Which Request Creates the Most Diet Change: A Reanalysis

Jacob Peacock and Harish Sethu

Humane League Labs Report E006R02
November 13, 2017

Abstract

This document presents a reanalysis of the data and the conclusions reported on September 20, 2015, by Humane League Labs in the blog post titled *Report: Which request creates the most diet change* and an accompanying report titled *Report: Does Encouraging The Public To "Eat Vegan," "Eat Vegetarian," "Eat Less Meat," or "Cut Out Or Cut Back On" Meat And Other Animal Products Lead To The Most Diet Change?* The study examined four different messages encouraging reduced animal product consumption and a control message delivered to college students via a leaflet. Subsequent self-reported dietary changes were measured with a food frequency questionnaire. Based on our reanalysis of the data, we cannot conclude that any of the messages differs significantly from any others in inspiring self-reported changes in the consumption of animal products. Further, the study was insufficiently powered for us to place any confidence in the alternative claim that the four messages are equally effective.

DOI 10.17605/OSF.IO/TFJ9Z

Contents

1 Introduction

This document presents a reanalysis of the data and conclusions reported on September 20, 2015, by Humane League Labs in the blog post titled *Report: Which request creates the most change?* [2] and the accompanying report [3]. This reanalysis aims to address statistical or numerical oversights and errors, assess the methodology in the original report, and revise conclusions as necessary. In keeping with our commitment to open code, the Appendix to this document includes all code used in our reanalysis along with explanatory comments. The code can also be downloaded from the GitLab account of Humane League Labs [4].

2 Methods

In the original study, students on college campuses were randomly approached and asked to complete a food frequency questionnaire on the number of meals consumed containing beef or pork, chicken or turkey, fish, eggs and dairy in a "typical week." Respondents' name, email address, phone number, age bracket and gender were recorded. The respondents were then assigned to one of twelve treatment groups receiving leaflets or to a control group receiving no leaflet. The twelve treatment groups received leaflets with messages encouraging the reader to eat vegan, vegetarian, less meat or to cut out or cut back on meat (henceforth referred to as vegan, vegetarian, less meat and combination messages, respectively). Each message group was divided into three subgroups receiving leaflets 4, 8 or 16 pages in length.

Respondents were given as much time as they desired to read the leaflet. Respondents were then asked to complete an identical food frequency questionnaire but to "picture yourself a couple months in the future." Between two and four months after this initial survey, using recorded emails and phone numbers of the respondents, a follow-up survey with the same food frequency questionnaire was conducted. Respondents were prompted to complete the study with three emails and finally a phone call. Of the 1,594 respondents completing the initial two surveys, 601 (38%) completed the follow-up survey. The links to materials used in the conduct of the study, the leaflets and the survey questionnaire, can be found on the original blog post [2].

3 Verification of numerical claims

The second table in the original report presents the reported percentage reductions in animal product consumption and the number of days of farm animal suffering spared. Our reanalysis found a numerical error in how the entries in this table are computed. The percentage reductions in the original report were computed on the basis of the ending value, rather than the starting value. For example, a decrease from 10 meals of chicken to 8 meals of chicken per week was reported as a 25% decrease instead of 20%. While this is a significant numerical error, it is limited to the

table and not one that affected the ANOVA analysis in the rest of the report. The Appendix to this reanalysis report includes the corrected table.

In some of the post-hoc t-test comparisons subsequent to the ANOVA analysis, we found discrepancies between some of the p values reported in the original report and those computed by us. These differences were not large enough to rise to the level of changing any conclusions.

4 Assessment of the methodology

During our reanalysis we found that, while three food frequency questionnaires are described in the original report (before reading the pamphlet, immediately after reading the pamphlet and a 2–4 month follow-up), the results of the survey immediately after reading the pamphlet are not reported. The authors of this reanalysis do not have information on whether the intermediate survey was not actually conducted but mistakenly mentioned in the report or if the analysis was subsequently limited to exclude the intermediate survey and the responses left out of the released data. However, since the survey results before and months after reading the leaflet are reported, the experiment does allow a study of changes in self-reported consumption of animal products.

The study described uses a 2-factor (message and leaflet length) full-factorial design. The sampling across factors is highly unbalanced, with some combinations of factors having only one sample, and likely too small to reasonably utilize the experiment as designed. A complete analysis of such an experiment would consider both factors and their interactions, while the original analysis only examines the messaging factor without regard to the leaflet length factor. The original report incorrectly states that the 42 respondents who received the 16-page pamphlets were excluded from further analysis; in fact, we found that these respondents were indeed included in all of the analysis.

Finally, the study relies on an unvalidated food frequency questionnaire, which is limited in its accuracy by self-reporting and recall biases.

5 Assessment of inferential claims

In this section, we report on the weaknesses in the inferential logic used in the analysis described in the original report.

5.1 The use of total change as a metric

The original report analyzes the differences between different messages in terms of the change they inspire in self-reported number of meals per week that include an animal product from each of five categories: red meat, poultry, fish, eggs, and dairy. The original report also considers the

sum of these five entries for each respondent and uses this *total change* as an additional category for inclusion in the analysis.

However, the total change must not be treated as a reliable indicator of booklet effectiveness as it misleadingly counts a meal multiple times if it contains multiple categories of animal products. For example, a bacon, egg and cheese sandwich would be counted as a meal each in dairy, egg and red meat categories and contribute three meals to the total. The total is thus dominated by the animal product category most frequently consumed (in this case, dairy).

5.2 Data quality

The study had a response rate of 38%. However, since the response rate alone is not a reliable predictor of non-response bias, we cannot adequately judge the representativeness of the survey respondents. Furthermore, for reasons not known to the authors of this reanalysis report, the data for dairy was missing for many respondents, including 6.5% of the followup respondents.

5.3 The use of post hoc t-tests

When comparing the means of multiple groups, as in this study, it is generally recommended that one first use a conventional ANOVA test to detect any differences in means between groups. Post-hoc t-tests, to determine which groups in particular differ and how, are recommended only if the ANOVA detects significant differences. This recommendation exists to discourage data dredging by performing a multitude of tests producing false positives purely by chance. For example, in this study with 5 groups and 6 outcome categories, 60 comparisons are possible. With so many possible comparisons, it is highly likely that, by mere chance alone, several of these comparisons will yield a significant result at a p value of 0.05.

The original report does not follow the above recommendation. Its ANOVA analysis showed that the differences between group means in none of the five categories (red meat, poultry, fish, eggs and dairy) were significant. Therefore, inferences made in the report based on t-test comparisons between the groups for any of these five categories do not merit our confidence. For example, we *cannot* conclude that the *vegetarian* request was less effective than the *less meat* message at getting people to change their red meat consumption.

6 Reanalysis

In the following, we summarize the reanalysis conducted on the data for this report while the Appendix to this document provides the more complete detail along with the code. (The code can also be downloaded from the GitLab account of Humane League Labs [4].)

The scope of this reanalysis is limited to assessing and revising the methodology used and the conclusions reached in the original report to help clarify the position of Humane League Labs. As

such, we consider the messaging factor but not the leaflet length factor, as in the original report. Given the unbalanced, small-sample experiment design, and the biased and unbiased errors typical of self-reported data based on food frequency questionnaires, the authors of this report judge an expanded scope to consider both factors to be unwarranted.

6.1 Effect sizes and confidence intervals

We begin our reanalysis by computing the effect sizes (Cohen's **f** [1], which is the standard deviation of the standardized group means). We find them to be generally low. We proceed with a visual inspection of 95% Bayesian intervals for changes in consumption of each category of animal products. We find significant overlap amongst these 95% intervals.

6.2 Differences between groups

We begin with tests of the assumptions underlying conventional ANOVA analysis, namely homoscedasticity and normality. Using Levene's test for homoscedasticity, we do not find cause to reject the hypothesis that the variances are equal across groups. Based on the Shapiro-Wilk test of normality and visual inspection of quantile-quantile plots, we conclude that there is moderate non-normality, especially in some animal product categories (red meat, poultry, fish and eggs). Since ANOVA is considered robust to such violations, we proceed with conventional ANOVA analysis but also complement it with a non-parametric Kruskal-Wallis test.

