COMP809 Data Mining and Machine Learning

Patricio Maturana-Russel

p.maturana.russel@aut.ac.nz

Department of Mathematical Sciences and Computer Science and Software Engineering Departments, Auckland University of Technology, Auckland, New Zealand

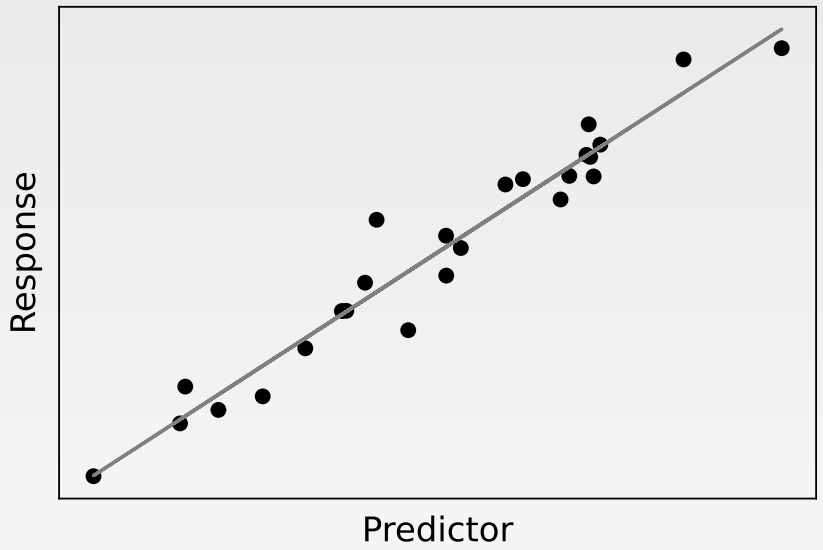
Semester 1, 2024

Contents

Regression analysis

In statistical modeling, regression analysis is a set of statistical processes for estimating the relationships between a dependent variable (response) and one or more independent variables (predictors).

The most common form of regression analysis is linear regression, in which one finds the line (or a more complex linear combination) that most closely fits the data according to a specific mathematical criterion.



Note: the model is linear in its parameters, but not necessarily in its predictors. For instance: $y = \beta_0 + \beta_1 \times \log x + \epsilon$ is a linear model.

Linear model

A linear model is defined as

$$Y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i,$$

where	Y_i:	response for the ith observation, with $i = 1, \ldots, n$
	x_{ij}:	jth predictor for the ith observation, for $j = 0, \ldots, p$
	β_j:	parameter
	ϵ_i:	error term

Y is the dependent variable and x is an independent one.

In matrix notation, this model can be written as

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\top$, with design matrix $\boldsymbol{X}^\top = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^\top$ $\boldsymbol{x}_i = (1, x_{i1}, \ldots, x_{ip})^\top$ for $i = 1, \ldots, n$, $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^\top$, and $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)^\top$.

Linear model

Without assuming any distribution for ϵ, the parameters can be estimated through the *least squares* method. However, this does not allow to make inferences about the parameters, unless the sample size is large.

That is why we usually assume that $\epsilon_i \sim \mathsf{N}(0, \sigma^2)$. Thus

$$Y_i \sim \mathsf{N}\left(\mu_{\boldsymbol{x}_i}, \sigma^2\right)$$
$$\mu_{\boldsymbol{x}_i} = \mathsf{E}\left(Y_i | \boldsymbol{x}_i\right) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}.$$

The model assumes that the errors are:

1. independent,
2. identically normally distributed,
3. with the same variance σ^2 (Homoscedasticity).

Assumptions 2 and 3 are not required for large sample sizes.

Linear model

Consider the linear model

$$Y = \beta_0 + \beta_1 \times x_1 + \beta_2 \times x_2 + \epsilon,$$

where x_1 is a continuous predictor, and x_2 is a factor with 2 levels (0 and 1) known as dummy variable, for instance, male and female.

Parameter interpretation:

- β_0: it is the intercept, the value of Y when the predictors are 0.
- β_1: it is a slope, i.e., the change of Y for a one unit increase in x_1, keeping the rest of predictors fixed.

