ASTA

The ASTA team

Contents

1 Contingency tables

1.1 A contingency table

- We return to the dataset `popularKids`, where we study **association** between 2 **factors**: `Goals` and `Urban.Rural`.
- Based on a sample we make a cross tabulation of the factors and we get a so-called **contingency table** (*krydstabel*).

```
import pandas as pd

popKids = pd.read_csv("https://asta.math.aau.dk/datasets?file=PopularKids.dat", sep="\t")
popKids = popKids.rename(columns={'Urban/Rural': 'Urban.Rural'})

tab_totals = pd.crosstab(popKids['Urban.Rural'], popKids['Goals'], margins=True)
tab_totals
```

```
## Goals        Grades  Popular  Sports  All
## Urban.Rural
## Rural            57       50      42  149
## Suburban         87       42      22  151
## Urban           103       49      26  178
## All             247      141      90  478
```

1.1.1 A conditional distribution

- Another representation of data is the percent-wise distribution of `Goals` for each level of `Urban.Rural`, i.e. the sum in each row of the table is 100 (up to rounding):

```
tab = pd.crosstab(popKids['Urban.Rural'], popKids['Goals'], margins=False)
tab_pct = (tab.div(tab.sum(axis=1), axis=0) * 100)
tab_pct['All'] = tab_pct.sum(axis=1)
tab_pct.round()
```

```
## Goals        Grades  Popular  Sports    All
## Urban.Rural
## Rural          38.0     34.0    28.0  100.0
## Suburban       58.0     28.0    15.0  100.0
## Urban          58.0     28.0    15.0  100.0
```

- Here we will talk about the **conditional distribution** of `Goals` given `Urban.Rural`.
- An important question could be:
  - Are the goals of the kids different when they come from urban, suburban or rural areas? I.e. are the rows in the table significantly different?
- There is (almost) no difference between urban and suburban, but it looks like rural is different.

2 Independence

2.1 Independence

- Recall, that two factors are **independent**, when there is no difference between the population's distributions of one factor given the levels of the other factor.
- Otherwise the factors are said to be **dependent**.
- If we e.g. have the following conditional **population distributions** of `Goals` given `Urban.Rural`:

```
##            Goals
## Urban.Rural Grades Popular Sports
##    Rural       500     300    200
##    Suburban    500     300    200
##    Urban       500     300    200
```

- Then the factors `Goals` and `Urban.Rural` are independent.
- We take a sample and "measure" the factors F_1 and F_2. E.g. `Goals` and `Urban.Rural` for a random child.
- The hypothesis of interest today is:

$$H_0 : F_1 \text{ and } F_2 \text{ are independent}, \quad H_a : F_1 \text{ and } F_2 \text{ are dependent.}$$

2.2 The Chi-squared test for independence

- The relative frequencies in the sample gives an estimate of the unconditional distribution of `Goals`:

```
tab = pd.crosstab(popKids['Urban.Rural'], popKids['Goals'])
n = tab.values.sum()
pctGoals = (tab.sum(axis=0) / n * 100).round(1)
pctGoals
```

```
## Goals
## Grades     51.7
## Popular    29.5
## Sports     18.8
## dtype: float64
```

- If we assume independence, then this is also a guess of the conditional distributions of `Goals` given `Urban.Rural`.
- The corresponding expected counts in the sample are then:

```
##            Goals
## Urban.Rural Grades         Popular        Sports         Sum
##    Rural     77.0 (51.7%)  44.0 (29.5%)  28.1 (18.8%) 149.0 (100%)
##    Suburban  78.0 (51.7%)  44.5 (29.5%)  28.4 (18.8%) 151.0 (100%)
##    Urban     92.0 (51.7%)  52.5 (29.5%)  33.5 (18.8%) 178.0 (100%)
##    Sum      247.0 (51.7%) 141.0 (29.5%)  90.0 (18.8%) 478.0 (100%)
```

2.3 Calculation of expected table

```
##            Goals
## Urban.Rural Grades         Popular        Sports         Sum
##    Rural     77.0 (51.7%)  44.0 (29.5%)  28.1 (18.8%) 149.0 (100%)
##    Suburban  78.0 (51.7%)  44.5 (29.5%)  28.4 (18.8%) 151.0 (100%)
##    Urban     92.0 (51.7%)  52.5 (29.5%)  33.5 (18.8%) 178.0 (100%)
##    Sum      247.0 (51.7%) 141.0 (29.5%)  90.0 (18.8%) 478.0 (100%)
```

- We note that

  - The relative frequency for a given column is columnTotal divided by tableTotal. For example `Grades`, which is $\frac{247}{478} = 51.7\%$.
  - The expected value in a given cell in the table is then the cell's relative column frequency multiplied by the cell's rowTotal. For example `Rural` and `Grades`: $149 \times 51.7\% = 77.0$.
- This can be summarized to:
  - The expected value in a cell is the product of the cell's rowTotal and columnTotal divided by tableTotal.

2.4 Chi-squared (χ^2) test statistic

- We have an **observed table**:

```
tab
```

```
## Goals        Grades  Popular  Sports
## Urban.Rural
## Rural            57       50      42
## Suburban         87       42      22
## Urban           103       49      26
```

- And an **expected table**, if H_0 is true:

```
##            Goals
## Urban.Rural Grades Popular Sports Sum
##    Rural     77.0   44.0    28.1  149.0
##    Suburban  78.0   44.5    28.4  151.0
##    Urban     92.0   52.5    33.5  178.0
##    Sum      247.0  141.0    90.0  478.0
```

