# Lecture 3

Deeper into Generalized Linear Models

# Project FAQs

- Should we use a classification or regression dataset?
  - The project should look at the dataset from multiple angles
  - In the context of statistical distributions, the difference is not as defined
- What if our data doesn’t follow one of the distributions / doesn’t have linear relationships
  - Try it and demonstrate that it doesn’t work, discuss why
  - We will cover non-linear regressions in lecture 4+
  - We will cover non-parametric methods in lectures 5+

# Review of GLMs

- Generalized Linear Models (GLMs) predict a distribution in response to independent variables

- A GLM has three properties
  - A value $y$ (dependent variable) is generated from a distribution
  - The mean of the distribution depends on some independent variables $X$
  - The link between $X$ and the mean $\mu$ is called a **link function**

# Example

| Awards | Program | Math score |
|---|---|---|
| 6 | Vocational | 69 |
| 2 | Academic | 61 |
| 4 | Vocational | 71 |
| 3 | General | 62 |
| 0 | Academic | 70 |
| 1 | General | 50 |
| 2 | Academic | 58 |
| 2 | Vocational | 60 |
| 0 | Academic | 64 |
| 0 | General | 57 |
| 1 | Vocational | 65 |
| 2 | Academic | 57 |
| 0 | General | 64 |
| 1 | Academic | 73 |

2 features

14 obs

I want to predict the number of awards a student will win over their academic career.

I have two data points

1. Their academic program, either "academic", "general", or "vocational"
2. Their most recent score in a math class

How can I determine if each of these features is helpful in my predictive model?

14-2 = 12 DOF

# Deviance

- Deviance is a model for the lack of fit. It is defined by the formula

$$D = 2(l(\theta) - l(\hat{\theta})$$

  - D is the deviance
  - $l(\theta)$ is the log-likelihood of the fitted model
  - $l(\hat{\theta})$ is the log-likelihood of the saturated model

- Notice that this looks a lot like the formula for the likelihood ratio test in lecture 1!

# Scaled, Residual, and Null Deviance

$N_{obs} - N_{features}$

- **Scaled deviance** is the deviance divided by the degrees of freedom
  - When introducing new features, both deviance and degrees of freedom go down
  - If scaled deviance goes down, the model is probably getting better!

- **Null deviance** is the deviance of a model with no predictors (i.e. only an intercept)

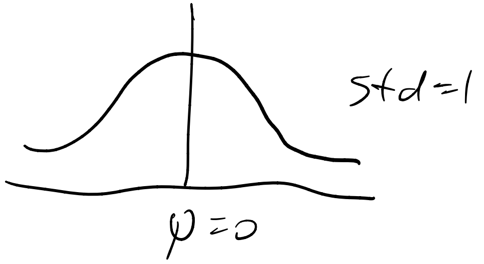
- **Residual deviance** is the deviance of a fitted model with predictors

# Analysis of Deviance using Chi-Square

- We can compare our residual deviance against a chi-square distribution for **some** distributions.
  - **Notable exception:** logistic regression

- The expected value of a χ2 random variable is equal to its degrees of freedom, a basic model check consists of comparing the residual deviance to its degrees of freedom

Lower is better



# Fitting our example data

```
endog = pd.concat([pd.get_dummies(df['program'], drop_first=True),
                      df['math']], axis=1)
model = sm.GLM(df['awards'], endog, family=sm.families.Poisson())
result = model.fit()

print(result.deviance)
>>> 189.4496199102934
print(result.null_deviance)
>>> 287.6722344528648
```

DOF ≈ 250

result. DOF

→ residual deviance

| X | Y |
|---|---|
| Black | .... |
| Blue | .... |
| Red | .... |
| Black | .... |

| Black | Blue | Red |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| 1 | 0 | 0 |

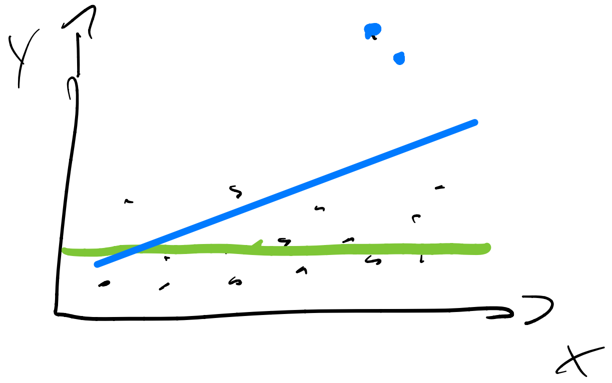
$y = Ax + b$

$2 = 1 \cdot 1 + 1$

$\quad \dots$

$= 5 \cdot 1 - 4$

# Bootstrapping




- Bootstrapping is the process of taking many samples from a dataset and generating a model on each sample

