S1: Principles of Data Science

Problem Sheet 1 Solutions

MPhil in Data Intensive Science

Matt Kenzie
mk652@cam.ac.uk

Michealmas Term 2023

Problem Sheet 1

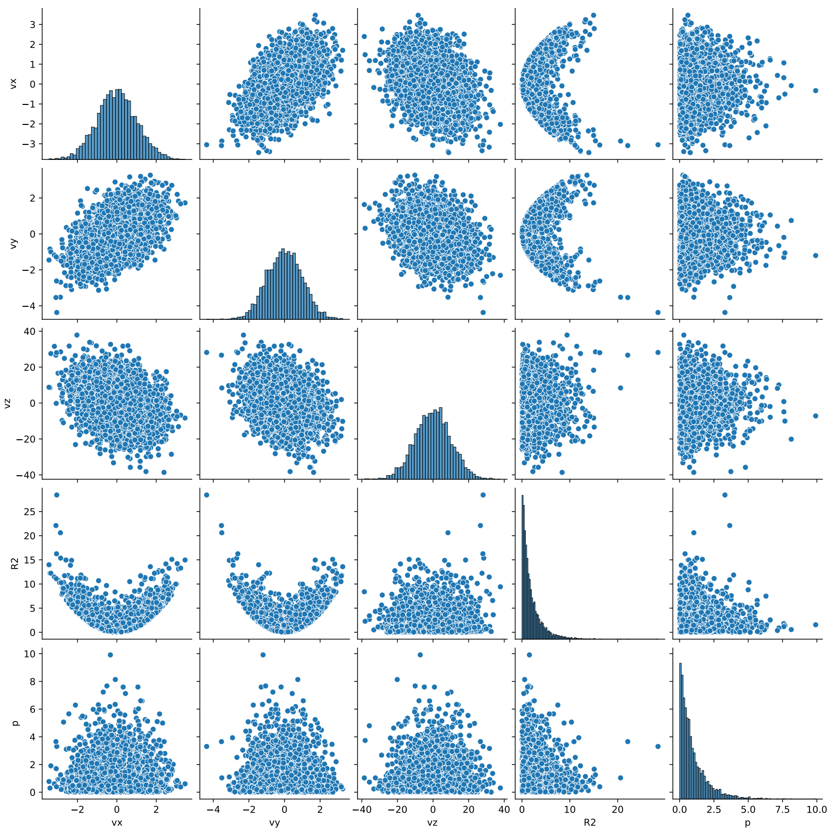
Lectures 1 – 6

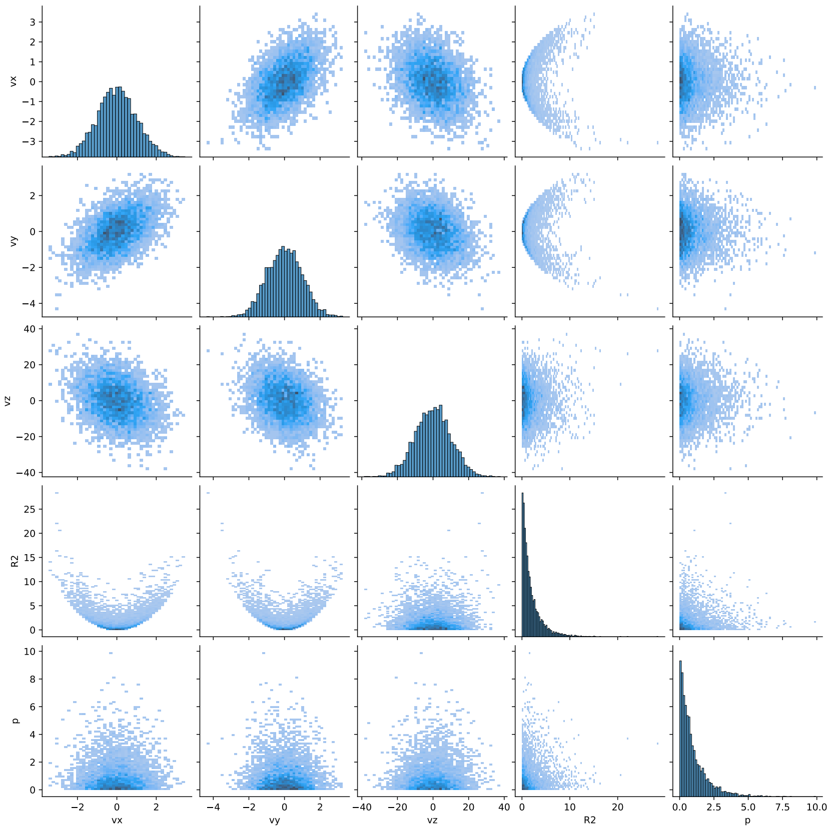
1. Here is a straightforward piece of code (`visualisation.py`) that should do the job

