Which Leaflet is More Effective: A Reanalysis

Jacob Peacock and Harish Sethu

Humane League Labs Report E001R02
August 11, 2017

Abstract

This document presents a reanalysis of the data and the conclusions reported by Humane League Labs on July 19, 2013, in the blog post titled *Report: Which leaflet is more effective?* Our reanalysis finds that, assuming no methodological concerns and based on the data alone, one may reasonably conclude that the leaflet titled *Something Better* is likely more effective in promoting dietary change away from animal products than *Compassionate Choices*. However, given methodological concerns which include the lack of a control group, a reliance on self-reported dietary data and the likely presence of social desirability bias, substantial uncertainty is warranted around this conclusion. Activism strategies related to leafleting must not be guided by the results of this study alone.

DOI 10.17605/OSF.IO/HQZY6

Contents

1 Introduction

This document presents a reanalysis of the data and the conclusions reported on July 19, 2013, by Humane League Labs in the blog post titled *Report: Which leaflet is more effective?* The data reported in the blog post was generated as part of a study conducted by Farm Sanctuary in the Fall of 2012 in collaboration with leafleters from The Humane League [1,2].

The study involved distributing thousands of leaflets outside dining halls at two large schools, the University of Delaware and the University of Maryland. Two different leaflets were distributed in equal numbers: Farm Sanctuary's *Something Better* and Vegan Outreach's *Compassionate Choices*. The leafleters returned to the same spots on the campuses two months later to survey 489 students about changes in their consumption of animal products. Students were approached and asked to agree to take a survey before they were told what it was about; only those who acknowledged receiving one of the two leaflets were asked to complete the survey. Preliminary results were reported by Farm Sanctuary and a further analysis of the data, focusing on the differences between the two leaflets, was reported by Humane League Labs in the aforementioned blog post. The scope of this reanalysis is limited to the claims made in the blog post by Humane League Labs [2].

This reanalysis is intended to serve two purposes: (i) to address any statistical and numerical oversights and errors in the original analysis, and (ii) to identify any methodological limitations of the previously conducted study and clarify the range of possible inferences given the observations.

In keeping with our commitment to open code, the Appendix to this document includes all of the code used in our reanalysis along with explanatory comments as necessary. The code can also be downloaded from the GitLab account of Humane League Labs [3].

2 Verification of numerical claims

Our reanalysis discovered some sources of numerical errors which are described in more detail, along with the correct results, in the Appendix. We summarize here three numerical claims made in the post which most warrant a revision:

- The original post stated that about one out of every 50 students who received a leaflet indicated they became vegetarian or pescatarian as a result. The answers given by the survey respondents to the specific multiple-choice questions, however, only justify this claim for one out of every 70 students.
- The original post stated that, among those who had not previously received either leaflet before this study, *Something Better* appeared to spare 35% more animals than *Compassionate Choices*. However, given the data and the assumptions made, the correct percentage is 36.38%.

- The original post stated that, among those who had previously received the *Compassionate Choices* leaflet, *Something Better* appeared to spare 49% more animals than *Compassionate Choices*. However, given the data and the assumptions made, the correct percentage is 38.16%.

3 Assessment of methodology

The design of this study lacks a control group, treated with either a leaflet on a non-animal issue or no leaflet at all. In the absence of a control group, we do not have a baseline against which we can contextualize the observed effect sizes. Lack of a control group also restricts our ability to infer a causal link between receiving a leaflet and reporting a substantive dietary change.

Furthermore, the study's reliance on self-reported data for the magnitude of dietary changes may encompass unknown biased and unbiased errors. This data is also likely to have been influenced by social desirability bias for at least two reasons: (i) the inclusion, in the survey questionnaire, of a reminder of the leaflet advocating eating fewer animals, and (ii) the nearby physical presence of the survey administrator while the survey questions were being answered.

4 Assessment of inferential claims

4.1 Proportion who report a substantive dietary change

The blog post drew inferences based on the proportions of three subgroups of surveyed students who reported substantial dietary changes (eating "a lot less" of an animal product or "stopped" eating it entirely):

- Subgroup A: Those who had not previously received the *Compassionate Choices* leaflet.
- Subgroup B: Those who had previously received the *Compassionate Choices* leaflet.
- Subgroup C: Those who had not previously received either leaflet.

Our reanalysis found that, in each of these three subgroups and for each category of animal products, the reported percentage of students who substantially reduced their consumption of that product was higher for the subset that received the *Something Better* leaflet than for the subset that received the *Compassionate Choices* leaflet.

As part of our reanalysis, we computed the confidence level at which the MOVER-B Bayesian interval of the difference between the proportions for the two leaflets does not contain zero [4]. Table 1 shows these confidence levels for each of the subgroups and the five categories of animal products. Since these are Bayesian intervals (as opposed to the classical confidence intervals), for each product category, they can be interpreted as the confidence level with which we can infer from the data that one of the leaflets is more effective than the other.

Table 1: Estimated probability that the true proportion of students who would report a substantial dietary change pertaining to a product was higher among those who received the *Something Better* leaflet than among those who received the *Compassionate Choices* leaflet.

	Subgroup A	Subgroup B	Subgroup C
Chicken	85%	59%	97%
Beef/pork	61%	91%	50%
Fish/seafood	77%	96%	68%
Eggs	92%	92%	96%
Dairy	85%	92%	88%

As shown in Table 1, the confidence with which we can claim that the *Something Better* leaflet is more effective than the *Compassionate Choices* leaflet ranges from 50% to 97% depending on the subgroup of students and the category of animal product. In translating these numerical estimates of confidence levels into plain language, we are guided by the scale adopted by the Intergovernmental Panel on Climate Change [5], which suggests that the range of confidence levels in Table 1 reflect a "likely" difference.[1]

Based solely on the data reported in this study and ignoring any methodological issues that may raise concerns about the accuracy of the data, we would characterize it as *likely* that *Something Better* is more effective than *Compassionate Choices*.