Our results based on conventional ANOVA and the Kruskal-Wallis test found no significant differences between groups in inspiring change in the self-reported consumption of animal products of any of the five categories: red meat, poultry, fish, eggs and dairy. Based on these results, we conclude that we cannot reject the hypothesis that there is no difference between any pair of messages.

The total change is the only category for which the conventional ANOVA and the Kruskal-Wallis tests report a significant result. We do not consider this category as a valid indicator of changes in animal product consumption for the reasons mentioned in Sections 5.1 and 5.2. However, for the sake of completeness, we proceed to use Tukey's honestly significant difference (HSD) post hoc test to ascertain which groups are different from which others. The results, however, indicate no significant differences between any pair of message groups for the total change category, weakening the confidence we can place in the only significant result obtained from our conventional ANOVA and the Kruskal-Wallis tests.

We conclude that, based on any of the metrics considered in the original report, we cannot reject the hypothesis that the message groups are equally effective at inspiring self-reported changes in animal product consumption.

6.3 Are all messages equally effective?

The reanalysis above fails to find support for the claim that any pair of messages are different in the change they inspire. We now assess the confidence we can place in the hypothesis that the booklet messages are all equally effective in inspiring self-reported changes in animal product consumption.

We assess this through a post hoc power analysis. ANOVA power analysis shows that the detectable effect size (Cohen's **f**) with 80% probability (statistical power) at a significance level corresponding to $\alpha = 0.05$ (Type I error) is higher than any of the effect sizes actually observed in the data for the five animal product categories (red meat, poultry, fish, eggs and dairy).

However, we also note that the robustness of ANOVA power analysis depends on assumptions of equal sample sizes, the distribution of the group means, and a single common within-group standard deviation across all groups. These assumptions do not hold in our context and, so, we proceed to use a more robust technique, a permutation test, to assess the statistical power of our ANOVA results. We use synthetic normally distributed data with group sizes, group means and group standard deviations identical to those in our study. The permutation test results further confirm that the study had low statistical power in most of the change categories.

While we do not find support for the claim that any pair of messages are different, we also do not find adequate support for the claim that the messages are all equivalent.

7 Conclusion

The study did not detect any significant differences between the vegan, vegetarian, less meat, combination or control groups on any of the measured changes in diet. Based on post hoc power analysis, we cannot place confidence in the claim that the different booklet messages are all equivalent. The study must be declared inconclusive.

References

[1] COHEN, J. *Statistical Power Analysis for the Behavioral Sciences*, 2 ed. Routledge, 1988.

[2] COONEY, N. Report: Which request creates the most diet change? http://www.humaneleaguelabs.org/blog/2015-09-20-which-request-creates-most-diet-change/, September 2015.

[3] DOEBEL, S., GABRIEL, S., AND COONEY, N. Report: Does encouraging the public to *eat vegan*, *eat vegetarian*, *eat less meat*, *cut out or cut back on* meat and other animal products lead to the most diet change? http://www.humaneleaguelabs.org/static/reports/E006R01-which-request-creates-the-most-diet-change.pdf, August 2015.

[4] HUMANE LEAGUE LABS. Gitlab account. https://gitlab.com/humane-league-labs/E006R02-which-request-creates-the-most-diet-change.git, 2017.

Appendix A: Reanalysis Code and Commentary

In this Appendix, we present the code, along with the results, used to reanalyze the data and conclusions described in the Humane League Labs report titled *Report: Does Encouraging The Public To "Eat Vegan," "Eat Vegetarian," "Eat Less Meat," or "Cut Out Or Cut Back On" Meat And Other Animal Products Lead To The Most Diet Change?* released on September 20, 2015. It covers the quantitative and inferential claims made in the original post and adds brief commentary as necessary. The code is listed in indexed cells labeled In[] and the results are generally reported in the identically indexed Out[] cells.

This document was produced using Jupyter notebook, an interactive computing environment, running the programming language Python. The Jupyter source file corresponding to this document may be found in the Humane League Labs GitLab account.

```
In [1]: from itertools import combinations
        import matplotlib.pyplot as plt
        import math
        import numpy as np
        import pandas as pd
        import random
        from scipy.stats import (ttest_1samp, kruskal, shapiro, f_oneway, levene,
                                 bayes_mvs, norm)
        from statsmodels.formula.api import ols
        from statsmodels.graphics.gofplots import qqplot
        from statsmodels.sandbox.stats.multicomp import MultiComparison
        from statsmodels.stats.power import tt_ind_solve_power, FTestAnovaPower
        from statsmodels.stats.weightstats import ttest_ind

In [2]: %matplotlib inline
        %config InlineBackend.figure_format = 'retina'
```

A.1 Data cleanup

```
In [3]: change_columns = ['RedMeat_Chg', 'Poultry_Chg', 'Fish_Chg',
                          'Eggs_Chg', 'Dairy_Chg']
        followup_columns = ['FollowupRedMeat', 'FollowupPoultry', 'FollowupFish',
                            'FollowupEggs', 'FollowupDairy']
        current_columns = ['CurrentRedMeat', 'CurrentPoultry', 'CurrentFish',
                           'CurrentEggs', 'CurrentDairy']

In [4]: def anonymize_data():
            """
            This function is included only to document our anonymization process
            and is not intended for general use.

            Prepare the anonymized data from the full data, if available.
            """
```

```
        f = 'data/raw/private/Which_request_creates_the_most_diet_change.xlsx'
        try:
            df = pd.read_excel(f)
        except FileNotFoundError:
            return "Full data not found."

        df.drop(['Location', 'Fname', 'PhoneNumber', 'Email'], axis=1,
            inplace=True)
        df.to_csv('data/raw/Which_request_creates_the_most_diet_change.csv',
            index=False)

    anonymize_data()
```

In [5]:
```
def load_data():
    df = pd.read_csv('data/raw/Which_request_creates_the_most_diet_change.csv')

    # make columns into proper booleans
    df.ChangeValueYN.replace({' ': False, '1': True}, inplace=True)
    df.ReceivedBefore.replace({' ': False, '1': True}, inplace=True)

    # fix null values
    fix_nan_cols = change_columns + followup_columns + current_columns + \
        ['Total_Chg']

    for col in fix_nan_cols:

        # replace any single space strings with null
        df[col].replace(' ', float('nan'), inplace=True)
        # now convert all columns to float (some may be object or int)
        df[col] = df[col].astype(float)
        # replace any 99.0, also used as nulls
        df[col].replace(99.0, float('nan'), inplace=True)

    # fix some age typos. A few truly ambiguous typos persist.
    df.Age.replace({
        '18-2':'18-22',
        '30 above': '30 or above',
        '30 0r above': '30 or above',
        '30 or aboe': '30 or above',
        '23-27':'23-29',
        '18-23': '18-22',
        '18 to 22': '18-22'
    }, inplace=True)

    # extract just the message type of the booklet
    df['BookletMessage'] = df.BookletDescrp.str.replace('4|8|16', '')
    # set BookletMessage as a categorical variable
    message_categories = ['vegan', 'veget', 'lessmeat', 'comb', 'control']
    df['BookletMessage'] = pd.Categorical(
        df['BookletMessage'], categories=message_categories, ordered=True)

    # some statistics use the sum of changes for fish, poultry and eggs
```

```
        df['SmallAnimal_Chg'] = df[['Fish_Chg', 'Poultry_Chg', 'Eggs_Chg']]\
            .sum(axis=1, skipna=False)

        return df

    df = load_data()
```