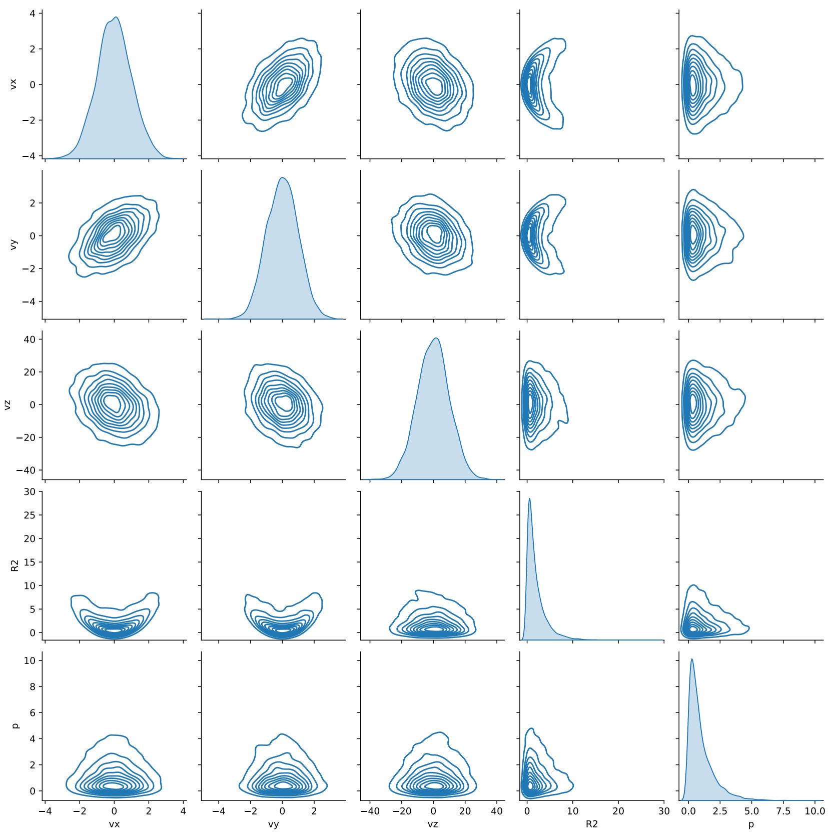
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# read the dataset
df = pd.read_pickle("../s1_principles_of_data_science/datasets/ps1.pkl")

# Part A
# make a pair plot using seaborn
sns.pairplot( df )
plt.savefig("figs/s1_pairplot.pdf")
sns.pairplot( df, kind='kde')
plt.savefig("figs/s1_pairplot_kde.pdf")
sns.pairplot( df, kind='hist')
plt.savefig("figs/s1_pairplot_hist.pdf")

# Part B
# make a publication style plot
plt.style.use('code/mphil.mplstyle')
fig, ax = plt.subplots()
x, y = df['vx'], df['vz']
sns.scatterplot(x=x, y=y, s=5, color="0.15")
sns.histplot(x=x,y=y, bins=25, pthresh=0.1, cmap="mako", alpha=0.9)
sns.kdeplot(x=x, y=y, levels=5, color='w', linewidths=1)
ax.set_xlabel('Variable A')
ax.set_ylabel('Variable B')
fig.savefig("figs/s1_pubplot.pdf")