4.2 Number of animals spared

In the following, as is conventional, we set the cutoff for statistical significance at a p value of 0.05.

To further compare the effectiveness of the two leaflets, we ran a two-sample t-test on the mean of the number of animals spared among the two groups of students. We found a p value of 0.14, indicating a lack of statistical significance.

Since the distribution of the animals spared was not normal, we also conducted the Wilcoxon-Mann-Whitney U test, which does not make an assumption of normality. This test checks if a random student chosen from the group that received one leaflet would spare more animals than a random student chosen from the group that received the other leaflet. We found a p value of 0.25, again indicating a lack of statistical significance.

[1] The other terms in IPCC's likelihood scale expressing affirmative confidence are *virtually certain*, *extremely likely*, *very likely*, and *about as likely as not*.

4.3 Compensating for social desirability bias

The blog post makes the argument that if "social desirability bias is accounted for, the difference [in animals spared] between the two booklets grows a lot larger." This argument hinges in part on the assumed uniformity of social desirability bias between the two leaflets and among the categories of animal products. However, the two leaflets differ in content, including in their covers displayed alongside the survey, likely eliciting socially desirable responses at different rates. Also, people's perceptions of different animals are known to vary; as a result, the magnitude of the social desirability bias contributing to the number of animals spared is not a fixed quantity independent of the composition of the different animals constituting that number. In light of these issues and in the absence of a baseline control group or supporting data on the distribution of social desirability bias, it is not possible to test whether social desirability bias may be masking the claimed superior performance of the *Something Better* leaflet.

5 Conclusion

Based on the reported data alone and ignoring methodological concerns, it is reasonable to conclude that the *Something Better* leaflet is *likely* more effective than the *Compassionate Choices* leaflet. However, the study has a few methodological weaknesses which include the lack of a control group, the presence of social desirability bias and the use of self-reported dietary data. Given these methodological issues which add a substantial but an unquantifiable amount of uncertainty to this conclusion, we do not recommend that activism strategies related to leafleting be guided by the results of this study alone.

References

[1] COONEY, N. The powerful impact of college leafleting (part 1). https://ccc.farmsanctuary.org/the-powerful-impact-of-college-leafleting-part-1/, January 2013.

[2] COONEY, N. Report: Which leaflet is more effective? http://www.humaneleaguelabs.org/blog/2013-07-19-which-leaflet-is-more-effective/, July 2013.

[3] HUMANE LEAGUE LABS. Gitlab account. https://gitlab.com/humane-league-labs/E001R02-which-leaflet-is-more-effective.git, 2017.

[4] LAUD, P. J. Equal-tailed confidence intervals for comparison of rates. *Pharmaceutical Statistics* (June 2017).

[5] MASTRANDREA, M. D., ET AL. Guidance note for lead authors of the IPCC fifth assessment report on consistent treatment of uncertainties. In *IPCC Cross-Working Group Meeting on Consistent Treatment of Uncertainties* (2010). https://www.ipcc.ch/pdf/supporting-material/uncertainty-guidance-note.pdf.

Appendix A: Reanalysis Code and Commentary

A.1 Introduction

In this Appendix, we present the code, along with the results, used to reanalyse the data reported in the Humane League Labs blog post titled *Report: Which leaflet is more effective*, originally posted on July 19, 2013. 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 the following, the two leaflets compared, *Something Better* and *Compassionate Choices*, are abbreviated either as 'SB' and 'CC', or as 'better' and 'compassionate', respectively.

A.2 Imports, globals and utilities

A.2.1 Imports

```
In [1]: from collections import namedtuple
        import math
        import pandas as pd
        from rpy2.robjects.packages import importr
        import rpy2.robjects as robjects
        from scipy.stats import ttest_ind
        from scipy.stats import mannwhitneyu
        from statsmodels.stats.proportion import proportion_confint
        import warnings
        %matplotlib inline

        # We use the moverbci() function in the ratesci package in R to compute
        # the MOVER-B Bayesian interval.
        ratesci = importr('ratesci')
        rmoverbci = robjects.r['moverbci']
```

A.2.2 Globals

```
In [2]: product_columns = ['ChangeChicken', 'ChangeBeefPork', 'ChangeFishSeafood',
                           'ChangeEggs', 'ChangeDairy']
        # The conversion of the change in the amount of a product eaten to the
        # number of animals spared is based on average per capita consumption of
        # each product in the US by consumers of meat who are not vegetarians or
        # meat-reducers. The numbers used in the original blog post are inferred
        # from the diet change Excel spreadsheet included with the raw data
        # released with the post. These numbers were derived from an early, less
```

```
        # correct, version of [^1] for chickens (28.0), beef cows (0.125),
        # pigs (0.5), turkeys (1.0) and fish (1.0). The numbers for dairy
        # cows (1/30) and egg industry hens (2.0) are also mentioned in [^2].
        #
        # [^1]:
        # http://countinganimals.com/how-many-animals-does-a-vegetarian-save/
        # [^2]:
        # https://ccc.farmsanctuary.org/the-powerful-impact-of-college-leafleting/

        product_to_animal = dict(zip(product_columns, [28.0, 1.0/8 + 1.0/2, 1.0,
                                                       2.0, 1.0/30]))

        change_categories = [
            "No response",
            "I ALREADY DID NOT EAT this product when I got the booklet",
            "I eat MORE",
            "I eat the SAME",
            "I eat a LITTLE LESS",
            "I eat a LOT LESS",
            "I STOPPED EATING this product",
        ]

        change_codes = ['nr','na', 1, 0, -1, -2, -3]

        change_categories_to_codes = dict(zip(change_categories, change_codes))

        change_code_to_multiplier = dict(zip(change_codes, [0.0, 0.0, -0.3, 0, 0.1,
                                                            0.4, 1.0]))
```

