stats_ch14_anova

August 6, 2024

1 Modern statistics: Intuition, Math, Python, R

1.1 Mike X Cohen (sincxpress.com)

1.1.1 https://www.amazon.com/dp/B0CQRGWGLY

Code for Chapter 14 (ANOVA)

---

2 About this code file:

2.0.1 This notebook will reproduce most of the figures in this chapter (some figures were made in Inkscape), and illustrate the statistical concepts explained in the text. The point of providing the code is not just for you to recreate the figures, but for you to modify, adapt, explore, and experiment with the code.

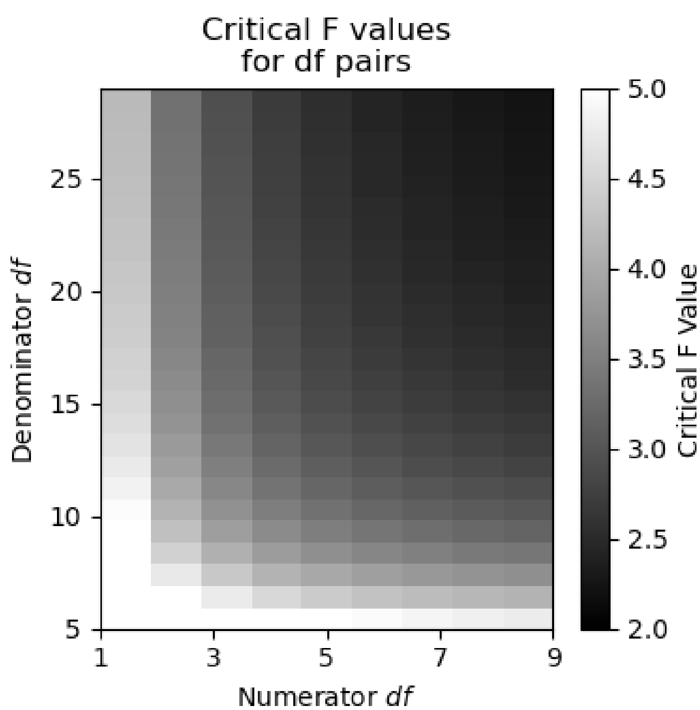
2.0.2 Solutions to all exercises are at the bottom of the notebook.

This code was written in google-colab. The notebook may require some modifications if you use a different IDE.

[2]:
```
# import libraries and set settings
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
from IPython.display import display
from matplotlib.font_manager import FontProperties # for making tables

# pingouin isn't pre-installed on colab
#!pip install pingouin
import pingouin as pg
import pandas as pd
import seaborn as sns

import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.anova import AnovaRM

# define global figure properties used for publication
import matplotlib_inline.backend_inline
```

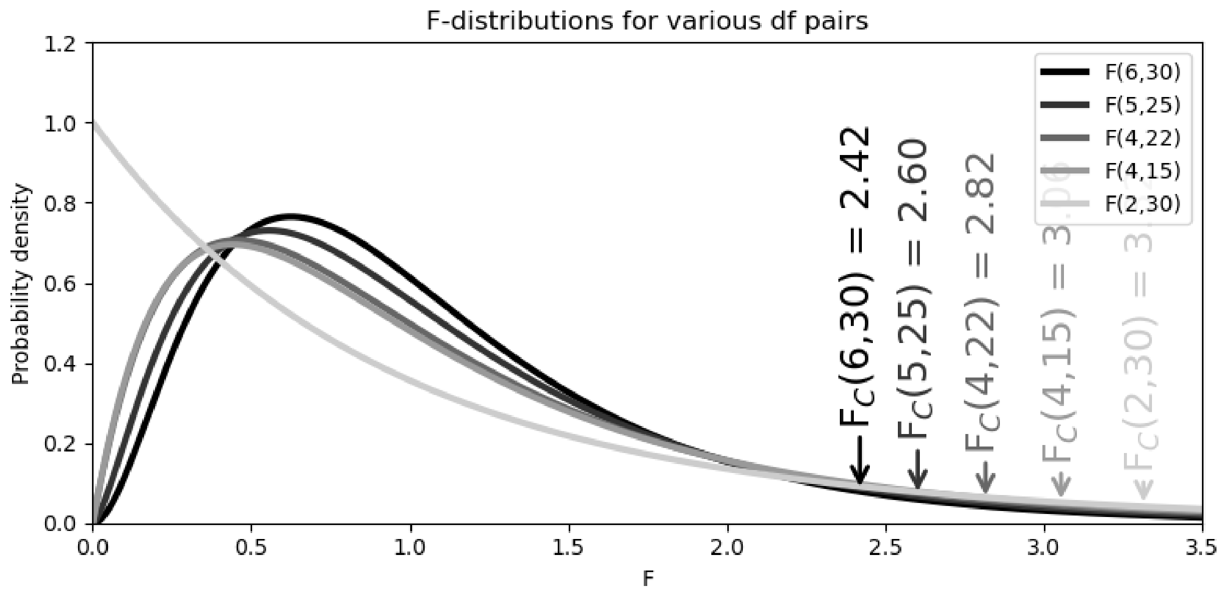
3 Figure 14.3: Critical F by df's

[5]:
```
# Define the degrees of freedom
df1_values = np.arange(1,10)
df2_values = np.arange(5,30)

# Create a 2D numpy array to store the critical F values
critFvals = np.zeros((len(df2_values),len(df1_values)))

# critical F values for each df pair
for i, df1 in enumerate(df1_values):
  for j, df2 in enumerate(df2_values):
    critFvals[j,i] = stats.f.ppf(.95, df1, df2)

# Plot the matrix as a heatmap
plt.figure(figsize=(4,4))
plt.imshow(critFvals, origin='lower', cmap='gray',␣
 ↪interpolation='nearest',aspect='auto',
        ␣
 ↪extent=[df1_values[0],df1_values[-1],df2_values[0],df2_values[-1]],vmin=2,vmax=5)
plt.colorbar(label='Critical F Value')
plt.xlabel(r'Numerator $df$')
plt.ylabel(r'Denominator $df$')
plt.xticks(df1_values[::2])
plt.title(f'Critical F values\nfor df pairs',loc='center')
plt.tight_layout()
plt.show()
```



4 Figure 14.4: F-distributions

[7]:
```
# Define the x range
x = np.linspace(0,3.5,1000)

# Define the degrees of freedom pairs
df_pairs = [(6,30), (5,25), (4,22), (4,15), (2,30)]

plt.figure(figsize=(8,4))
for i,(df1,df2) in enumerate(df_pairs):
  # F pdf
  F = stats.f.pdf(x, df1, df2)

  # color
  c = i/len(df_pairs)

  # plot the distribution
  plt.plot(x,F,linewidth=3,color=(c,c,c),label=fr'F({df1},{df2})')

  # critical F value for p=.05
  crit_f_x = stats.f.ppf(.95,df1,df2) # this is the F value
  crit_f_y = stats.f.pdf(crit_f_x,df1,df2) # this is the y-axis coordinate (prob␣
↪density)

  # Add annotation for the critical F value
  plt.annotate(text=fr'F$_C$({df1},{df2}) = {crit_f_x:.
↪2f}',color=(c,c,c),xy=(crit_f_x,crit_f_y),rotation=90,
               xytext=(crit_f_x,crit_f_y*3),fontsize=18,
               arrowprops=dict(color=(c,c,c), arrowstyle='->',linewidth=2),
               ha='center', va='bottom')
# some niceties
plt.title('F-distributions for various df pairs',loc='center')
plt.xlabel('F')
plt.xlim([0,x[-1]])
plt.ylim([0,1.2])
plt.ylabel('Probability density')
plt.legend()

plt.tight_layout()
#plt.savefig('anova-FDists.png')
plt.show()
```



5 Figure 14.5: One-way ANOVA table

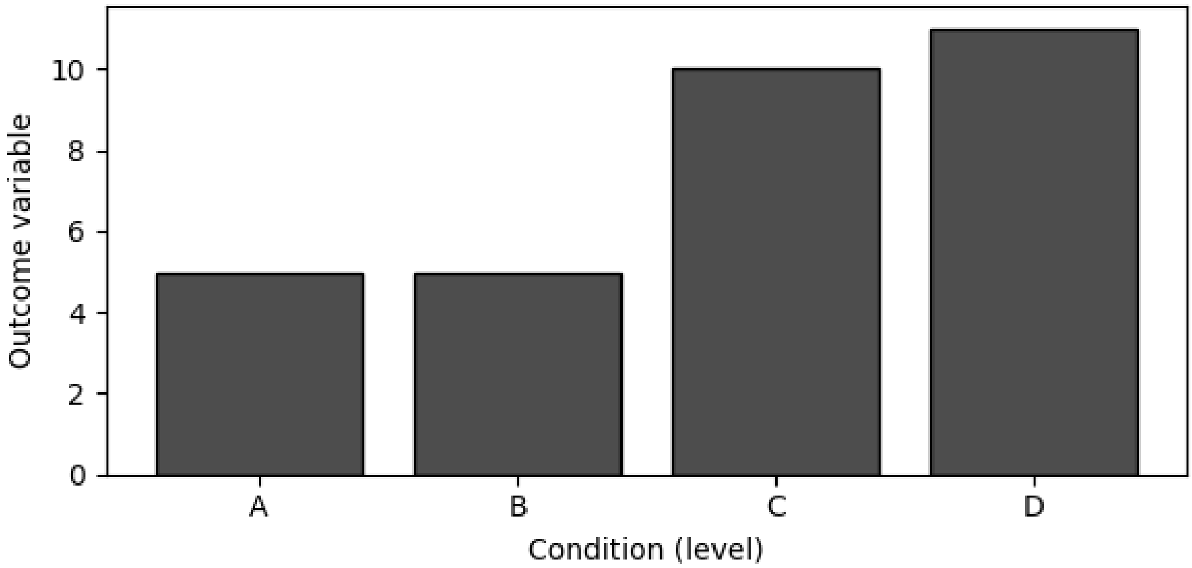
[8]:
```python
# Data
rows = ['Between', 'Within', 'Total']
columns = ['Source', 'SS', 'df', 'MS', 'F']
cell_text = [
    ['Between', r'$\sum_{j=1}^{k}n_j(\overline{x_j}-\overline{x})^2$', r'$k-1$',␣
 ↪r'$\frac{SS_{Between}}{k-1}$', r'$\frac{MS_{Between}}{MS_{Within}}$'],
    ['Within', r'$\sum_{j=1}^{k}\sum_{i=1}^{n_j}(x_{ij}-\overline{x_j})^2$',␣
 ↪r'$N-k$', r'$\frac{SS_{Within}}{N-k}$',''],
    ['Total', r'$\sum_{j=1}^{k}\sum_{i=1}^{n_j}(x_{ij}-\overline{x})^2$',␣
 ↪r'$N-1$', '', '']
]
# Create table
fig, ax = plt.subplots()
ax.axis('off')
table = ax.table(cellText   = cell_text,
                 colLabels  = columns,
                 colColours = [(.8,.8,.8)] * len(columns),
                 cellLoc    = 'center',
                 loc        = 'center')
# adjustments
for (row, col), cell in table.get_celld().items():
  cell.set_text_props(fontproperties=FontProperties(family='serif'))
  if row==0: cell.
 ↪set_text_props(fontproperties=FontProperties(weight='bold',size=16))
  if row>0 and col>2: cell.set_text_props(fontproperties=FontProperties(size=20))
```

```
table.auto_set_font_size(False)
table.scale(1.8,4)

# export
#plt.savefig('anova_ANOVAtable.png',bbox_inches='tight')
plt.show()
```

Source	SS	df	MS	F
Between	$\sum_{j=1}^{k} n_j(\overline{x_j}-\overline{x})^2$	$k-1$	$\frac{SS_{Between}}{k-1}$	$\frac{MS_{Between}}{MS_{Within}}$
Within	$\sum_{j=1}^{k}\sum_{i=1}^{n_j}(x_{ij}-\overline{x_j})^2$	$N-k$	$\frac{SS_{Within}}{N-k}$	
Total	$\sum_{j=1}^{k}\sum_{i=1}^{n_j}(x_{ij}-\overline{x})^2$	$N-1$		