- Repeating this process a high number of times can give a distribution of each parameter value

- Analyzing these distributions can give a better understanding of the model results

# Example

Endog (enous) = y
= variable from the model
Exog (enous) = X
= variable outside the model

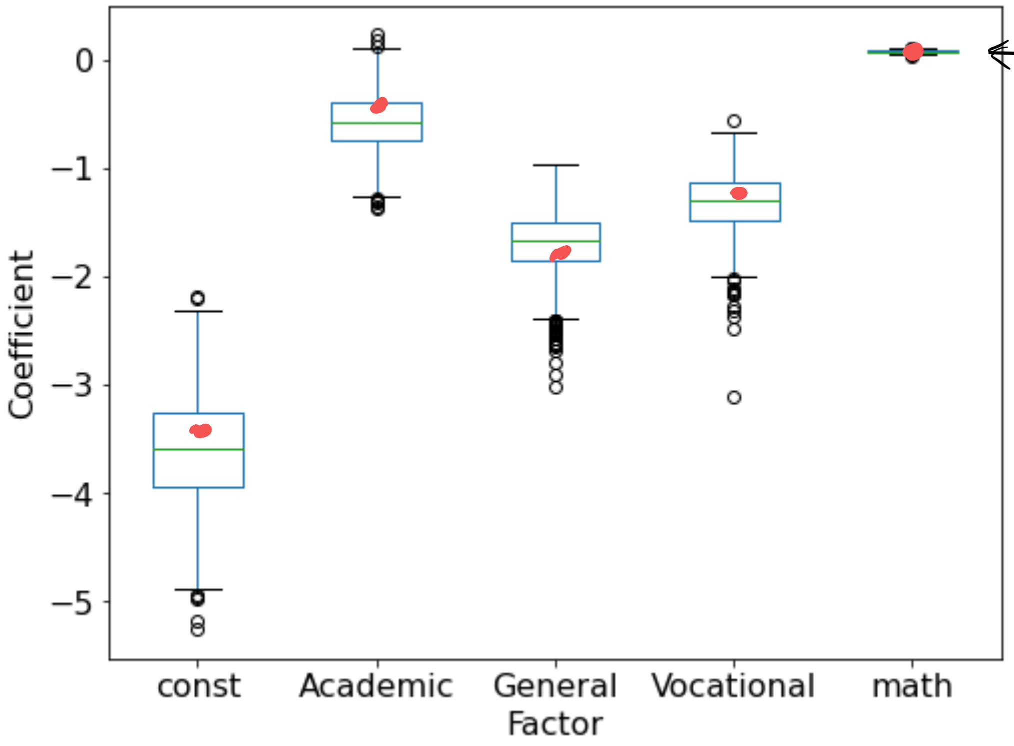
dataset size

```
all_results = {}
for e in range(1000):
    sample = df.sample(200, replace=True)
    endog = pd.concat([pd.get_dummies(sample['program']),
    exog
                       sample['math']],
    exog       axis=1)
                                    exog
    endog = sm.add_constant(endog)      exog
    model = sm.GLM(sample['awards'], endog, family=sm.families.Poisson())
    result = model.fit()  exog
    params = dict(zip(endog.columns, result._results.params))
    all_results[e] = params
```

# Bootstrapping Results

● → Value from the model trained on all data

Can adjust coefficient by range of variable



# Another Example

I am a data scientist working in a (cookie) manufacturing environment. The factory has a piece of equipment that has frequent stops and I am tasked with predicting when these stops will occur.

Stops are broken down in minor (a few seconds), short (a few minutes), and major (a few hours).

I am concerned with predicting time until the next major downtime. I want to know if some shifts or products are having more breakdowns than others.