# Part C
print( df.corr() )
```

```
print( df.cov() )
print( df.corr().to_latex() )
print( df.cov().to_latex() )

plt.show()
```

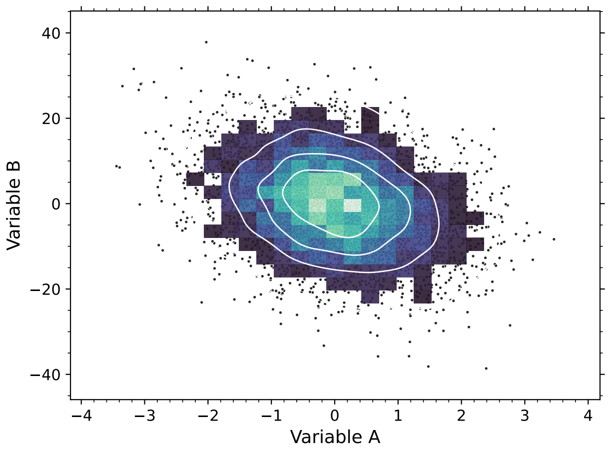
(a) Make something like one of the following (any fancier versions will do):







(b) Make something like the following, do we consider this publication quality?



(c) Straight-forward use of `pandas` built-in methods is fine, but also great if they do this themselves with some code (or any other built-in from e.g. `numpy`).
Correlation:

	vx	vy	vz	R2	p
vx	1.000000	0.499943	-0.283132	0.001504	-0.001291
vy	0.499943	1.000000	-0.274027	-0.001331	-0.024314
vz	-0.283132	-0.274027	1.000000	0.008939	0.015736
R2	0.001504	-0.001331	0.008939	1.000000	0.002401
p	-0.001291	-0.024314	0.015736	0.002401	1.000000

Covariance:

	vx	vy	vz	R2	p
vx	1.002608	0.493081	-2.852446	0.003324	-0.001334
vy	0.493081	0.970208	-2.715740	-0.002893	-0.024699
vz	-2.852446	-2.715740	101.233585	0.198513	0.163291
R2	0.003324	-0.002893	0.198513	4.871490	0.005466
p	-0.001334	-0.024699	0.163291	0.005466	1.063632

2. Let's see what happens if we try to use simply the average deviation rather than the average deviation squared:

$$\begin{aligned}\frac{1}{N}\sum_i (X_i - \bar{X}) &= \frac{1}{N}\sum_i X_i - \frac{1}{N}\sum_i \bar{X} && (1)\\ &= \bar{X} - \bar{X} && (2)\\ &= 0. && (3)\end{aligned}$$

So we always get a spread of zero, hence why we use the squares.

You do occasionally see use of the *mean absolute deviation*:

$$\frac{1}{N}\sum_i \left| X_i - \bar{X} \right| . \tag{4}$$

The reason this isn't widely used is because of its rather nasty mathematical behaviour. Differentials of squared quantities give nice linear terms. Differentials of modulus terms is horrible.

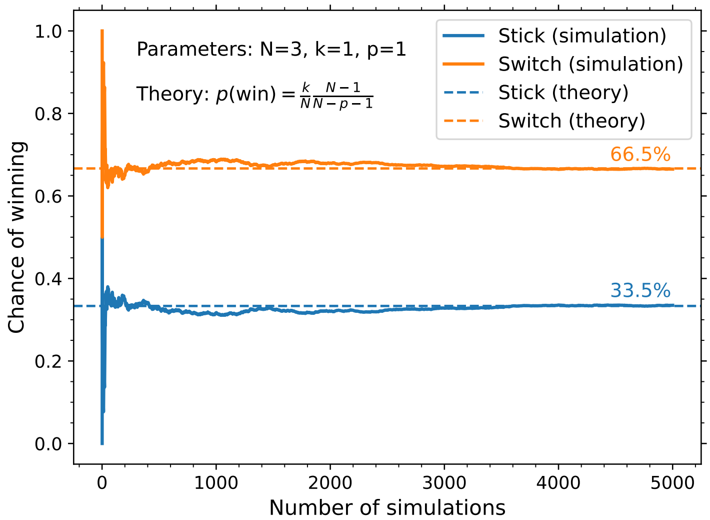
3. (a) Here is a snippet of code (`monty_hall_sim.py`) which runs a Monty Hall simulator for any number of boxes, N, prizes, k, and box reveals, p.