A.2.3 Utilities

```
In [3]: def make_respondent(c=0, b=0, f=0, e=0, d=0):
            """
            Makes a Respondent object for testing purposes.

            Up to 5 arguments will be assigned in order as attributes:
                'ChangeChicken', 'ChangeBeefPork', 'ChangeFishSeafood',
                'ChangeEggs' and 'ChangeDairy'.
            Values must be a valid change code from among:
                'nr', 'na', 1, 0, -1, -2, -3
            and will be converted to the corresponding change category. Missing
            arguments default to 0.
            """

            Respondent = namedtuple('respondent', ['ChangeChicken',
                                                   'ChangeBeefPork',
                                                   'ChangeFishSeafood',
                                                   'ChangeEggs',
                                                   'ChangeDairy'])
            return Respondent(c, b, f, e, d)

In [4]: def is_converted_pescatarian(r):
```

```
        """
        Returns a boolean of whether the respondent r was converted into a
        pescatarian.

        As per our interpretation of the assumptions made, to qualify,
        r must:
            (1) stop eating or already not eat chicken and beef/pork, and
            (2) stop eating at least one of chicken and beef/pork, and
            (3) eat fish.

        Without qualification (2), respondent was not converted, but rather
        started out as a pescatarian. Without (3), respondent is, in fact,
        a vegetarian.
        """

        changes = set([r.ChangeChicken, r.ChangeBeefPork])

        return (
            # did not eat chicken/beef/pork
            (changes.issubset(set(['na', -3]))) &
            # but they actually reduced consumption of at least one
            (changes != set(['na'])) &
            # and they still eat fish, otherwise they'd be vegetarian
            (r.ChangeFishSeafood not in set(['na', -3]))
        )

    # Tests
    # stopped eating chicken and beef, did not change fish consumption.
    assert is_converted_pescatarian(make_respondent(-3, -3, 0))
    # stopped eating chicken and beef, reduced fish
    assert is_converted_pescatarian(make_respondent(-3, -3, -2))
    # stopped eating chicken and beef and fish, is not pescatarian, but
    # vegetarian
    assert not is_converted_pescatarian(make_respondent(-3, -3, -3))
    # stopped eating chicken and beef but never ate fish, is not a converted
    # pescatarian, but vegetarian
    assert not is_converted_pescatarian(make_respondent(-3, -3, 'na'))
    # never ate any chicken, beef or fish. Not converted.
    assert not is_converted_pescatarian(make_respondent('na', 'na', 'na'))
    # stopped eating chicken, beef and fish. Converted vegetarian, not
    # pescatarian
    assert not is_converted_pescatarian(make_respondent(-3, -3, -3))
    # Only reduced chicken consumption. Not converted pescatarian.
    assert not is_converted_pescatarian(make_respondent(-2, -3, -1))
```

In [5]:
```
def is_converted_vegetarian(r):
    """
    Returns a boolean of whether respondent r is vegetarian.

    As per our interpretation of the assumptions made, to qualify,
    r must:
        (1) stop eating or already not eat chicken, beef/pork and
```

```
            fish, and
        (2) have stopped eating as least one of those categories.

    If respondent did not stop eating at least one category, they were
    already vegetarian and do not qualify.
    """

    changes = set([r.ChangeChicken, r.ChangeBeefPork, r.ChangeFishSeafood])

    return (
        # stopped or did not eat all meats
        changes.issubset(set(['na', -3])) &
        # stopped eating at least one meat; otherwise we didn't actually
        # convert them
        (changes != set(['na']))
    )

# Tests
# stopped eating chicken, but didn't eat beef/pork or fish.
assert is_converted_vegetarian(make_respondent(-3, 'na', 'na'))
# stopped eating chicken and fish, didn't eat beef/pork
assert is_converted_vegetarian(make_respondent(-3, 'na', -3))
# already vegetarian, so not converted
assert not is_converted_vegetarian(make_respondent('na', 'na', 'na'))
# increased all
assert not is_converted_vegetarian(make_respondent(1, 1, 1))
# no change in chicken and beef/pork, but reduced fish
assert not is_converted_vegetarian(make_respondent(0, 0, -2))
# reduced chicken, didn't eat beef/pork, increased fish
assert not is_converted_vegetarian(make_respondent(-1, 'na', 1))
```

In [6]:
```
def animals_spared(r, _product_to_animal=product_to_animal):
    """
    For a respondent r, calculate the number of animals spared based
    on their changes in consumption. Non-responses ('nr') and missing
    responses ('na') are counted as zero.

    The percent changes corresponding to each change response and the
    number of animals consumed on average are the same as those used in
    the original study.
    """
    animals_spared = 0

    for product in product_columns:
        change_code = getattr(r, product)
        animals_spared += _product_to_animal[product] \
                          * change_code_to_multiplier[change_code]

    return animals_spared


# Tests
```

```
    # doing nothing saves no animals
    assert animals_spared(make_respondent()) == 0.0
    # respondents who are already vegan also have no change
    assert animals_spared(make_respondent('na', 'na', 'na', 'na', 'na')) == 0.0
    # increasing consumption saves negative animals
    assert animals_spared(make_respondent(1, 1, 1, 1, 1)) < 0
    # reducing consumption saves positive animals
    assert animals_spared(make_respondent(-3, -3, -3, -3, -3)) > 0
    # all category codes work
    assert animals_spared(make_respondent('na', -3, -2, -1, 0))
    # handles nan correctly
    assert animals_spared(make_respondent('nr', 'nr', 'nr', 'nr', 'nr')) == 0.0
```