A.2 Data validation

```
In [6]: # check if the calculated change columns are correct
        chg_col_correct = []
        for change, start, end in zip(change_columns, current_columns,
                                      followup_columns):

            # calculate our own change columns. We fill nan with the string 'None'
            # so we can use equality to compare correctly. (Recall nan != nan)
            validate_change = (df[start] - df[end]).fillna('None')

            # record if validate_change agrees with the given change column
            chg_col_correct.append((df[change].fillna('None') == validate_change)\
                                   .all())

        chg_col_correct

Out[6]: [True, True, True, True, True]

In [7]: # check if the calculated total change column is correct
        df.Total_Chg.fillna('None').eq(
            df[change_columns].sum(axis=1, skipna=False).fillna('None')).all()

Out[7]: True

In [8]: # Check the ChangeValueYN column, which indicates whether a respondent
        # has completed the follow-up survey. No respondent completing the
        # follow-up should have null values in all the followup_columns.
        df[df.ChangeValueYN][followup_columns].isnull().all(axis=1).any()

Out[8]: False

In [9]: # All respondents not completing the follow-up survey should have null
        # values in all the followup_columns.
        df[~df.ChangeValueYN][followup_columns].isnull().all(axis=1).all()

Out[9]: True

In [10]: # What's the distribution of null responses among
         # those who were followed up?
         df[df.ChangeValueYN].isnull().mean().T

Out[10]: •ÈÀDateApproached                    0.000000
         Gender                               0.000000
```

```
Age                                              0.000000
BookletDescrp                                    0.014975
BookletLetter                                    0.000000
Request                                          0.000000
Length                                           0.000000
CurrentRedMeat                                   0.001664
CurrentPoultry                                   0.001664
CurrentFish                                      0.003328
CurrentEggs                                      0.000000
CurrentDairy                                     0.039933
CommentsOnBooklet                                0.000000
ReceivedBefore                                   0.000000
FollowupRedMeat                                  0.000000
FollowupPoultry                                  0.001664
FollowupFish                                     0.000000
FollowupEggs                                     0.001664
FollowupDairy                                    0.019967
RememberMostfromBooklet                          0.004992
Haveyousoughtoutanyadditionalinformation         0.000000
OtherComments                                    0.000000
RedMeat_Chg                                      0.001664
Poultry_Chg                                      0.003328
Fish_Chg                                         0.003328
Eggs_Chg                                         0.001664
Dairy_Chg                                        0.059900
Total_Chg                                        0.064892
Total_Current                                    0.000000
Total_FollowupYN                                 0.000000
ChangeValueYN                                    0.000000
BookletMessage                                   0.014975
SmallAnimal_Chg                                  0.006656
dtype: float64
```

A.3 Verification of claims

A.3.0.1 A limited number of respondents were provided with one of four 16-page booklets (one for each of the four message types), but their data was excluded from this study due to the small number of respondents and their uneven distribution across the message types (most respondents who received a 16-page booklet received the version with "vegan" messaging).

While the original report stated that respondents who received the 16-page booklet were excluded from the analysis, we found all the results presented in fact included those respondents. The analysis that follows includes all booklets.

```
In [11]: df[df.ChangeValueYN].BookletDescrp.value_counts(dropna=False)
```

```
Out[11]: comb4         84
         veget8        76
```

```
lessmeat4     70
lessmeat8     70
control       57
veget4        56
vegan4        52
comb8         44
vegan8        41
vegan16       33
NaN            9
lessmeat16     7
comb16         1
veget16        1
Name: BookletDescrp, dtype: int64
```

A.3.0.2 A total of 1,594 respondents completed the two initial surveys.

```
In [12]: len(df)
```

```
Out[12]: 1594
```

A.3.0.3 A total of 601 respondents completed the follow-up survey

```
In [13]: df.ChangeValueYN.sum()
```

```
Out[13]: 601
```

A.3.0.4 Of those who completed the follow up survey, 126 received the vegan message, 133 received the vegetarian message, 147 received the "eat less meat" message, and 129 received the combination "cut out or cut back on" meat and other animal products message. 57 did not receive any message (control participants). 9 people were followed-up but there was no data on which survey they received.

```
In [14]: df[df.ChangeValueYN].BookletMessage.value_counts(dropna=False)
```

```
Out[14]: lessmeat    147
         veget       133
         comb        129
         vegan       126
         control      57
         NaN           9
         Name: BookletMessage, dtype: int64
```

A.3.0.5 There was a fair amount of missing data for dairy during at least one time point: vegan = 6, vegetarian = 4, control = 25 (first time point).

```
In [15]: df[df.ChangeValueYN].groupby('BookletMessage').Dairy_Chg.apply(
             lambda g: g.isnull().sum())
```

```
Out[15]: BookletMessage
         vegan        6
```

```
veget        4
lessmeat     1
comb         0
control     25
Name: Dairy_Chg, dtype: int64
```

A.3.0.6 Animal-Friendly Changes In Diet (positive numbers indicate a reduction in consumption)

```
In [16]: df[df.ChangeValueYN].groupby('BookletMessage')[change_columns + ['Total_Chg']]\
             .mean().round(2)
```

Out[16]:

BookletMessage	RedMeat_Chg	Poultry_Chg	Fish_Chg	Eggs_Chg	Dairy_Chg	Total_Chg
vegan	0.17	0.46	0.17	-0.24	-0.37	0.19
veget	-0.23	0.12	-0.24	-0.22	-0.41	-0.88
lessmeat	0.50	0.41	0.19	-0.01	0.27	1.33
comb	0.70	0.80	0.13	0.43	0.57	2.44
control	0.95	1.12	-0.02	0.16	1.06	4.25

A.3.0.7 Only the ANOVA with "total change" as the dependent variable was significant, F(4, 548)=3.01, p=.018. The ANOVA with red meat as the dependent measure was marginally significant, F(4, 586)=2.31, p=.056

```
In [17]: def run_anovas():
             results = []
             for col in change_columns + ['Total_Chg']:
                 model = ols(col + '~ BookletMessage', df[df.ChangeValueYN]).fit()
                 results.append([col, model.fvalue, model.f_pvalue])

             return pd.DataFrame(results, columns=['change_column', 'F', 'p'])

         run_anovas().round(3)
```

Out[17]:

	change_column	F	p
0	RedMeat_Chg	2.319	0.056
1	Poultry_Chg	1.240	0.293
2	Fish_Chg	0.862	0.487
3	Eggs_Chg	0.892	0.468
4	Dairy_Chg	1.808	0.126
5	Total_Chg	3.014	0.018

A.3.0.8 Inspection of the post-hoc t-tests confirmed that there were no significant differences among any paired comparisons for the poultry, fish, dairy, and egg diet change variables.

We omit the verification of the above statement since post-hoc *t*-tests after ANOVA are not recommended for those dependent variables for which a statistically significant result was not found.

A.3.0.9 Post-hoc t-test comparisons indicate that the vegan message was less effective than the control message (p=.037) at getting people to reduce overall animal product consumption. The vegetarian message was less effective than the combination message (p=.007) and the control message (p=.008). There was a non-significant trend such that the vegan message was less effective than the combination message (p = .071). There was also a non-significant trend such that the vegetarian message was less effective than the eat less meat message (p=.062) All other differences were not in the ballpark of significance

```
In [18]: def all_pairs_t_test(df):
             """
             For each pair of (label, values) tuples in an iterable,
             perform a t-test comparing the pair of values
             """
             results = []
             for a, b in combinations(df, 2):

                 # use the index as labels
                 title = '{} vs. {}'.format(a[0], b[0])

                 # run t-test returning t statistic and p-value
                 t, p, _ = ttest_ind(a[1], b[1])

                 results.append([title, p, t])

             return pd.DataFrame(results, columns=['title', 'p', 't'])

In [19]: all_pairs_t_test(df[df.ChangeValueYN].dropna(subset=['Total_Chg'])\
                          .groupby('BookletMessage').Total_Chg.__iter__()).round(3)
```

Out[19]:

	title	p	t
0	vegan vs. veget	0.368	0.902
1	vegan vs. lessmeat	0.314	-1.008
2	vegan vs. comb	0.087	-1.719
3	vegan vs. control	0.045	-2.021
4	veget vs. lessmeat	0.043	-2.031
5	veget vs. comb	0.009	-2.634
6	veget vs. control	0.009	-2.658
7	lessmeat vs. comb	0.356	-0.926
8	lessmeat vs. control	0.116	-1.579
9	comb vs. control	0.413	-0.820

Our computed p values do not match those in the original report. For *vegan* vs. *control*, we

find a p value of 0.045 (instead of 0.037). For *vegetarian* vs. *combination*, we find a p value of 0.009 (instead of 0.007). For *vegan* vs. *combination*, we find a p value of 0.087 (instead of 0.071). For *vegetarian* vs. *lessmeat*, we find a p value of 0.043 (instead of 0.062).

A.3.0.10 Post-hoc t-test comparisons indicate that the vegetarian request was less effective than the less meat message at getting people to change their red meat consumption (p=.05; although). It was also less effective than the combination (p=.02) and control (p=.02) requests

We omit the verification of the above statement since post-hoc t-tests after ANOVA are not recommended for those dependent variables for which a statistically significant result was not found.

