 Resampling Methods

Both permutation tests and bootstrapping of repeated measures designs would have to focus on resampling the data points within each case/subject/participant. Resampling of cases as a whole would not make sense since that variation is what is "controlled for" by doing the repeated measures analysis. I have not found many illustrations of these methods for repeated measures designs.

5.2.1 Howell's permutation approach

David Howell has shared R code for manually performing a permutation test with the 1-factor repeated measures design. I have adapted it here for our five-category design.

https://www.uvm.edu/~statdhtx/StatPages/Randomization%20Tests/RepeatedMeasuresAnovaR.html

The approach finds a way to reshuffle the five DV values for each case, randomly and within each case. The primary output is a frequency histogram of all of the F values produced by this procedure - 1000 of them since I specifed the number of resamplings as that value. Note that our observed F value for the data set is actually larger than any of the 1000 permuted samples and thus the empirical p-value is zero, an outcome that is rather extreme.

The function requires the long-format data frame which is imported here once again and given a different name so as to not confuse it with any of the earlier data frames in this document.

A note of caution about this method: I have carefully examined Howell's code and found that it does accomplish the permutations as intended. However, for our data set, the empirical distribution of F values has a large and suprising majority of values well less than 1.0. This strikes me as an unexpected outcome. Perhaps the non-sphericity in our data set is contributing to the extreme position of our observed F value (8.847) relative to the empirical sampling distribution based on permutations. This requires more exploration.

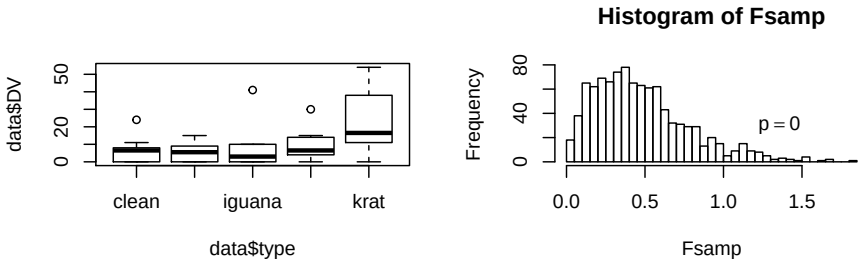
```
data <- read.csv("data/1facrpt_long.csv")
# make sure that the case variable is a factor
data$snum <- factor(data$snum)
```

```r
# change the order of the factor levels of type
# to match the original order
# and match our prior SPSS work, including setting
# up contrasts
data$type <- ordered(data$type,
                     levels=c("clean","homecage","iguana",
                              "lizard", "krat"))
# specify sum to zero contrasts for the type factor
options(contrasts=c("contr.sum","contr.poly"))
aovbase <- aov(DV~type + Error(snum/type), data = data)
#str(summary(aovbase))
obtF <- summary(aovbase)$"Error: snum:type"[[1]][[4]][1]
#obtF
par( mfrow = c(2,2))
boxplot(data$DV~data$type)
nreps <- 1000
counter <- 0
Fsamp <- numeric(nreps)
levels <- c(1:5)
permTypes <-NULL
orderedType <- data[order(data$type),]
set.seed(12357)
for (i in 1:nreps) {
for (j in 1:10) {
  permTypes <- c(permTypes, sample(levels,5,replace = FALSE))
  }
orderedType$permTypes <- permTypes
sampAOV <- aov(DV~factor(permTypes)+Error(factor(snum)),
               data = orderedType)
Fsamp[i] <- summary(sampAOV)$"Error: Within"[[1]][[4]][1]
if (Fsamp[i] > obtF) counter = counter + 1
permTypes <- NULL
}

p <- counter/nreps
cat("The probability of sampled F greater than obtained F is = ", p, '\n')
```

```
## The probability of sampled F greater than obtained F is =  0
```

```r
hist(Fsamp, breaks = 50, bty = "n")
 legend(1,50,bquote(paste(p  == .(p))), bty = "n" )
```



5.3 Bootstrappping

I don't see much usage of bootstrapping with repeated measures designs, or with more complex factorial designs either. But it can be done although with more complex designs, multiple approaches have to be considered. With our simple 1-factor repeated measures design, the bootstrapping is more straight forward, requiring resampling within each case, across the five conditions. One very easy implementation uses a function provided in the **WRS2** package. The `rmanovab` function implements bootstrapping along with robust estimation using trimmed means. The first illustration here sets the trimming to zero, thus matching the core non-robust approach.