```python
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('code/mphil.mplstyle')
from tqdm import tqdm
from argparse import ArgumentParser

def run_sim( N, k, p ):
    """
    Run Monty Hall simulation

    Parameters
    ----------
    N : int
        The number of boxes
    k : int
        The number of boxes which contain prizes, must have k < N-1
    p : int
        The number of boxes which are revealed by the game show host
        after the initial selection, must have p < N-k
    """

    # make sure number of prizes is less than N-1
    if not k < (N-1):
        raise ValueError(f"Cannot have more prizes than number of boxes minus 1, k={k}, N={N}")

    # make sure number of door reveals is less than N-k
    if not p < (N-k):
        raise ValueError(f"Cannot have more box reveals than number of boxes minus number of prizes, p={p}, k={k}, N={N}")

    # box labels
    box_choices = np.arange(1, N+1, dtype=np.int32)

    # randomly choose which prizes boxes are in
    prizes = np.random.choice(box_choices, size=k, replace=False)

    # figure out which boxes are not prizes
    not_prizes = np.asarray( [ a for a in box_choices if a not in prizes ] )

    # randomly make an initial choice of box
    initial_choice = np.random.choice(box_choices, size=1)

    # possible reveal boxes
    poss_reveals = [ a for a in not_prizes if a not in initial_choice ]

    # randomly reveal other boxes not containing prize
```

```
    reveals = np.random.choice( poss_reveals, size=p,  replace=False )

    # which boxes could now be switched to
    switch_options = [ a for a in box_choices if a not in initial_choice and a not in reveals ]

    # pick a switch box
    switch_choice = np.random.choice( switch_options, size=1 )

    # let's run some checks we didn't make a mistake
    assert( len(box_choices) == len(prizes)+len(not_prizes) )
    assert( sorted(box_choices) == sorted( np.concatenate( [prizes, not_prizes] ) ) )

    # win outcome
    win_stick = initial_choice in prizes
    win_switch = switch_choice in prizes

    return win_stick, win_switch

def theory( N, k, p ):
    prob_win_switch = (k/N)*( (N-1)/(N-p-1) )
    prob_win_stick  = 1 - prob_win_switch
    return prob_win_stick, prob_win_switch

def run_ntrials( Ntrials, N, k, p, seed=None ):

    if seed:
        np.random.seed(seed)

    Nwin_stick  = 0
    Nwin_switch = 0

    stick_win_tracker = []
    switch_win_tracker = []

    for i in tqdm(range(Ntrials)):
        win_stick, win_switch = run_sim( N, k, p)
        if win_stick:
            Nwin_stick += 1
        if win_switch:
            Nwin_switch += 1

        stick_win_tracker.append( Nwin_stick )
        switch_win_tracker.append( Nwin_switch )

    stick_win_tracker = np.asarray( stick_win_tracker )
    switch_win_tracker = np.asarray( switch_win_tracker )

    fig, ax = plt.subplots()
    x = np.arange(1, Ntrials+1, 1)

    stick_win_frac = stick_win_tracker / x
    switch_win_frac = switch_win_tracker / x

```

```
    res_stick = stick_win_frac[-1]
    res_switch = switch_win_frac[-1]