A.3.0.11 We conducted a oneway ANOVA with message type as the between-subjects variable and total change for fish, eggs, and poultry (summing change across these categories) as our dependent variable. The ANOVA was not significant ...

```
In [20]: ols('SmallAnimal_Chg ~ BookletMessage', df).fit().f_pvalue.round(3)
```

```
Out[20]: 0.26400000000000001
```

A.3.0.12 Comparing Change Observed in Each Message Group to Zero Change ... the combination message generated change that differed significantly from zero on total change, red meat, and poultry, ps< .02. The eat less meat message was associated with change that differed from zero in the red meat category only, p=.023. The control message also was associated with change across red meat and poultry, ps < .02. Change in the vegan and vegetarian categories did not significantly differ from zero on any of the measures of change.

```
In [21]: def apply_df_ttest_1samp(df):
             """
             Get the pvalue of a one-sample t-test against zero for each column
             of DataFrame df.
             """
             def apply_col_ttest_1samp(d):
                 _, pvalue = ttest_1samp(d, popmean=0, nan_policy='omit')
                 return pvalue

             return df.apply(apply_col_ttest_1samp)

         df[df.ChangeValueYN].groupby('BookletMessage')[change_columns + ['Total_Chg']]\
             .apply(apply_df_ttest_1samp).round(3)
```

```
Out[21]:                RedMeat_Chg  Poultry_Chg  Fish_Chg  Eggs_Chg  Dairy_Chg  \
         BookletMessage
         vegan                0.525        0.124     0.400     0.413      0.323
         veget                0.442        0.689     0.186     0.478      0.287
         lessmeat             0.023        0.145     0.250     0.975      0.370
         comb                 0.016        0.006     0.560     0.183      0.120
```

```
control                0.017        0.007     0.950     0.728      0.238
```

```
                Total_Chg
BookletMessage
vegan               0.826
veget               0.279
lessmeat            0.072
comb                0.013
control             0.054
```

A.3.0.13 Percentage reductions

```
In [22]: def percentage_meat_reduction(reproduce_original=False):
             """Calculate the percentage change in the average diet by message"""

             followup = df[df.ChangeValueYN][followup_columns].mean()
             # rename the index so the later division will work
             followup.index = followup.index.str.replace('Followup', '')

             current = df[df.ChangeValueYN][current_columns].mean()
             # rename the index so the later division will work
             current.index = current.index.str.replace('Current', '')

             change = df[df.ChangeValueYN].groupby('BookletMessage')\
                 [change_columns].mean()
             # rename columns so tables will align for division
             change.columns = change.columns.str.replace('_Chg', '')

             if reproduce_original:
                 #XXX Note this unusual calculation that reports the percent change
                 # on a basis of the *ending* rather than starting value!
                 return change/followup

             # otherwise return the correct calculation
             return change/current

         percentage_meat_reduction().multiply(100).round(1)
```

```
Out[22]:                RedMeat  Poultry  Fish  Eggs  Dairy
         BookletMessage
         vegan              4.3      8.2   8.2  -5.5   -5.1
         veget             -5.5      2.2 -11.3  -5.0   -5.7
         lessmeat          12.2      7.4   9.0  -0.2    3.8
         comb              17.2     14.5   6.3   9.8    7.9
         control           23.2     20.3  -0.8   3.6   14.7
```

These do not match the numbers in the original report which incorrectly uses the followup numbers for the base in computing the percentages. We demonstrate this by using the incorrect calculation, (followUp - current)/followUp, and showing that we can reach better agreement:

```
In [23]: percentage_meat_reduction(reproduce_original=True).multiply(100).round(1)
```

```
Out[23]:                 RedMeat  Poultry  Fish  Eggs  Dairy
         BookletMessage
         vegan               4.7      9.1   8.5  -5.6   -5.1
         veget              -6.0      2.4 -11.8  -5.1   -5.7
         lessmeat           13.3      8.1   9.4  -0.2    3.8
         comb               18.8     16.0   6.5   9.9    8.0
         control            25.3     22.4  -0.9   3.7   14.9
```

Note that the change in fish consumption for the *vegan* message is 8.5%, but stated as 8.8% in the original report. This is attributable to a round-off error propagated through the calculations.

A.3.0.14 Number of days of farm animal suffering spared

```
In [24]: average_diet_suffering = (113, 1220, 1500, 365, 12)
```

First we propagate the correct calculation from above, which disagrees with the original analysis:

```
In [25]: percentage_meat_reduction().multiply(average_diet_suffering)\
             .sum(axis=1).round(0)
```

```
Out[25]: BookletMessage
         vegan       208.0
         veget      -169.0
         lessmeat    239.0
         comb        328.0
         control     276.0
         dtype: float64
```

Second, we reproduce the original incorrect figures using the followup figures as the base:

```
In [26]: percentage_meat_reduction(reproduce_original=True)\
             .multiply(average_diet_suffering).sum(axis=1).round(0)
```

```
Out[26]: BookletMessage
         vegan       223.0
         veget      -173.0
         lessmeat    255.0
         comb        351.0
         control     304.0
         dtype: float64
```

The differences here are attributable to round-off errors as well.

A.4 Reanalysis

It is worth noting that only 38% of those initially surveyed followed up:

```
In [27]: df.ChangeValueYN.mean().round(3)
```

```
Out[27]: 0.377
```

Even among those who followed up, about 6.5% were missing the change data for at least one product category, usually dairy.

```
In [28]: df[df.ChangeValueYN].Total_Chg.isnull().mean().round(3)
```

```
Out[28]: 0.065000000000000002
```

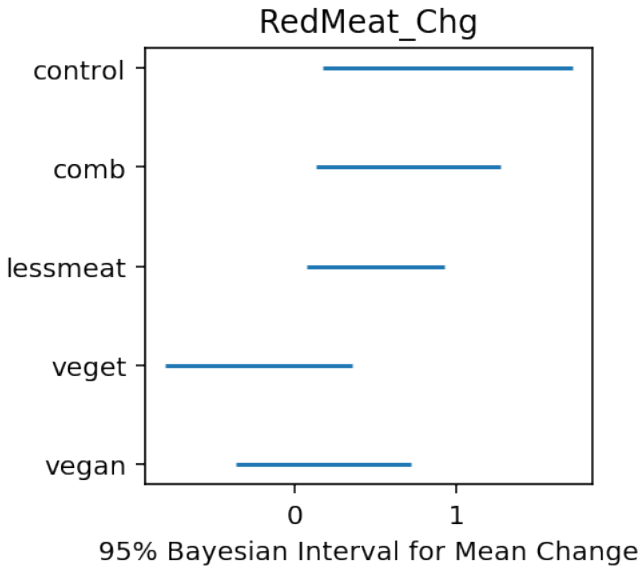
A.4.1 Effect sizes and confidence intervals

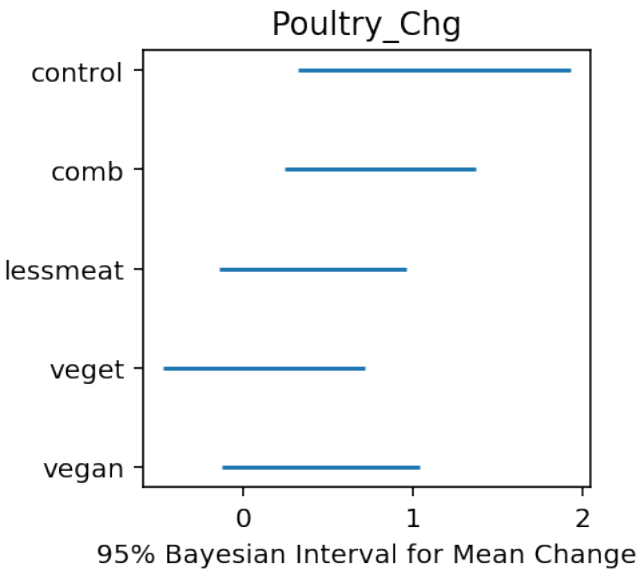
```
In [29]: def cohens_f(factor_col, dependent_col, df):
             """
             Calculate the effect size, f, using Cohen's definition as
             the standard deviation of the standardized group means.
             """

             group_df = df.groupby(factor_col)[dependent_col]

             grand_mean = df[df[factor_col].notnull()][dependent_col].mean()
             group_means = group_df.mean()
             group_variances = group_df.var()
             group_sizes = group_df.count()
             n_groups = len(group_df.groups)
             n_grand = group_df.count().sum()

             num = (group_sizes * (group_means - grand_mean)**2).sum()
             num = (num / n_grand)**0.5

             den = (group_variances * (group_sizes - 1)).sum()
             den = (den / (n_grand- n_groups))**0.5

             return num/den

         def run_cohens_f():
             results = []

             for col in change_columns + ['Total_Chg']:
                 results.append([col, cohens_f('BookletMessage', col, df)])

             return pd.DataFrame(results,
                                 columns=['change_column', 'cohens_f_effect_size'])

         run_cohens_f()
```