The `rmanovab` function can use the same long-format data frame that has been the commonly used one in this document. Here, I use the long-format data frame object created just above for the permutation illustration. The function requires two initial arguments specifying the DV and the IV. The number of bootstrap samples is set to 1000 here (probably overkill) and the trimming to zero. The analysis reports the original F value and compares it to the 95th percentile empirical F distribution cutoff. In this illustration our omnibus F exceeds that critical value, so the null is rejected.

Note that in this illustration, trimming is probably not a good idea with only n=10. Doing trimming results in a substantial power loss here. The function help does not make it clear exactly how the bootstrapping is done, and I have some curiosity about whether bootstrapping whole cases has been performed.

If that is the case, then it is not clear that it is the best approach. It requires a bit more exploration of Wilcox' algorithms before I can be fully confident in what the results mean.

```
library(WRS2)
set.seed(12345)
rmanovab(data$DV, data$type, data$snum, nboot = 1000, tr=0)
```

```
## Call:
## rmanovab(y = data$DV, groups = data$type, blocks = data$snum,
##     tr = 0, nboot = 1000)
##
## Test statistic: 8.8447
## Critical value: 6.9944
## Significant:  TRUE
```

The `pairdepb` function is another **WRS2** function that does bootstrapping. It produces pairwise comparison tests for all possible pairs. From the function help, it is not clear exactly how the bootstrapping is done and it is surprising that all of the CV are identical and none significant. I need to work through Wilcox' textbook a bit more on this function before I can recommend it.

```
## post hoc
set.seed(12354)
pairdepb(data$DV, data$type, data$snum, nboot = 1000, tr=0)
```

```
## Call:
## pairdepb(y = data$DV, groups = data$type, blocks = data$snum,
##     tr = 0, nboot = 1000)
##
##                      psihat  ci.lower ci.upper     test    crit   sig
## clean vs. homecage      0.8  -6.05004  7.65004  0.45083 6.17647 FALSE
## clean vs. iguana       -0.5 -11.20390 10.20390  1.05789 6.17647 FALSE
## clean vs. lizard       -2.6 -16.06131 10.86131 -1.14708 6.17647 FALSE
## clean vs. krat        -16.3 -55.36506 22.76506 -2.23985 6.17647 FALSE
## homecage vs. iguana    -1.3 -11.53631  8.93631  0.80452 6.17647 FALSE
## homecage vs. lizard    -3.4 -14.69917  7.89917 -1.63989 6.17647 FALSE
## homecage vs. krat     -17.1 -54.42551 20.22551 -2.42698 6.17647 FALSE
## iguana vs. lizard      -2.1 -12.16089  7.96089 -2.66027 6.17647 FALSE
## iguana vs. krat       -15.8 -51.53824 19.93824 -2.76520 6.17647 FALSE
## lizard vs. krat       -13.7 -43.26975 15.86975 -2.43691 6.17647 FALSE
```

Chapter 6

Bayesian Approaches

With the many and varied styles of Bayesian Analysis, it is difficult to find succinct advice on the relative merits of the various approaches. But the availability of BayesFactor-based algorithms and their ease of use has made them a go-to approach.

6.1 Bayes Factor analysis

The approach used in the **BayesFactor** package is somewhat analogous to the way we specified LMM models. The `anovaBF` function simply requires specification of a standard model, including the case variable. The key is the specification of that case variable (snum) as a random factor with the "whichRandom" argument. The default prior for the effect with **BayesFactor** functions is Cauchy with a scale parameter $r = \sqrt{2}/2$.

Note that the `anovaBF` function uses the long-format data frame that was initially introduced in chapter 2.

```
mod1.bf = anovaBF(DV ~type + snum, data = rpt1.df,
             whichRandom="snum")
mod1.bf
```

```
## Bayes factor analysis
## --------------
## [1] type + snum : 549.0552 ±0.36%
##
## Against denominator:
##   DV ~ snum
## ---
## Bayes factor type: BFlinearModel, JZS
```

In this simple model there is only one model comparison of interest, the full model with "type" vs an intercept only model. That is the comparison reported by the code above, so no further model comparison strategies are required.