    # compute the theory line
    th_win_stick, th_win_switch = theory( N, p, k )

    ax.plot(x, stick_win_frac, lw=2, label='Stick (simulation)')
    ax.plot(x, switch_win_frac, lw=2, label='Switch (simulation)')
    ax.axhline( th_win_stick, c='C0', ls='--', label='Stick (theory)')
    ax.axhline( th_win_switch, c='C1', ls='--', label='Switch (theory)')
    ax.text(0.1,0.9, f'Parameters: N={N}, k={k}, p={p}', transform=ax.transAxes)
    ax.text(0.1,0.8, r'Theory: $p(\mathrm{win}) = \frac{k}{N}\frac{N-1}{N-p-1}$', transform=ax.transAxes)
    ax.text(Ntrials, res_stick+0.01, f'${100*res_stick:3.1f}\%$', color='C0', va='bottom', ha='right')
    ax.text(Ntrials, res_switch+0.01, f'${100*res_switch:3.1f}\%$', color='C1', va='bottom', ha='right')

    ax.set_xlabel('Number of simulations')
    ax.set_ylabel('Chance of winning')
    ax.legend()

    fig.savefig('figs/monty_hall.pdf')

if __name__=="__main__":

    # I'd like to have an argument please
    parser = ArgumentParser()
    parser.add_argument('-t','--nTrials', default=5000, type=int, help='Number of simulation trials to run')
    parser.add_argument('-N','--nBoxes' , default=3, type=int, help='Number of boxes in the game show')
    parser.add_argument('-k','--nPrizes', default=1, type=int, help='Number of boxes which contain prizes')
    parser.add_argument('-p','--nReveal', default=1, type=int, help='Number of boxes which are revealed')
    parser.add_argument('-s','--seed', default=0, type=int, help='The initial random seed of the generation')
    args = parser.parse_args()

    run_ntrials( args.nTrials, args.nBoxes, args.nPrizes, args.nReveal, args.seed )