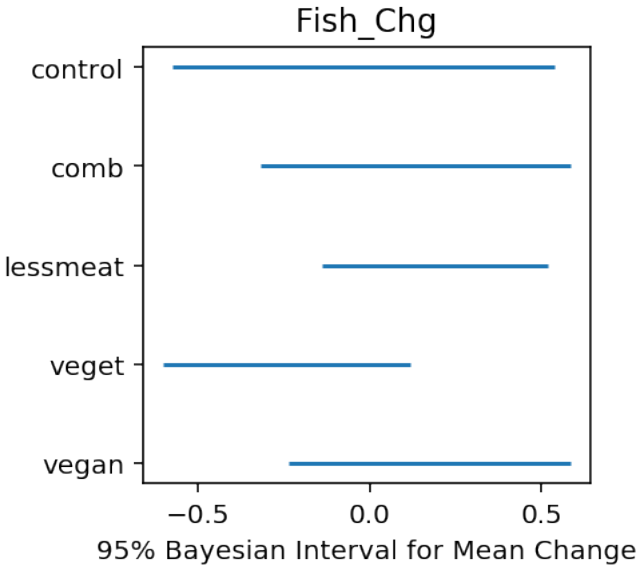
```
Out[29]:   change_column  cohens_f_effect_size
         0   RedMeat_Chg              0.125270
         1   Poultry_Chg              0.091700
         2      Fish_Chg              0.076445
         3      Eggs_Chg              0.077717
         4     Dairy_Chg              0.114060
         5     Total_Chg              0.147644
```

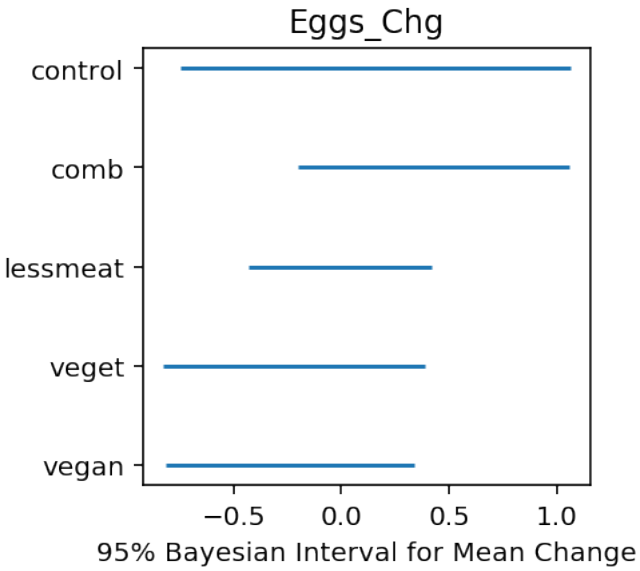
To gain better insight into the effect size distributions, we examine 95% Bayesian intervals for the mean of each change category for each treatment group. Note that these are not conventional confidence intervals; we can say with 95% confidence that the true mean lies within the intervals reported.

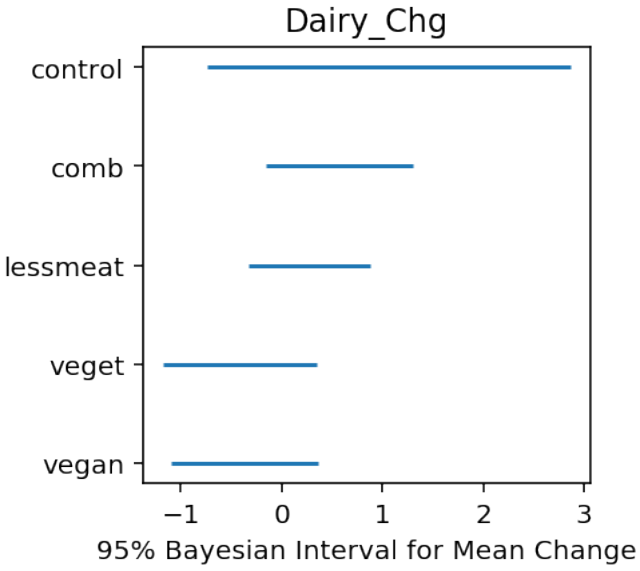
```
In [30]: def plot_mean_bcis(df, title):
             means = []
             means_low = []
             means_up = []
             y_locations = range(len(df.groups))

             for i, (_, group) in enumerate(df):
                 (mean, (m_low, m_up)), _, _ = bayes_mvs(group.dropna(), 0.95)
                 means.append(mean)
                 means_low.append(abs(m_low - mean))
                 means_up.append(abs(m_up - mean))

             plt.figure(figsize=(3,3))
             plt.errorbar(means, y_locations, xerr=(means_low, means_up), linestyle='')
             plt.yticks(y_locations, df.groups)
             plt.title(title)
             plt.xlabel('95% Bayesian Interval for Mean Change')
             plt.show()

         for change_column in change_columns + ['Total_Chg']:
             plot_mean_bcis(df.groupby('BookletMessage')[change_column], change_column)
```

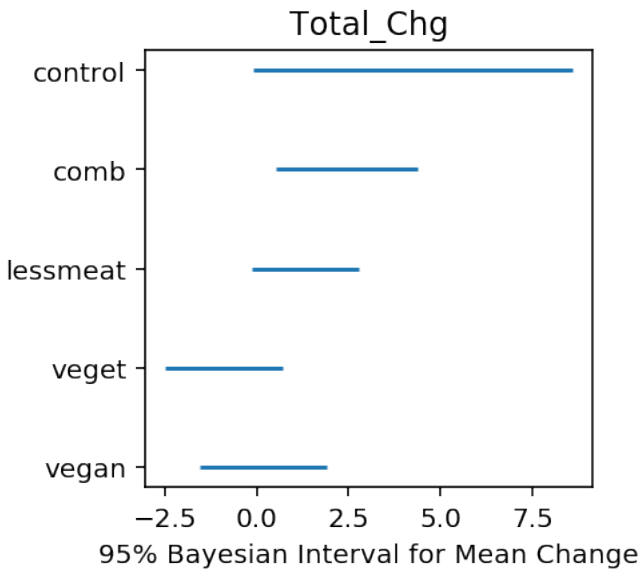



Poultry_Chg
control
comb
lessmeat
veget
vegan
0
1
2
95% Bayesian Interval for Mean Change


Fish_Chg
control
comb
lessmeat
veget
vegan
−0.5
0.0
0.5
95% Bayesian Interval for Mean Change


Eggs_Chg
control
comb
lessmeat
veget
vegan
−0.5
0.0
0.5
1.0
95% Bayesian Interval for Mean Change


Dairy_Chg
control
comb
lessmeat
veget
vegan
−1
0
1
2
3
95% Bayesian Interval for Mean Change



A.4.2 Differences between groups

We analyze the differences between the experiment groups as before, using an ANOVA. However, we first check the assumptions of the ANOVA test: homoskedasticity across groups and normality within each group. We employ Levene's test to test the null hypothesis that variance is equal across experimental groups and conclude the groups are sufficiently homoskedastic.

```
In [31]: def run_levenes_test():

             results = []

             for col in change_columns:

                 prod_change_series = \
                     df.groupby('BookletMessage')[col].apply(lambda df: df.dropna())

                 # list of lists containing the changes for each booklet message
                 prod_change_values = \
                     [df for _, df in prod_change_series.groupby('BookletMessage')]

                 f, p = levene(*prod_change_values)

                 results.append([col, p, f])

             return pd.DataFrame(results, columns=['title', 'p_value', 'f_statistic'])

         run_levenes_test()
```

```
Out[31]:          title   p_value  f_statistic
         0  RedMeat_Chg  0.462030     0.902592
         1  Poultry_Chg  0.996800     0.041043
         2     Fish_Chg  0.940647     0.195671
         3     Eggs_Chg  0.287551     1.252583
         4    Dairy_Chg  0.272227     1.291244
```