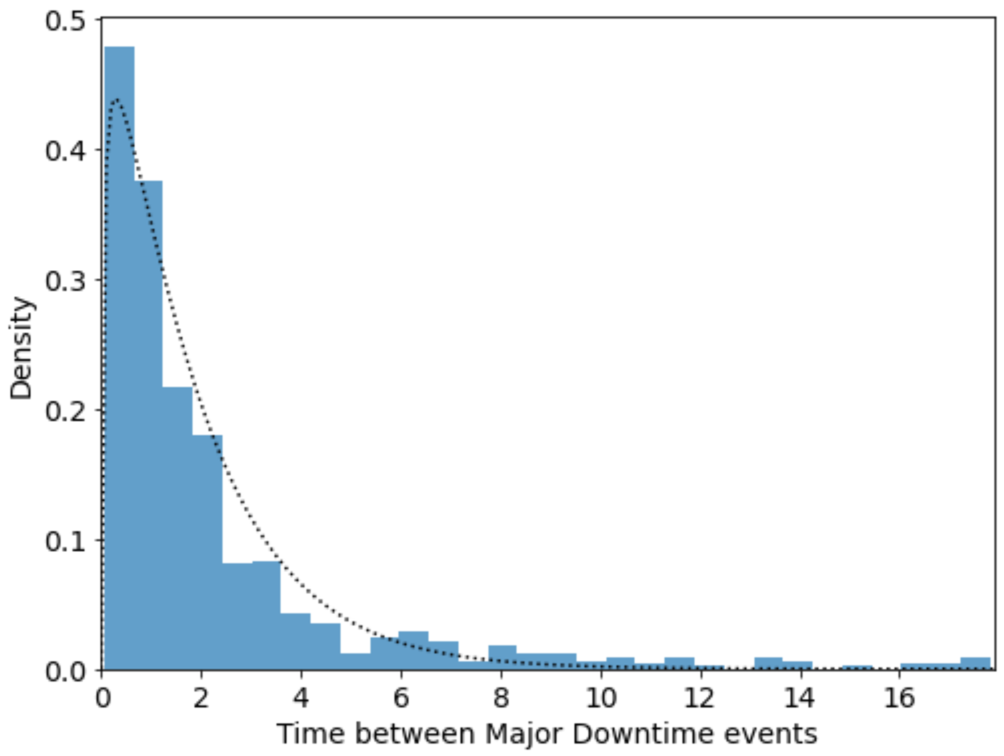
| startTime | downtimeType |
|---|---|
| 8/24/2023 15:57 | Minor |
| 8/24/2023 15:58 | Short |
| 8/24/2023 16:02 | Short |
| 8/24/2023 16:13 | Minor |
| 8/24/2023 16:57 | Minor |
| 8/24/2023 17:05 | Minor |
| 8/24/2023 17:23 | Minor |
| 8/24/2023 17:25 | Minor |
| 8/24/2023 17:26 | Short |
| 8/24/2023 17:29 | Minor |
| 8/24/2023 17:34 | Short |
| 8/24/2023 17:41 | Short |
| 8/24/2023 17:47 | Minor |
| 8/24/2023 18:26 | Short |
| 8/24/2023 18:29 | Minor |
| 8/24/2023 18:42 | Minor |
| 8/24/2023 18:43 | Short |
| 8/24/2023 18:47 | Minor |
| **8/24/2023 18:54** | **Major** |
| … | … |