    plt.show()
```

It produces this kind of simulation plot:



(b) You can do this by Bayes' theorem or by intuition or by the simulation. This question is instructive because people attempt to assert their intuition that in the original Monty-Hall problem the chances are 50:50. However given this situtation its hard to imagine how it could still be 50:50. The answer is 1/100 if you stick and 99/100 if you switch. The Bayes theorem proofs I put below, first for $N = 3$, $k = 1$ and $p = 1$ that we saw in the lectures, and second for $N = 100$, $k = 1$, $p = 98$ as asked in this question.

For $N = 3$, $k = 1$, $p = 1$:

There are 3 boxes A, B, C. Let's assume the we pick box A and the host opens box B. Note, that this covers all possibilities because we can simply re-name the boxes for a different ordering.

Let's first compute the probability the car is in box A. From Bayes' theorem:

$$p(\text{car in A}|\text{host opens B}) = \frac{p(\text{host opens B}|\text{car in A})p(\text{car in A})}{p(\text{host opens B})} \tag{5}$$

The terms on the right hand numerator are trivial, but what about the right hand side denominator? How do we compute the probability the host opens B because it depends on different outcomes, i.e. whether the prize is already in A or not. This is where we can exploit the theory of total probability (the sum over all possibilities). The law of total probability states that:

$$p(X) = \sum_i p(X|Y_i)Y_i. \tag{6}$$

So that in our case

$$
\begin{align}
p(\text{host opens B}) &= p(\text{host opens B}|\text{car in A})p(\text{car in A}) \tag{7}\\
&\quad + p(\text{host opens B}|\text{car in C})p(\text{car in C}) \tag{8}\\
&= \frac{1}{2} \times \frac{1}{3} + 1 \times \frac{1}{3} \tag{9}\\
&= \frac{1}{6} + \frac{1}{3} \tag{10}\\
&= \frac{1}{2}. \tag{11}
\end{align}
$$

Incidentally many find this probability easy to intuit. Either the host will open B or C with equal probability.

Now we can plug the numbers back into Bayes' theorem so that:

$$
\begin{align}
p(\text{car in A}|\text{host opens B}) &= \frac{p(\text{host opens B}|\text{car in A})p(\text{car in A})}{p(\text{host opens B})} \tag{12}\\
&= \frac{\frac{1}{2} \times \frac{1}{3}}{\frac{1}{2}} = \frac{1}{3}. \tag{13}
\end{align}
$$

Can follow the same logic for the switch probability:

$$
\begin{align}
p(\text{car in C}|\text{host opens B}) &= \frac{p(\text{host opens B}|\text{car in C})p(\text{car in C})}{p(\text{host opens B})} \tag{14}\\
&= \frac{1 \times \frac{1}{3}}{\frac{1}{2}} = \frac{2}{3}. \tag{15}
\end{align}
$$

For $N = 100$, $k = 1$, $p = 98$:
There are 100 boxes $i = 1, \dots, 100$. Let's assume the we pick box 1 and the host opens boxes $2, \dots, 99$. Note, that this covers all possibilities because we can simply re-name the boxes for a different ordering.

$$
p(\text{car in 1}|\text{host opens 2-99}) = \frac{p(\text{host opens 2-99}|\text{car in 1})p(\text{car in 1})}{p(\text{host opens 2-99})}. \tag{16}
$$

For the denominator:

$$
\begin{align}
p(\text{host open 2-99}) &= \sum_i p(B|A_i)p(A_i) \tag{17}\\
&= p(\text{host opens 2-99}|\text{car in 1})p(\text{car in 1}) \tag{18}\\
&\quad + p(\text{host opens 2-99}|\text{car in 100})p(\text{car in 100}) \tag{19}\\
&= \frac{1}{99} \times \frac{1}{100} + 1 \times \frac{1}{100} \tag{20}\\
&= \frac{1}{100}\left(\frac{1}{99} + \frac{99}{99}\right) \tag{21}\\
&= \frac{1}{99}. \tag{22}
\end{align}
$$

So for the total probability if you stick:

$$p(\text{car in 1}|\text{host opens 2-99}) = \frac{p(\text{host opens 2-99}|\text{car in 1})p(\text{car in 1})}{p(\text{host opens 2-99})} \tag{23}$$

$$= \frac{\frac{1}{99} \times \frac{1}{100}}{\frac{1}{99}} \tag{24}$$

$$= \frac{1}{100}. \tag{25}$$

Thus the probabilities in this case are:

$$p(\text{win on stick}) = \frac{1}{100} \tag{26}$$

$$p(\text{win on switch}) = \frac{99}{100}. \tag{27}$$

(c) The above can then fairly easily be generalised to N doors with $N - 2$ being revealed to give

$$p(\text{win on stick}) = \frac{1}{N} \tag{28}$$

$$p(\text{win on switch}) = \frac{N-1}{N}. \tag{29}$$