6 Figure 14.6: Bar plot used for Tukey test description

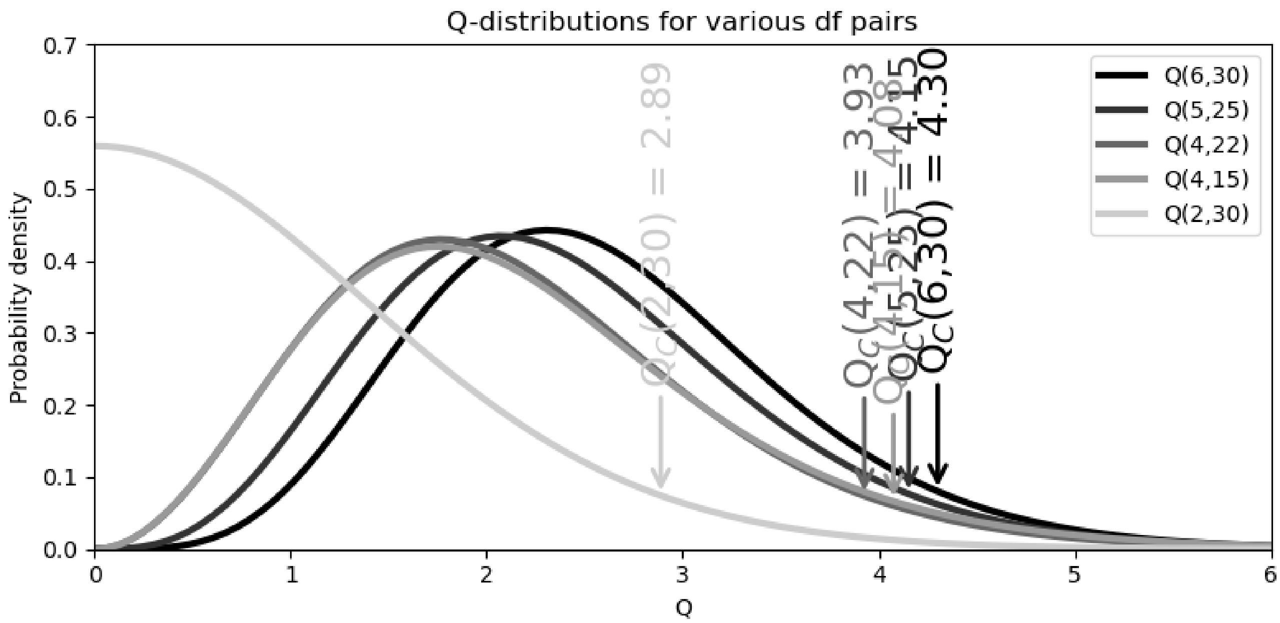
[9]:
```
y = [ 5,5,10,11]
L = ['A','B','C','D']

plt.figure(figsize=(6,3))
plt.bar(range(len(L)),y,color=(.3,.3,.3),edgecolor='k')
plt.xticks(range(len(L)),labels=L)
plt.xlabel('Condition (level)')
plt.ylabel('Outcome variable')

plt.tight_layout()
#plt.savefig('anova-4tukey.png')
plt.show()
```



7 Figure 14.7: Q-distributions with various df pairs

[11]:
```
# Define the x range
x = np.linspace(0,6,100)

# Define the degrees of freedom pairs
df_pairs = [(6,30), (5,25), (4,22), (4,15), (2,30)]

plt.figure(figsize=(8,4))
for i,(df1,df2) in enumerate(df_pairs):
  # Q pdf
  Q = stats.studentized_range.pdf(x,df1,df2)

  # color
  c = i/len(df_pairs)

  # plot the distribution
  plt.plot(x,Q,linewidth=3,color=(c,c,c),label=fr'Q({df1},{df2})')

  # critical Q value for p=.05
  crit_q_x = stats.studentized_range.ppf(.95,df1,df2) # this is the F value
  crit_q_y = stats.studentized_range.pdf(crit_q_x,df1,df2) # this is the y-axis␣
↪coordinate (prob density)

  # Add annotation for the critical Q value
  plt.annotate(text=fr'Q$_C$({df1},{df2}) = {crit_q_x:.
↪2f}',color=(c,c,c),xy=(crit_q_x,crit_q_y),rotation=90,
```

```
                 xytext=(crit_q_x,crit_q_y*3),fontsize=18,
                 arrowprops=dict(color=(c,c,c), arrowstyle='->',linewidth=2),
                 ha='center', va='bottom')
# some niceties
plt.title('Q-distributions for various df pairs',loc='center')
plt.xlabel('Q')
plt.xlim([0,x[-1]])
plt.ylim([0,.7])
plt.ylabel('Probability density')
plt.legend()

plt.tight_layout()
#plt.savefig('anova-QDists.png')
plt.show()
```



8 Figure 13.14: rmANOVA table

[12]:
```
# Data
rows = ['Between', 'Subjects', 'Within', 'Total']
columns = ['Source', 'SS', 'df', 'MS', 'F']
cell_text = [
    ['Between', r'$N\sum_{j=1}^{k} (\overline{x_j} - \overline{x})^2$',␣
 ↪r'$k-1$', r'$\frac{SS_{Between}}{k-1}$',␣
 ↪r'$\frac{MS_{Between}}{MS_{Within}}$'],
    ['Subjects', r'$\sum_{i=1}^{N}(\overline{x_i}-\overline{x})^2$', r'$N-1$',␣
 ↪r'$\frac{SS_{Subjects}}{N-1}$', r'$\frac{MS_{Subjects}}{MS_{Within}}$'],
```

```python
    ['Within', r'$SS_{T} - SS_{B} - SS_{S}$', r'$(N-1)(k-1)$', r'$\frac{SS_{Within}}{(N-1)(k-1)}$',''],
    ['Total', r'$\sum_{j=1}^{k}\sum_{i=1}^{n_j}(x_{ij}-\overline{x})^2$', r'$Nk-1$', '', '']
]
# Create table
fig, ax = plt.subplots()
ax.axis('off')
table = ax.table(cellText   = cell_text,
                 colLabels  = columns,
                 colColours = [(.8,.8,.8)] * len(columns),
                 cellLoc    = 'center',
                 loc        = 'center')
# adjustments
from matplotlib.font_manager import FontProperties
for (row, col), cell in table.get_celld().items():
  cell.set_text_props(fontproperties=FontProperties(family='serif'))
  if row==0: cell.set_text_props(fontproperties=FontProperties(weight='bold',size=16))
  if row>0 and col>2: cell.set_text_props(fontproperties=FontProperties(size=20))

table.auto_set_font_size(False)
table.scale(1.8,4)

# export
#plt.savefig('anova_rmANOVAtable.png',bbox_inches='tight')
plt.show()
```

Source	**SS**	**df**	**MS**	**F**
Between	$N\sum_{j=1}^{k}(\overline{x_j}-\overline{x})^2$	$k-1$	$\frac{SS_{Between}}{k-1}$	$\frac{MS_{Between}}{MS_{Within}}$
Subjects	$\sum_{i=1}^{N}(\overline{x_i}-\overline{x})^2$	$N-1$	$\frac{SS_{Subjects}}{N-1}$	$\frac{MS_{Subjects}}{MS_{Within}}$
Within	$SS_T-SS_B-SS_S$	$(N-1)(k-1)$	$\frac{SS_{Within}}{(N-1)(k-1)}$	
Total	$\sum_{j=1}^{k}\sum_{i=1}^{n_j}(x_{ij}-\overline{x})^2$	$Nk-1$		

9 Figures 14.15 - 14.18: Example rmANOVA (the "snacks study")

[13]:
```
data = {
    'Participant': ['P1', 'P2', 'P3', 'P4', 'P5', 'P6', 'P7', 'P8']*4,
    'Snack': ['Baseline']*8 + ['Chocolate']*8 + ['Chips']*8 + ['Ice Cream']*8,
    'Mood': [5, 7, 6, 6, 5, 8, 7, 6,  # Baseline
             6, 8, 8, 7, 8, 9, 8, 7,  # Chocolate
             5, 7, 6, 5, 4, 6, 4, 6,  # Chips
             7, 9, 7, 8, 7, 9, 8, 9]  # Ice Cream
}
df = pd.DataFrame(data)

# show the data in "long" format
df[::4]
```

[13]:

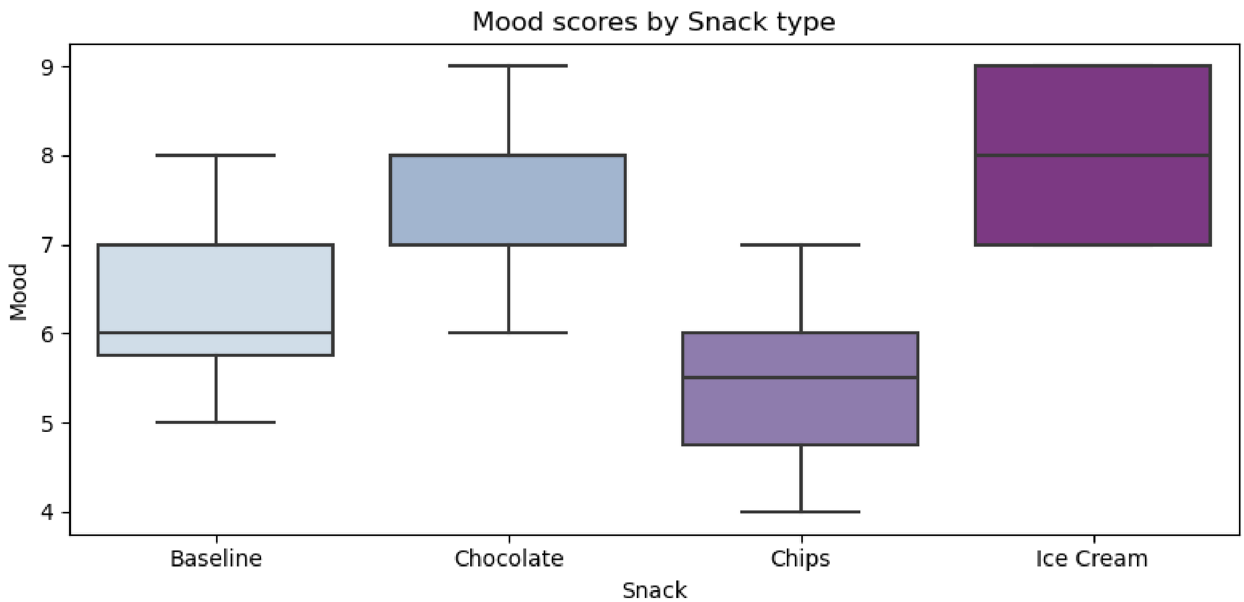
	Participant	Snack	Mood
0	P1	Baseline	5
4	P5	Baseline	5
8	P1	Chocolate	6
12	P5	Chocolate	8
16	P1	Chips	5
20	P5	Chips	4
24	P1	Ice Cream	7
28	P5	Ice Cream	7

[14]:
```
# show the data in "wide" format
df.pivot(index='Participant', columns='Snack', values='Mood')
```

[14]:

Snack	Baseline	Chips	Chocolate	Ice Cream
Participant				
P1	5	5	6	7
P2	7	7	8	9
P3	6	6	8	7
P4	6	5	7	8
P5	5	4	8	7
P6	8	6	9	9
P7	7	4	8	8
P8	6	6	7	9

[15]:
```
# Plot the data
plt.figure(figsize=(8,4))
sns.boxplot(x='Snack', y='Mood', data=df, palette='BuPu')
plt.title('Mood scores by Snack type',loc='center')

plt.tight_layout()
#plt.savefig('anova_rmSnackRes.png')
plt.show()
```



[16]:
```python
rmANOVA = pg.rm_anova(data=df, dv='Mood', within='Snack',
                      subject='Participant', detailed=True)
rmANOVA
```

[16]:

	Source	SS	DF	MS	F	p-unc	ng2	eps
0	Snack	35.625	3	11.875000	24.036145	5.456379e-07	0.5666	0.701599
1	Error	10.375	21	0.494048	NaN	NaN	NaN	NaN

[17]:
```python
# pairwise comparisons
pairwise_tests = pg.pairwise_tests(data=df, dv='Mood', within='Snack',
                                   subject='Participant',padjust='bonferroni')
print(pairwise_tests)
```

```
  Contrast          A          B  Paired  Parametric         T  dof  \
0    Snack   Baseline      Chips    True        True  2.197950  7.0
1    Snack   Baseline  Chocolate    True        True -5.227101  7.0
2    Snack   Baseline  Ice Cream    True        True -7.000000  7.0
3    Snack      Chips  Chocolate    True        True -4.965096  7.0
4    Snack      Chips  Ice Cream    True        True -8.104372  7.0
5    Snack  Chocolate  Ice Cream    True        True -1.000000  7.0

  alternative     p-unc    p-corr    p-adjust     BF10    hedges
0   two-sided  0.063924  0.383545  bonferroni    1.582  0.789415
1   two-sided  0.001216  0.007298  bonferroni   35.164 -1.330027
2   two-sided  0.000212  0.001269  bonferroni   147.28 -1.684907
3   two-sided  0.001628  0.009769  bonferroni   27.768 -2.146524
4   two-sided  0.000084  0.000503  bonferroni  317.037 -2.492972
5   two-sided  0.350617  1.000000  bonferroni      0.5 -0.384963
```

[18]:
```
# FYI, this is the code to implement a Tukey test using statsmodels.
# The Tukey test is not appropriate for repeated-measures factors,
# although the conclusions here are the same as in the previous cell.
m_comp = sm.stats.multicomp.MultiComparison(df['Mood'],df['Snack'])
tukey_result = m_comp.tukeyhsd()

print(tukey_result)
```

```
   Multiple Comparison of Means - Tukey HSD, FWER=0.05
=========================================================
  group1    group2  meandiff p-adj   lower  upper  reject
---------------------------------------------------------
 Baseline     Chips   -0.875 0.3066 -2.2217 0.4717  False
 Baseline Chocolate    1.375  0.044  0.0283 2.7217   True
 Baseline Ice Cream     1.75 0.0072  0.4033 3.0967   True
    Chips Chocolate     2.25 0.0005  0.9033 3.5967   True
    Chips Ice Cream    2.625 0.0001  1.2783 3.9717   True
Chocolate Ice Cream    0.375 0.8715 -0.9717 1.7217  False
---------------------------------------------------------
```