Note that,

if $x_2 = 0 \rightarrow Y = \beta_0 + \beta_1 \times x_1 + \epsilon$
if $x_2 = 1 \rightarrow Y = (\beta_0 + \beta_2) + \beta_1 \times x_1 + \epsilon$

So, β_2 is the change in the intercept for $x_2 = 1$.

Linear model

Consider the model

$$Y = \beta_0 + \beta_1 \times x_1 + \beta_2 \times x_2 + \beta_3 \times x_1 \times x_2 + \epsilon,$$

where $x_1 \times x_2$ is an interaction term.

Note that,

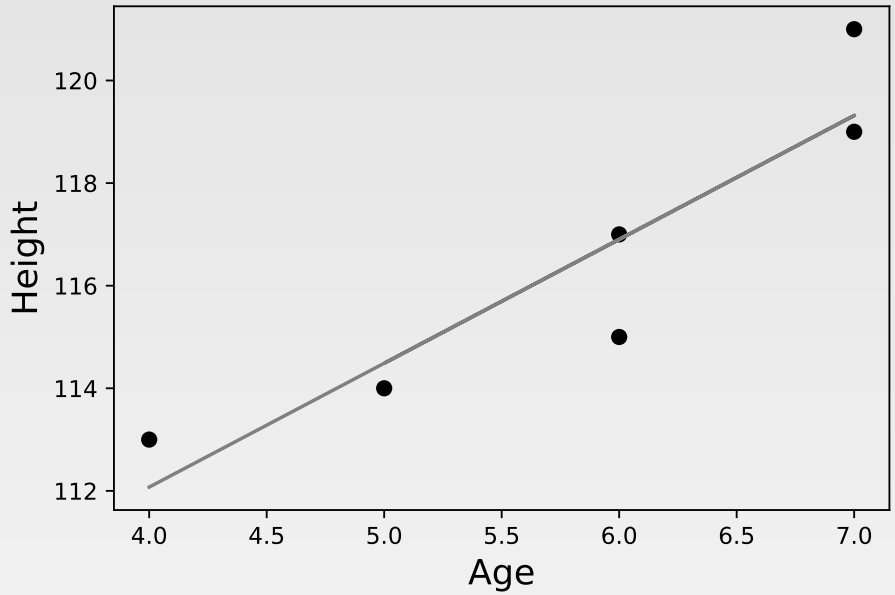
if $x_2 = 0 \rightarrow Y = \beta_0 + \beta_1 \times x_1 + \epsilon$

if $x_2 = 1 \rightarrow Y = (\beta_0 + \beta_2) + (\beta_1 + \beta_3) \times x_1 + \epsilon$

So, β_3 is the change in the slope for $x_2 = 1$.

Linear model

Child	Age	Height
1	4	113
2	7	121
3	6	117
4	5	114
5	6	115
6	7	119
7	4	?



We fit a simple linear model, i.e., $Height = \beta_0 + \beta_1 Age + \epsilon$, obtaining $\widehat{\beta}_0 = 102.41$ and $\widehat{\beta}_1 = 2.41$.

We expect that for an increase in a year, the child is expected to increase their height in 2.41 units on average.

The interpretation of β_0 in this context does not make sense.

We can estimate the height for someone 4 years old as $102.41 + 2.41 \times 4 = 112.07$ (useful for missing data).

Case study

FEV (forced expiratory volume) is an index of pulmonary function that measures the volume of air expelled after one second of constant effort. The data contains determinations of FEV on 654 children ages 6-22 who were seen in the Childhood Respiratory Disease Study in 1980 in East Boston, Massachusetts. The data are part of a larger study to follow the change in pulmonary function over time in children.

ID	ID number
Age	years
FEV	litres
Height	inches
Sex	Male or Female
Smoker	Non = nonsmoker, Current = current smoker

The data with its corresponding information is available on http://www.statsci.org/.