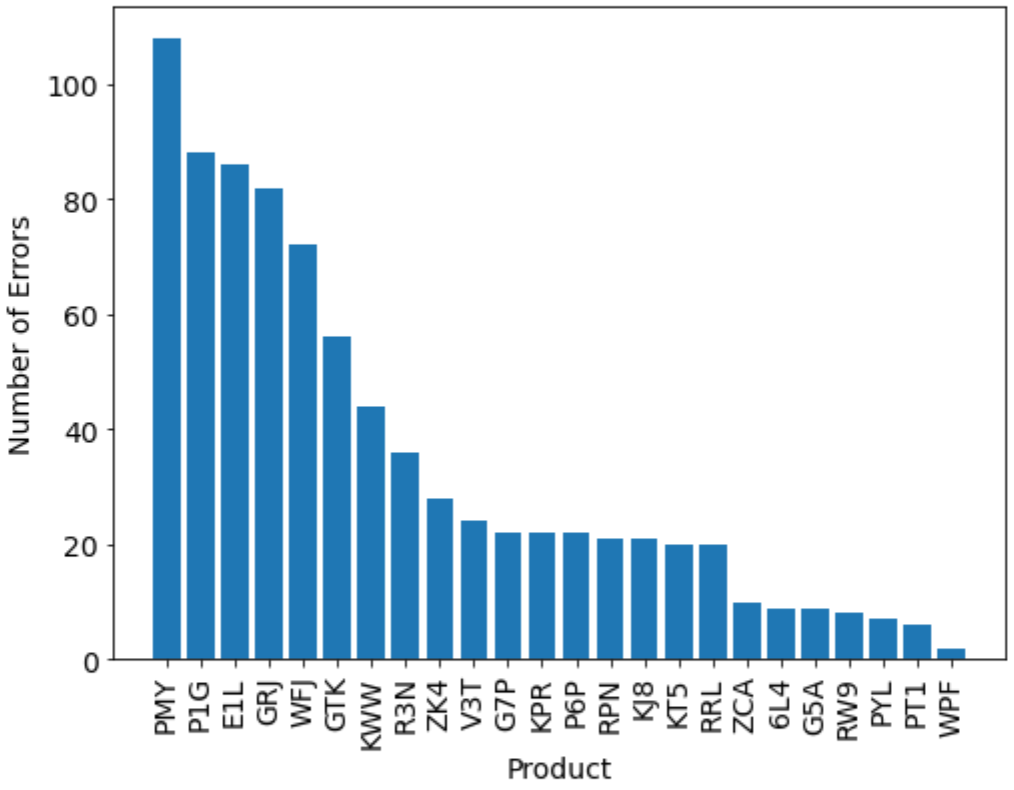
# Example – Data Processing

```
# df is the original data
reason = 'Major Downtime’
downtime_df = df.loc[df['reasonLvl1']==reason]
time_between_failures = downtime_df['startDT'].diff().iloc[1:].dt.seconds / 3600

shift_values = downtime_df['shift'].iloc[1:]
product_keys = downtime_df['productCode’].iloc[1:]

print(pd.concat([shift_values, product_keys, time_between_failures], axis=1))
```

| Shift | Product | Time Between Failures |
|---|---|---|
| 1 | KT5 | 16.2731 |
| 1 | KT5 | 1.39139 |
| 1 | KT5 | 0.31306 |
| 1 | KT5 | 1.14111 |
| 1 | KT5 | 0.7375 |

# Example – Data Exploration



# Example – Fitting GLM

```
from statsmodels.genmod.families.links import Identity
exog
endog = pd.concat([pd.get_dummies(shift_values),
                      pd.get_dummies(product_keys)],
                      axis=1)
endog
exog = time_between_failures

model = sm.GLM(exog, endog, family=sm.families.Gamma(link=Identity()))

result = model.fit()
result.summary()
```

# GLM Results – First Attempt

Generalized Linear Model Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | startDT | No. Observations: | 823 |
| Model: | GLM | Df Residuals: | 797 |
| Model Family: | Gamma | Df Model: | 25 |
| Link Function: | Identity | Scale: | 1.8311 |
| Method: | IRLS | Log-Likelihood: | -1634.0 |
| Date: | Mon, 12 Feb 2024 | Deviance: | 982.91 |
| Time: | 11:15:09 | Pearson chi2: | 1.45e+03 |
| No. Iterations: | 100 | Pseudo R-squ. (CS): | 0.04799 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| 1.0 | 2.9149 | 0.489 | 5.956 | 0.000 | 1.956 | 3.874 |
| 2.0 | 2.8772 | 0.497 | 5.787 | 0.000 | 1.903 | 3.852 |
| 3.0 | 2.9600 | 0.483 | 6.122 | 0.000 | 2.012 | 3.908 |
| 6L4 | 3.5166 | 2.918 | 1.205 | 0.228 | -2.203 | 9.236 |
| E1L | -0.5144 | 0.571 | -0.901 | 0.368 | -1.634 | 0.605 |
| G5A | -1.4648 | 0.789 | -1.856 | 0.064 | -3.012 | 0.082 |
| G7P | -0.8927 | 0.734 | -1.216 | 0.224 | -2.331 | 0.546 |
| GRJ | -1.1441 | 0.533 | -2.146 | 0.032 | -2.189 | -0.099 |
| GTK | -1.0653 | 0.574 | -1.857 | 0.063 | -2.190 | 0.059 |
| KJ8 | 0.2789 | 1.002 | 0.278 | 0.781 | -1.684 | 2.242 |
| KPR | 0.1295 | 0.959 | 0.135 | 0.893 | -1.750 | 2.009 |