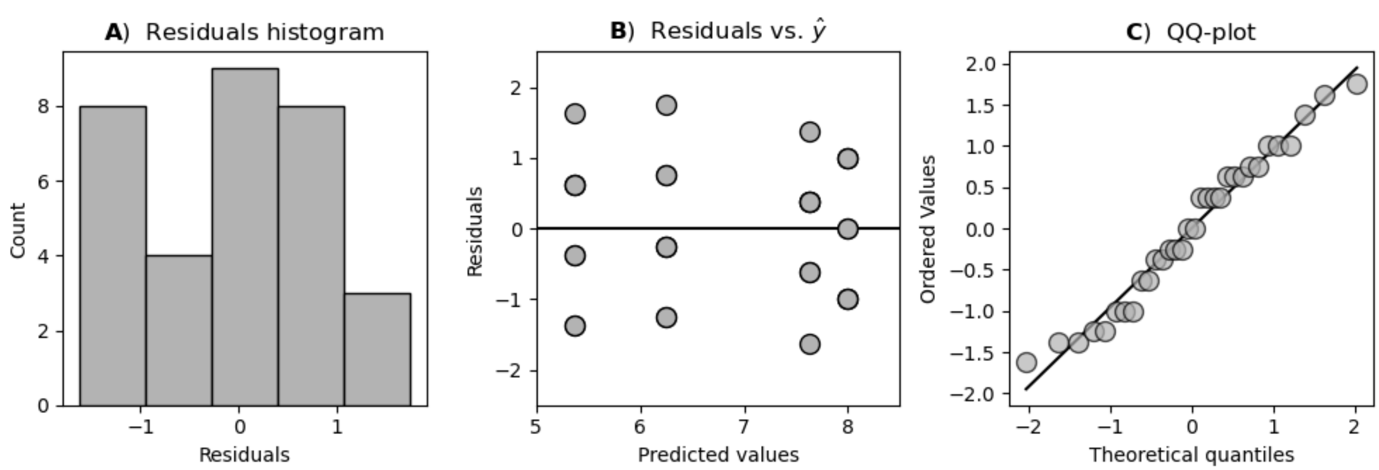
[19]:
```
# calculate the mean for each group
group_means = df.groupby('Snack')['Mood'].mean()

# column of predicted data
df['Predicted'] = df['Snack'].map(group_means)

# column of residuals
df['Residual'] = df['Mood'] - df['Predicted']

# show a few rows
df[::4]
```

[19]:

	Participant	Snack	Mood	Predicted	Residual
0	P1	Baseline	5	6.250	-1.250
4	P5	Baseline	5	6.250	-1.250
8	P1	Chocolate	6	7.625	-1.625
12	P5	Chocolate	8	7.625	0.375
16	P1	Chips	5	5.375	-0.375
20	P5	Chips	4	5.375	-1.375
24	P1	Ice Cream	7	8.000	-1.000
28	P5	Ice Cream	7	8.000	-1.000

10 Figure 14.21: Inspecting ANOVA results

[20]:
```
_,axs = plt.subplots(1,3,figsize=(10,3.5))

# histogram
axs[0].hist(df['Residual'],bins=5,facecolor=(.7,.7,.7),edgecolor='k')
axs[0].set(xlabel='Residuals',ylabel='Count')
axs[0].set_title(r'$\bf{A}$)  Residuals histogram')

# residuals by fitted values
axs[1].plot(df['Predicted'], df['Residual'],'ko',markersize=10,markerfacecolor=(.
↪7,.7,.7))
axs[1].axhline(y=0, color='k', linestyle='-', zorder=-2)
axs[1].set(xlabel='Predicted values',ylabel='Residuals',xlim=[5,8.5],ylim=[-2.
↪5,2.5])
axs[1].set_title(r'$\bf{B}$)  Residuals vs. $\hat{y}$')

# QQ plot
stats.probplot(df['Residual'],dist='norm',plot=axs[2])
axs[2].get_lines()[0].set(markerfacecolor=(.7,.7,.7),
                          markeredgecolor='k',
                          markersize=10,
                          alpha=.7)
axs[2].get_lines()[1].set(zorder=-1,color='k')
axs[2].set_title(r'$\bf{C}$)  QQ-plot')

plt.tight_layout()
#plt.savefig('anova_residuals.png')
plt.show()
```



11 Figure 14.23: 2-way ANOVA table

[21]:
```
rows = ['Between A', 'Between B', 'Interaction AB', 'Within', 'Total']
columns = ['Source', 'SS', 'df', 'MS', 'F']

cell_text = [
    ['Between A', r'$SS_A$', r'$A-1$', r'$\frac{SS_A}{df_A}$',␣
 ↪r'$\frac{MS_A}{MS_W}$'],
    ['Between B', r'$SS_B$', r'$B-1$', r'$\frac{SS_B}{df_B}$',␣
 ↪r'$\frac{MS_B}{MS_W}$'],
    ['Interaction AB', r'$SS_{AB}$', r'$(A-1)(B-1)$',␣
 ↪r'$\frac{SS_{AB}}{df_{AB}}$', r'$\frac{MS_{AB}}{MS_W}$'],
    ['Within', r'$SS_W$', r'$N-AB$', r'$\frac{SS_W}{df_W}$', ''],
    ['Total', r'$SS_T$', r'$N-1$', '', '']
]
# Create table
fig, ax = plt.subplots()
ax.axis('off')
table = ax.table(cellText   = cell_text,
                 colLabels  = columns,
                 colColours = [(.8,.8,.8)] * len(columns),
                 cellLoc    = 'center',
                 loc        = 'center')
# adjustements
from matplotlib.font_manager import FontProperties
for (row, col), cell in table.get_celld().items():
  cell.set_text_props(fontproperties=FontProperties(family='serif'))
  if row==0: cell.
 ↪set_text_props(fontproperties=FontProperties(weight='bold',size=16))
  if row>0 and col>2: cell.set_text_props(fontproperties=FontProperties(size=20))

table.auto_set_font_size(False)
table.scale(1.8,4)

# export
#plt.savefig('anova_2ANOVAtable.png',bbox_inches='tight')
plt.show()
```

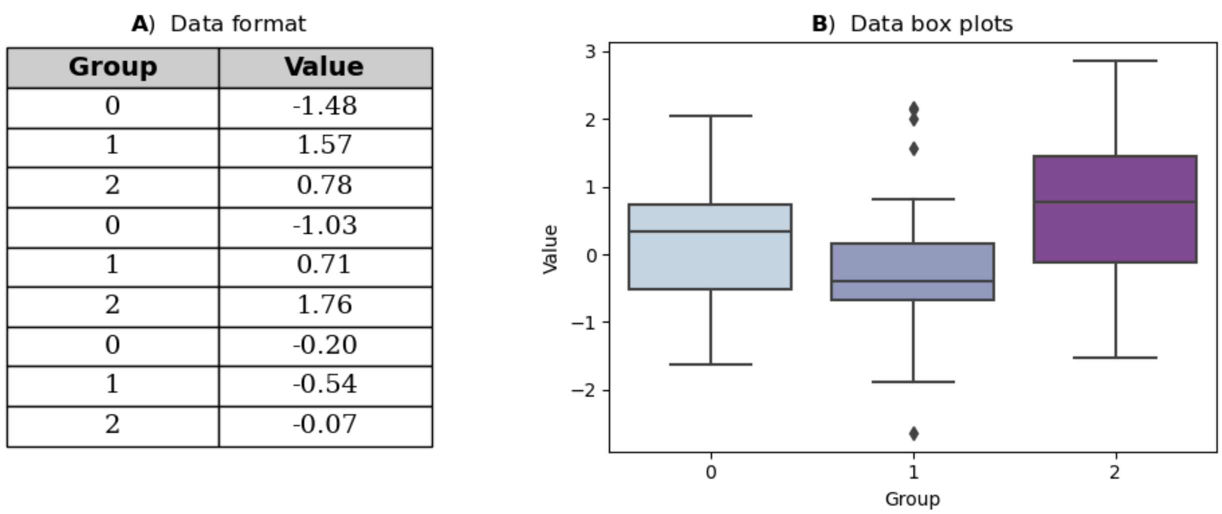
Source	SS	df	MS	F
Between A	SS_A	$A-1$	$\frac{SS_A}{df_A}$	$\frac{MS_A}{MS_W}$
Between B	SS_B	$B-1$	$\frac{SS_B}{df_B}$	$\frac{MS_B}{MS_W}$
Interaction AB	SS_{AB}	$(A-1)(B-1)$	$\frac{SS_{AB}}{df_{AB}}$	$\frac{MS_{AB}}{MS_W}$
Within	SS_W	$N-AB$	$\frac{SS_W}{df_W}$	
Total	SS_T	$N-1$		

12 Figure 14.24: Simulate data for a one-way ANOVA

[22]:
```
# group means and number of levels
level_means = [ 0,.1,.5 ]

# sample size and dataset size
nLevels = len(level_means)
samplesize = 34
nDataRows = samplesize*nLevels # total rows in the dataset

# create the column with group assignments
group_column = np.tile(np.arange(nLevels), samplesize)

# column data (initialize as zeros, then modulate by level_means)
col_data = np.zeros(nDataRows)
for i in range(nLevels):
  # row selection
  whichrows = group_column==i

  # population cell mean
  cellMean = level_means[i]

  # random data for those rows
  col_data += np.random.normal(loc=cellMean,scale=1,size=nDataRows)*whichrows

# import data into a dataframe
df = pd.DataFrame({
        'Group'  : group_column,
        'Value'  : col_data   })
```

[24]:
```
# visualization
_,axs = plt.subplots(1,2,figsize=(10,4))

### example data showing formatting

# need a copy for formatting
dfd = df.copy()
dfd['Group'] = dfd['Group'].map('{:.0f}'.format)
dfd['Value'] = dfd['Value'].map('{:.2f}'.format)

table = axs[0].table(cellText   = dfd[:9].values,
                     colLabels  = dfd.columns,
                     colColours = [(.8,.8,.8)] * len(dfd.columns),
                     cellLoc    = 'center',
                     loc        = 'center')

# adjustments
for (row, col), cell in table.get_celld().items():
  cell.set_text_props(fontproperties=FontProperties(family='serif'))
```

```
  if row==0: cell.
↪set_text_props(fontproperties=FontProperties(weight='bold',size=14))

table.scale(.7,1.8)
table.auto_set_font_size(False)
table.set_fontsize(14)
axs[0].axis('off')
axs[0].set_title(r'$\bf{A}$)  Data format')

### boxplots of data
sns.boxplot(x='Group', y='Value', data=df, palette='BuPu',ax=axs[1])
axs[1].set_title(r'$\bf{B}$)  Data box plots')

plt.tight_layout()
#plt.savefig('anova_sim1b.png')
plt.show()
```



A) Data format

Group	Value
0	-1.48
1	1.57
2	0.78
0	-1.03
1	0.71
2	1.76
0	-0.20
1	-0.54
2	-0.07

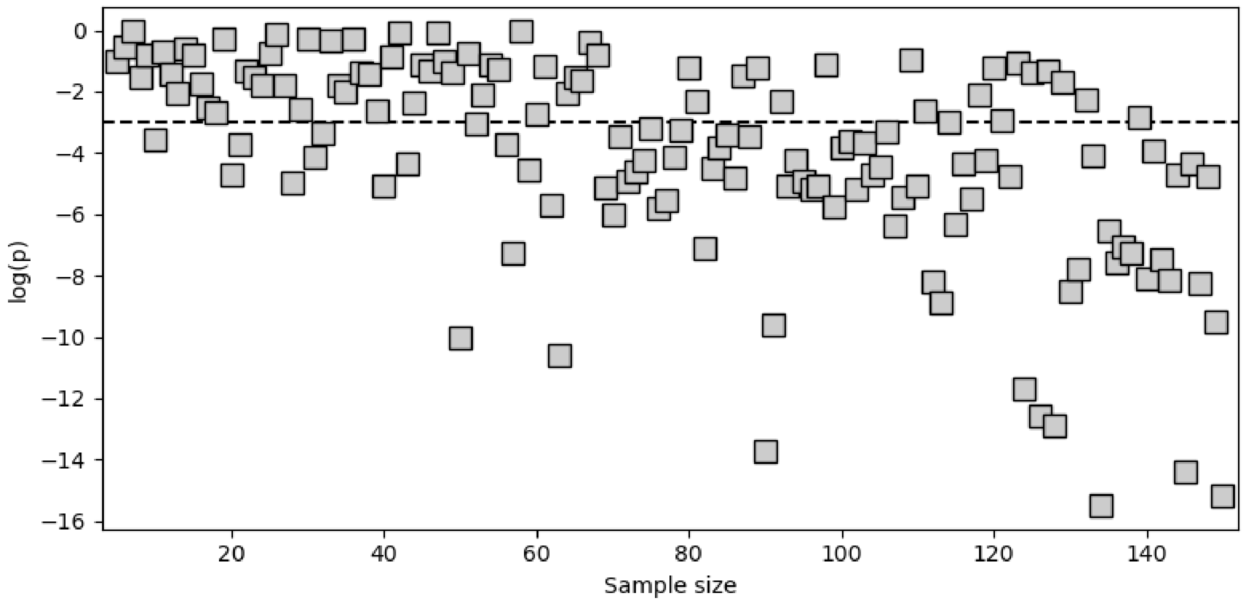
[25]:
```
# One-way ANOVA
pg.anova(dv='Value', between='Group', data=df, detailed=True)
```