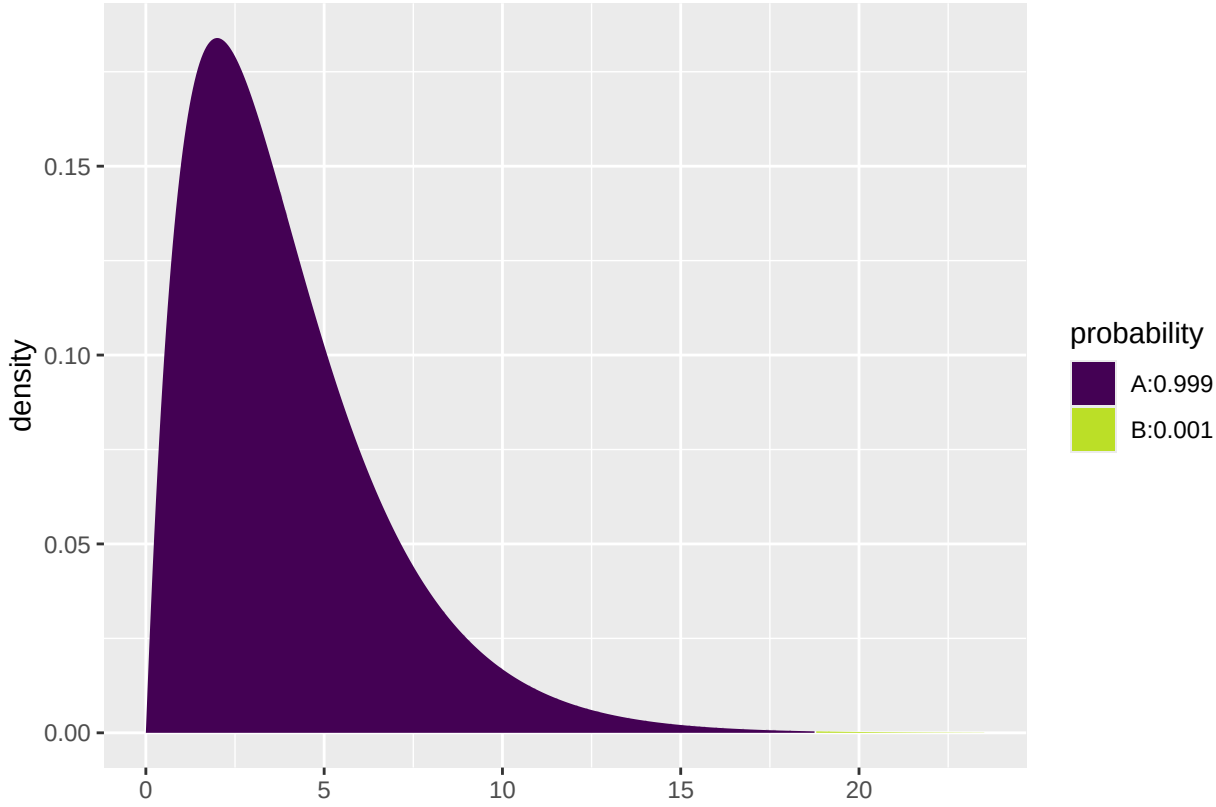
- If these tables are "far from each other", then we reject H_0. We want to measure the distance via the Chi-squared test statistic:
  - $X^2 = \sum \frac{(f_o - f_e)^2}{f_e}$: Sum over all cells in the table
  - f_o is the frequency in a cell in the observed table
  - f_e is the corresponding frequency in the expected table.
- We have:

$$X^2_{obs} = \frac{(57-77)^2}{77} + \ldots + \frac{(26-33.5)^2}{33.5} = 18.8$$

- Is this a large distance??

2.5 χ^2-test template.

- We want to test the hypothesis H_0 of independence in a table with r rows and c columns:
  - We take a sample and calculate X^2_{obs} - the observed value of the test statistic.
  - p-value: Assume H_0 is true. What is then the chance of obtaining a larger X^2 than X^2_{obs}, if we repeat the experiment?
- This can be approximated by the **χ^2-distribution** with $df = (r-1)(c-1)$ degrees of freedom.
- For `Goals` and `Urban.Rural` we have $r = c = 3$, i.e. $df = 4$ and $X^2_{obs} = 18.8$, so the p-value is:




```
from scipy.stats import chi2

1 - chi2.cdf(18.8, 4)
```

```
## np.float64(0.0008603302817890013)
```

- There is clearly a significant association between `Goals` and `Urban.Rural`.

2.6 Perform the test using software

- All of the above calculations can be obtained as follows:

```
import statsmodels.api as sm

tab = pd.crosstab(popKids['Urban.Rural'], popKids['Goals'])
chisqtab = sm.stats.Table(tab)
chisqtab.fittedvalues
```

```
## Goals           Grades    Popular     Sports
## Urban.Rural
## Rural        76.993724  43.951883  28.054393
## Suburban     78.027197  44.541841  28.430962
## Urban        91.979079  52.506276  33.514644
```

```
print(chisqtab.test_nominal_association())
```

```
## df          4
## pvalue      0.0008496551610398528
## statistic   18.827626180696555
```

- The frequency data can also be put directly into a matrix.

```
import numpy as np

data = np.array([
  57, 50, 42,
  87, 42, 22,
  103, 49, 26]).reshape(3,3)
tab = pd.DataFrame(data,
                   index=["Rural", "Suburban", "Urban"],
                   columns=["Grades", "Popular", "Sports"])
tab
```

```
##           Grades  Popular  Sports
## Rural         57       50      42
## Suburban      87       42      22
## Urban        103       49      26
```

```
chisqtab = sm.stats.Table(tab)
chisqtab.fittedvalues
```

```
##              Grades    Popular     Sports
## Rural     76.993724  43.951883  28.054393
## Suburban  78.027197  44.541841  28.430962
## Urban     91.979079  52.506276  33.514644
```

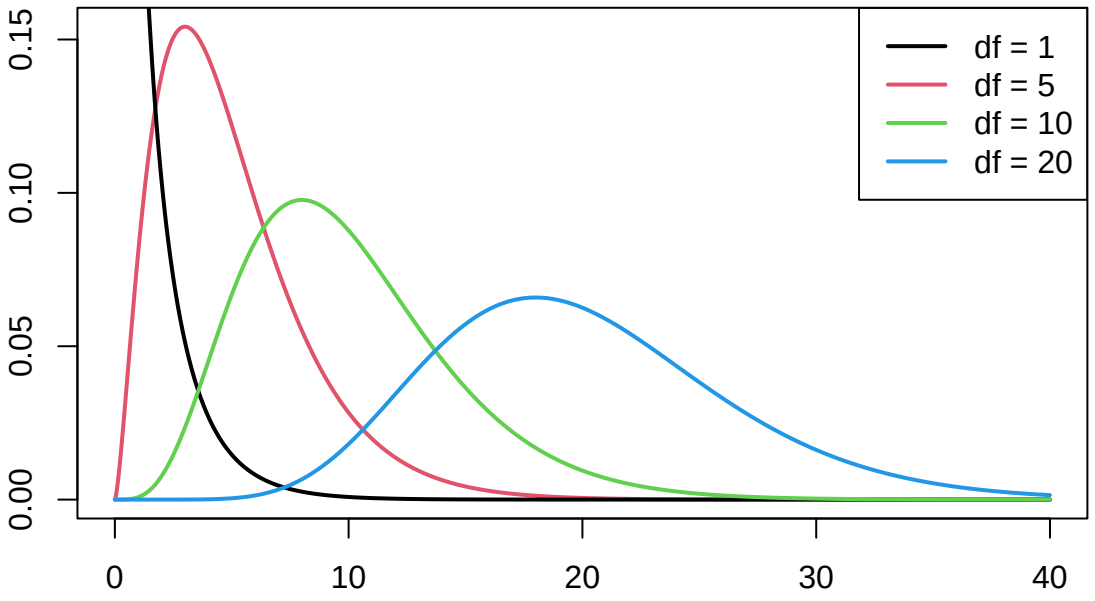
```
print(chisqtab.test_nominal_association())
```

```
## df          4
## pvalue      0.0008496551610398528
## statistic   18.827626180696555
```

3 The χ^2-distribution

3.1 The χ^2-distribution

- The χ^2-distribution with df degrees of freedom:
  - Is never negative.
  - Has mean $\mu = df$
  - Has standard deviation $\sigma = \sqrt{2df}$
  - Is skewed to the right, but approaches a normal distribution when df grows.