Model fitting

```
>>> import pandas as pd
>>> data = pd.read_csv("fev.csv")
>>> print(data)
          ID  Age    FEV  Height     Sex   Smoker
0        301    9  1.708    57.0  Female      Non
1        451    8  1.724    67.5  Female      Non
2        501    7  1.720    54.5  Female      Non
3        642    9  1.558    53.0    Male      Non
4        901    9  1.895    57.0    Male      Non
..       ...  ...    ...     ...     ...      ...
649    73041   16  4.270    67.0    Male  Current
650    73042   15  3.727    68.0    Male  Current
651    73751   18  2.853    60.0  Female      Non
652    75852   16  2.795    63.0  Female  Current
653    77151   15  3.211    66.5  Female      Non

[654 rows x 6 columns]
```

The data contains two numerical predictors (Age and Height) and 2 factors (Sex and Smoker). FEV is the response.

Model fitting

We fit the following linear model

$$FEV = \beta_0 + \beta_1 \times Sex + \beta_2 \times Smoker + \beta_3 \times Age + \beta_4 \times Height + \epsilon,$$

where $\epsilon \sim \mathsf{N}(0, \sigma^2)$.

In python,

```
>>> import statsmodels.formula.api as smf
>>> mod1 = smf.ols("FEV ~ C(Sex) + C(Smoker) + Age + Height", data=data)
>>> mod1_res = mod1.fit()
>>> mod1_res.summary()
```

Model fitting

OLS Regression Results

Dep. Variable:	FEV	R-squared:	0.775
Model:	OLS	Adj. R-squared:	0.774
Method:	Least Squares	F-statistic:	560.0
Date:	Mon, 27 Feb 2023	Prob (F-statistic):	9.10e-209
Time:	17:53:36	Log-Likelihood:	-345.90
No. Observations:	654	AIC:	701.8
Df Residuals:	649	BIC:	724.2
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.5442	0.232	-19.583	0.000	-5.000	-4.089
C(Sex)[T.Male]	0.1571	0.033	4.731	0.000	0.092	0.222
C(Smoker)[T.Non]	0.0872	0.059	1.472	0.141	-0.029	0.204
Age	0.0655	0.009	6.904	0.000	0.047	0.084
Height	0.1042	0.005	21.901	0.000	0.095	0.114

Omnibus:	22.758	Durbin-Watson:	1.645
Prob(Omnibus):	0.000	Jarque-Bera (JB):	43.271
Skew:	0.207	Prob(JB):	4.02e-10
Kurtosis:	4.190	Cond. No.	899.

Model fitting

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
 specified.
```

The p-value for the variable $Smoker$ is 0.141, so it is not statistically significant. Therefore, it can be excluded. The hypotheses are $H_0 : \beta_2 = 0$ and $H_1 : \beta_2 \neq 0$. This is the backward selection method.

The model explains 77% (Adj. R-squared) of FEV variability.

We fit a linear model without the variable $Smoker$.