However, the Shapiro-Wilk test for normality reveals the values within each group are not normally distributed for all of the change categories. Here, low p values indicate rejection of the null hypothesis of normality, in favor of the alternative that the values are not normally distributed:

```
In [32]: def apply_df_shapiro_wilk(df):

             def apply_col_shapiro_wilk(col):
                 return shapiro(col.dropna())[1]

             return df.apply(apply_col_shapiro_wilk)

         df.groupby('BookletMessage')[change_columns + ['Total_Chg']]\
             .apply(apply_df_shapiro_wilk).round(5)
```

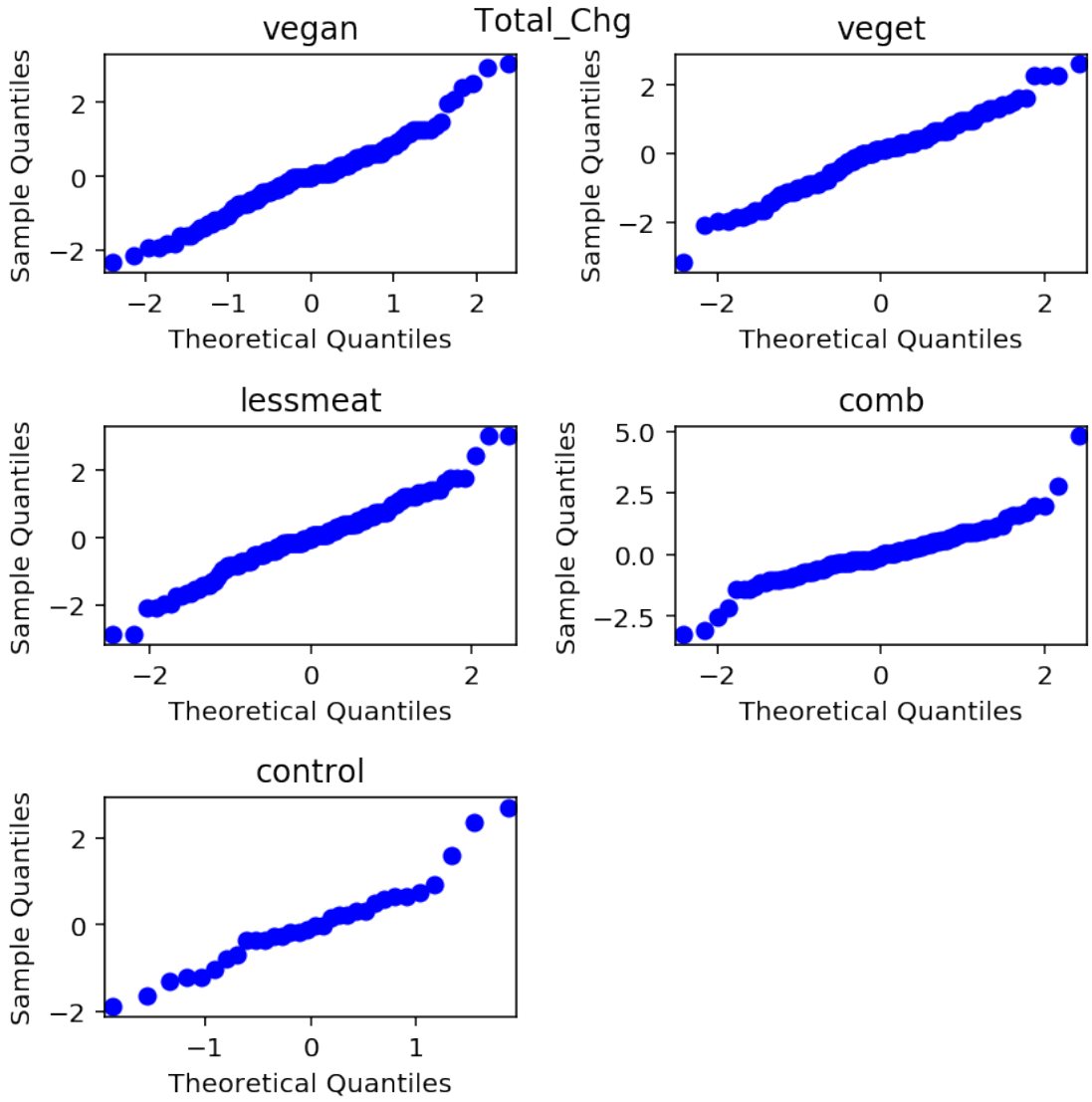
```
Out[32]:                 RedMeat_Chg  Poultry_Chg  Fish_Chg  Eggs_Chg  Dairy_Chg  \
         BookletMessage
         vegan               0.00001      0.00536   0.00000   0.00001    0.03615
         veget               0.00000      0.00001   0.00000   0.00012    0.05056
         lessmeat            0.00000      0.00223   0.00000   0.00009    0.03179
         comb                0.00000      0.00557   0.00000   0.00000    0.19372
         control             0.00001      0.14454   0.00043   0.00033    0.32234

                         Total_Chg
         BookletMessage
         vegan             0.03555
         veget             0.26705
         lessmeat          0.06962
         comb              0.00001
         control           0.20176
```

The above shows the assumption of normality is violated for red meat, poultry, fish and eggs, while dairy and total change categories are sometimes normal. To understand the magnitude of non-normality in the latter cases, we examine the QQ-plot of each group, which compares the observed quantiles to those of a corresponding best-fit normal distribution. If a group were perfectly normally distributed, the plot would appear as a straight line.

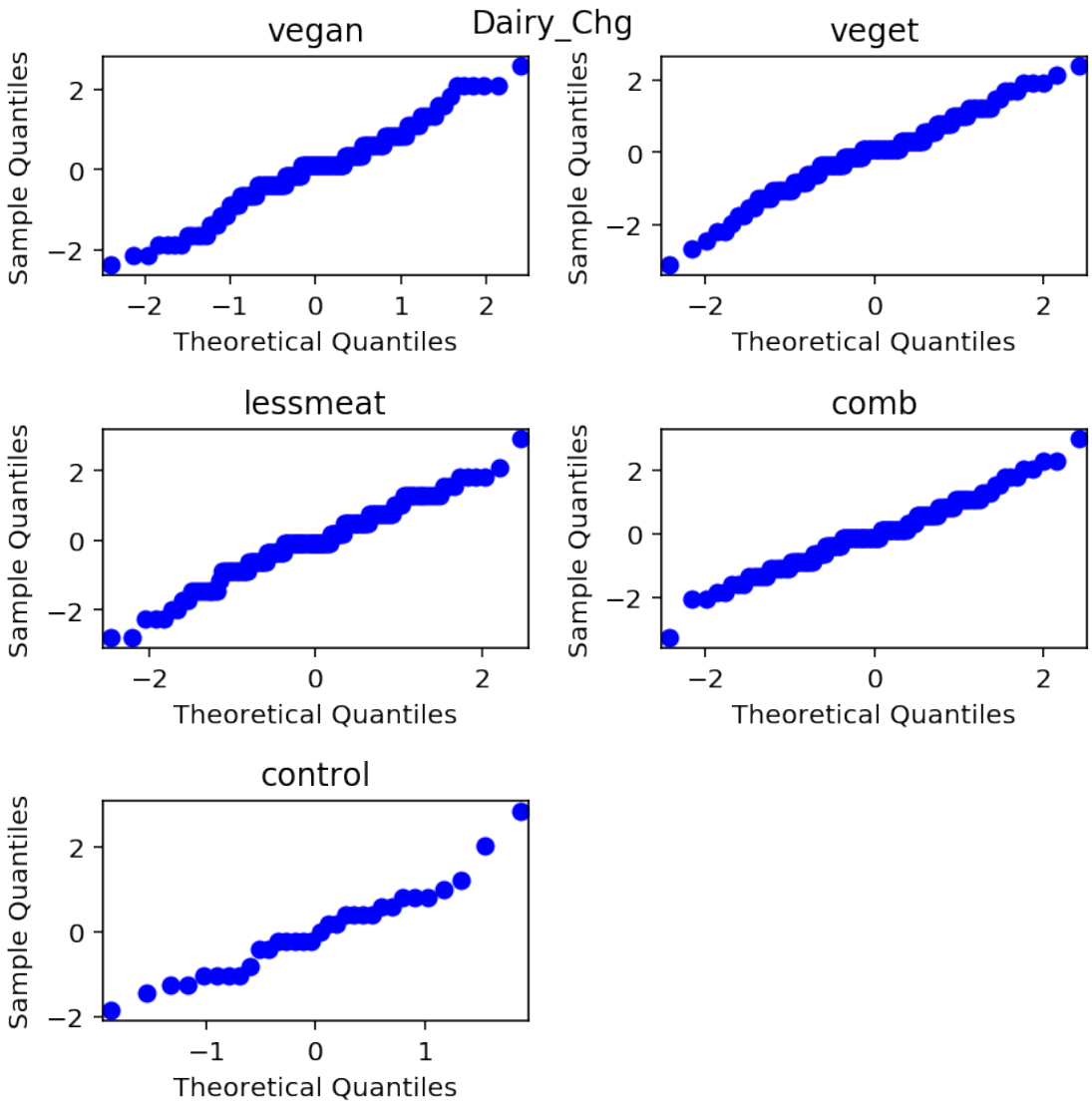
```
In [33]: def make_qqplots(col):
             f, _ = plt.subplots(3, 2, figsize=(6, 6))
             f.suptitle(col)

             for ax, (group, data) in zip(f.axes, df.groupby('BookletMessage')):
                 ax.set_title(group)
```

```
            qqplot(data[col].dropna(), dist=norm, fit=True, ax=ax)

        f.delaxes(f.axes[-1])
        f.tight_layout()
```