4 Agresti - Summary

4.1 Summary

- For the the Chi-squared statistic, X^2, to be appropriate we require that the expected values have to be $f_e \geq 5$.
- Now we can summarize the ingredients in the Chi-squared test for independence.

TABLE 8.5: The Five Parts of the Chi-Squared Test of Independence

1. Assumptions: Two categorical variables, random sampling, $f_e \geq 5$ in all cells
2. Hypotheses: H_0: Statistical independence of variables
   H_a: Statistical dependence of variables
3. Test statistic: $\chi^2 = \sum \frac{(f_o - f_e)^2}{f_e}$, where $f_e = \frac{(\text{Row total})(\text{Column total})}{\text{Total sample size}}$
4. P-value: P = right-tail probability above observed χ^2 value,
   for chi-squared distribution with $df = (r - 1)(c - 1)$
5. Conclusion: Report P-value
   If decision needed, reject H_0 at α-level if $P \leq \alpha$

5 Standardized residuals

5.1 Residual analysis

- If we reject the hypothesis of independence it can be of interest to identify the significant deviations.
- In a given cell in the table, $f_o - f_e$ is the deviation between data and the expected values under the null hypothesis.
- We assume that $f_e \geq 5$.
- If H_0 is true, then the standard error of $f_o - f_e$ is given by

$$se = \sqrt{f_e(1 - \text{rowProportion})(1 - \text{columnProportion})}$$

- The corresponding z-score
$$z = \frac{f_o - f_e}{se}$$
should in 95% of the cells be between ± 2. Values above 3 or below -3 should not appear.
- In popKids table cell `Rural and Grade` we got $f_e = 77.0$ and $f_o = 57$. Here columnProportion= 51.7% and rowProportion= $149/478 = 31.2\%$.
- We can then calculate
$$z = \frac{57 - 77}{\sqrt{77(1 - 0.517)(1 - 0.312)}} = -3.95$$
.
- Compared to the null hypothesis there are way too few rural kids who find grades important.
- In summary: The standardized residuals allow for cell-by-cell (f_e vs f_o) comparision.

5.2 Residual analysis

- We can calculate the standardized residuals:

```
import statsmodels.api as sm

tab = pd.crosstab(popKids['Urban.Rural'], popKids['Goals'])
table = sm.stats.Table(tab)
table.standardized_resids
```

```
## Goals          Grades   Popular    Sports
## Urban.Rural
## Rural       -3.950845  1.309623  3.522500
## Suburban     1.766661 -0.548407 -1.618521
## Urban        2.086578 -0.727433 -1.818622
```

5.3 Why not just use two-way ANOVA ?

- number of persons in different categories are *not* normally distributed
- variance typically larger the larger expected frequency
- underlying data are discrete (for each person, which column and row category does person belong to)
- these discrete variables are naturally modelled in terms of probabilies for different categories
- therefore hypothesis of independence becomes natural null hypothesis
- it is possible to model table frequencies as dependent variable using a regression model but then we need the framework of *generalized linear models* (see last slides)

Contingency table:

- *counts* of how many individuals fall within different categories for two (or more) categorical variables

Two-way ANOVA:

- a number of individuals/objects/... available for each combination of two categorical variables
- next a continuous variable is measured for each individual or object (this becomes the response variable)

6 Models for table data

6.1 Example

- We will study the dataset `HairEyeColor`.

```
HairEyeColor = pd.read_csv("https://asta.math.aau.dk/datasets?file=HairEyeColor.txt", sep="\t")
HairEyeColor.head(6)
```

```
##     Hair    Eye   Sex  Freq
## 0  Black  Brown  Male    32
## 1  Brown  Brown  Male    53
## 2    Red  Brown  Male    10
## 3  Blond  Brown  Male     3
## 4  Black   Blue  Male    11
## 5  Brown   Blue  Male    50
```

- Data is organized such that the variable `Freq` gives the frequency of each combination of the factors `Hair`, `Eye` and `Sex`.
- For example: 32 observations are men with black hair and brown eyes.
- We are interested in the association between eye color and hair color ignoring the sex
- We aggregate data, so we have a table with frequencies for each combination of `Hair` and `Eye`.

```
HairEye = HairEyeColor.groupby(['Eye', 'Hair'], as_index=False)['Freq'].sum()
HairEye
```

```
##       Eye   Hair  Freq
## 0    Blue  Black    20
## 1    Blue  Blond    94
## 2    Blue  Brown    84
## 3    Blue    Red    17
## 4   Brown  Black    68
## 5   Brown  Blond     7
## 6   Brown  Brown   119
## 7   Brown    Red    26
## 8   Green  Black     5
## 9   Green  Blond    16
## 10  Green  Brown    29
## 11  Green    Red    14
## 12  Hazel  Black    15
## 13  Hazel  Blond    10
## 14  Hazel  Brown    54
## 15  Hazel    Red    14
```

6.2 Model specification

- We can write down a model for (the logarithm of) the expected frequencies by using dummy variables z_{e1}, z_{e2}, z_{e3} and z_{h1}, z_{h2}, z_{h3}
- To denote the different levels of `Eye` and `Hair` (the reference level has all dummy variables equal to 0):

$$\log(f_e) = \alpha + \beta_{e1} z_{e1} + \beta_{e2} z_{e2} + \beta_{e3} z_{e3} + \beta_{h1} z_{h1} + \beta_{h2} z_{h2} + \beta_{h3} z_{h3}.$$

- Note that we haven't included an interaction term, which is this case implies, that we assume independence between `Eye` and `Hair` in the model.
- Since our response variable now is a count it is no longer a linear model as we have been used to (linear regression).
- Instead it is a so-called generalized linear model and the relevant command is `glm`.