Two remaining issues could be addressed in future revisions of this chapter. The first is the logic for choice of priors in repeated measures designs - we have used the commonly employed default method in `anovaBF`. The second is a return to the question of how to do contrast analysis or post hoc pairwise tests. An indirect way of doing contrasts from a Bayesian perspective is addressed next.

6.2 Contrasts with BayesFactor methods?

One way of approaching contrasts with the BF method is to return to the manually created contrast variables initially examined in chapter 3. I will repeat the creation of those objects here using the wide format data frame.

```
rpt1w.df <- mutate(rpt1w.df,
            ac1=(4*clean+(-1)*homecage+(-1)*iguana+(-1)*lizard+(-1)*krat))
rpt1w.df <- mutate(rpt1w.df,
            ac2=(0*clean+(3)*homecage+(-1)*iguana+(-1)*lizard+(-1)*krat))
rpt1w.df <- mutate(rpt1w.df,
            ac3=(0*clean+(0)*homecage+(-1)*iguana+(-1)*lizard+(2)*krat))
rpt1w.df <- mutate(rpt1w.df,
            ac4=(0*clean+(0)*homecage+(-1)*iguana+(1)*lizard+(0)*krat))
gt(rpt1w.df)
```

snum	clean	homecage	iguana	lizard	krat	ac1	ac2	ac3	ac4
1	24	15	41	30	50	-40	-76	29	-11
2	6	6	0	6	13	-1	-1	20	6
3	4	0	5	4	9	-2	-18	9	-1
4	11	9	10	14	18	-7	-15	12	4
5	0	0	0	0	0	0	0	0	0
6	8	15	10	15	38	-46	-18	51	5
7	8	5	2	6	15	4	-8	22	4
8	0	0	0	11	54	-65	-65	97	11
9	0	3	1	1	11	-16	-4	20	0
10	7	7	4	7	23	-13	-13	35	3

Originally we did four separate one-sample t-tests (null was that each linear combination contrast had a mean of zero). We can do that again, but using BF methods for the 1-sample t-test analog. I illustrate here by examining the third contrast which compares the kangaroo rat condition to the average of the two lizard conditions. The alternative hypothesis that the effect size is not zero is favored moderately.

```r
ttestBF(rpt1w.df$ac3, mu=0)
```

```
## Bayes factor analysis
## --------------
## [1] Alt., r=0.707 : 7.261164 ±0%
##
## Against denominator:
##   Null, mu = 0
## ---
## Bayes factor type: BFoneSample, JZS
```

Chapter 7

Nonparametric Approaches

Non-parametric methods are often employed when variables are not measured with interval-level measurement or when the normality assumptions cannot be satisfied. They typically transform the data to ranks prior to analysis. A good source to review these methods is the Hollander, et al, textbook (Hollander et al., 2013).

There is one easilty implemented non-parametric test for the 1-factor repeated measures omnibus test. It is the Friedman's Rank Sum test and is implemented in the base R installation with the `friedman.test` function.

The function requires the data in wide format but also in matrix form, where the only variables are the repeated measure factor levels. The matrix is arranged so that the levels are columns. We already used this wide format matrix for multivariate linear modeling in chapter 2, so that matrix is still available.

```
gt(rpt1w.mat)
```

clean	homecage	iguana	lizard	krat
24	15	41	30	50
6	6	0	6	13
4	0	5	4	9
11	9	10	14	18
0	0	0	0	0
8	15	10	15	38
8	5	2	6	15
0	0	0	11	54
0	3	1	1	11
7	7	4	7	23

Execution of the function is accomplished by passing only one argument, the name of the data matrix. The test statistic is a chi-squared variable and it is significant for this data set. Degrees of freed are found as the number of levels of the repeated factor minus one.

```
friedman.test(rpt1w.mat)
```

```
##
##  Friedman rank sum test
##
## data:  rpt1w.mat
## Friedman chi-squared = 22.061, df = 4, p-value = 0.0001949
```