[25]:
```
   Source          SS  DF        MS         F     p-unc       np2
0   Group   13.859466   2  6.929733  6.105913  0.003158  0.109807
1  Within  112.357253  99  1.134922       NaN       NaN       NaN
```

13 Figure 14.25: Parametric experiment on a one-way ANOVA

[26]:
```
samplesizes = np.arange(5,151)

# group means and number of levels
level_means = [ 0,.2,.4 ]
nLevels = len(level_means)
```

```
## run the experiment!
pvals = np.zeros(len(samplesizes))

for expi,N in enumerate(samplesizes):
  # setup
  nDataRows = N*nLevels # total rows in the dataset

  # create the column subject and group assignments
  group_column = np.tile(np.arange(nLevels), N)

  # column data (initialize as zeros, then modulate by group_mean)
  col_data = np.zeros(nDataRows)
  for i in range(nLevels):
    col_data += np.random.normal(loc=level_means[i],
                                 size=nDataRows)*(group_column==i)
  # import data into a dataframe
  df = pd.DataFrame({ 'Group':group_column, 'Value':col_data })

  # run the ANOVA and store the p-value
  anova = pg.anova(dv='Value', between='Group', data=df)

  pvals[expi] = anova['p-unc'].item()

## visualization
plt.figure(figsize=(8,4))
plt.plot(samplesizes,np.log(pvals),'ks',markersize=10,markerfacecolor=(.8,.8,.8))
plt.axhline(y=np.log(.05),color='k',linestyle='--',zorder=-1)
plt.xlabel('Sample size')
plt.ylabel('log(p)')
plt.xlim([samplesizes[0]-2,samplesizes[-1]+2])

plt.tight_layout()
#plt.savefig('anova_sim1b_exp.png')
plt.show()
```



14 Figure 14.26: Simulate data for a one-way repeated-measures ANOVA

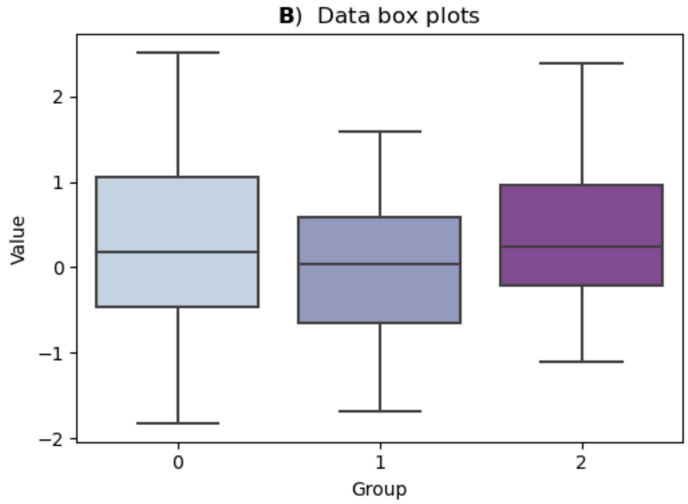
[27]:
```
# group means and number of levels
level_means = [ 0,.1,.5 ]

# sample size and dataset size
samplesize = 34
nLevels = len(level_means)
nDataRows = samplesize*nLevels # total rows in the dataset

# create the column subject and group assignments
subject_column = np.repeat(np.arange(samplesize), nLevels)
group_column = np.tile(np.arange(nLevels), samplesize)

# column data (initialize as zeros, then modulate by group_mean)
col_data = np.zeros(nDataRows)
for i in range(nLevels):
  # row selection
  whichrows = (group_column==i)

  # population cell mean
  cellMean = level_means[i]

  # random data for those rows
  col_data += np.random.normal(loc=cellMean,scale=1,size=nDataRows)*whichrows
```

```python
# import data into a dataframe
df = pd.DataFrame({
        'Subject': subject_column,
        'Group'  : group_column,
        'Value'  : col_data   })
```

[28]:
```python
# visualization
_,axs = plt.subplots(1,2,figsize=(10,4))

### example data showing formatting

# need a copy for formatting
dfd = df.copy()
dfd['Subject'] = dfd['Subject'].map('{:.0f}'.format)
dfd['Group'] = dfd['Group'].map('{:.0f}'.format)
dfd['Value'] = dfd['Value'].map('{:.2f}'.format)

table = axs[0].table(cellText   = dfd[:9].values,
                     colLabels  = dfd.columns,
                     colColours = [(.8,.8,.8)] * len(dfd.columns),
                     cellLoc    = 'center',
                     loc        = 'center')

# adjustments
for (row, col), cell in table.get_celld().items():
  cell.set_text_props(fontproperties=FontProperties(family='serif'))
  if row==0: cell.
 ↪set_text_props(fontproperties=FontProperties(weight='bold',size=14))

table.scale(.7,1.8)
table.auto_set_font_size(False)
table.set_fontsize(14)
axs[0].axis('off')

axs[0].set_title(r'$\bf{A}$)  Data format')

### boxplots of data
sns.boxplot(x='Group', y='Value', data=df, palette='BuPu',ax=axs[1])
axs[1].set_title(r'$\bf{B}$)  Data box plots')

plt.tight_layout()
#plt.savefig('anova_sim1r.png')
plt.show()
```

A) Data format

Subject	Group	Value
0	0	2.21
0	1	-0.02
0	2	2.38
1	0	-0.08
1	1	-0.77
1	2	0.19
2	0	-0.24
2	1	-0.60
2	2	-0.20



[29]:
```
# One-way repeated measures ANOVA
pg.rm_anova(dv='Value', within='Group', subject='Subject', data=df,␣
↪detailed=True)
```

[29]:
```
  Source         SS  DF        MS         F     p-unc       ng2      eps
0  Group   3.333521   2  1.666760  2.326228  0.105621  0.035569  0.90896
1  Error  47.289518  66  0.716508       NaN       NaN       NaN      NaN
```

15 Figure 14.27: Simulate data for a two-way between-subjects ANOVA

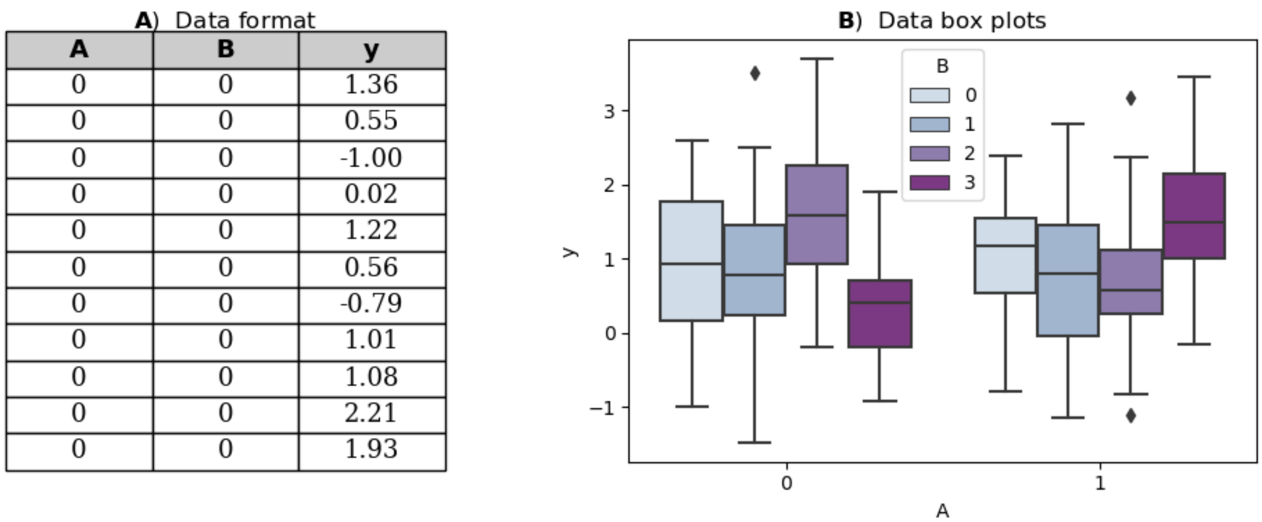
[30]:
```
# subjects per group
n = 30

# population cell means
# "factor A" is the number of rows, "factor B" is the number of columns
group_means = [ [ 1,1,1.5,.5 ],
                [ 1,1,.5,1.5 ] ]

factA,factB = np.shape(group_means)
nDataRows = n*factA*factB # total rows in the dataset

# create the column subject and group assignments
colA = np.repeat(np.arange(factA), n*factB)
colB = np.repeat(np.tile(np.arange(factB), factA), n)

# column data (initialize as zeros, then modulate by group_mean)
col_data = np.zeros(nDataRows)
for a in range(factA):
  for b in range(factB):
    # row selection
```

```
    whichrows = (colA==a) & (colB==b)

    # population cell mean
    cellMean = group_means[a][b]

    # random data for those rows
    col_data += np.random.normal(loc=cellMean,scale=1,size=nDataRows)*whichrows

# Create dataframe
df = pd.DataFrame({
      'A' : colA,
      'B' : colB,
      'y' : col_data
})
# print dataframe
#print(df.to_string())
```

[31]:

```
# visualization
_,axs = plt.subplots(1,2,figsize=(10,4))

### example data showing formatting

# need a copy for formatting
dfd = df.copy()
dfd['A'] = dfd['A'].map('{:.0f}'.format)
dfd['B'] = dfd['B'].map('{:.0f}'.format)
dfd['y'] = dfd['y'].map('{:.2f}'.format)

table = axs[0].table(cellText  = dfd[:11].values,
                    colLabels  = dfd.columns,
                    colColours = [(.8,.8,.8)] * len(dfd.columns),
                    cellLoc    = 'center',
                    loc        = 'center')

# adjustments
for (row, col), cell in table.get_celld().items():
  cell.set_text_props(fontproperties=FontProperties(family='serif'))
  if row==0: cell.
 ↪set_text_props(fontproperties=FontProperties(weight='bold',size=14))

table.scale(.7,1.6)
table.auto_set_font_size(False)
table.set_fontsize(13)
axs[0].axis('off')
axs[0].set_title(r'$\bf{A}$)  Data format')

### boxplots of data
```

```
sns.boxplot(x='A', y='y', hue='B', data=df, palette='BuPu',ax=axs[1])
axs[1].set_title(r'$\bf{B}$)  Data box plots')

plt.tight_layout()
#plt.savefig('anova_sim2b.png')
plt.show()
```



[32]:
```
# two-way ANOVA
print(pg.anova(data=df, dv='y', between=['A','B'], detailed=True))
```

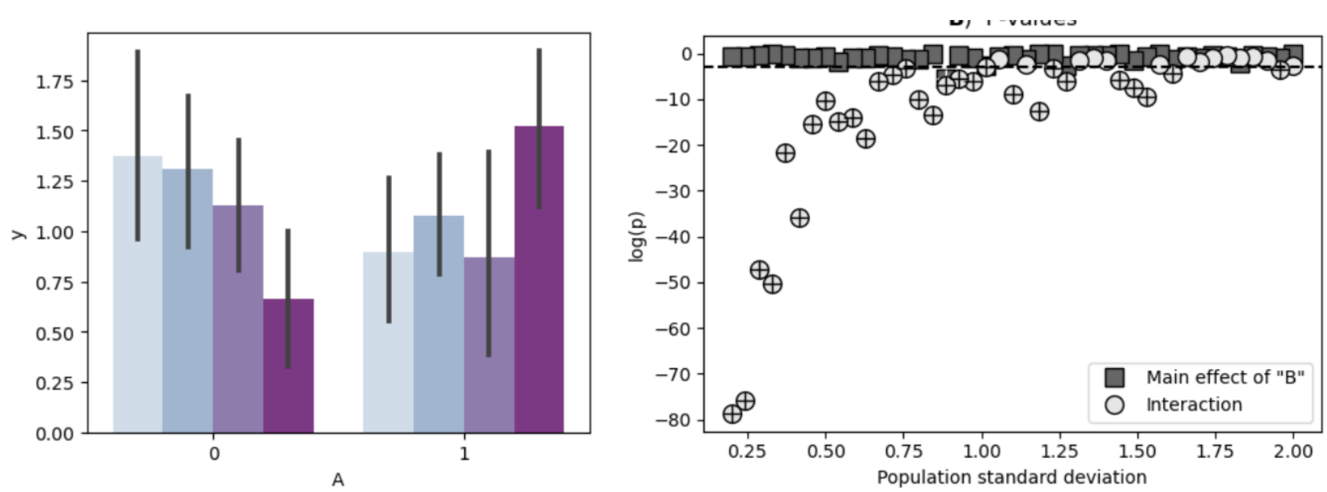
```
     Source          SS   DF         MS          F         p-unc       np2
0         A    0.200560    1   0.200560   0.240773  6.241118e-01  0.001037
1         B    4.178691    3   1.392897   1.672179  1.737309e-01  0.021165
2     A * B   34.988044    3  11.662681  14.001103  2.021587e-08  0.153295
3  Residual  193.252060  232   0.832983        NaN           NaN       NaN
```