6.3 Model specification

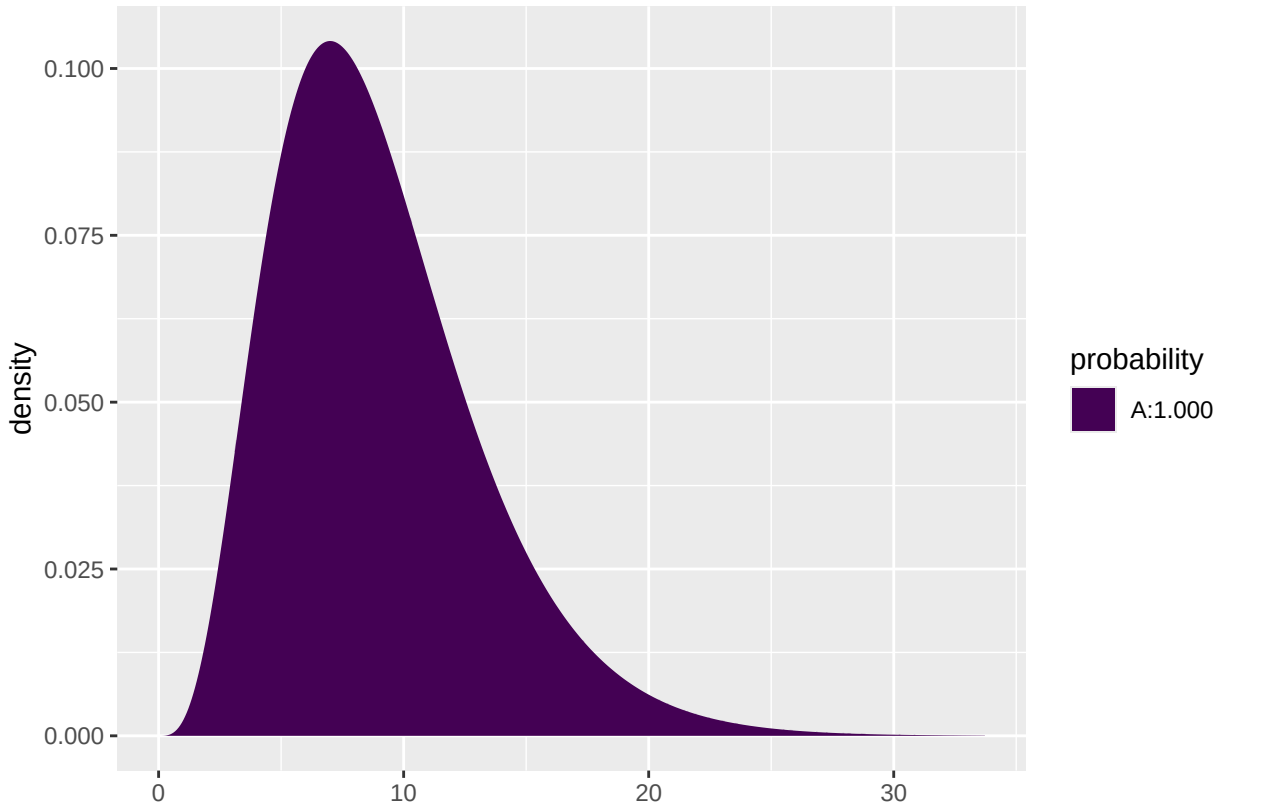
```
import statsmodels.formula.api as smf

model = smf.glm('Freq ~ Hair + Eye', data=HairEye, family=sm.families.Poisson()).fit()
```

- The `family` argument (Poisson) ensures that data are interpreted as discrete counts and not a continuous variable.

```
model.summary()
```

```
## <class 'statsmodels.iolib.summary.Summary'>
## """
##                  Generalized Linear Model Regression Results
## ==============================================================================
## Dep. Variable:                   Freq   No. Observations:                   16
## Model:                            GLM   Df Residuals:                        9
## Model Family:                 Poisson   Df Model:                           6
## Link Function:                    Log   Scale:                          1.0000
## Method:                          IRLS   Log-Likelihood:                -113.52
## Date:                Mon, 03 Nov 2025   Deviance:                       146.44
## Time:                        18:12:23   Pearson chi2:                     138.
## No. Iterations:                     5   Pseudo R-squ. (CS):              1.000
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          z      P>|z|      [0.025      0.975]
## ------------------------------------------------------------------------------
## Intercept      3.6693      0.111     33.191      0.000       3.453       3.886
## Hair[T.Blond]  0.1621      0.131      1.238      0.216      -0.094       0.419
## Hair[T.Brown]  0.9739      0.113      8.623      0.000       0.752       1.195
## Hair[T.Red]   -0.4195      0.153     -2.745      0.006      -0.719      -0.120
## Eye[T.Brown]   0.0230      0.096      0.240      0.811      -0.165       0.211
## Eye[T.Green]  -1.2118      0.142     -8.510      0.000      -1.491      -0.933
## Eye[T.Hazel]  -0.8380      0.124     -6.752      0.000      -1.081      -0.595
## ==============================================================================
## """
```

- A deviance value of $X^2 = 146.44$ with $df = 9$ shows that there is very clear significance and we reject the null hypothesis of independence between hair and eye color.




```
1 - chi2.cdf(146.44, 9)
```

```
## np.float64(0.0)
```

6.4 Expected values and standardized residuals

- We also want to look at expected values and standardized (studentized) residuals.
- The null hypothesis predicts $e^{3.67+0.02} = 40.1$ with brown eyes and black hair, but we have observed 68.
- This is significantly too many, since the standardized residual is 6.1.
- The null hypothesis predicts 47.2 with brown eyes and blond hair, but we have seen 7. This is significantly too few, since the standardized residual is -8.3.

```
HairEye['fitted'] = model.fittedvalues
HairEye['resid'] = model.get_influence().resid_studentized
HairEye
```

```
##       Eye   Hair  Freq      fitted     resid
## 0    Blue  Black    20   39.222973 -4.253816
## 1    Blue  Blond    94   46.123311  9.967550
## 2    Blue  Brown    84  103.868243 -3.397883
## 3    Blue    Red    17   25.785473 -2.311052
## 4   Brown  Black    68   40.135135  6.136520
## 5   Brown  Blond     7   47.195946 -8.328248
## 6   Brown  Brown   119  106.283784  2.164282
## 7   Brown    Red    26   26.385135 -0.100824
## 8   Green  Black     5   11.675676 -2.287896
## 9   Green  Blond    16   13.729730  0.732023
## 10  Green  Brown    29   30.918919 -0.508263
```

```
## 11  Green    Red    14    7.675676  2.576569
## 12  Hazel  Black    15   16.966216 -0.575026
## 13  Hazel  Blond    10   19.951014 -2.737977
## 14  Hazel  Brown    54   44.929054  2.050216
## 15  Hazel    Red    14   11.153716  0.989512
```

7 Introduction to logistic regression

7.1 Binary response

- We consider a binary response y with outcome 1 or 0. This might be a code indicating whether a person is able or unable to perform a given task.
- Furthermore, we are given an explanatory variable x, which is numeric, e.g. age.
- We shall study models for
$$P(y = 1 \,|\, x)$$
i.e. the probability that a person of age x is able to complete the task.
- We shall see methods for determining whether or not age actually influences the probability, i.e. is y independent of x?