In [7]:
```
def conf_bci_different(x1=0, n1=0, x2=0, n2=0):
    """
    For two proportions x1/n1 and x2/n2, return the maximum confidence
    level at which a one-sided difference interval does not include
    zero. This function uses the Bayesian method, MOVER-B. This is an
    approximate computation.

    The quantity returned may be interpreted as the confidence we can
    have that the true value sampled by one proportion (x1/n1) is
    larger than the true value sampled by the other proportion (x2/n2).
    """
    # force x1/n1 to be greater than x2/n2
    if x1 / n1 < x2 / n2:
        x1, x2 = x2, x1
        n1, n2 = n2, n1

    # iterate over different confidence levels (starting at 99%) until
    # the lower limit of the interval is greater than zero. Since x1/n1
    # is forced to be larger than x2/n2, we know the iteration need not
    # continue below a confidence level of 50%. If the iteration
    # goes all the way to 51% and the lower limit is still below zero,
    # we can stop the iteration and return 50.
    percent_confidence = 99
    lower = -1.0

    while percent_confidence > 51 and lower <= 0.0:

        percent_confidence -= 1

        # reduce the confidence level to get a one-sided interval
        level = 2 * percent_confidence / 100 - 1

        lower = rmoverbci(x1, n1, x2, n2, level=level)[0]

    # Meanwhile, as soon as the lower limit is still below zero at 51%
    # confidence, we can stop the iteration and return 50.
    if lower < 0:
        percent_confidence = 50
    return percent_confidence
```

```
In [8]: def fraction_to_percent(d):
            """Multiply d by 100 and round to 2 decimal places"""

            if isinstance(d, (float, int)):
                # convert to string and back to float to make sure result is
                # pretty!
                return float(format(100 * d, '.2f'))

            elif isinstance(d, (pd.DataFrame, pd.Series)):
                return (100 * d).round(2)

            else:
                raise ValueError('Data of type {} cannot be rounded.'.format(type(d)))

In [9]: def one_out_of(x):
            """Express a fraction less than 1 in the form 'one out of'"""
            return '1 out of %0.1f' % (1/x)
```

A.3 Data loading

```
In [10]: def anonymize_data(f):
             """
             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.
             """

             try:
                 df = pd.read_csv('data/private/' + f)
             except FileNotFoundError:
                 return "Full data not found."
             else:
                 pii_columns = ['IPAddress', 'EmailAddress', 'FirstName', 'LastName']

                 df.drop(pii_columns, axis=1).to_csv('data/raw/' + f, index=False)

         anonymize_data('compassionate-choices-details.csv')
         anonymize_data('something-better-details.csv')

In [11]: def load_data():
             data_compassionate = pd.read_csv(
                 "data/raw/compassionate-choices-details.csv",
                 index_col=0, parse_dates=[1, 2])
             data_compassionate['PamphletReceived'] = 'compassionate'

             data_better = pd.read_csv(
                 "data/raw/something-better-details.csv",
                 index_col=0, parse_dates=[1, 2])
             data_better['PamphletReceived'] = 'better'
```

```
    data = pd.concat([data_better, data_compassionate])

    # convert "Yes" and "No" to True and False, respectively
    yes_no__boolean = {'Yes': True, 'No': False}
    data.ReceivedBPreviously.replace(yes_no__boolean, inplace=True)
    data.ChangeSomeoneElse.replace(yes_no__boolean, inplace=True)

    # convert the long change categories to codes
    for product_column in product_columns:
        data[product_column].replace(change_categories_to_codes, inplace=True)
        # code 'nan's as 'nr' for non-response
        data[product_column].fillna('nr', inplace=True)

    return data
```

```
In [12]: data = load_data()
```

A.4 Data validation

```
In [13]: # check what values appear in each column
         for col in data.columns:
             # ignore the datetime and free response columns
             if col not in ['StartDate', 'EndDate', 'WhyFreeResponse']:
                 print('{}: {}'.format(col, data[col].unique()))
```

```
CustomData: [ nan]
ReceivedBPreviously: [True False nan]
GradeLevel: ['Sophomore' 'Freshman' 'Junior' nan 'Senior']
ChangeChicken: ['na' 1 0 -1 -2 -3 'nr']
ChangeBeefPork: ['na' 0 -2 -1 -3 1 'nr']
ChangeFishSeafood: ['na' 0 -1 1 -2 -3 'nr']
ChangeEggs: [0 -1 -2 'na' 1 -3 'nr']
ChangeDairy: [-2 0 -1 'na' -3 1 'nr']
ChangeSomeoneElse: [True False nan]
ExaminationDuration: ['1 to 5 minutes' '10 seconds to a minute' 'More than 5 minutes'
 'Less than 10 seconds' nan]
PamphletReceived: ['better' 'compassionate']
```

```
In [14]: # Each StartDate and EndDate pair is either identical or off by a minute
         # (in a single case)
         (data.EndDate - data.StartDate).value_counts()
```

```
Out[14]: 00:00:00    488
         00:01:00      1
         dtype: int64
```

```
In [15]: # fraction of nans in each columns
         fraction_to_percent(data.isnull().mean())
```

```
Out[15]: StartDate                0.00
         EndDate                  0.00
         CustomData             100.00
         ReceivedBPreviously      2.25
         GradeLevel               6.75
         ChangeChicken            0.00
         ChangeBeefPork           0.00
         ChangeFishSeafood        0.00
         ChangeEggs               0.00
         ChangeDairy              0.00
         ChangeSomeoneElse        3.07
         ExaminationDuration      0.82
         WhyFreeResponse         93.87
         PamphletReceived         0.00
         dtype: float64
```

```
In [16]: # Who did not respond for some change in consumption?
         data[data[product_columns].eq('nr').any(axis=1)];
```