Chapter 8

Reproducibility

Version 1.0 April 21, 2020

```
sessionInfo()
```

```
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18362)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] WRS2_1.0-0             sjstats_0.17.9         sciplot_1.2-0
##  [4] rmarkdown_2.1          psych_1.9.12.31        plyr_1.8.6
##  [7] phia_0.2-1             permuco_1.1.0          nortest_1.0-4
## [10] nlme_3.1-145           multcomp_1.4-12        TH.data_1.0-10
## [13] MASS_7.3-51.5          survival_3.1-11        mvtnorm_1.1-0
## [16] knitr_1.28             kableExtra_1.1.0       gt_0.1.0
## [19] granova_2.1            ggthemes_4.2.0         ggplot2_3.3.0
## [22] foreign_0.8-76         ez_4.4-0               emmeans_1.4.5
## [25] car_3.0-7              carData_3.0-3          BayesFactor_0.9.12-4.2
```

```
## [28] coda_0.19-3            afex_0.27-2            lme4_1.1-21
## [31] Matrix_1.2-18
##
## loaded via a namespace (and not attached):
##  [1] minqa_1.2.4         colorspace_1.4-1    ellipsis_0.3.0
##  [4] rio_0.5.16          sjlabelled_1.1.3    estimability_1.3
##  [7] parameters_0.6.0    mc2d_0.1-18         rstudioapi_0.11
## [10] farver_2.0.3        MatrixModels_0.4-1  fansi_0.4.1
## [13] xml2_1.2.5          codetools_0.2-16    splines_3.6.3
## [16] mnormt_1.5-6        sjmisc_2.8.3        nloptr_1.2.2.1
## [19] broom_0.5.3.9000    effectsize_0.2.0    readr_1.3.1
## [22] compiler_3.6.3      httr_1.4.1          backports_1.1.5
## [25] assertthat_0.2.1    cli_2.0.2           htmltools_0.4.0
## [28] tools_3.6.3         lmerTest_3.1-1      gtable_0.3.0
## [31] glue_1.3.2          reshape2_1.4.3      dplyr_0.8.5
## [34] Rcpp_1.0.4          cellranger_1.1.0    vctrs_0.2.4
## [37] insight_0.8.2       xfun_0.12           stringr_1.4.0
## [40] openxlsx_4.1.4      rvest_0.3.5         lifecycle_0.2.0
## [43] gtools_3.8.1        zoo_1.8-7           scales_1.1.0
## [46] hms_0.5.3           parallel_3.6.3      sandwich_2.5-1
## [49] yaml_2.2.1          curl_4.3            pbapply_1.4-2
## [52] sass_0.2.0          reshape_0.8.8       stringi_1.4.6
## [55] bayestestR_0.5.3    checkmate_2.0.0     permute_0.9-5
## [58] boot_1.3-24         zip_2.0.4           rlang_0.4.5
## [61] pkgconfig_2.0.3     commonmark_1.7      evaluate_0.14
## [64] lattice_0.20-40     purrr_0.3.3         labeling_0.3
## [67] tidyselect_1.0.0    magrittr_1.5        bookdown_0.18
## [70] R6_2.4.1            generics_0.0.2      pillar_1.4.3
## [73] haven_2.2.0         withr_2.1.2         mgcv_1.8-31
## [76] abind_1.4-5         performance_0.4.5   tibble_3.0.0
## [79] modelr_0.1.6        crayon_1.3.4        grid_3.6.3
## [82] readxl_1.3.1        data.table_1.12.8   forcats_0.5.0
## [85] digest_0.6.25       webshot_0.5.2       xtable_1.8-4
## [88] tidyr_1.0.2         numDeriv_2016.8-1.1 munsell_0.5.0
## [91] viridisLite_0.3.0
```

Bibliography

Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W., and Iannone, R. (2020). *rmarkdown: Dynamic Documents for R*. R package version 2.1.

Bates, D., Maechler, M., Bolker, B., and Walker, S. (2019). *lme4: Linear Mixed-Effects Models using 'Eigen' and S4*. R package version 1.1-21.

Boik, R. J. (1981). A priori tests in repeated measures designs: Effects of nonsphericity. *Psychometrika*, 46(3):241–255.