16 Figure 14.28: Experiment: Interaction by standard deviation

[37]:
```
stdevs = np.linspace(2,.2,43)

# subjects per group
n = 30

# population cell means
# "factor A" is the number of rows, "factor B" is the number of columns
group_means = [ [ 1,1,1.3,.7 ],
                [ 1,1,.7,1.3 ] ]

factA,factB = np.shape(group_means)
nDataRows = n*factA*factB # total rows in the dataset
```

```
# create the column subject and group assignments
colA = np.repeat(np.arange(factA), n*factB)
colB = np.repeat(np.tile(np.arange(factB), factA), n)

### run the experiment
intpvals = np.zeros((len(stdevs),2))

for expi,std in enumerate(stdevs):
  # column data (initialize as zeros, then modulate by level_mean)
  col_data = np.zeros(nDataRows)
  for a in range(factA):
    for b in range(factB):
      whichrows = (colA==a) & (colB==b)
      cellMean = group_means[a][b]
      col_data += np.random.normal(loc=cellMean,scale=std, # modulate the␣
↪standard deviation
                                   size=nDataRows)*whichrows
  # Create dataframe
  df = pd.DataFrame({
        'A' : colA,
        'B' : colB,
        'y' : col_data
  })

  # store interaction p-value ("[2]" b/c the interaction term is the 3rd row of␣
↪the table
  intpvals[expi,:] = pg.anova(data=df,dv='y',between=['A','B'])['p-unc'][1:3]

  if expi==len(stdevs)//2: df2plot=df.copy()

## visualization
_,axs = plt.subplots(1,2,figsize=(11,4))

# boxplots
#sns.barplot(x='A', y='y', hue='B', data=df2plot, palette='BuPu',ax=axs[0])
#axs[0].set_title(fr'$\bf{{A}}$)  Bar plot of data (std={stdevs[len(stdevs)//2]:.
↪2f})')

# plot the p-values with a + for p<.05
axs[1].plot(stdevs,np.log(intpvals[:,0]),'ks',markersize=10,markerfacecolor=(.4,.
↪4,.4),label='Main effect of "B"')
axs[1].plot(stdevs[intpvals[:,0]<.05],np.log(intpvals[intpvals[:,0]<.
↪05,0]),'w+',markersize=10)
axs[1].plot(stdevs,np.log(intpvals[:,1]),'ko',markersize=10,markerfacecolor=(.9,.
↪9,.9),label='Interaction')
```

```
axs[1].plot(stdevs[intpvals[:,1]<.05],np.log(intpvals[intpvals[:,1]<.
↪05,1]),'k+',markersize=10)

# some other adjustments
axs[1].axhline(y=np.log(.05),color='k',linestyle='--')
axs[1].set(xlabel='Population standard deviation',ylabel='log(p)')
axs[1].legend()
axs[1].set_title(r'$\bf{B}$)  P-values')

plt.tight_layout()
#plt.savefig('anova_sim2b_std.png')
plt.show()
```



17 Figure 14.29: Two-way mixed-effects ANOVA

[38]:
```
# subjects per group
n = 30

# population cell means
# "factor A" is the number of rows, "factor B" is the number of columns
# Factor B is repeated-measures; Factor A is between-subjects
group_means = [ [1.1,1.2,1.3],
                [2,2.2,2.5] ]

factA,factB = np.shape(group_means)
nDataRows = n*factA*factB # total rows in the dataset

# create the column subject and group assignments
colA = np.repeat(np.arange(factA), n*factB)#,np.repeat(np.arange(factA), n*factB)
colB = np.tile(np.arange(factB), n*factA)#,np.repeat(np.tile(np.arange(factB),␣
↪factA), n)
colS = np.floor(np.arange(nDataRows)/factB)
```

```python
# column data
col_data = np.zeros(nDataRows)
for a in range(factA):
  for b in range(factB):
    # row selection
    whichrows = (colA==a) & (colB==b)

    # population cell mean
    cellMean = group_means[a][b]

    # random data for those rows
    col_data += np.random.normal(loc=cellMean,scale=1,size=nDataRows)*whichrows

# Create data
df = pd.DataFrame({
          'A' : colA, # between-subjects levels
          'B' : colB, # within-subjects level
         'ID' : colS, # subject ID (to know which data values are repeated)
          'y' : col_data
})
# print dataframe
print(df.to_string())
```

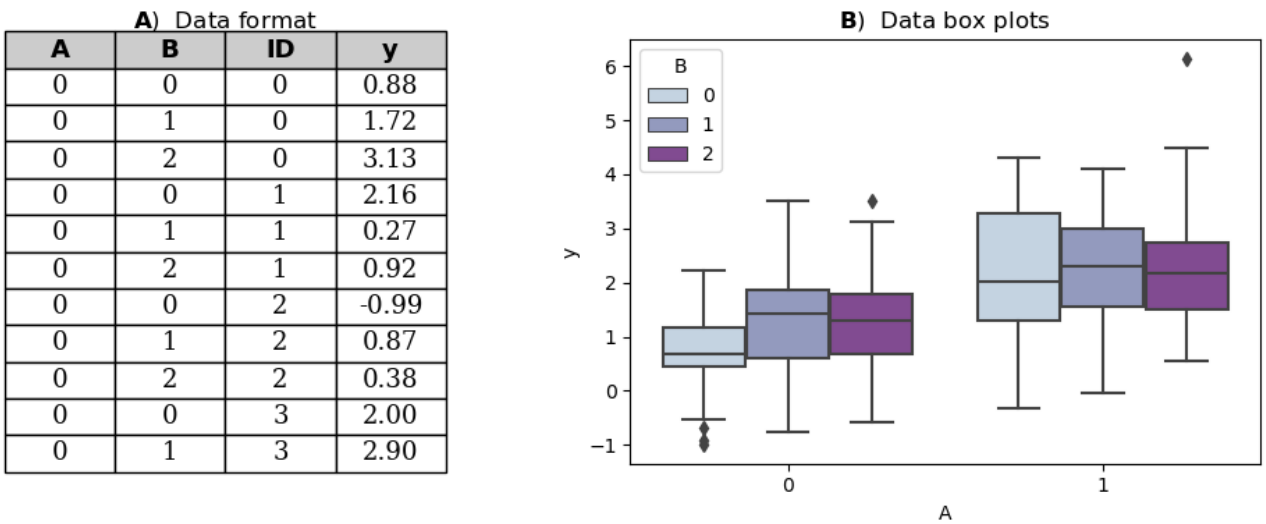
```
     A  B    ID         y
0    0  0   0.0  0.879740
1    0  1   0.0  1.716846
2    0  2   0.0  3.130247
3    0  0   1.0  2.164964
4    0  1   1.0  0.268591
5    0  2   1.0  0.919037
6    0  0   2.0 -0.987720
7    0  1   2.0  0.873084
8    0  2   2.0  0.377791
9    0  0   3.0  1.998192
.............
.............

170  1  2  56.0  1.474859
171  1  0  57.0  3.912927
172  1  1  57.0  3.013809
173  1  2  57.0  0.794705
174  1  0  58.0 -0.314862
175  1  1  58.0  3.737599
176  1  2  58.0  2.981355
177  1  0  59.0  1.297371
178  1  1  59.0  2.515145
179  1  2  59.0  0.893591
```

[39]:
```
# Run the mixed-design ANOVA
pg.mixed_anova(data=df, dv='y', between='A', within='B', subject='ID')
```

[39]:
```
        Source         SS  DF1  DF2         MS          F         p-unc  \
0            A  56.142958    1   58  56.142958  50.365783  2.015070e-09
1            B   3.865748    2  116   1.932874   1.780759  1.730857e-01
2  Interaction   3.389224    2  116   1.694612   1.561248  2.142515e-01

        np2       eps
0  0.464776       NaN
1  0.029788  0.995453
2  0.026212       NaN
```

[40]:
```
# visualization
_,axs = plt.subplots(1,2,figsize=(10,4))

### example data showing formatting

# need a copy for formatting
dfd = df.copy()
dfd['A']  = dfd['A'].map('{:.0f}'.format)
dfd['B']  = dfd['B'].map('{:.0f}'.format)
dfd['ID'] = dfd['ID'].map('{:.0f}'.format)
dfd['y']  = dfd['y'].map('{:.2f}'.format)

table = axs[0].table(cellText   = dfd[:11].values,
                     colLabels  = dfd.columns,
                     colColours = [(.8,.8,.8)] * len(dfd.columns),
                     cellLoc    = 'center',
                     loc        = 'center')
# adjustments
for (row, col), cell in table.get_celld().items():
  cell.set_text_props(fontproperties=FontProperties(family='serif'))
  if row==0: cell.
↪set_text_props(fontproperties=FontProperties(weight='bold',size=14))

table.scale(.7,1.6)
table.auto_set_font_size(False)
table.set_fontsize(13)
axs[0].axis('off')
axs[0].set_title(r'$\bf{A}$)  Data format')

### boxplots of data
sns.boxplot(x='A', y='y', hue='B', data=df, palette='BuPu',ax=axs[1])
axs[1].set_title(r'$\bf{B}$)  Data box plots')

plt.tight_layout()
```

```
#plt.savefig('anova_sim2w.png')
plt.show()
```



A) Data format

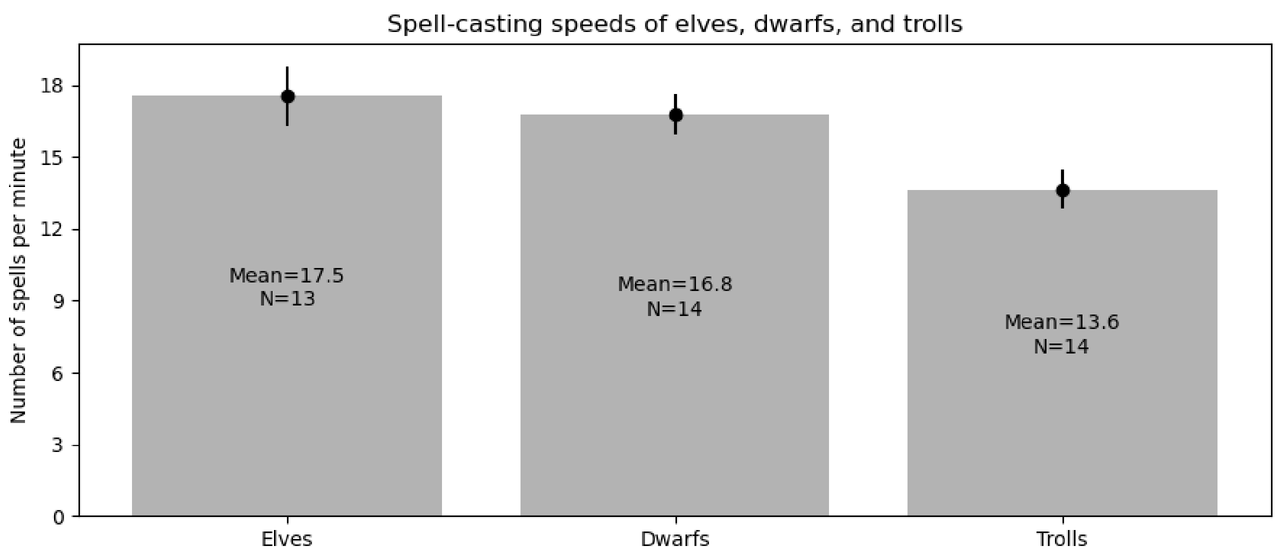
A	B	ID	y
0	0	0	0.88
0	1	0	1.72
0	2	0	3.13
0	0	1	2.16
0	1	1	0.27
0	2	1	0.92
0	0	2	-0.99
0	1	2	0.87
0	2	2	0.38
0	0	3	2.00
0	1	3	2.90

18 Exercise 1

[41]:
```
### the raw data
elves  = np.array([17, 20, 16, 22, 20, 12, 15, 23,  9, 22, 21, 19, 12    ])
dwarfs = np.array([15, 14, 15, 25, 19, 16, 20, 18, 18, 15, 18, 13, 14, 15])
trolls = np.array([14, 16, 11, 17, 12, 13, 10, 12, 10, 18, 13, 14, 11, 20])

### descriptive statistics

# sample sizes
Nelves  = len(elves)
Ndwarfs = len(dwarfs)
Ntrolls = len(trolls)

# means
mean_elves  = np.mean(elves)
mean_dwarfs = np.mean(dwarfs)
mean_trolls = np.mean(trolls)

# standard errors
sem_elves  = np.std(elves, ddof=1) / np.sqrt(Nelves)
sem_dwarfs = np.std(dwarfs,ddof=1) / np.sqrt(Ndwarfs)
sem_trolls = np.std(trolls,ddof=1) / np.sqrt(Ntrolls)
```

[43]:
```
# create an error bar plot
plt.figure(figsize=(9,4))

# the bars
```

```
plt.bar(range(3),[mean_elves,mean_dwarfs,mean_trolls],color=(.7,.7,.7))
plt.errorbar(range(3),[mean_elves,mean_dwarfs,mean_trolls],
             yerr=[sem_elves,sem_dwarfs,sem_trolls],fmt='ko')
# text in bars
plt.text(0,mean_elves/2, f'Mean={mean_elves:.1f}\nN={Nelves}',ha='center')
plt.text(1,mean_dwarfs/2,f'Mean={mean_dwarfs:.1f}\nN={Ndwarfs}',ha='center')
plt.text(2,mean_trolls/2,f'Mean={mean_trolls:.1f}\nN={Ntrolls}',ha='center')

plt.xticks(range(3),['Elves', 'Dwarfs', 'Trolls'])
plt.yticks(np.arange(19,step=3))
plt.ylabel('Number of spells per minute')
plt.title('Spell-casting speeds of elves, dwarfs, and trolls',loc='center')

plt.tight_layout()
#plt.savefig('anova_magicalMeans.png')
plt.show()
```