A.5 Verification of numerical claims

```
In [17]: data['AnimalsSpared'] = data.apply(animals_spared, axis=1)
         data['ConvertedVegetarian'] = data.apply(is_converted_vegetarian, axis=1)
         data['ConvertedPescatarian'] = data.apply(is_converted_pescatarian, axis=1)
```

A.5.1 Approximately 450 students at two major east coast state universities filled out a survey two to three months after receiving a leaflet.

```
In [18]: len(data)
```

```
Out[18]: 489
```

A.5.2 About one out of every 50 students who received a leaflet indicated they became vegetarian or pescatarian as a result.

```
In [19]: # One last test: no two rows are both vegetarian and pescatarian
         assert not (data.ConvertedVegetarian & data.ConvertedPescatarian).any()
```

```
In [20]: # One out of how many people were converted to either vegetarian or
         # pescatarian diet?
         one_out_of((data.ConvertedVegetarian | data.ConvertedPescatarian).mean())
```

```
Out[20]: '1 out of 69.9'
```

The source of this discrepancy remains unknown. It may likely be due to the original post counting additional respondents whose written comments in the column `WhyFreeResponse` were interpreted to mean that they or someone they knew went vegetarian or pescatarian as a result of receiving a leaflet. The answers to the more specific multiple-choice questions, however, only justify the claim for one out of every 70 students.

A.5.3 7% of students said they now ate "a lot less" chicken, a lot fewer eggs, and a lot less dairy as a result of getting the leaflet. 6% ate a lot less fish, and 12% ate a lot less red meat.

We interpret references to "red meat" in the original post to mean beef and pork.

```
In [21]: fraction_to_percent(data[product_columns].eq(-2).mean())
```

```
Out[21]: ChangeChicken         6.75
         ChangeBeefPork       11.66
         ChangeFishSeafood     6.34
         ChangeEggs            7.36
         ChangeDairy           7.36
         dtype: float64
```

A.5.4 And one out of every five students said they shared the leaflet with someone else who then began to eat less meat.

```
In [22]: one_out_of(data.ChangeSomeoneElse.mean())
```

```
Out[22]: '1 out of 5.6'
```

We attribute the discrepancy to a rounding error.

A.5.5 The data suggested that one farm animal was spared for every two leaflets distributed.

```
In [23]: data.groupby('PamphletReceived').AnimalsSpared.describe().transpose()
```

```
Out[23]: PamphletReceived     better  compassionate
         count             94.000000     395.000000
         mean               2.499778       1.368122
         std                7.093177       4.394045
         min               -9.300000      -9.497500
         25%                0.000000       0.000000
         50%                0.008333       0.000000
         75%                2.937500       0.615833
         max               30.428333      29.025000
```

While this claim is important, it is not substantially developed in the blog post. However, the claim is not inconsistent with the data and the assumptions made in the post about the numbers of animals spared by each type of diet change. As can be noted from the second row in the table above, this data and the accompanying assumptions imply that each SB leaflet spares an average of 2.50 animals and each CC leaflet spares an average of 1.37 animals.

A.5.6 Table 1

```
In [24]: def percent_change_by_pamphlet(d, change_types):

             def percent_in_change_type(g):
                 return g.isin(change_types).mean()

             return fraction_to_percent(
                 d.groupby("PamphletReceived")\
                  .apply(percent_in_change_type)[product_columns].transpose())
```

```
In [25]: # What percent of respondents who had not previously received the
         # "Compassionate Choices" booklet (so had never received either booklet)
         # stopped eating or ate a lot less of each product, grouped by which
         # pamphlet they received in the study.
         percent_change_by_pamphlet(data[~data.ReceivedBPreviously.fillna(True)],
                                    set([-2, -3]))
```

```
Out[25]: PamphletReceived   better  compassionate
         ChangeChicken       14.75           9.72
         ChangeBeefPork      19.67          18.06
         ChangeFishSeafood   14.75          11.11
         ChangeEggs          13.11           6.94
         ChangeDairy         13.11           8.33
```

We attribute the differences between the table in the original post and the table above to errors in rounding.

A.5.7 *Something Better* appeared to spare 35% more animals among this group of students.

```
In [26]: def percent_change(df):
             """
             Of two items in an iterable, calculate the percent
             change of the second item over the first
             """
             return fraction_to_percent(df[0]/df[1]-1.0)
```

```
In [27]: percent_change(data[~data.ReceivedBPreviously.fillna(True)]\
                       .groupby('PamphletReceived').AnimalsSpared.mean())
```

```
Out[27]: 36.38
```

The difference between the stated percentage of 35% in the original post and the actual percentage of 36.38% in our analysis is attributed to the following: 1. The use of rounded percentages from the table before computation of the number of animals spared in the original study. In our reanalysis, we compute final impact on animals spared without rounding. 2. An error in the spreadsheet used for the original post where the total number of animals spared by someone who stops eating pork and beef is entered as 0.275 animals, while the intended number was actually 0.625 animals (= 0.125 cows + 0.5 pigs).

A.5.8 Table 2

```
In [28]: # What percentage of respondents who previously received the
         # "Compassionate Choices" pamphlet began eating "a lot less" or
         # stopped eating each animal product?
         percent_change_by_pamphlet(data[data.ReceivedBPreviously.fillna(False)],
                                    set([-2, -3]))
```

```
Out[28]: PamphletReceived   better  compassionate
         ChangeChicken        6.90          6.15
         ChangeBeefPork      20.69         11.48
         ChangeFishSeafood   10.34          2.87
         ChangeEggs          13.79          6.15
         ChangeDairy         13.79          6.15
```

We attribute the differences between the table in the original post and the table above to errors in rounding.