```
>>> mod2  =  smf.ols("FEV ~ C(Sex) + Age + Height", data=data)
>>> mod2_res = mod2.fit()
>>> mod2_res.summary()
```

Model fitting

OLS Regression Results

Dep. Variable:	FEV	R-squared:	0.775
Model:	OLS	Adj. R-squared:	0.774
Method:	Least Squares	F-statistic:	744.6
Date:	Mon, 27 Feb 2023	Prob (F-statistic):	9.05e-210
Time:	17:55:56	Log-Likelihood:	-346.99
No. Observations:	654	AIC:	702.0
Df Residuals:	650	BIC:	719.9
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.4486	0.223	-19.952	0.000	-4.886	-4.011
C(Sex)[T.Male]	0.1611	0.033	4.864	0.000	0.096	0.226
Age	0.0614	0.009	6.766	0.000	0.044	0.079
Height	0.1046	0.005	21.986	0.000	0.095	0.114

Omnibus:	23.453	Durbin-Watson:	1.636
Prob(Omnibus):	0.000	Jarque-Bera (JB):	47.454
Skew:	0.189	Prob(JB):	4.96e-11
Kurtosis:	4.264	Cond. No.	861.

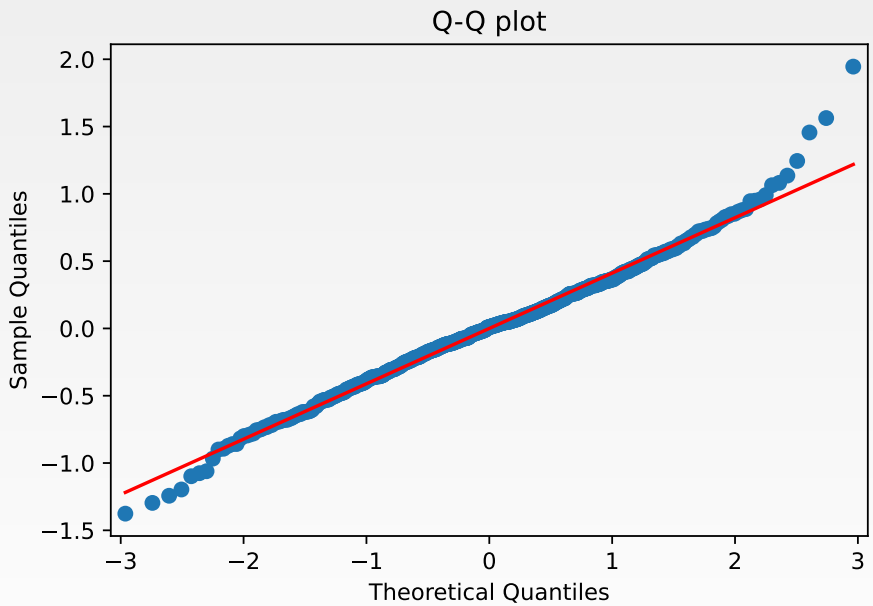
Checking assumption

(1) Independence: it is reasonable to assume that the individuals are independent.
(2) Residuals normally distributed: for this we can use the QQ plot.

```
>>> import statsmodels.api as sm
>>> sm.qqplot(mod2_res.resid, line="s");
```

Most of the points fall on the 45-degree line in the center of the distribution. It is reasonable to so assume that the residuals follow a normal distribution.

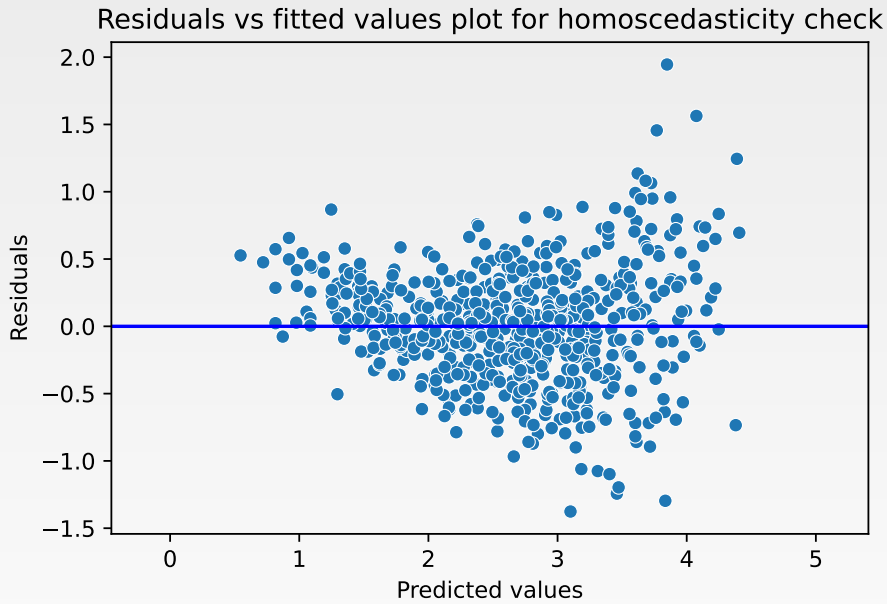
Note: Normality assumption is not required for large sample sizes due to Central Limit Theorem.



Checking assumption

(3) Homoscedasticity: for this we can study the relationship between the predicted values and the residuals.

In the case of homoscedasticity, non pattern should be observed. However, as the predicted values increase, the variability of the residuals increase too. The constant variance assumption is not met. This phenomenom is called heteroscedasticity.



Checking assumption

Python code to produce homoscedasticity checking plot:

```
ypred = mod2_res.predict(data[["Age","Height","Sex"]])
residuals = np.array(data["FEV"]) - np.array(ypred) # from list to an array
p = plt.scatter(ypred,residuals, s=15)
plt.xlabel("Predicted values")
plt.ylabel("Residuals")
plt.axhline(y = 0.0, color = "b", linestyle = "-")
p = plt.title("Residuals vs fitted values plot for homoscedasticity check")
plt.show()
```

Heteroscedasticity

Main consequences:

- The least squares estimator is still a linear and unbiased estimator, but there is another estimator with a smaller variance.
- The errors of the estimators are incorrect, consequently the p-values too.

Remarks:

- If the sample size is large enough, the variance of the least squares estimator may still be sufficiently small to obtain precise estimates (see results below).
- Assuming that ordinary least squares no longer produces the best linear unbiased estimators (BLUE), **robust standard errors** can be used to correct the issue of incorrect standard errors so that interval estimates and hypothesis tests are valid.
- Or the response can be transformed, e.g., log transformation, or re-defined.

Model fitting - Robust Linear Regression

```
>>> from statsmodels.formula.api import rlm
>>> rlm_mod1 = rlm("FEV ~ C(Sex) + C(Smoker) + Age + Height",
                   data=data).fit()
>>> rlm_mod1.summary()
```

Robust linear Model Regression Results

Dep. Variable:	FEV	No. Observations:	654
Model:	RLM	Df Residuals:	649
Method:	IRLS	Df Model:	4
Norm:	HuberT		
Scale Est.:	mad		
Cov Type:	H1		
Date:	Tue, 28 Feb 2023		
Time:	10:45:01		
No. Iterations:	20		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-4.3951	0.224	-19.621	0.000	-4.834	-3.956
C(Sex)[T.Male]	0.1266	0.032	3.949	0.000	0.064	0.189
C(Smoker)[T.Non]	0.0500	0.057	0.874	0.382	-0.062	0.162
Age	0.0600	0.009	6.547	0.000	0.042	0.078
Height	0.1034	0.005	22.505	0.000	0.094	0.112

Model fitting - Robust Linear Regression

```
>>> from statsmodels.formula.api import rlm
>>> rlm_mod2 = rlm("FEV ~ C(Sex) + Age + Height",data=data).fit()
>>> rlm_mod2.summary()
```

```
                    Robust linear Model Regression Results
==============================================================================
Dep. Variable:                    FEV   No. Observations:                  654
Model:                            RLM   Df Residuals:                      650
Method:                          IRLS   Df Model:                            3
Norm:                         HuberT
Scale Est.:                      mad
Cov Type:                         H1
Date:                Tue, 28 Feb 2023
Time:                        10:46:36
No. Iterations:                    24
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept         -4.3382      0.213    -20.362      0.000      -4.756      -3.921
C(Sex)[T.Male]     0.1286      0.032      4.063      0.000       0.067       0.191
Age                0.0579      0.009      6.687      0.000       0.041       0.075
Height             0.1035      0.005     22.772      0.000       0.095       0.112
==============================================================================
If the model instance has been used for another fit with different fit parameters,
then the fit options might not be the correct ones anymore.
```

Model fitting - Robust Linear Regression

The `rml` implements by default a Huber regression that weights the observations. This M-estimator assigns outliers less weight.

The standard errors are similar to the ones estimated in the first approach, due to the large number of observations. The final model is

$$FEV = \beta_0 + \beta_1 \times Age + \beta_2 \times Height + \beta_3 \times Sex + \epsilon.$$

The estimated model is

$$FEV = -4.34 + 0.13 \times Sex + 0.06 \times Age + 0.10 \times Height.$$

Model comparison

Models can be compared through:

- R^2: coefficient of determination. It represents the percentage of the response variability explained by the model (higher is better).
- AIC: Akaike Information Criterion (smaller is better).
- BIC: Bayesian Information Criterion (smaller is better).

Remarks:

- AIC and BIC values are not very informative on their own, but useful to compare models.
- AIC and BIC penalize the inclusion of parameters.
- R^2 tends to increase with the inclusion of parameters. That is why the Adjusted R^2 is preferred.
- The `rml` function does not yield these statistics. Probably, it is not suitable to compare models with different weights.

Prediction

We can generate a prediction for a 10 years old female, with height 65, as follows