[44]:
```
# Stack the data into a single array for convenience
all_data = np.hstack((elves,dwarfs,trolls))

# Calculate the overall mean
total_mean = np.mean(all_data)

# Calculate SS_Between
ss_between = 0
for group in [elves,dwarfs,trolls]:
  ss_between += len(group) * (group.mean() - total_mean)**2

# Could also use list comprehension, but I think a loop is more readable.
```

```
#ss_between = np.sum([len(group) * (group.mean() - overall_mean)**2 for group in␣
 ↪[elves,dwarfs,trolls]])

# Calculate SS_Within
ss_within = np.sum( (elves  - elves.mean())**2  ) + \
            np.sum( (dwarfs - dwarfs.mean())**2 ) + \
            np.sum( (trolls - trolls.mean())**2 )

# Calculate SS Total
ss_total = ss_between + ss_within

# Calculate degrees of freedom for between, within, and total
df_between = 3 - 1  # number of groups minus 1
df_within = len(all_data) - 3  # number of observations minus number of groups
df_total = len(all_data) - 1  # number of observations minus 1

# Calculate MS_Between and MS_Within
ms_between = ss_between / df_between
ms_within = ss_within / df_within

# Calculate F statistic and associated p-value
f_stat = ms_between / ms_within
p_value = 1 - stats.f.cdf(f_stat, df_between, df_within)

# Print out the ANOVA table
print('Source\t|    SS\t\tdf\t  MS\t F\tp-value')
print('-'*56)
print(f'Between\t| {ss_between:6.2f}\t {df_between}\t{ms_between:.2f}\t{f_stat:.
 ↪2f}\t{p_value:.4f}')
print(f'Within\t| {ss_within:6.2f}\t{df_within}\t{ms_within:.2f}')
print(f'Total\t| {ss_total:6.2f}\t{df_total}')
```

```
Source  |    SS          df        MS      F       p-value
--------------------------------------------------------
Between | 117.10          2      58.55    4.51     0.0174
Within  | 492.80         38      12.97
Total   | 609.90         40
```

[45]:
```
# effect sizes
eta2 = ss_between / ss_total
omega2 = (ss_between - df_between*ms_within) / (ss_total+ms_within)

print(f'eta^2   = {eta2:.3f}')
print(f'omega^2 = {omega2:.3f}')
```

```
eta^2   = 0.192
omega^2 = 0.146
```

19 Exercise 2

[46]:
```
# Combine the data into one numpy array
data = np.concatenate([elves,dwarfs,trolls])

# Create group labels
group_labels = ['Elves']*Nelves + ['dwarfs']*Ndwarfs + ['trolls']*Ntrolls

# Create a DataFrame from the data
df = pd.DataFrame({'Spells':data, 'Creature':group_labels})

# print the dataframe
df[::6]
```

[46]:

	Spells	Creature
0	17	Elves
6	15	Elves
12	12	Elves
18	16	dwarfs
24	13	dwarfs
30	17	trolls
36	18	trolls

[47]:
```
# Perform the one-way ANOVA
result = pg.anova(data=df, detailed=True,
                  dv='Spells', between='Creature')
result
```

[47]:

	Source	SS	DF	MS	F	p-unc	np2
0	Creature	117.100241	2	58.550121	4.514802	0.017411	0.191998
1	Within	492.802198	38	12.968479	NaN	NaN	NaN

[48]:
```
# Compare with detailed=False
result = pg.anova(data=df, dv='Spells', between='Creature', detailed=False)
print(result)
```

```
     Source  ddof1  ddof2         F     p-unc       np2
0  Creature      2     38  4.514802  0.017411  0.191998
```

[49]:
```
# all pairwise comparisons using Tukey method
df.pairwise_tukey(dv='Spells', between='Creature').round(3)
```

[49]:

	A	B	mean(A)	mean(B)	diff	se	T	p-tukey	hedges
0	Elves	dwarfs	17.538	16.786	0.753	1.387	0.543	0.851	0.190
1	Elves	trolls	17.538	13.643	3.896	1.387	2.809	0.021	0.992
2	dwarfs	trolls	16.786	13.643	3.143	1.361	2.309	0.067	0.977

[50]:

```
## FYI, corresponding statsmodels code (not part of this exercise):

# create and define the model
model = ols('Spells ~ C(Creature)', data=df).fit()

# Performing ANOVA
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

[50]:

```
                 sum_sq    df         F    PR(>F)
C(Creature)  117.100241   2.0  4.514802  0.017411
Residual     492.802198  38.0       NaN       NaN
```

20 Exercise 3

[51]:

```
## data parameters

# group means
mean1 = 4
mean2 = 6

# samples per group
N1 = 30
N2 = 35

## now to simulate the data
data1 = np.random.normal(mean1,2,size=N1)
data2 = np.random.normal(mean2,2,size=N2)

datacolumn = np.hstack((data1,data2))

# group labels
groups = ['1']*N1 + ['2']*N2

# convert to a pandas dataframe
df = pd.DataFrame({'TheData':datacolumn,'Group':groups})
df
```

[51]:

```
     TheData Group
0   7.267539     1
1   4.875795     1
2   3.200373     1
..       ...   ...
63  6.535049     2
64  5.765951     2

[65 rows x 2 columns]
```

[52]:
```python
# run the ANOVA and t-test
anova = pg.anova(data=df,dv='TheData',between='Group')
ttest = stats.ttest_ind( df['TheData'][df['Group']=='1'],
                         df['TheData'][df['Group']=='2'] )
```

[53]:
```python
# compare against t-test
print(f"ANOVA: F{anova['ddof1'].item(),anova['ddof2'].item()} = {anova['F'].
 ↪item():.3f}, p = {anova['p-unc'].item():.3f}")
print(f'\nT-test: t({N1+N2-2}) = {ttest.statistic:.2f}, p = {ttest.pvalue:.3f}')
print(f'\nt^2 = {ttest.statistic**2:.3f}')
```

```
ANOVA: F(1, 63) = 11.353, p = 0.001

T-test: t(63) = -3.37, p = 0.001

t^2 = 11.353
```

21 Exercise 4

[54]:
```python
## data parameters

# sample size
N = 20

## simulate the data
data = np.random.normal(0,1,size=3*N)

# replace the final two data points with outliers (fixed to 10)
data[-2:] = 10

# group labels
groups = ['1']*N + ['2']*N + ['3']*N

# convert to a pandas dataframe
df = pd.DataFrame({'TheData':data,'Group':groups})

# run an ANOVA
pg.anova(data=df,dv='TheData',between='Group')
```

[54]:

	Source	ddof1	ddof2	F	p-unc	np2
0	Group	2	57	1.728323	0.186757	0.057176

[55]:
```python
## data parameters

# sample size
N = 50
nOutliers = 3
```

```
# group labels
groups = ['1']*N + ['2']*N + ['3']*N

# experiment params
isSig = 0  # counter
nTests = 300 # number of tests to simulate

# now for the experiment!
for i in range(nTests):
  ##simulate the data
  data = np.random.normal(0,1,size=3*N)
  data[-nOutliers:] = np.random.normal(10,1,size=nOutliers)
  # run an ANOVA
  df = pd.DataFrame({'TheData':data,'Group':groups})
  anova = pg.anova(data=df,dv='TheData',between='Group')

  # count if significant
  isSig += anova['p-unc'].item()<.05

# print the results
print(f'{isSig} of {nTests} tests ({isSig*100/nTests:.2f}%) had p<.05 with N={N}␣
 ↪and {nOutliers} outliers in group 3.')
```

```
73 of 300 tests (24.33%) had p<.05 with N=50 and 3 outliers in group 3.
```

22 Exercise 5

[ ]:
```
# Here is one possible way to do it:
# 10 factors, each with only 1 sample, and one additional group with 20 samples.

# Numerator (between-group) df: (number of groups - 1) = (10+1 - 1) = 10
# Denominator (within-group) df: (total number of observations - number of␣
 ↪groups) = (10 + 20 - 11) = 19
# So in this contrived example, the numerator df (10) is smaller than the␣
 ↪denominator df (19).
```

23 Exercise 6

[56]:
```
## data parameters
N = 10000

## simulate the data
data1 = np.random.normal(0,1,size=N)
data2 = np.random.normal(.1,1,size=N)
data  = np.concatenate((data1,data2),axis=0)
```

```python

# group labels
groups = ['1']*N + ['2']*N

# convert to a pandas dataframe
df = pd.DataFrame({'TheData':data,'Group':groups})

# run an ANOVA
pg.anova(data=df,dv='TheData',between='Group')
```

[56]:

	Source	ddof1	ddof2	F	p-unc	np2
0	Group	1	19998	73.027995	1.368619e-17	0.003638

[57]:

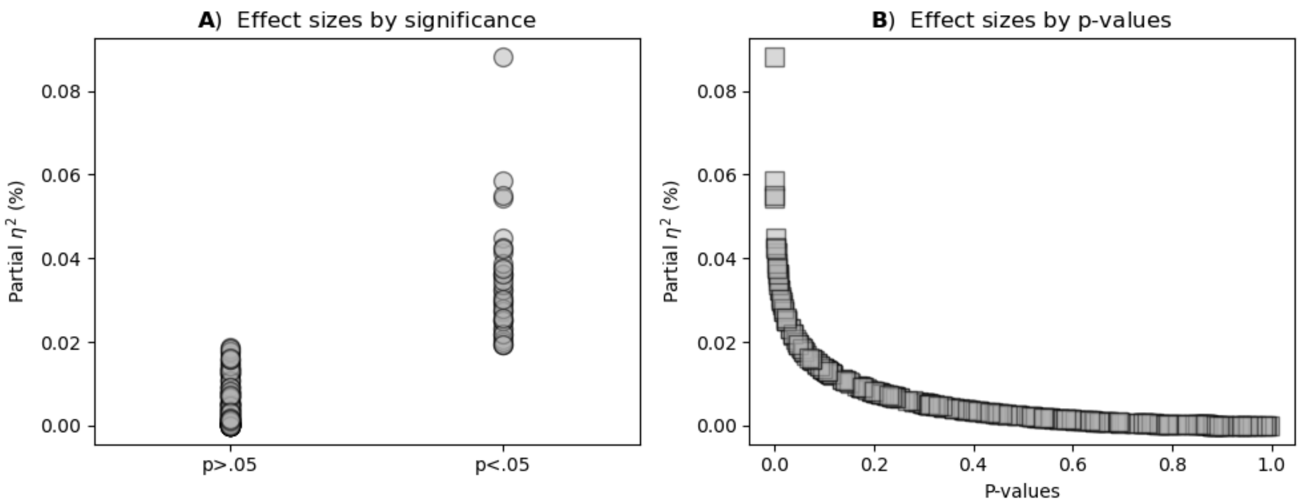
```python
# sample size
N = 10000

# experiment params
nTests = 300 # number of tests to simulate
groups = ['1']*N + ['2']*N
pvals = np.zeros(nTests)  # counter
peta2 = np.zeros(nTests)

# now for the experiment!
for i in range(nTests):
  ##simulate the data
  data1 = np.random.normal(0,1,size=N)
  data2 = np.random.normal(.01,1,size=N)
  data  = np.concatenate((data1,data2),axis=0)
  # run an ANOVA
  df = pd.DataFrame({'TheData':data,'Group':groups})
  anova = pg.anova(data=df,dv='TheData',between='Group')

  # count if significant
  pvals[i] = anova['p-unc'].item()
  peta2[i] = 100*anova['np2'].item()

# print the results
print(f'{np.sum(pvals<.05)} of {nTests} tests ({np.sum(pvals<.05)*100/nTests:.
 ↪2f}%) had p<.05 with N={N}.')
```

```
42 of 300 tests (14.00%) had p<.05 with N=10000.
```

[58]:

```python
_,axs = plt.subplots(1,2,figsize=(10,4))
axs[0].plot(pvals<.05,peta2,'ko',markersize=10,markerfacecolor=(.7,.7,.7),alpha=.
 ↪5)
axs[0].set(xlim=[-.5,1.5],xticks=[0,1],xticklabels=['p>.05','p<.
 ↪05'],ylabel=r'Partial $\eta^2$ (%)')
```