7.2 A linear model

$$P(y = 1 \,|\, x) = \alpha + \beta x$$

is simple, but often inappropiate. If β is positive and x sufficiently large, then the probability exceeds 1.

8 Simple logistic regression

8.1 Logistic model

Instead we consider the **odds** that the person is able to complete the task

$$\texttt{Odds}(y = 1 \,|\, x) = \frac{P(y = 1 \,|\, x)}{P(y = 0 \,|\, x)} = \frac{P(y = 1 \,|\, x)}{1 - P(y = 1 \,|\, x)}$$

which can have any positive value.

The logistic model is defined as:

$$\texttt{logit}(P(y = 1 \,|\, x)) = \log(\texttt{Odds}(y = 1 \,|\, x)) = \alpha + \beta x$$

The function $\texttt{logit}(p) = \log(\frac{p}{1-p})$ - i.e. **log of odds** - is termed **the logistic transformation**.

Remark that log odds can be any number, where zero corresponds to $P(y = 1 \,|\, x) = 0.5$. Solving $\alpha + \beta x = 0$ shows that at age $x_0 = -\alpha/\beta$ you have fifty-fifty chance of solving the task.

8.2 Logistic transformation

```
import numpy as np
from scipy.special import logit, expit

p = np.arange(0.1, 1.0, 0.2)  # stop is exclusive, so use 1.0
p
```

```
## array([0.1, 0.3, 0.5, 0.7, 0.9])
```

```
l = logit(p)
l.round(3)
```

```
## array([-2.197, -0.847,  0.   ,  0.847,  2.197])
```

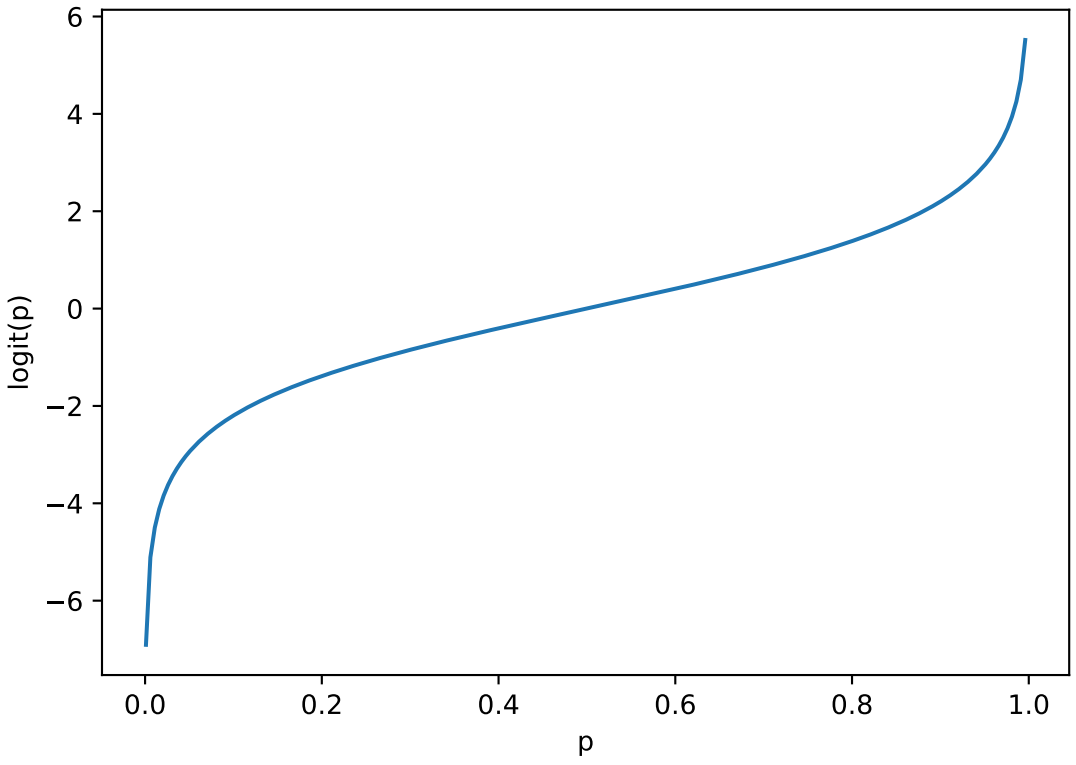
```
expit(l)
```

```
## array([0.1, 0.3, 0.5, 0.7, 0.9])
```

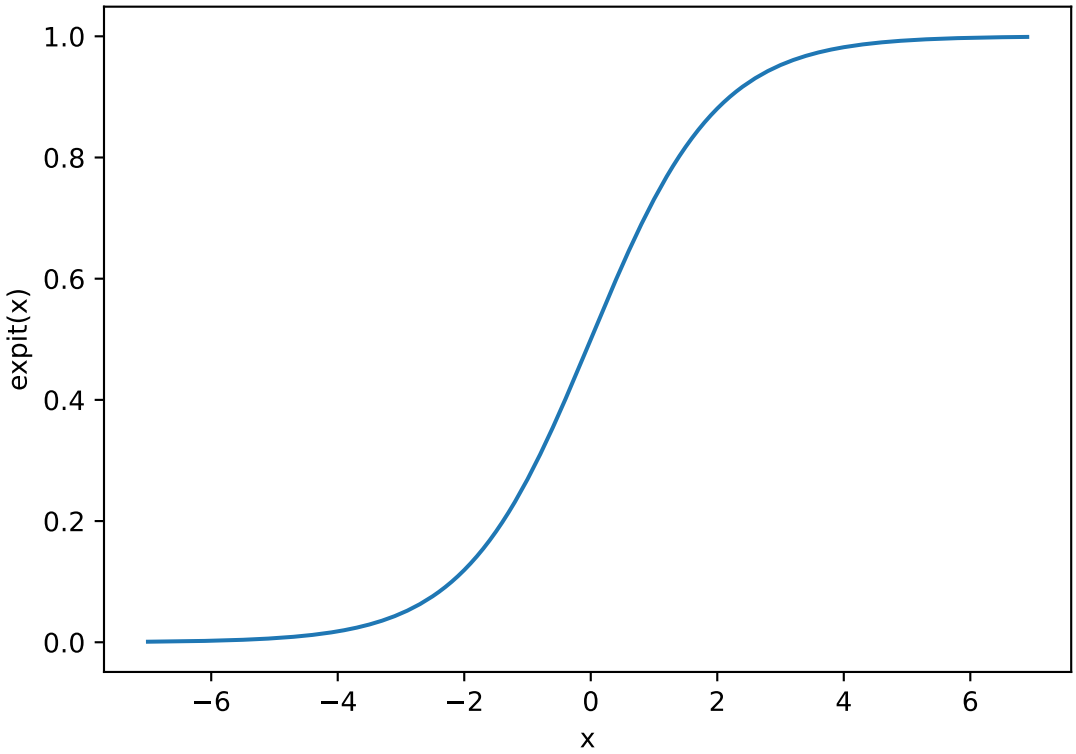
- The inverse logistic transformation `expit()` applied to the transformed values can recover the original probabilities:

Plot of logistic function and inverse logistic

```
import matplotlib.pyplot as plt

p = np.arange(0.001, 0.999, 0.005)
fit = plt.plot(p, logit(p), label='logit(p)')
plt.xlabel('p')
plt.ylabel('logit(p)')
plt.show()
```




```
x = np.arange(-7, 7, 0.1)
fig = plt.plot(x, expit(x), label='inverse logit (expit)')
plt.xlabel('x')
plt.ylabel('expit(x)')
```



8.3 Odds-ratio

Interpretation of β:

What happens to odds, if we increase age by 1 year?