Deviance = 983
Null deviance = 1057

# GLM Results – Second Attempt

Generalized Linear Model Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | startDT | No. Observations: | 823 |
| Model: | GLM | Df Residuals: | 820 |
| Model Family: | Gamma | Df Model: | 2 |
| Link Function: | Identity | Scale: | 2.0645 |
| Method: | IRLS | Log-Likelihood: | -1683.5 |
| Date: | Mon, 12 Feb 2024 | Deviance: | 1055.0 |
| Time: | 11:20:19 | Pearson chi2: | 1.69e+03 |
| No. Iterations: | 4 | Pseudo R-squ. (CS): | 0.001178 |
| Covariance Type: | nonrobust | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| 1.0 | 2.4069 | 0.200 | 12.035 | 0.000 | 2.015 | 2.799 |
| 2.0 | 2.5590 | 0.242 | 10.578 | 0.000 | 2.085 | 3.033 |
| 3.0 | 2.7039 | 0.227 | 11.913 | 0.000 | 2.259 | 3.149 |

| 1.0 | 2.0 | 3.0 | sum |
|---|---|---|---|

Deviance = 1055

Null deviance = 1057

# Example Takeaways

- The time between errors follows a Gamma distribution with a rate parameter of ~2.5

- Neither shift nor product type has much impact on the error rate

# Collinearity

- Collinearity occurs when two or more variables are closely related

**Related measurements**

| Speed (m/s) | Time (s) |
|---|---|
| 6.22 | 1.61 |
| 5.33 | 1.88 |
| 3.64 | 2.74 |
| 4.99 | 2.01 |
| 6.22 | 1.61 |
| 6.61 | 1.51 |
| 5.17 | 1.93 |
| 3.33 | 3.01 |
| 4.75 | 2.11 |
| 4.66 | 2.15 |
| 3.43 | 2.92 |

**Linear relationships**

| Lap 1 Time (s) | Lap 2 Time (s) | Lap 3 Time (s) | Total Time (s) |
|---|---|---|---|
| 29.30 | 29.22 | 33.05 | 91.57 |
| 29.40 | 32.06 | 32.95 | 94.41 |
| 32.09 | 28.06 | 27.38 | 87.54 |
| 32.20 | 29.87 | 30.84 | 92.92 |
| 32.80 | 27.96 | 28.38 | 89.14 |
| 29.66 | 31.33 | 31.37 | 92.36 |
| 28.47 | 28.09 | 28.83 | 85.39 |
| 30.26 | 29.17 | 34.32 | 93.75 |
| 26.08 | 29.69 | 29.13 | 84.89 |
| 29.55 | 29.86 | 31.04 | 90.45 |
| 26.32 | 29.79 | 28.36 | 84.46 |

# Detecting Collinearity

- We can detect collinearity by fitting a linear regression model of one exogenous variable to the others

$$X_1 = \beta_0 + \beta_2 X_2 + \beta_3 X_3 \ldots$$

- Using this model's coefficient of determination (R^2) we can calculate the **variance inflation factor**

Higher = more collinear

range is $(1, \infty)$

$\geq 10$ is a problem

$$VIF = \frac{1}{1 - R_i^2}$$

# Detecting Collinearity

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

X15 = np.random.normal(10,2,size=100).reshape(20,5)
X6 = np.sum(X15,axis=1) + np.random.uniform(-2,2,20)

lr = LinearRegression()
lr.fit(X15, X6)
r2 = r2_score(X6, lr.predict(X15))
print(1/(1-r2))
>>> 11.83