In [34]:
```
make_qqplots('Total_Chg')
```



In [35]:
```
make_qqplots('Dairy_Chg')
```



These plots suggest deviations from normality are moderate and primarily in the tails. While ANOVA is generally considered robust to moderate deviations from normality, we confirm the ANOVA results with a non-parametric Kruskal-Wallis ANOVA.

```
In [36]: def run_kruskal_wallis():
             results = []

             for col in change_columns + ['Total_Chg']:
                 prod_change_series = \
                     df.groupby('BookletMessage')[col].apply(lambda df: df.dropna())
```

```
            # list of lists containing the changes for each booklet message
            prod_change_values = \
            [df.values for _, df in prod_change_series.groupby('BookletMessage')]

            result = kruskal(*prod_change_values)
            results.append([col, result.statistic, result.pvalue])

        return pd.DataFrame(results, columns=['title', 'statistic', 'p_value'])

    run_kruskal_wallis().round(3)
```

Out[36]:

	title	statistic	p_value
0	RedMeat_Chg	6.131	0.190
1	Poultry_Chg	6.734	0.151
2	Fish_Chg	3.143	0.534
3	Eggs_Chg	5.309	0.257
4	Dairy_Chg	5.916	0.205
5	Total_Chg	10.813	0.029

As in the traditional ANOVA, the above finds total change to be the only category in which we find a significant difference in means. Since we find that the means may be unequal in the case of the Total_Chg category, we now seek to ascertain which means are different at a significance level of $alpha = 0.05$. In order not to increase the likelihood of a Type I error, we choose not to perform pairwise t-tests but instead apply the Tukey HSD (Honestly Significant Difference) test on the Total_Chg category:

```
In [37]: def run_tukey_test():

            c = df[df.ChangeValueYN][['Total_Chg', 'BookletMessage']].dropna()

            return MultiComparison(c.Total_Chg.values, c.BookletMessage).tukeyhsd()

        print(run_tukey_test())
```

```
Multiple Comparison of Means - Tukey HSD,FWER=0.05
=================================================
 group1   group2  meandiff  lower   upper  reject
-------------------------------------------------
  comb   control   1.8091  -3.4783  7.0964 False
  comb   lessmeat -1.1099  -4.3587  2.1389 False
  comb    vegan   -2.2493  -5.6524  1.1538 False
  comb    veget   -3.3169  -6.6584  0.0246 False
control  lessmeat  -2.919  -8.1399  2.302  False
control   vegan   -4.0583  -9.3767   1.26  False
control   veget    -5.126  -10.4051 0.1532 False
lessmeat  vegan   -1.1394  -4.4383  2.1595 False
lessmeat  veget    -2.207  -5.4423  1.0283 False
 vegan    veget   -1.0676  -4.4579  2.3226 False
-------------------------------------------------
```

The Tukey HSD test above finds no significant differences between any of the booklet messages. The study does not suggest conclusions about relative effectiveness of messages with adequate statistical signifcance.

A.4.3 Post-hoc power analysis

While the above reanalysis fails to reject the null hypothesis that the mean change inspired by the different booklets are the same, we now seek to develop an insight into the confidence we should place in the hypothesis. Following Cohen in *Statistical Power Analysis for the Behavioral Sciences (1988)*, we conduct a *post hoc* power analysis of the ANOVA results by using the size of each group as the arithmetic mean of the 5 group sizes.

```
In [38]: def run_anova_pwr():
             results = []
             for col in change_columns + ['Total_Chg']:
                 effect_size = FTestAnovaPower()\
                     .solve_power(effect_size=None,
                                  alpha=0.05,
                                  power=0.8,
                                  nobs=df.groupby('BookletMessage')[col].count().sum(),
                                  k_groups=5)
                 results.append([col, effect_size])

             return pd.DataFrame(results, columns=['change_column', 'effect_size'])

         run_anova_pwr().round(3)
```

Out[38]:

	change_column	effect_size
0	RedMeat_Chg	0.143
1	Poultry_Chg	0.143
2	Fish_Chg	0.143
3	Eggs_Chg	0.143
4	Dairy_Chg	0.147
5	Total_Chg	0.148

The Cohen's f effect size reported above is the standard deviation of the standardized means detectable with 80% probability at a significance level corresponding to a Type I error probability of 0.05. The minor differences in the effect sizes for different change categories arise from differences in sample sizes caused by NaNs. Following Cohen in *Statistical Power Analysis for the Behavioral Sciences (1988)*, we describe an effect size of 0.1 as small and one of 0.25 as medium. Note that the detectable effect size of 0.14 or above indicates that the survey was not powered enough to detect small effect sizes.

We hasten to add that the robustness of ANOVA power analysis depends on assumptions of equal sample sizes, the pattern of distribution of the group means, and a single common within-group standard deviation across all groups. These assumptions do not hold in our context and so,

we proceed to use a more robust technique, a permutation test, to assess the statistical power of our ANOVA results. We use synthetic normally distributed data with group sizes, group means and group standard deviations identical to those in our study.

```
In [39]: def get_power(g_mean, g_std, g_size):
             """
             Use Fisher's permutation test to return the statistical power on
             an ANOVA given three lists of identical length as parameters:
                 g_mean: the list of means of groups
                 g_std:  the list of standard deviations of groups
                 g_size: the list of sample sizes for each group

             Note: this calculation may take quite some time!
             """

             NUM_SIMS_POWER = 1000
             NUM_SIMS_P = 1000

             k = len(g_mean)
             power_count = 0
             for i in range(NUM_SIMS_POWER):
                 g_all = []
                 g = []
                 for j in range(k):
                     g.append(np.random.normal(loc=g_mean[j],
                                               scale=g_std[j],
                                               size=g_size[j]))
                     g_all.extend(g[j])
                 (f_obs,p_obs) = f_oneway(*g)

                 p_count = 0
                 for s in range(NUM_SIMS_P):
                     random.shuffle(g_all)
                     start = 0
                     for j in range(k):
                         g[j] = g_all[start:start+g_size[j]]
                         start = start + g_size[j]
                     (f,discard) = f_oneway(*g)
                     if f > f_obs:
                         p_count = p_count + 1
                 p = p_count*1.0/NUM_SIMS_P
                 if p < 0.05:
                     power_count = power_count + 1
             return(power_count*1.0/NUM_SIMS_POWER)

         def run_anova_permutation_test():
             results = []
             for col in change_columns + ['Total_Chg']:
                 group_mean = df.groupby('BookletMessage')[col].mean()
                 group_std = df.groupby('BookletMessage')[col].std()
                 group_size = df.groupby('BookletMessage')[col].count()
```

```
            power = get_power(group_mean, group_std, group_size)
            results.append([col, power])

        return pd.DataFrame(results, columns=['change_column', 'power'])

    run_anova_permutation_test()
```

Out[39]:

	change_column	power
0	RedMeat_Chg	0.671
1	Poultry_Chg	0.378
2	Fish_Chg	0.304
3	Eggs_Chg	0.271
4	Dairy_Chg	0.547
5	Total_Chg	0.795

The low statistical power of this study in most change categories suggests that, while we cannot reject the null hypothesis that the means are all equal, we cannot place any confidence in the truth of the hypothesis either. The study must be declared inconclusive.