Consider the so-called **odds-ratio**:

$$\frac{\texttt{Odds}(y = 1 \,|\, x + 1)}{\texttt{Odds}(y = 1 \,|\, x)} = \frac{\exp(\alpha + \beta(x + 1))}{\exp(\alpha + \beta x)} = \exp(\beta)$$

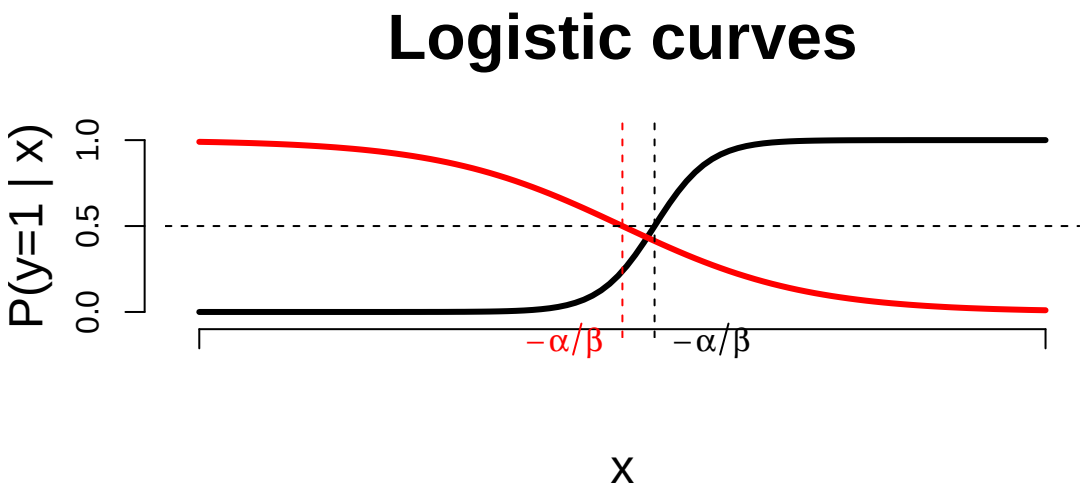
where we see, that $\exp(\beta)$ equals the odds for age $x + 1$ relative to odds at age x.

This means that when age increase by 1 year, then the relative change

$$\frac{\exp(\alpha + \beta(x + 1)) - \exp(\alpha + \beta x)}{\exp(\alpha + \beta x)}$$

in odds is given by $100(\exp(\beta) - 1)\%$.

8.4 Simple logistic regression



Examples of logistic curves for $P(y = 1|x)$. The black curve has a positive β-value (=10), whereas the red has a negative β (=-3).

In addition we note that:

- Increasing the absolute value of β yields a steeper curve.
- When $P(y = 1 \,|\, x) = \frac{1}{2}$ then logit is zero, i.e. $\alpha + \beta x = 0$.

This means that at age $x = -\frac{\alpha}{\beta}$ you have 50% chance to perform the task.

8.5 Example: Credit card data

We shall investigate if income is a good predictor of whether or not you have a credit card.

- Data structure: For each level of income, we let `n` denote the number of persons with that income, and `credit` how many of these that carries a credit card.

```
import pandas as pd

creInc = pd.read_csv("https://asta.math.aau.dk/datasets?file=income-credit.csv", sep=',')
creInc.head(6)
```

```
##    Income   n  credit
## 0      12   1       0
## 1      13   1       0
## 2      14   8       2
## 3      15  14       2
## 4      16   9       0
## 5      17   8       2
```

8.6 Example: Fitting the model

```
import statsmodels.api as sm
import statsmodels.formula.api as smf

modelFit = smf.glm(formula='credit + I(n - credit) ~ Income',
                   data=creInc,
                   family=sm.families.Binomial()).fit()
modelFit.summary()
```

```
## <class 'statsmodels.iolib.summary.Summary'>
## """
##                      Generalized Linear Model Regression Results
## ==============================================================================================
## Dep. Variable:     ['credit', 'I(n - credit)']   No. Observations:                   24
## Model:                                     GLM   Df Residuals:                       22
## Model Family:                         Binomial   Df Model:                           1
## Link Function:                           Logit   Scale:                         1.0000
## Method:                                   IRLS   Log-Likelihood:                -27.417
## Date:                         Mon, 03 Nov 2025   Deviance:                       39.276
## Time:                                 18:12:26   Pearson chi2:                     32.3
## No. Iterations:                              5   Pseudo R-squ. (CS):             0.6698
## Covariance Type:                     nonrobust
## ==============================================================================
##                  coef    std err          z      P>|z|      [0.025      0.975]
## ------------------------------------------------------------------------------
## Intercept     -3.5179      0.710     -4.953      0.000      -4.910      -2.126
## Income         0.1054      0.026      4.030      0.000       0.054       0.157
## ==============================================================================
## """
```

- The response has the form `credit + I(n - credit)`.
- We need to use the function `glm` (generalized linear model).
- The argument `family=sm.families.Binomial()` tells the function that the data has binomial variation. Leaving out this argument will lead Python to believe that data follows a normal distribution (linear regression).
- The `params` extracts the coefficients (estimates of parameters) from the model:

```
modelFit.params
```

```
## Intercept   -3.517947
## Income       0.105409
## dtype: float64
```

8.7 Test of no effect

```
modelFit.summary2().tables[1]
```

```
##               Coef.  Std.Err.         z         P>|z|    [0.025    0.975]
## Intercept -3.517947  0.710336 -4.952513  7.326117e-07 -4.910179 -2.125714
## Income     0.105409  0.026157  4.029788  5.582714e-05  0.054141  0.156677
```