De Rosario-Martinez, H. (2015). *phia: Post-Hoc Interaction Analysis*. R package version 0.2-1.

Fox, J., Weisberg, S., and Price, B. (2020). *car: Companion to Applied Regression*. R package version 3.0-7.

Frossard, J. and Renaud, O. (2019). *permuco: Permutation Tests for Regression, (Repeated Measures) ANOVA/ANCOVA and Comparison of Signals*. R package version 1.1.0.

Gross, J. and Ligges, U. (2015). *nortest: Tests for Normality*. R package version 1.0-4.

Hays, W. L. (1994). *Statistics*. Harcourt College Publishers, Fort Worth, 5th edition.

Hollander, M., Wolfe, D. A., and Chicken, E. (2013). *Nonparametric statistical methods*. John Wiley and Sons, Inc., Hoboken, New Jersey, 3rd edition.

Howell, D. C. (2013). *Statistical methods for psychology*. Wadsworth Cengage Learning, Belmont, CA, 8th edition.

Iannone, R., Cheng, J., and Schloerke, B. (2019). *gt: Easily Create Presentation-Ready Display Tables*. R package version 0.1.0.

Keppel, G. and Wickens, T. D. (2004). *Design and analysis : a researcher's handbook*. Pearson Prentice Hall, Upper Saddle River, N.J, 4th edition.

Kirk, R. E. (2013). *Experimental design : procedures for the behavioral sciences*. Sage Publications, Thousand Oaks, 4th edition.

Lawrence, M. A. (2016). *ez: Easy Analysis and Visualization of Factorial Experiments*. R package version 4.4-0.

Lüdecke, D. (2020). *sjstats: Collection of Convenient Functions for Common Statistical Computations*. R package version 0.17.9.

Lenth, R. (2020). *emmeans: Estimated Marginal Means, aka Least-Squares Means*. R package version 1.4.5.

Mair, P. and Wilcox, R. (2019). Robust Statistical Methods in R Using the WRS2 Package. *Behavior Research Methods*. Forthcoming.

Maxwell, S. E., Delaney, H. D., and Kelley, K. (2017). *Designing experiments and analyzing data : a model comparison perspective*. Routledge, New York, NY, third edition / edition.

McCulloch, C. E. (2005). Repeated measures anova, rip? *Chance*, 18(3):29–33.

Morales, M., with code developed by the R Development Core Team, with general advice from the R-help listserv community, and especially Duncan Murdoch. (2020). *sciplot: Scientific Graphing Functions for Factorial Designs*. R package version 1.2-0.

Morey, R. D. and Rouder, J. N. (2018). *BayesFactor: Computation of Bayes Factors for Common Designs*. R package version 0.9.12-4.2.

Myers, J. L., Well, A., and Lorch, R. F. (2010). *Research design and statistical analysis*. Routledge, New York, 3rd edition.

Pruzek, R. M. and Helmreich, J. E. (2014). *granova: Graphical Analysis of Variance*. R package version 2.1.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Revelle, W. (2020). *psych: Procedures for Psychological, Psychometric, and Personality Research*. R package version 1.9.12.31.

Rom, D. M. (1990). A sequentially rejective test procedure based on a modified bonferroni inequality. *Biometrika*, 77(3):663–665.

RStudio Team (2015). *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA.

Singmann, H., Bolker, B., Westfall, J., Aust, F., and Ben-Shachar, M. S. (2020). *afex: Analysis of Factorial Experiments*. R package version 0.27-2.

Wickham, H., Chang, W., Henry, L., Pedersen, T. L., Takahashi, K., Wilke, C., Woo, K., Yutani, H., and Dunnington, D. (2020). *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. R package version 3.3.0.

Wilcox, R. R. (2017). *Introduction to robust estimation and hypothesis testing*. Elsevier, Waltham, MA, 4th edition. edition.

Winer, B. J., Brown, D. R., and Michels, K. M. (1991). *Statistical principles in experimental design*. McGraw-Hill series in psychology. McGraw-Hill, New York, 3rd edition.

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2020a). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.18.

Xie, Y. (2020b). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.28.

Zhu, H. (2019). *kableExtra: Construct Complex Table with 'kable' and Pipe Syntax*. R package version 1.1.0.