```
>>> predX = {"Age":[10], "Height": [65], "Sex": ["Female"]};
>>> predX = pd.DataFrame(data=predX);
>>> rlm_mod2.predict(predX)
    2.96764
```

Multicollinearity

The linear relationship among the predictors (columns of the design matrix X) can potentially generate some difficulties, particularly, in case we want to study the relationship between the response y and the predictors.

When the predictors are correlated, a change in one predictor is associated with shifts in another predictor. The stronger the correlation, the more difficult is to change one variable without altering another. Therefore, the parameter estimation becomes difficult.

This is known as **multicollinearity** and causes the following problems:

- The estimates become very sensitive to small changes in the model.
- Reduction of the precision of the estimates. As a result, the p-values are unreliable.

In the case the aim of the model is the prediction of the response, multicollinearity is not a problem.

Variance inflation factor

The VIF is used to measure the severity of the multicollinearity. It is calculated using the each predictor as a response in the linear model. For instance, if we have 3 predictors, the VIF value for x_1 is given

$$\text{VIF}_1 = \frac{1}{1 - R^2},$$

where the R^2 is the coefficient of determination for model $x_1 = \beta_0 + \beta_2 \times x_2 + \beta_3 \times x_3 + \epsilon$.

A rule of thumb for interpreting the variance inflation factor:

Value	Interpretation
$\text{VIF} < 1$	not correlated
$1 \leqslant \text{VIF} \leqslant 5$	moderately correlated
$\text{VIF} > 5$	highly correlated

Exactly how large a VIF has to be before it causes issues is a subject of debate (www.statisticshowto.com).

Variance inflation factor

```
>>> from patsy import dmatrices
>>> y, X = dmatrices("FEV ~ Age + Height + C(Sex)",
                     data, return_type="dataframe")

>>> # For each X, calculate VIF and save in dataframe
>>> vif = pd.DataFrame()
>>> from statsmodels.stats.outliers_influence import variance_inflation_factor
>>> vif["VIF Factor"]=[variance_inflation_factor(X.values,i)
                                          for i in range(X.shape[1])]
>>> vif["Feature"] = X.columns
>>> vif.round(2)
      VIF Factor         Feature
   0      191.00       Intercept
   1        1.05  C(Sex)[T.Male]
   2        2.75             Age
   3        2.82          Height
```

Remarks:

- We are mainly interested in the continuous features.
- High VIFs for dummy variables representing nominal variables with three or more categories are usually not a problem.

Multicollinearity

In the case multicollinearity is a problem:

- remove highly correlated predictors,
- a principal component analysis can be applied to the continuous variables, reducing its dimension and generating independent new predictors (linear combination of the original ones), or
- fit a LASSO regression model.

Prediction using PCA

If you want to generate a prediction using PCs, you have to apply the corresponding transformations:

```
>>> from sklearn.preprocessing import StandardScaler
>>> from sklearn.decomposition import PCA

>>> scaler = StandardScaler();     # Creating object
>>> fitted = scaler.fit(X);        # Calculating means and SDs
>>> X_std  = fitted.transform(X); # Standardising data

#   pred is a data frame with the predictor values
#   Mean and SD defined in "fitted" are used to standardise "pred"
>>> pred_std = fitted.transform(pred);

>>> pca        = PCA(n_components=X_std.shape[1]); # Specifying number of PCs
>>> pca_fitted = pca.fit(X_std);                   # Calculating PC transformation
>>> PCAs       = pca_fitted.transform(X_std);      # Generating PCs

#   Transforming "pred_std" according to "pca_fitted"
>>> pred_pca = pca_fitted.transform(pred_std);

#   Then we can generate the prediction
>>> model_pca.predict(pred_pca) # model_pca is the model using PCs as predictors
```

End