A.5.9 Of the 7% who received the *Something Better* leaflet and began eating "a lot less" chicken, all 7% stopped eating chicken entirely. Of the 6% who received the *Compassionate Choices* leaflet and began eating "a lot less" chicken, 1% stopped eating chicken entirely.

```
In [29]: # What percentage receiving each pamphlet stopped eating each animal
         # product?
         percent_change_by_pamphlet(data[data.ReceivedBPreviously\
                                         .fillna(False)], set([-3]))
```

```
Out[29]: PamphletReceived   better  compassionate
         ChangeChicken        6.90          1.23
         ChangeBeefPork      10.34          2.46
         ChangeFishSeafood    0.00          0.41
         ChangeEggs           3.45          0.41
         ChangeDairy          3.45          0.82
```

A.5.10 When the overall results were translated to number of animals spared, *Something Better* appeared to spare 49% more animals.

```
In [30]: percent_change(data[data.ReceivedBPreviously.fillna(False)]\
                        .groupby('PamphletReceived').AnimalsSpared.mean())
```

```
Out[30]: 38.16
```

The difference between the stated percentage of 49% in the original post and the actual percentage of 38.16% in our analysis is attributed to the following: 1. The use of rounded percentages from the table before computation of the number of animals spared in the original study. In our reanalysis, we compute final impact on animals spared without rounding. 2. An error in the spreadsheet used for the original post where the total number of animals spared by someone who stops eating pork and beef is entered as 0.275 animals, while the intended number was actually 0.625 animals (= 0.125 cows + 0.5 pigs).

A.5.11 Table 3

```
In [31]: # We compare freshmen who received "Compassionate Choices" and had not
         # received it previously with all respondents who received "Something
         # Better" but not "Compassionate Choices" previously. These two groups
         # ostensibly had their first exposure to any pamphlet during the study.
         data_first_exposure = data[
             ~data.ReceivedBPreviously.fillna(True) &
             (data.PamphletReceived.eq('better') |
              (data.GradeLevel.eq('Freshman') &
               data.PamphletReceived.eq('compassionate')))]

         percent_change_by_pamphlet(data_first_exposure, set([-2, -3]))
```

```
Out[31]: PamphletReceived   better  compassionate
         ChangeChicken       14.75          2.78
         ChangeBeefPork      19.67         19.44
         ChangeFishSeafood   14.75         11.11
         ChangeEggs          13.11          2.78
         ChangeDairy         13.11          5.56
```

We attribute the differences between Table 3 in the original post and the table above to errors in rounding.

A.5.12 Table 4

```
In [32]: fraction_to_percent(data.groupby('PamphletReceived')\
             [['ConvertedVegetarian', 'ConvertedPescatarian']].mean().transpose())
```

```
Out[32]: PamphletReceived      better  compassionate
         ConvertedVegetarian     1.06           0.00
         ConvertedPescatarian    3.19           0.76
```

A.5.13 In 16 out of the 17 product/diet comparisons above, *Something Better* performed significantly better, often 50-100% better.

We found that *Something Better* exceeds *Compassionate Choices* in all 17 comparisons, after correcting a rounding error in Table 3. However, these comparisons are not independent and often use the same groups of respondents. A larger number of favorable but non-independent comparisons do not imply higher confidence in the result.

A.6 Assessment of inferential claims

A.6.1 Bayesian comparison of proportions of respondents reporting significantly less animal product consumption

A.6.1.1 Table 1

```
In [33]: def get_major_changes_by_pamphlet(d):
             """
```

```
        Returns a table with rows for each change category and two columns
        for each pamphlet: the number of respondents eating significantly
        less of the change category and the number of respondents.
        """
        # take only the columns of interest
        d = d[["PamphletReceived"] + product_columns].copy()

        for col in product_columns:
            d[col] = d[col].isin([-2, -3])

        return d.groupby("PamphletReceived").agg(['sum', 'count'])\
            .transpose().unstack()

    major_changes_by_pamphlet = get_major_changes_by_pamphlet(
        data[~data.ReceivedBPreviously.fillna(True)])

    major_changes_by_pamphlet
```

Out[33]:

PamphletReceived	better		compassionate	
	sum	count	sum	count
ChangeChicken	9.0	61.0	14.0	144.0
ChangeBeefPork	12.0	61.0	26.0	144.0
ChangeFishSeafood	9.0	61.0	16.0	144.0
ChangeEggs	8.0	61.0	10.0	144.0
ChangeDairy	8.0	61.0	12.0	144.0