from statsmodels.stats.outliers_influence import variance_inflation_factor
X = np.hstack([X15,X6.reshape(-1,1)])
X = sm.add_constant(X)
variance_inflation_factor(X, 6)
>> 11.83
```

index of prediction to check

predictors

# Non-Independent Data

- So far we have looked at models where each observation is independent

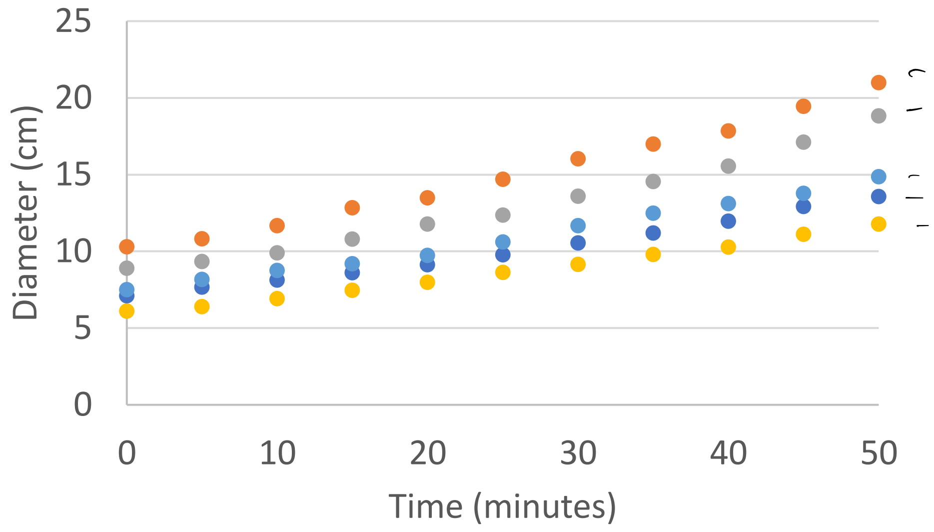
- What about the case when our observations share some characteristic?
  - Multiple observations of the same item (*e.g.* growth measurements of individuals)
  - Data with natural clusters (*e.g.* students within schools)

# Sample use case

My cookie factory is branching out and starting to make bread.

I want to measure how fast my dough rises so that I know how long I have to let it rest.

I made some test batches and observed their growth.



# Linear Mixed Effects Model

- A linear mixed effects model has both **fixed effects** and **random effects**
  - A fixed effects something that has a constant effect not matter which group the observation is in
  - A random effect has a different effect for each group of observations.
  - Random effects are normally distributed with a mean of 0

- Linear mixed models have the form:

$$y = \underbrace{X\beta}_{\text{fixed}} + \overbrace{Z\gamma}^{\text{random}} + \epsilon$$

# Linear Mixed Effects Model

```
data = pd.read_excel("lecture3_figures.xlsx", sheet_name="bread growth")
data = pd.melt(data, id_vars='Time')

endog = data["value"]
exog = data[["Time"]]
exog = sm.add_constant(exog)
md = sm.MixedLM(endog, exog, groups=data['variable'])
mdf = md.fit()
print(mdf.summary())
```

$\rightarrow (N_{obs}, N_{feat})$

$y = A(\text{time})$

add_constant

$y = A \cdot \text{time} + b$

Mixed Linear Model Regression Results

| | | | |
|---|---|---|---|
| Model: | MixedLM | Dependent Variable: | value |
| No. Observations: | 55 | Method: | REML |
| No. Groups: | 5 | Scale: | 0.2176 |
| Min. group size: | 11 | Log-Likelihood: | -52.5400 |
| Max. group size: | 11 | Converged: | Yes |
| Mean group size: | 11.0 | | |

| | Coef. | Std.Err. | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | 7.588 | 0.984 | 7.714 | 0.000 | 5.660 | 9.516 |
| Time | 0.169 | 0.004 | 42.554 | 0.000 | 0.162 | 0.177 |
| Group Var | 4.769 | 7.546 | | | | |

| Time | Bread 1 | Bread 2 | ... |
|---|---|---|---|
| 0 | 5 | 8 | |
| 5 | 6 | 10 | |
| 10 | 7 | 11 | |
| 15 | 8 | 12 | |

| x1 | x2 | x3 | x4 |
|---|---|---|---|
| 1 | 2 | 2 | 1 |
| 2 | 1 | 1 | 3 |
| ... | ... | ... | ... |

wide

| idx | variable | value |
|---|---|---|
| 0 | $x_1$ | 1 |
| 0 | $x_2$ | 2 |
| 0 | $x_3$ | 2 |
| | ... | ... |
| 1 | $x_1$ | 2 |
| 1 | $x_2$ | 1 |

tall

| Time | Variable | Value |
|---|---|---|
| 0 | Bread1 | 5 |
| 0 | Bread2 | 8 |