Our model for dependence of odds of having a credit card related to income(x) is

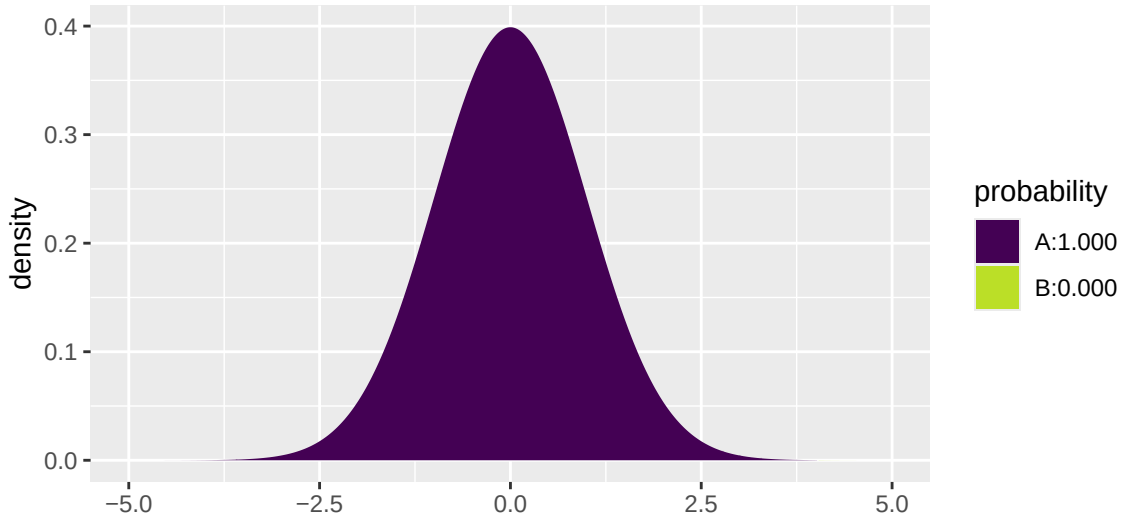
$$\texttt{logit}(x) = \alpha + \beta x$$

The hypothesis of no relation between income and ability to obtain a credit card corresponds to

$$H_0: \quad \beta = 0$$

with the alternative $\beta \neq 0$. Inspecting the summary reveals that $\hat{\beta} = 0.1054$ is more than 4 standard errors away from zero.

With a z-score equal to 4.03 we get the tail probability




```
from scipy.stats import norm

ptail = 2 * (1 - norm.cdf(4.03))
ptail
```

```
## np.float64(5.577685288105094e-05)
```

Which is very significant - as reflected by the p-value.

8.8 Confidence interval for odds ratio

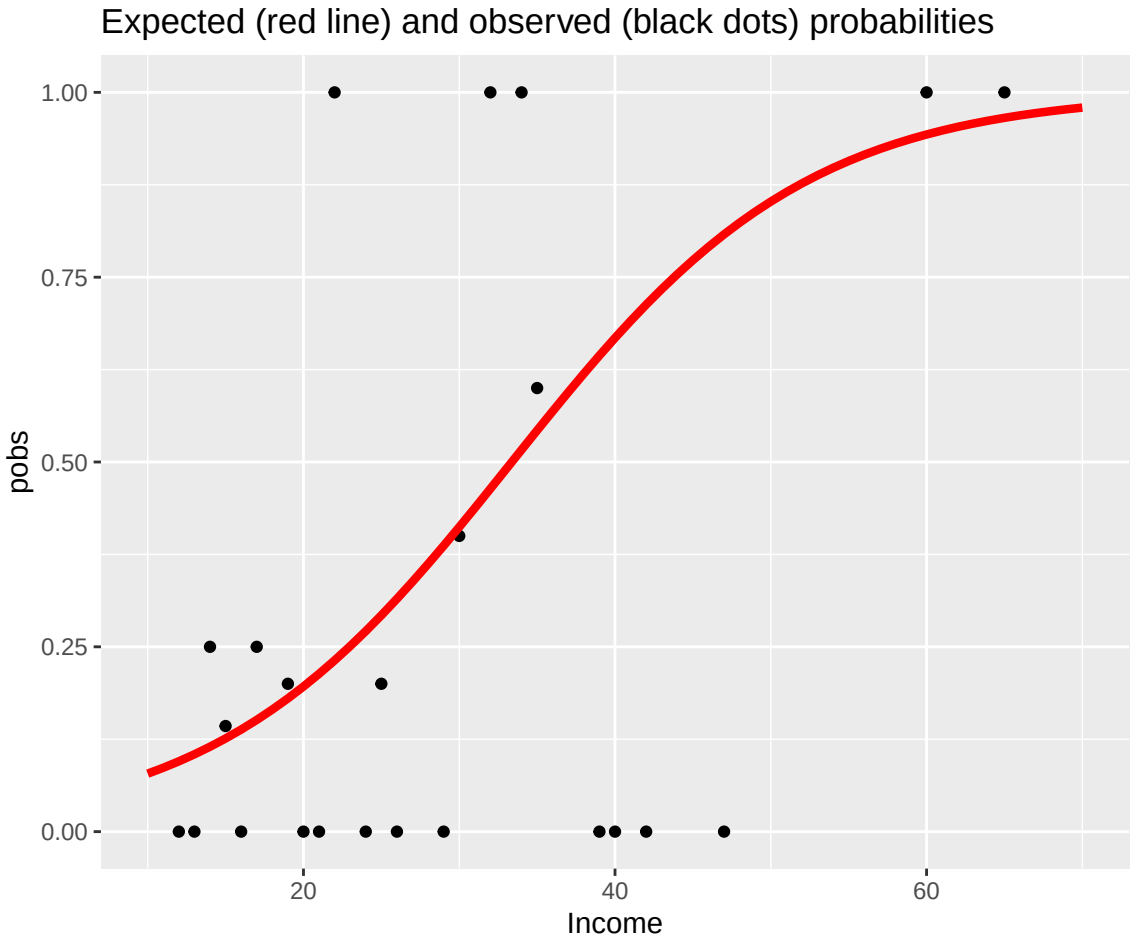
From the summary:

- $\hat{\beta} = 0.10541$ and hence $\exp(\hat{\beta}) - 1 = 0.11$. If income increases by 1000 euro, then odds increases by 11%.
- Standard error on $\hat{\beta}$ is 0.02616 and hence a 95% confidence interval for log-odds ratio is $\hat{\beta} \pm 1.96 \times 0.02616 = (0.054; 0,157)$.
- Corresponding interval for odds ratio: $\exp((0.054; 0,157)) = (1.056; 1.170)$,

  i.e. the increase in odds is - with confidence 95% - between 5.6% and 17%.

8.9 Plot of model predictions against actual data

```
## Ignoring unknown labels:
## * ylab : "Probability of credit card"
## * xlab : "Income"
```



- Tendency is fairly clear and very significant.
- Due to low sample size at some income levels, the deviations are quite large.