```
axs[0].set_title(r'$\bf{A}$)  Effect sizes by significance')

axs[1].plot(pvals,peta2,'ks',markersize=10,markerfacecolor=(.7,.7,.7),alpha=.5)
axs[1].set(xlabel='P-values',ylabel=r'Partial $\eta^2$ (%)')
axs[1].set_title(r'$\bf{B}$)  Effect sizes by p-values')

plt.tight_layout()
#plt.savefig('anova_ex6.png')
plt.show()
```



[59]:

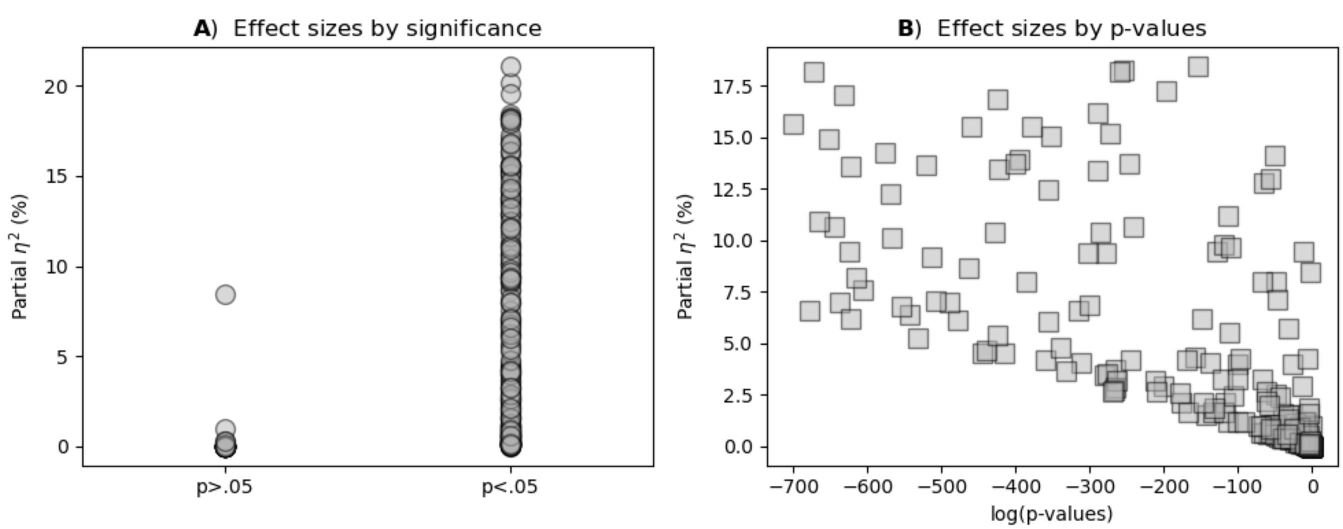
```
### repeat for random sample size

# experiment params
nTests = 300 # number of tests to simulate
groups = ['1']*N + ['2']*N
pvals = np.zeros(nTests)  # counter
peta2 = np.zeros(nTests)

# now for the experiment!
for i in range(nTests):
  # sample size
  N = np.random.randint(10,10000)
  groups = ['1']*N + ['2']*N

  ##simulate the data
  data1 = np.random.normal(0,1,size=N)
  data2 = np.random.normal(np.random.rand()**2,1,size=N)
  data  = np.concatenate((data1,data2),axis=0)

  # run an ANOVA
  df = pd.DataFrame({'TheData':data,'Group':groups})
```

```
    anova = pg.anova(data=df,dv='TheData',between='Group')

    # count if significant
    pvals[i] = anova['p-unc'].item()
    peta2[i] = 100*anova['np2'].item()

# print the results
print(f'{np.sum(pvals<.05)} of {nTests} tests ({np.sum(pvals<.05)*100/nTests:.
 ↪2f}%) had p<.05 with N={N}.')
```

```
246 of 300 tests (82.00%) had p<.05 with N=1778.
```

[60]:
```
_,axs = plt.subplots(1,2,figsize=(10,4))

axs[0].plot(pvals<.05,peta2,'ko',markersize=10,markerfacecolor=(.7,.7,.7),alpha=.
 ↪5)
axs[0].set(xlim=[-.5,1.5],xticks=[0,1],xticklabels=['p>.05','p<.
 ↪05'],ylabel=r'Partial $\eta^2$ (%)')
axs[0].set_title(r'$\bf{A}$)  Effect sizes by significance')

axs[1].plot(np.log(pvals),peta2,'ks',markersize=10,markerfacecolor=(.7,.7,.
 ↪7),alpha=.5)
axs[1].set(xlabel='log(p-values)',ylabel=r'Partial $\eta^2$ (%)')
axs[1].set_title(r'$\bf{B}$)  Effect sizes by p-values')

plt.tight_layout()
#plt.savefig('anova_ex6b.png')
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_13112\1203015327.py:7:
RuntimeWarning: divide by zero encountered in log
  axs[1].plot(np.log(pvals),peta2,'ks',markersize=10,markerfacecolor=(.7,.7,.7),
alpha=.5)
```



24 Exercise 7

[61]:
```
# create data
n_subjects = 30
n_conditions = 3
data = np.random.normal(size=(n_subjects,n_conditions))
data[:,1] += .25 # small offset to measurement #2
data[:,2] += .5  # small offset to measurement #3

# Create a DataFrame
df1 = pd.DataFrame(data, columns=['Cond1','Cond2','Cond3'])

# Convert to long format
df = pd.melt(df1.reset_index(), id_vars=['index'],␣
 ↪value_vars=['Cond1','Cond2','Cond3'])
df.columns = ['Subject', 'Condition', 'Value']

# repeated-measures ANOVA
rmANOVA = pg.rm_anova(data=df, dv='Value', within='Condition',␣
 ↪subject='Subject', detailed=True)
print('Results of a repeated-measures ANOVA:')
display(rmANOVA)

# between-subjects ANOVA
ANOVA = pg.anova(data=df,dv='Value', between='Condition',detailed=True)
print(f'\n\nResults of a between-subjects ANOVA')
display(ANOVA)
```

```
Results of a repeated-measures ANOVA:

      Source         SS  DF        MS         F     p-unc       ng2       eps
0  Condition   0.797082   2  0.398541  0.357717  0.700802  0.009683  0.996619
1      Error  64.619129  58  1.114123       NaN       NaN       NaN       NaN



Results of a between-subjects ANOVA

      Source         SS  DF        MS         F    p-unc       np2
0  Condition   0.797082   2  0.398541  0.425324  0.65491  0.009683
1     Within  81.521584  87  0.937030       NaN      NaN       NaN
```

[62]:
```
# FYI, using statsmodels (not part of the exercise)
# repeated measures ANOVA
rm_anova = AnovaRM(df, 'Value', 'Subject', within=['Condition'])
results = rm_anova.fit()
print(results)
```

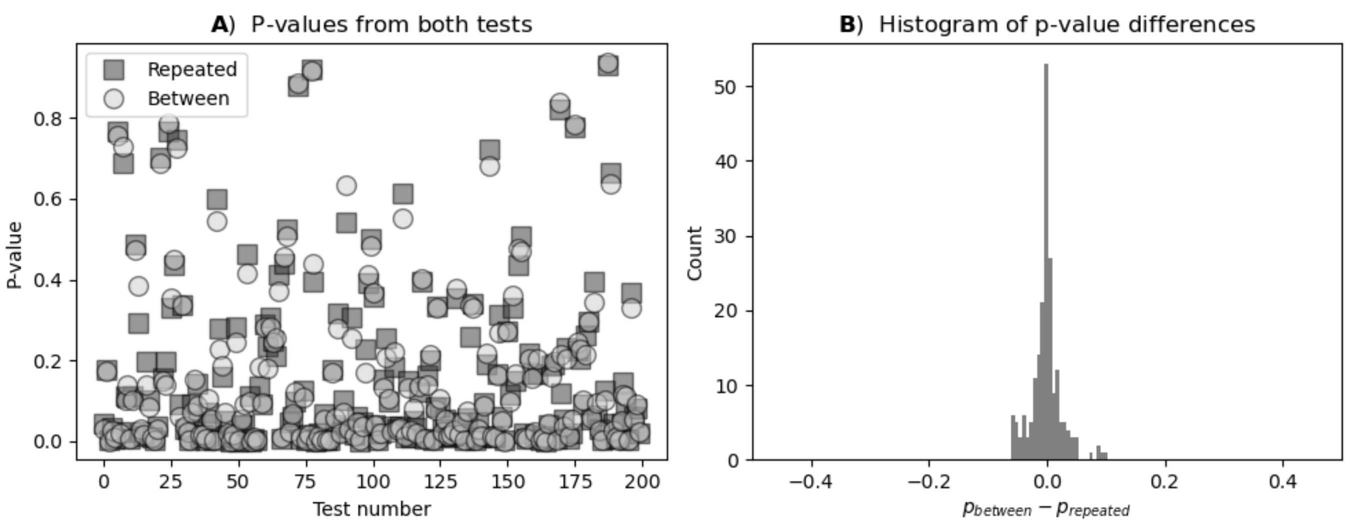
```
# between-subjects ANOVA
model = ols('Value ~ C(Condition)', data=df).fit()
anova_results = sm.stats.anova_lm(model, typ=1)
print(anova_results)
```

```
                 Anova
========================================
          F Value Num DF  Den DF Pr > F
----------------------------------------
Condition  0.3577 2.0000 58.0000 0.7008
========================================

                df     sum_sq   mean_sq         F    PR(>F)
C(Condition)   2.0   0.797082  0.398541  0.425324  0.65491
Residual      87.0  81.521584  0.937030       NaN      NaN
```

[63]:

```
# now for the experiment
nReps = 200

# initialize a matrix of p-values
pvals = np.zeros((nReps,2))

# start the experiment
for i in range(nReps):
  # generate the data (NOTE: the commented code at the end is for exercise 8)
  data = np.random.normal(size=(n_subjects,n_conditions)) #+ np.
↪arange(n_subjects)[:,None]
  data[:,1] += .25
  data[:,2] += .5

  # Create a DataFrame
  df1 = pd.DataFrame(data, columns=['Cond1','Cond2','Cond3'])
  df = pd.melt(df1.reset_index(), id_vars=['index'],␣
↪value_vars=['Cond1','Cond2','Cond3'])
  df.columns = ['Subject', 'Condition', 'Value']

  # the two ANOVAs on the same data
  rmANOVA = pg.rm_anova(data=df, dv='Value', within='Condition',␣
↪subject='Subject')
  ANOVA = pg.anova(data=df,dv='Value', between='Condition')

  # store the p-values
  pvals[i,0] = rmANOVA['p-unc'].item()
  pvals[i,1] = ANOVA['p-unc'].item()
```

```
[64]: # visualize the p-values
_,axs = plt.subplots(1,2,figsize=(10,4))
axs[0].plot(np.arange(200),pvals[:,0],'ks',markersize=10,markerfacecolor=(.2,.2,.
 ↪2),alpha=.5,label='Repeated')
axs[0].plot(np.arange(200),pvals[:,1],'ko',markersize=10,markerfacecolor=(.8,.8,.
 ↪8),alpha=.5,label='Between')
axs[0].set(xlabel='Test number',ylabel='P-value')
axs[0].set_title(r'$\bf{A}$)  P-values from both tests')
axs[0].legend()

axs[1].hist(np.diff(pvals,axis=1),bins='fd',color=(.5,.5,.5))
axs[1].set_title(r'$\bf{B}$)  Histogram of p-value differences')
axs[1].set(xlabel=r'$p_{between}-p_{repeated}$',ylabel='Count')
axs[1].set(xlim=[-.5,.5])

plt.tight_layout()
#plt.savefig('anova_ex7b.png')
plt.show()
```



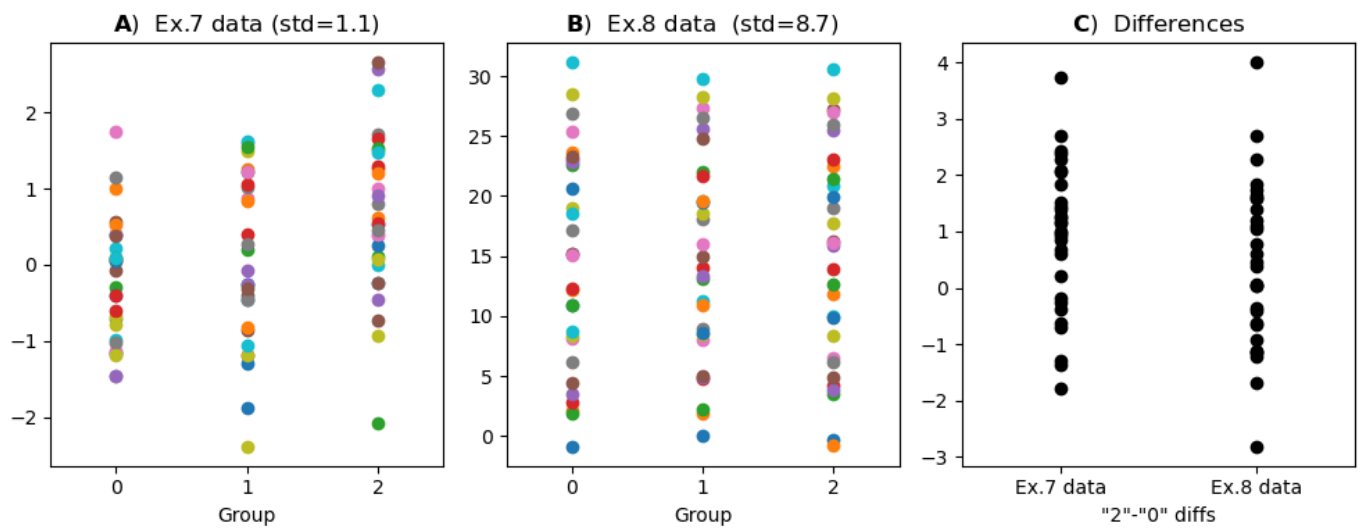
25 Exercise 8