A.5 Packages

```
In [40]: !conda list -e
```

```
# This file may be used to create an environment using:
# $ conda create --name <env> --file <this file>
# platform: osx-64
_license=1.1=py36_1
alabaster=0.7.10=py36_0
anaconda=4.4.0=np112py36_0
anaconda-client=1.6.3=py36_0
anaconda-navigator=1.6.2=py36_0
anaconda-project=0.6.0=py36_0
appnope=0.1.0=py36_0
appscript=1.0.1=py36_0
asn1crypto=0.22.0=py36_0
astroid=1.4.9=py36_0
astropy=1.3.2=np112py36_0
babel=2.4.0=py36_0
backports=1.0=py36_0
beautifulsoup4=4.6.0=py36_0
bitarray=0.8.1=py36_0
blaze=0.10.1=py36_0
bleach=1.5.0=py36_0
bokeh=0.12.5=py36_1
boto=2.46.1=py36_0
bottleneck=1.2.1=np112py36_0
cffi=1.10.0=py36_0
chardet=3.0.3=py36_0
click=6.7=py36_0
cloudpickle=0.2.2=py36_0
```

```
clyent=1.2.2=py36_0
colorama=0.3.9=py36_0
conda=4.3.21=py36_0
conda-env=2.6.0=0
contextlib2=0.5.5=py36_0
cryptography=1.8.1=py36_0
curl=7.52.1=0
cycler=0.10.0=py36_0
cython=0.25.2=py36_0
cytoolz=0.8.2=py36_0
dask=0.14.3=py36_1
datashape=0.5.4=py36_0
decorator=4.0.11=py36_0
distributed=1.16.3=py36_0
docutils=0.13.1=py36_0
entrypoints=0.2.2=py36_1
et_xmlfile=1.0.1=py36_0
fastcache=1.0.2=py36_1
flask=0.12.2=py36_0
flask-cors=3.0.2=py36_0
freetype=2.5.5=2
get_terminal_size=1.0.0=py36_0
gevent=1.2.1=py36_0
greenlet=0.4.12=py36_0
h5py=2.7.0=np112py36_0
hdf5=1.8.17=1
heapdict=1.0.0=py36_1
html5lib=0.999=py36_0
icu=54.1=0
idna=2.5=py36_0
imagesize=0.7.1=py36_0
ipykernel=4.6.1=py36_0
ipython=5.3.0=py36_0
ipython_genutils=0.2.0=py36_0
ipywidgets=6.0.0=py36_0
isort=4.2.5=py36_0
itsdangerous=0.24=py36_0
jbig=2.1=0
jdcal=1.3=py36_0
jedi=0.10.2=py36_2
jinja2=2.9.6=py36_0
jpeg=9b=0
jsonschema=2.6.0=py36_0
jupyter=1.0.0=py36_3
jupyter_client=5.0.1=py36_0
jupyter_console=5.1.0=py36_0
jupyter_core=4.3.0=py36_0
lazy-object-proxy=1.2.2=py36_0
libiconv=1.14=0
libpng=1.6.27=0
libtiff=4.0.6=3
```

```
libxml2=2.9.4=0
libxslt=1.1.29=0
llvmlite=0.18.0=py36_0
locket=0.2.0=py36_1
lxml=3.7.3=py36_0
markupsafe=0.23=py36_2
matplotlib=2.0.2=np112py36_0
mistune=0.7.4=py36_0
mkl=2017.0.1=0
mkl-service=1.1.2=py36_3
mpmath=0.19=py36_1
msgpack-python=0.4.8=py36_0
multipledispatch=0.4.9=py36_0
navigator-updater=0.1.0=py36_0
nbconvert=5.1.1=py36_0
nbformat=4.3.0=py36_0
networkx=1.11=py36_0
nltk=3.2.3=py36_0
nose=1.3.7=py36_1
notebook=5.0.0=py36_0
numba=0.33.0=np112py36_0
numexpr=2.6.2=np112py36_0
numpy=1.12.1=py36_0
numpydoc=0.6.0=py36_0
odo=0.5.0=py36_1
olefile=0.44=py36_0
openpyxl=2.4.7=py36_0
openssl=1.0.2l=0
packaging=16.8=py36_0
pandas=0.20.1=np112py36_0
pandocfilters=1.4.1=py36_0
partd=0.3.8=py36_0
path.py=10.3.1=py36_0
pathlib2=2.2.1=py36_0
patsy=0.4.1=py36_0
pep8=1.7.0=py36_0
pexpect=4.2.1=py36_0
pickleshare=0.7.4=py36_0
pillow=4.1.1=py36_0
pip=9.0.1=py36_1
ply=3.10=py36_0
prompt_toolkit=1.0.14=py36_0
psutil=5.2.2=py36_0
ptyprocess=0.5.1=py36_0
py=1.4.33=py36_0
pycosat=0.6.2=py36_0
pycparser=2.17=py36_0
pycrypto=2.6.1=py36_6
pycurl=7.43.0=py36_2
pyflakes=1.5.0=py36_0
pygments=2.2.0=py36_0
```

```
pylint=1.6.4=py36_1
pyodbc=4.0.16=py36_0
pyopenssl=17.0.0=py36_0
pyparsing=2.1.4=py36_0
pyqt=5.6.0=py36_1
pytables=3.3.0=np112py36_0
pytest=3.0.7=py36_0
python=3.6.1=2
python-dateutil=2.6.0=py36_0
python.app=1.2=py36_4
pytz=2017.2=py36_0
pywavelets=0.5.2=np112py36_0
pyyaml=3.12=py36_0
pyzmq=16.0.2=py36_0
qt=5.6.2=2
qtawesome=0.4.4=py36_0
qtconsole=4.3.0=py36_0
qtpy=1.2.1=py36_0
readline=6.2=2
requests=2.14.2=py36_0
rope=0.9.4=py36_1
ruamel_yaml=0.11.14=py36_1
scikit-image=0.13.0=np112py36_0
scikit-learn=0.18.1=np112py36_1
scipy=0.19.0=np112py36_0
seaborn=0.7.1=py36_0
setuptools=27.2.0=py36_0
simplegeneric=0.8.1=py36_1
singledispatch=3.4.0.3=py36_0
sip=4.18=py36_0
six=1.10.0=py36_0
snowballstemmer=1.2.1=py36_0
sortedcollections=0.5.3=py36_0
sortedcontainers=1.5.7=py36_0
sphinx=1.5.6=py36_0
spyder=3.1.4=py36_0
sqlalchemy=1.1.9=py36_0
sqlite=3.13.0=0
statsmodels=0.8.0=np112py36_0
sympy=1.0=py36_0
tblib=1.3.2=py36_0
terminado=0.6=py36_0
testpath=0.3=py36_0
tk=8.5.18=0
toolz=0.8.2=py36_0
tornado=4.5.1=py36_0
traitlets=4.3.2=py36_0
unicodecsv=0.14.1=py36_0
unixodbc=2.3.4=0
wcwidth=0.1.7=py36_0
werkzeug=0.12.2=py36_0
```

```
wheel=0.29.0=py36_0
widgetsnbextension=2.0.0=py36_0
wrapt=1.10.10=py36_0
xlrd=1.0.0=py36_0
xlsxwriter=0.9.6=py36_0
xlwings=0.10.4=py36_0
xlwt=1.2.0=py36_0
xz=5.2.2=1
yaml=0.1.6=0
zict=0.1.2=py36_0
zlib=1.2.8=3
```