9 Multiple logistic regression

9.1 Several numeric predictors

We generalize the model to the case, where we have k predictors $x_1, x_2, \ldots, x_k$. Where some might be dummies for a factor.

$$\texttt{logit}(P(y = 1 \mid x_1, x_2, \ldots, x_k)) = \alpha + \beta_1 x_1 + \cdots + \beta_k x_k$$

Interpretation of β-values is unaltered: If we fix $x_2, \ldots, x_k$ and increase x_1 by one unit, then the relative change in odds is given by $\exp(\beta_1) - 1$.

9.2 Example

Wisconsin Breast Cancer Database covers 683 observations of 10 variables in relation to examining tumors in the breast.

- Nine clinical variables with a score between 0 and 10.
- The binary variable `Class` with levels `benign/malignant`.
- By default `Python` orders the levels lexicografically and chooses the first level as reference ($y = 0$). Hence `benign` is reference, and we model odds of `malignant`.

We shall work with only 4 of the predictors, where two of these have been discretized.

```
BC = pd.read_csv("https://asta.math.aau.dk/datasets?file=BC0.dat", sep=' ')
BC.head(6)
```

```
##    nuclei  cromatin  Size.low  Size.medium  Shape.low      Class
## 0       1         3      True        False       True     benign
## 1      10         3     False         True      False     benign
## 2       2         3      True        False       True     benign
## 3       4         3     False        False      False     benign
## 4       1         3      True        False       True     benign
## 5      10         9     False        False      False  malignant
```

9.3 Global test of no effects

First we fit the model `mainEffects` with main effect of all predictors - remember the notation `~ .` for all predictors. Then we fit the model `noEffects` with no predictors.

```
BC['Class_numeric'] = (BC['Class'] == 'malignant').astype(int)
BC = BC.rename(columns={
    'Size.low': 'Size_low',
    'Size.medium': 'Size_medium',
    'Shape.low': 'Shape_low'
})
mainEffects = smf.glm('Class_numeric ~ nuclei + cromatin + Size_low + Size_medium + Shape_low', data=BC
noEffects = smf.glm('Class_numeric ~ 1', data=BC, family=sm.families.Binomial()).fit()
```

First we want to test, whether there is any effect of the predictors, i.e the null hypothesis

$$H_0: \quad \beta_1 = \beta_2 = \beta_3 = \beta_4 = \beta_5 = 0$$

9.4 Example

We test the hypothesis that all $\beta_i = 0$ using a χ^2-test.

```
from scipy.stats import chi2

ll_noEffects = noEffects.llf
ll_mainEffects = mainEffects.llf
test_stat = -2 * (ll_noEffects - ll_mainEffects)
df = mainEffects.df_model - noEffects.df_model
p_value = 1 - chi2.cdf(test_stat, df)
print(f"Test statistic: {test_stat}")
```

```
## Test statistic: 749.2859276261477
```

```
print(f"Degrees of freedom: {df}")
```

```
## Degrees of freedom: 5
```

```
print(f"P-value: {p_value}")
```

```
## P-value: 0.0
```

`mainEffects` is a much better model.

The test statistic is the Deviance (749.29), which should be small.

It is evaluated in a chi-square with 5 (the number of parameters equal to zero under the nul hypothesis) degrees of freedom.

The 95%-critical value for the $\chi^2(5)$ distribution is 11.07 and the p-value is in practice zero.

9.5 Test of influence of a given predictor

```
mainEffects.summary2().tables[1].round(4)
```

```
##                        Coef.  Std.Err.       z   P>|z|  [0.025  0.975]
## Intercept           -0.7090    0.8570 -0.8274  0.4080 -2.3887  0.9706
## Size_low[T.True]    -3.6154    0.8081 -4.4739  0.0000 -5.1992 -2.0315
## Size_medium[T.True] -2.3773    0.7188 -3.3073  0.0009 -3.7861 -0.9685
## Shape_low[T.True]   -2.1490    0.6054 -3.5496  0.0004 -3.3356 -0.9624
## nuclei               0.4403    0.0823  5.3483  0.0000  0.2790  0.6017
## cromatin             0.5058    0.1444  3.5025  0.0005  0.2228  0.7888
```

For each predictor p can we test the hypothesis:

$$H_0: \quad \beta_p = 0$$

- Looking at the z-values, there is a clear effect of all 5 predictors. Which of course is also supported by the p-values.

9.6 Prediction and classification

```
BC['pred'] = mainEffects.predict()
```

- We add the column `pred` to our dataframe `BC`.
- `pred` is the final model's estimate of the probability of `malignant`.

```
BC[['Class', 'pred']].head(6)
```

```
##        Class      pred
## 0     benign  0.010817
## 1     benign  0.944507
## 2     benign  0.016702
## 3     benign  0.928883
## 4     benign  0.010817
## 5  malignant  0.999738
```

Not good for patients 2 and 4.

We may classify by `round(BC$pred)`:

- 0 to denote `benign` (probability `BC$pred` less than 0.5)
- 1 to denote `malignant` (probability `BC$pred` more than 0.5)

```
BC['pred_class'] = np.where(BC['pred'] > 0.5, "pred_malignant", "pred_benign")
tab = pd.crosstab(BC['Class'], BC['pred_class'])
tab
```

```
## pred_class  pred_benign  pred_malignant
## Class
## benign              433              11
## malignant            11             228
```

11+11=22 patients are misclassified.

```
malignant_preds = BC.loc[BC['Class'] == 'malignant', 'pred']
malignant_preds_sorted = malignant_preds.sort_values().head(5)
malignant_preds_sorted
```

```
## 440    0.034703
## 216    0.036964
## 63     0.088544
## 474    0.189870
## 55     0.205024
## Name: pred, dtype: float64
```

There is a malignant woman with a predicted probability of malignancy, which is only 3.5%.

If we assign all women with predicted probability of malignancy above 5% to further investigation, then we only miss two malignant.

```
BC['pred_class'] = np.where(BC['pred'] > 0.05, "pred_malignant", "pred_benign")
tab = pd.crosstab(BC['Class'], BC['pred_class'])
tab
```

```
## pred_class  pred_benign  pred_malignant
## Class
## benign              394              50
## malignant             2             237
```

The expense is that the number of false positive increases from 11 to 50.

```
BC['pred_class'] = np.where(BC['pred'] > 0.1, "pred_malignant", "pred_benign")
tab = pd.crosstab(BC['Class'], BC['pred_class'])
tab
```

```
## pred_class  pred_benign  pred_malignant
## Class
## benign              417              27
## malignant             3             236
```

- If we instead set the alarm to 10%, then the number of false positives decreases from 50 to 27.
- But at the expense of 3 false negative (instead of 2 as before).