```
[65]: # adapt (or copy/paste) the code from Exercise 7, and replace
data = np.random.normal(size=(n_subjects, n_conditions))

# with
data = np.random.normal(size=(n_subjects, n_conditions)) + np.
 ↪arange(n_subjects)[:,None]

# The idea is to use 'broadcasting' to add the index number to each data row.
data
```

[65]: array([[0.52086785, -1.78815474, 2.16921924],
       [0.45088603, 1.27966865, 0.03857434],
       [0.65565056, 1.53546975, 3.7753272],
       [3.83671695, 3.21273414, 2.11173952],
       [3.60103249, 1.80543048, 1.87496199],
       [4.47504144, 3.84369237, 4.12242559],
       [3.62852543, 6.03293035, 6.91545762],
       [6.420611 , 6.23484879, 7.12571109],
       [7.82819648, 8.30485035, 7.21314677],
       [9.33292435, 9.56775843, 9.28567773],
       [8.50366625, 10.32901253, 9.00603252],
       [11.0564618 , 10.4414868 , 12.94684751],
       [12.0134852 , 11.16174934, 12.95583296],
       [11.30044446, 11.71551882, 13.81002308],
       [12.78974852, 14.27195132, 14.00613428],
       [15.82860785, 15.04018069, 13.91458248],
       [16.35439186, 16.79641261, 16.67453615],
       [15.84440915, 16.77721295, 18.27581224],
       [17.04127269, 18.46101297, 17.00160617],
       [18.511083 , 19.44848547, 16.10347217],
       [21.2503356 , 20.03001537, 19.54128888],
       [20.57675703, 20.57092008, 20.34432343],
       [20.45810874, 21.82892556, 22.23735385],
       [23.89126372, 21.91673688, 21.94877517],
       [24.38734544, 24.52326229, 23.9688552],
       [24.46562466, 26.36259006, 24.31862811],
       [27.1769679 , 26.62532782, 26.68161081],
       [26.81762788, 27.85219292, 26.92273792],
       [29.59286202, 27.20510761, 27.92217244],
       [28.7514172 , 29.06277626, 27.46272382]])

[66]:
```
# code to make the figure
data1 = np.random.normal(size=(n_subjects, n_conditions))
data1[:,1] += .25
data1[:,2] += .5

data2 = np.random.normal(size=(n_subjects, n_conditions)) + np.
 ↪arange(n_subjects)[:,None]
data2[:,1] += .25
data2[:,2] += .5

fig,axs = plt.subplots(1,3,figsize=(10,4))

axs[0].plot(data1.T,'o')
axs[0].set_title(fr'$\bf{{A}}$)  Ex.7 data (std={np.std(data1):.1f})')

axs[1].plot(data2.T,'o')
```

```
axs[1].set_title(fr'$\bf{{B}}$)  Ex.8 data  (std={np.std(data2):.1f})')

axs[2].plot(np.zeros(n_subjects),data1[:,2]-data1[:,0],'ko')
axs[2].plot(np.ones(n_subjects), data2[:,2]-data2[:,0],'ko')
axs[2].set(xlim=[-.5,1.5],xticks=[0,1],xticklabels=['Ex.7 data','Ex.8␣
↪data'],xlabel='"2"-"0" diffs')
axs[2].set_title(r'$\bf{C}$)  Differences')

for a in axs[:2]:
  a.set(xlim=[-.5,2.5],xticks=[0,1,2],xlabel='Group')

plt.tight_layout()
#plt.savefig('anova_ex8.png')
plt.show()
```



26 Exercise 9

[67]:
```
# population cell means
# "factor A" is the number of rows, "factor B" is the number of columns
group_means = [ [ 1,1,1.3,.7 ],
                [ 1,1,.7,1.3 ] ]

factA,factB = np.shape(group_means)

# per-cell sample sizes
cellCounts = np.random.choice(range(25,36),factA*factB)
nDataRows = np.sum(cellCounts) # total rows in the dataset

# data matrix in numpy (initialize as zeros, then modulate by group_mean)
datamat = np.zeros((nDataRows,3))
rowidx = 0
```

```
for idx in range(factA*factB):
  # convert linear to matrix index (to get group_means)
  a,b = np.unravel_index(idx,(factA,factB))

  # population cell mean
  cellMean = group_means[a][b]

  # random data
  celldata = np.random.normal(loc=cellMean,scale=1,size=cellCounts[idx])

  # add to matrix
  datamat[rowidx:rowidx+cellCounts[idx],0] = a
  datamat[rowidx:rowidx+cellCounts[idx],1] = b
  datamat[rowidx:rowidx+cellCounts[idx],2] = celldata

  # update row counter
  rowidx += cellCounts[idx]

# Create dataframe
df = pd.DataFrame(datamat,columns=['A','B','val'])

# two-way ANOVAs
for i in range(1,4):
  print(f'Type-{i} ANOVA table:')
  print(pg.anova(data=df, dv='val', between=['A','B'], ss_type=i))
  print(f'\n\n')
```

```
Type-1 ANOVA table:
     Source          SS     DF        MS         F     p-unc       np2
0         A    1.055462    1.0  1.055462  1.091098  0.297325  0.004722
1         B    2.656183    3.0  0.885394  0.915288  0.434225  0.011798
2     A * B   26.483828    3.0  8.827943  9.126004  0.000010  0.106373
3  Residual  222.488036  230.0  0.967339       NaN       NaN       NaN




Type-2 ANOVA table:
     Source          SS     DF        MS         F     p-unc       np2
0         A    0.820874    1.0  0.820874  0.848589  0.357917  0.003676
1         B    2.656183    3.0  0.885394  0.915288  0.434225  0.011798
2     A * B   26.483828    3.0  8.827943  9.126004  0.000010  0.106373
3  Residual  222.488036  230.0  0.967339       NaN       NaN       NaN
```

```
Type-3 ANOVA table:
     Source          SS     DF        MS         F     p-unc       np2
0         A    0.923010    1.0  0.923010  0.954174  0.329685  0.004131
1         B    3.870470    3.0  1.290157  1.333717  0.264139  0.017099
2     A * B   26.483828    3.0  8.827943  9.126004  0.000010  0.106373
3  Residual  222.488036  230.0  0.967339       NaN       NaN       NaN
```

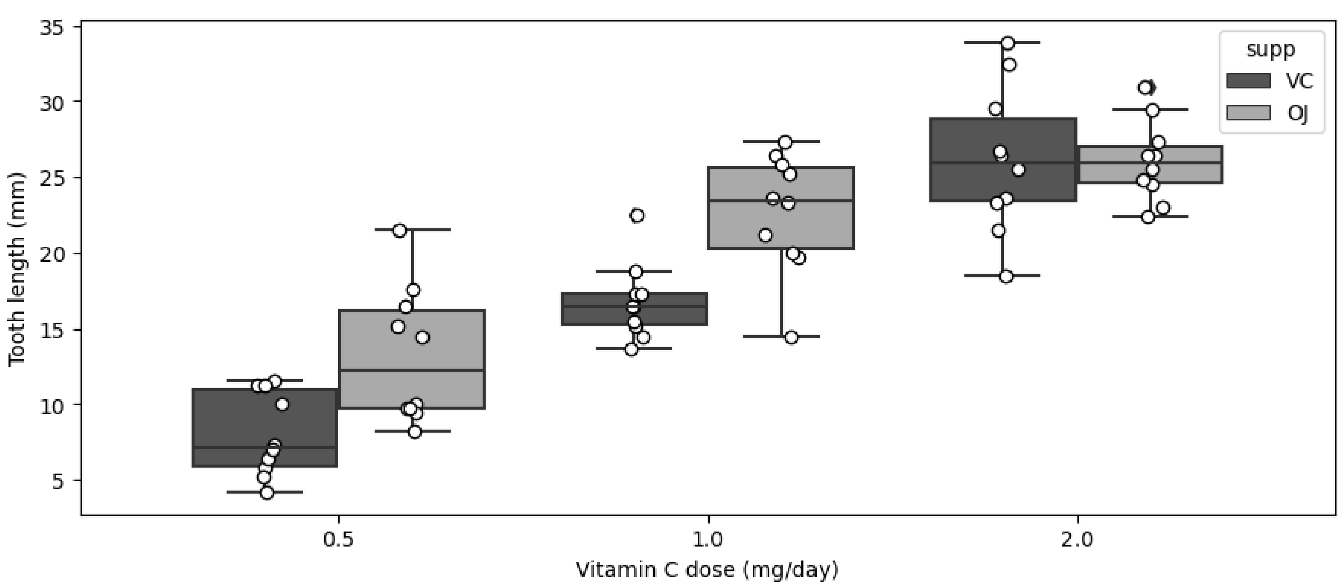
27 Exercise 10

[68]:
```
# Original source: https://www.rdocumentation.org/packages/datasets/versions/3.6.
↪2/topics/ToothGrowth

url = "https://sincxpress.com/ToothGrowth.csv"

data = pd.read_csv(url)
data
```

[68]:
```
    Unnamed: 0   len supp  dose
0            1   4.2   VC   0.5
1            2  11.5   VC   0.5
2            3   7.3   VC   0.5
3            4   5.8   VC   0.5
4            5   6.4   VC   0.5
5            6  10.0   VC   0.5
6            7  11.2   VC   0.5
7            8  11.2   VC   0.5
8            9   5.2   VC   0.5
9           10   7.0   VC   0.5
10          11  16.5   VC   1.0
........
........

50          51  25.5   OJ   2.0
51          52  26.4   OJ   2.0
52          53  22.4   OJ   2.0
53          54  24.5   OJ   2.0
54          55  24.8   OJ   2.0
55          56  30.9   OJ   2.0
56          57  26.4   OJ   2.0
57          58  27.3   OJ   2.0
58          59  29.4   OJ   2.0
59          60  23.0   OJ   2.0
```

[70]:

```
# show the data
plt.figure(figsize=(9,4))

# boxplot
sns.boxplot(x='dose', y='len', hue='supp', data=data, palette='gray')

# offsets (manually coded)
offsets = [.2,-.2, 1.2,.8, 2.2,1.8 ]
i=0 # counter

# loop through all conditions to plot individual data points
for d in np.unique(data['dose']):
  for s in np.unique(data['supp']):
    # the data just from this condition
    tmpY = data[(data['dose']==d) & (data['supp']==s)]['len']
    tmpX = np.random.normal(loc=offsets[i],scale=.02,size=len(tmpY))

    # plot those values, with a bit of offset
    plt.plot(tmpX,tmpY,'ko',markerfacecolor='w')
    i+=1 # update counter

plt.ylabel('Tooth length (mm)') # more informative
plt.xlabel('Vitamin C dose (mg/day)')
plt.tight_layout()
plt.savefig('anova_ex10.png')
plt.show()
```



```
[71]: # run the ANOVA!
pg.anova(data=data, dv='len', between=['supp','dose'])
```

```
[71]:        Source           SS  DF           MS          F         p-unc  \
0          supp   205.350000   1   205.350000  15.571979  2.311828e-04
1          dose  2426.434333   2  1213.217167  91.999965  4.046291e-18
2   supp * dose   108.319000   2    54.159500   4.106991  2.186027e-02
3      Residual   712.106000  54    13.187148        NaN           NaN

        np2
0  0.223825
1  0.773109
2  0.132028
3       NaN
```

28 Exercise 11

```
[72]: # calculate the mean for each group
data['predictions'] = data.groupby(['dose','supp'])['len'].transform('mean')

# Subtract the group means (predicted data) from the DV (observed data) to get␣
 ↪residuals
data['residuals'] = data['len'] - data['predictions']

# show a few rows
data[::4]
```

```
[72]:     Unnamed: 0   len supp  dose  predictions  residuals
0            1   4.2   VC   0.5         7.98      -3.78
4            5   6.4   VC   0.5         7.98      -1.58
8            9   5.2   VC   0.5         7.98      -2.78
12          13  15.2   VC   1.0        16.77      -1.57
16          17  13.6   VC   1.0        16.77      -3.17
20          21  23.6   VC   2.0        26.14      -2.54
24          25  26.4   VC   2.0        26.14       0.26
28          29  23.3   VC   2.0        26.14      -2.84
32          33  17.6   OJ   0.5        13.23       4.37
36          37   8.2   OJ   0.5        13.23      -5.03
40          41  19.7   OJ   1.0        22.70      -3.00
44          45  20.0   OJ   1.0        22.70      -2.70
48          49  14.5   OJ   1.0        22.70      -8.20
52          53  22.4   OJ   2.0        26.06      -3.66
56          57  26.4   OJ   2.0        26.06       0.34
```

[73]:
```
# empirical correlation
r = stats.pearsonr(data['predictions'],data['residuals'])

# scatter plot
plt.figure(figsize=(8,3))

plt.plot(data['predictions'], data['residuals'],'ko',
                markerfacecolor=(.8,.8,.8), markersize=12, alpha=.5)
plt.xlabel('Predicted length')
plt.ylabel('Residuals')
plt.title(f'Pearson r={r.statistic:.4f}, p={r.pvalue:.4f}',loc='center')

plt.tight_layout()
#plt.savefig('anova_ex11.png')
plt.show()
```