In [34]:
```
    def make_percent_diff_bcis_by_pamphlet(d):
        """
        Accepting the output from get_major_changes_by_pamphlet, report four
        columns:

        SB_bci, CC_bci: Bayesian credible intervals for the fraction of
            respondents eating significantly less of each product using
            the Jeffreys method. We use the Jeffreys method
            for its Bayesian approach allowing easier interpretation of the
            true value given the observed data and secondarily for its good
            coverage probability compared to more conservative intervals
            such as the 'exact' Clopper-Pearson interval.

        conf_diff_bci: Percent confidence that the pamphlet with the greater
            change proportion is indeed greater.

        diff_bci: Bayesian credible interval for the difference between the
            two pamphlets in the fraction of respondents eating
            significantly less of each animal product
        """

        # wrap some R credible interval functions for convenience
        def _rmoverbci(x1, n1, x2, n2):
            """
            For two proportions x1/n1 and x2/n2, compute the 95% two-sided
            Bayesian interval of the difference between the proportions,
```

```
        (x1/n1 - x2/n2). This function uses the Bayesian MOVER-B
        method [^3].

        [^3] Laud, P. J., Equal-tailed confidence intervals for
            comparison of rates. Pharmaceutical Statistics 2017.
        """
        bci = rmoverbci(x1, n1, x2, n2, level=0.95)
        return round(bci[0], 3), round(bci[2], 3)


    def _proportion_confint(x1, n1):
        """
        For a proportion x1/n1, compute the 95% two-sided Jeffreys
        Bayesian credible interval
        """
        l, u = proportion_confint(x1, n1, method='jeffrey')
        return l.round(3), u.round(3)


    return pd.DataFrame(
        # use .tolist() to convert the pandas Series of numpy
        # int/floats to plain python, because rpy2 can't accept numpy
        # int/float types!
        {'SB_bci':  d['better'].apply(
            lambda d: _proportion_confint(*d.tolist()), axis=1),
         'CC_bci': d['compassionate'].apply(
            lambda d: _proportion_confint(*d.tolist()), axis=1),
         'diff_bci': d.apply(
            lambda d: _rmoverbci(*d.tolist()), axis=1),
         'conf_diff_bci': d.apply(
            lambda d: conf_bci_different(*d.tolist()), axis=1)
        })

make_percent_diff_bcis_by_pamphlet(major_changes_by_pamphlet)
```

Out[34]:

	CC_bci	SB_bci	conf_diff_bci	\
ChangeChicken	(0.057, 0.154)	(0.076, 0.252)	85	
ChangeBeefPork	(0.124, 0.249)	(0.112, 0.309)	61	
ChangeFishSeafood	(0.068, 0.17)	(0.076, 0.252)	77	
ChangeEggs	(0.036, 0.12)	(0.064, 0.232)	92	
ChangeDairy	(0.046, 0.137)	(0.064, 0.232)	85	

	diff_bci
ChangeChicken	(-0.041, 0.162)
ChangeBeefPork	(-0.093, 0.142)
ChangeFishSeafood	(-0.057, 0.149)
ChangeEggs	(-0.022, 0.168)
ChangeDairy	(-0.038, 0.155)

See `make_percent_diff_bcis_by_pamphlet` documentation for complete column explanations. Note that the credible intervals for SB and CC ("SB_bci" and "SB_bci", respectively) overlap

substantially. The extent of this overlap is quantified by computing Bayesian credible intervals for the difference between the proportions observed for the two leaflets ("diff_bci") and a percent confidence that one proportion is greater ("conf_diff_bci").

A.6.1.2 Table 2

```
In [35]: major_changes_by_pamphlet_previous_exposure = \
             get_major_changes_by_pamphlet(data[data.ReceivedBPreviously.fillna(False)])

         major_changes_by_pamphlet_previous_exposure
```

Out[35]:

PamphletReceived	better		compassionate	
	sum	count	sum	count
ChangeChicken	2.0	29.0	15.0	244.0
ChangeBeefPork	6.0	29.0	28.0	244.0
ChangeFishSeafood	3.0	29.0	7.0	244.0
ChangeEggs	4.0	29.0	15.0	244.0
ChangeDairy	4.0	29.0	15.0	244.0

```
In [36]: make_percent_diff_bcis_by_pamphlet(major_changes_by_pamphlet_previous_exposure)
```

Out[36]:

	CC_bci	SB_bci	conf_diff_bci	\
ChangeChicken	(0.036, 0.097)	(0.015, 0.203)	59	
ChangeBeefPork	(0.079, 0.159)	(0.091, 0.378)	91	
ChangeFishSeafood	(0.013, 0.055)	(0.03, 0.251)	96	
ChangeEggs	(0.036, 0.097)	(0.048, 0.295)	92	
ChangeDairy	(0.036, 0.097)	(0.048, 0.295)	92	

	diff_bci
ChangeChicken	(-0.057, 0.144)
ChangeBeefPork	(-0.032, 0.266)
ChangeFishSeafood	(-0.004, 0.223)
ChangeEggs	(-0.02, 0.235)
ChangeDairy	(-0.02, 0.235)

Note that, except in the case of chickens, the confidence level that SB is more effective than CC is over 90% for this group of respondents.

A.6.1.3 Table 3

```
In [37]: major_changes_by_pamphlet_first_exposure = \
             get_major_changes_by_pamphlet(data_first_exposure)

         major_changes_by_pamphlet_first_exposure
```

Out[37]:

PamphletReceived	better		compassionate	
	sum	count	sum	count
ChangeChicken	9.0	61.0	1.0	36.0
ChangeBeefPork	12.0	61.0	7.0	36.0
ChangeFishSeafood	9.0	61.0	4.0	36.0
ChangeEggs	8.0	61.0	1.0	36.0
ChangeDairy	8.0	61.0	2.0	36.0

```
In [38]: make_percent_diff_bcis_by_pamphlet(major_changes_by_pamphlet_first_exposure)
```