(d) You can solve this using Bayes' theorem again but it gets a bit fiddly, so actually easier just to think in terms of intuitive probabilities once again. In this case I have N boxes with 1 prize and p empty doors revealed. Once I have initially picked a box and then p have been revelead as empty, there are $N - p - 1$ boxes remaining for me to chose to switch to. So then the probability of chosing correctly upon switching is:

$$p(\text{picked wrong}) \times p(\text{switched to prize}|\text{picked wrong}) \tag{30}$$

$$+ p(\text{picked right}) \times p(\text{switched to prize}|\text{picked right}) \tag{31}$$

$$= \frac{N-1}{N} \frac{1}{N-p-1} + \frac{1}{N} 0 \tag{32}$$

$$= \frac{N-1}{N} \frac{1}{N-p-1} \tag{33}$$

(e) I can readily extend the above because now my initial choice of picking right is k/N, which after switching means my chances are $(k-1)/(N-p-1)$. If I intially pick wrongly, which is $(N-k)/N$, then my chances of winning after a switch are

$k/(N-p-1)$. Thus my total probability is:

$$\begin{aligned}
&p(\text{picked wrong}) \times p(\text{switched to prize}|\text{picked wrong}) && (34)\\
&\qquad + p(\text{picked right}) \times p(\text{switched to prize}|\text{picked right}) && (35)\\
&= \frac{N-k}{N}\frac{k}{N-p-1} + \frac{k}{N}\frac{k-1}{N-p-1} && (36)\\
&= \frac{k(N-1)}{N(N-p-1)} && (37)\\
&= \frac{k}{N}\frac{N-1}{N-p-1}. && (38)
\end{aligned}$$

(f) i. Once again extending the above and considering only $m = 1$:

$$\begin{aligned}
&p(\text{picked wrong}) \times p(\text{switched to prize}|\text{picked wrong}) && (39)\\
&\qquad + p(\text{picked right}) \times p(\text{switched to prize}|\text{picked right}) && (40)\\
&= \frac{N-k}{N}\frac{k-r}{N-p-1} + \frac{k}{N}\frac{k-r-1}{N-p-1} && (41)\\
&= \frac{k(N-1)-rN}{N(N-p-1)}. && (42)
\end{aligned}$$

Switching is the best strategy if it gives a larger probability than the initial random choice of k/N so we get an equality for the switching of

$$\begin{aligned}
&\frac{k(N-1)-rN}{N(N-p-1)} > \frac{k}{N} && (43)\\
&\Rightarrow k(N-1) - rN > k(N-p-1) && (44)\\
&\Rightarrow -rN > -kp && (45)\\
&\Rightarrow rN < kp && (46)\\
&\Rightarrow \frac{r}{p} < \frac{k}{N}. && (47)
\end{aligned}$$

ii. Now let us consider the broader case when $m > 1$. What I want to try and do here is maximise my expected number of prizes. An alternative way of thinking about the scenario with m initial choices is that it is similar to having a single choice but where the contents don't contain zero or one (i.e. no prize or a single prize) but instead contain some fraction of prizes (or total number of prizes). So let's work out the expected number of prizes if I stick with my initial choice. Well I pick m doors and each of them has a chance of having a prize of k/N so therefore I would expect to get:

$$\langle \text{N prizes} \rangle = m\frac{k}{N} \tag{48}$$

prizes, if I stick with my initial choice.

Now let's think about the expected number of prizes in the doors I didn't initially choose, well that is simply

$$\langle \text{N prizes} \rangle = (N-m)\frac{k}{N} \tag{49}$$

because there are $N-m$ boxes I didn't choose each with a chance of having a prize $\frac{k}{N}$. However then what happens is that the game show host reveals p boxes and r of them have a prize in them, so those prizes are no longer winnable. This means the expected number of prizes that I didn't initially choose goes down by r:

$$\langle \text{N prizes} \rangle = (N-m)\frac{k}{N} - r. \tag{50}$$

Because some of those other boxes have also been revealed as empty I can now compute the expected number of prizes elsewhere per box remaining elsewhere which is then the above expectation divided by the number of boxes left, which is $N-p-m$:

$$\langle \text{N prizes / box} \rangle = \frac{(N-m)k/N - r}{N-p-m} = \frac{(N-m)k - rN}{N(N-p-m)}. \tag{51}$$

Now if I have to switch all of my m boxes for a set of m new ones then the total expected number of prizes after swithcing is the above multiplied by m, giving

$$\langle \text{N prizes} \rangle = m\frac{(N-m)k - rN}{N(N-p-m)}. \tag{52}$$