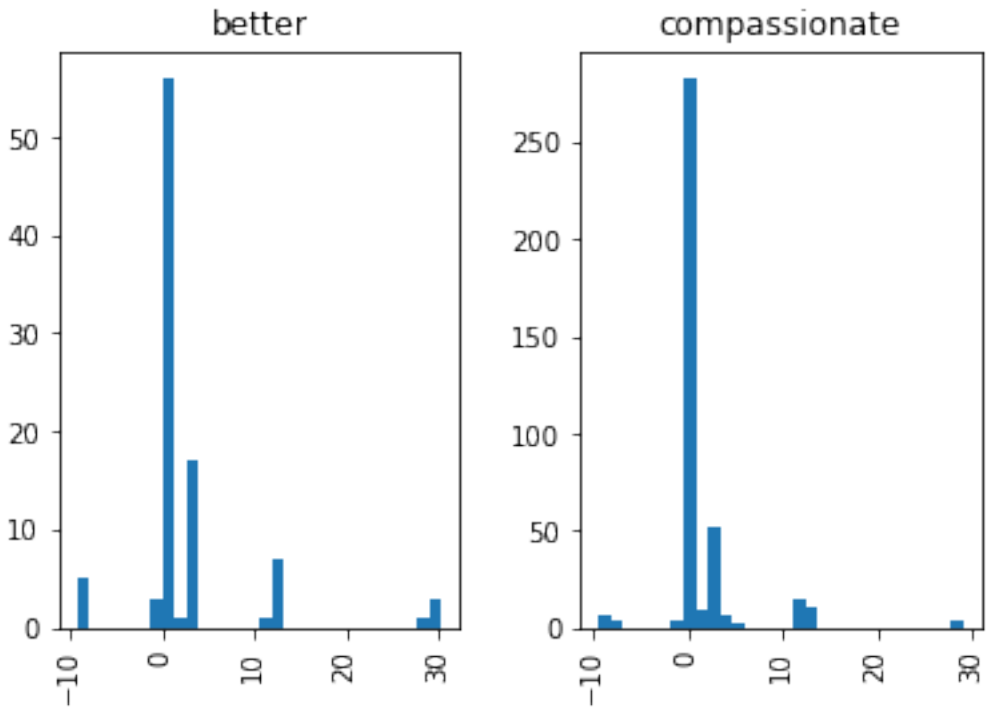
Out[38]:

	CC_bci	SB_bci	conf_diff_bci	diff_bci
ChangeChicken	(0.003, 0.123)	(0.076, 0.252)	97	(0.0, 0.224)
ChangeBeefPork	(0.091, 0.344)	(0.112, 0.309)	50	(-0.169, 0.154)
ChangeFishSeafood	(0.039, 0.243)	(0.076, 0.252)	68	(-0.113, 0.162)
ChangeEggs	(0.003, 0.123)	(0.064, 0.232)	96	(-0.013, 0.204)
ChangeDairy	(0.012, 0.166)	(0.064, 0.232)	88	(-0.054, 0.183)

Given the data, this table best supports the claim (at 97% confidence) that SB is more effective than CC for reducing the consumption of chickens, at least for the subgroup of students considered in this table. However, the observed difference in the changes in seafood and beef/pork consumption weaken a more general claim that the *Something Better* leaflet is overall more effective than the *Compassionate Choices* leaflet.

A.6.2 Are the two leaflets different in numbers of animals spared?

```
In [39]: # The distribution of the number of animals spared for each pamphlet
         data.AnimalsSpared.hist(by=data.PamphletReceived, bins=30);
```



Although the distribution of responses for both samples is not particularly normal, the t-test result is important because it tests the difference in means and we wish to maximize the mean number of animals spared. To test if the effectiveness of the two pamphlets differs, we use a two-sample, unequal variance t-test (we choose an unequal variance test partially because the two groups exhibit quite different sample variances):

```
In [40]: ttest_ind(data[data.PamphletReceived == 'better'].AnimalsSpared,
                   data[data.PamphletReceived == 'compassionate'].AnimalsSpared,
                   equal_var=False)
```

```
Out[40]: Ttest_indResult(statistic=1.4806792998449418, pvalue=0.14153665932047763)
```

Note that the p value of 0.14 indicates a lack of statistical significance.

We also run an exploratory Wilcoxon-Mann-Whitney U test, which may be more appropriate given the non-normality of the data since it does not require the assumption of normal distributions.

```
In [41]: mannwhitneyu(data[data.PamphletReceived == 'better'].AnimalsSpared,
                      data[data.PamphletReceived == 'compassionate'].AnimalsSpared,
                      alternative='two-sided')
```

```
Out[41]: MannwhitneyuResult(statistic=19903.0, pvalue=0.2492200114832871)
```

Note that the p value of 0.25 again indicates a lack of statistical significance.

A.7 Requirements

```
In [42]: # Python packages
         !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
```

```
In [43]: # R packages
         for package, version in robjects.r('installed.packages()[,c(3)]').items():
             print("{}={}".format(package, version))
```

```
assertthat=0.1
BH=1.62.0-1
colorspace=1.3-2
DescTools=0.99.21
dichromat=2.0-0
digest=0.6.10
ExactCIdiff=1.3
expm=0.999-2
ggplot2=2.2.0
gridExtra=2.2.1
gtable=0.2.0
labeling=0.3
lattice=0.20-34
lazyeval=0.2.0
magrittr=1.5
manipulate=0.98.1091
MASS=7.3-45
munsell=0.4.3
mvtnorm=1.0-6
nlme=3.1-128
plyr=1.8.4
PropCIs=0.2-5
ratesci=0.2-0
```

```
RColorBrewer=1.1-2
Rcpp=0.12.11
reshape2=1.4.2
RGraphics=2.0-14
rstudio=0.98.1091
scales=0.4.1
stringi=1.1.2
stringr=1.1.0
tibble=1.2
base=3.3.2
boot=1.3-18
class=7.3-14
cluster=2.0.5
codetools=0.2-15
compiler=3.3.2
datasets=3.3.2
foreign=0.8-67
graphics=3.3.2
grDevices=3.3.2
grid=3.3.2
KernSmooth=2.23-15
lattice=0.20-34
MASS=7.3-45
Matrix=1.2-7.1
methods=3.3.2
mgcv=1.8-15
nlme=3.1-128
nnet=7.3-12
parallel=3.3.2
rpart=4.1-10
spatial=7.3-11
splines=3.3.2
stats=3.3.2
stats4=3.3.2
survival=2.39-5
tcltk=3.3.2
tools=3.3.2
utils=3.3.2
```