Now all I want to do is determine if this is greater than the expected number of boxes if I stick with my original choice so:

$$m\frac{(N-m)k - rN}{N(N-p-m)} > m\frac{k}{N} \tag{53}$$

$$\frac{(N-m)k - rN}{N-p-m} > k \tag{54}$$

$$Nk - mk - rN > kN - pk - mk \tag{55}$$

$$-rN > -pk \tag{56}$$

$$\frac{r}{p} < \frac{k}{N} \tag{57}$$

iii. Now what if I am allowed to swap some subset of m, ℓ, of the boxes I intially chose? Well in this case the same equality is true. One way of intuitively thinking about this is to summarise the problem in terms of 3 boxes (or 3 spaces). My first space (or box) is M which contains the boxes I chose, the second set is the space P which contains the boxes revealed, and the third set is $N-M-P$ which contains the boxes remaining that I can switch to. I know that the space of my original choice, M, will be populated with prizes in the fraction k/N (it is just a random subsample of the wider N space). So if I am shown the space P to contain a higher fraction than k/N I know that the remaining space $N-M-P$ must contain a lower fraction and therefore I should stay where I am. If I am shown that the space P contains a lower density of prizes then I should switch and I should switch as many boxes as I am allowed to because that space contains a higher density of prizes than my original space. Therefore the switch condition of $\frac{r}{p} < \frac{k}{N}$ holds regardless of the values of m or ℓ.

4. The binomial p.m.f. is given by:

$$P(k; p, n) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}. \tag{58}$$

To check this is properly normalised we can sum it over all possibilities $k \in [0, n]$. There are a few ways to do this but I'm going to exploit Newton's Binomial theorem which states that:

$$\begin{aligned} (1+x)^n &= 1 + nx + \frac{n(n-1)}{2!}x^2 + \frac{n(n-1)(n-2)}{3!}x^3 + \cdots + nx^{n-1} + x^n & (59) \\ &= \sum_{k=0}^{n} \frac{n!}{k!(n-k)1} x^k. & (60) \end{aligned}$$

Using this theorem we can now simplify the sum of the Binomial p.d.f. over all $k \in [0, n]$:

$$\begin{aligned} \sum_{k=0}^{n} \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k} &= \sum_{k=0}^{n} \frac{n!}{k!(n-k)!} \frac{p^k (1-p)^n}{(1-p)^k} & (61) \\ &= (1-p)^n \sum_{k=0}^{n} \frac{n!}{k!(n-k)!} \left(\frac{p}{1-p} \right)^k & (62) \\ &= (1-p)^n \left(1 + \frac{p}{1-p} \right)^n & (63) \\ &= (1-p)^n \left(\frac{1-p+p}{1-p} \right)^n & (64) \\ &= (1-p)^n \left(\frac{1}{1-p} \right)^n & (65) \\ &= 1. & (66) \end{aligned}$$

Equally valid and even more straightforward is to use the general Binomial formula

$$(a+b)^n = \sum_{i=0}^{n} \frac{n!}{i!(n-i)!} a^i b^{n-i} \tag{67}$$

in which case the proof is a one-liner.

5. (a) Probability of exactly three hits with three stations is

$$P(3; 0.95, 3) = 0.95^3 = 0.857, \tag{68}$$

so 85.7%.

(b) Probability of three or more hits with four stations is

$$P(3; 0.95, 4) + P(4; 0.95, 4) = 0.171 + 0.815 = 98.6\%. \tag{69}$$

The probability of three or more hits with give stations is

$$P(3; 0.95, 5) + P(4; 0.95, 5) + P(5; 0.95, 5) = 0.021 + 0.204 + 0.774 = 99.9\%. \tag{70}$$

6. Poisson p.m.f. given by

$$P(k; \lambda) = \frac{e^{-\lambda}\lambda^k}{k!}. \tag{71}$$

Going to exploit the Binomial expansion formula again:

$$(a+b)^n = \sum_{i=0}^{n} \frac{n!}{i!(n-i)!} a^i b^{n-i}. \tag{72}$$

For the sum of two Poissons $Z = X + Y$ then they could all be from X or all from Y so we need to sum over several possibilities:

$$\begin{aligned}
P(Z) &= \sum_{X=0}^{Z} P(X; \lambda_X) P(Z - X; \lambda_Y) && (73) \\
&= \sum_{X=0}^{Z} \frac{e^{-\lambda_X}\lambda_X^X}{X!} \frac{e^{-\lambda_Y}\lambda_Y^{Z-X}}{(Z-X)!} && (74) \\
&= e^{-\lambda_X} e^{-\lambda_Y} \sum_{X=0}^{Z} \frac{\lambda_X^X \lambda_Y^{Z-X}}{X!(Z-X)!} && (75) \\
&= \frac{e^{-\lambda_X} e^{-\lambda_Y}}{Z!} \sum_{X=0}^{Z} \frac{Z!}{X!(Z-X)!} \lambda_X^X \lambda_Y^{Z-X} && (76) \\
&= \frac{e^{-(\lambda_X+\lambda_Y)}(\lambda_X+\lambda_Y)^Z}{Z!} && (77)
\end{aligned}$$

which is a Poisson distribution with expectation $\lambda_Z = \lambda_X + \lambda_Y$.