Human vs. Computer: Can we teach the computer to read residual plots?

A thesis submitted for the degree of

Master

by

Shuofan Zhang

Master, Monash University

Department of Econometrics and Business Statistics

Monash University

Australia

May 2018

Contents

Acknowledgements iii

Declaration v

Abstract vii

1 Introduction and literature review 1

1.1 Lineup protocol . . . 3
1.2 Computer vision . . . 4
1.3 Comparing human vs. computer . . . 7

2 Comparing computer performance against database of human evaluation 9

2.1 Amazon Mechanical Turk study explanation . . . 9
2.2 Linear relationship simulation . . . 12
2.3 Null plot simulation . . . 15
2.4 Computer model . . . 18
2.5 Comparing results . . . 23
2.6 Aside discussion related to the comparing results . . . 26

3 New experiment comparing human vs. computer on reading heteroskedasticity 27

3.1 Human experiment explanation . . . 27
3.2 Heteroskedasticity simulation . . . 28
3.3 Null plot simulation . . . 30
3.4 White test . . . 31
3.5 Computer model . . . 31
3.6 Comparing results . . . 32

4 Conclusion and discussion 35

Bibliography 37

Acknowledgements

I would like to thank my supervisor, Di, for being patient with me as always.

This thesis was written using R markdown with relevant code and data accessible with the text.

https://github.com/shuofan18/ETF5550

Software used to conduct this research is R (R Core Team, 2013), Keras (Chollet et al., 2015), ggplot2 (Wickham, 2009)

Declaration

I hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma in any university or equivalent institution, and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Shuofan Zhang

Abstract

Residuals plots are a primary means to diagnose statistical models. It requires human evaluation to determine if structure in the plot is consistent with random variation or not. If not, then the diagnosis is that the model has not adequately captured the relationships between response and explanatory variable in the data. This thesis develops a computer vision model to read residual plots. It compares results with a large database of human evaluations. The evaluations were conducted using a protocol called the "lineup" which places residual plots in a formal framework for statistical hypothesis testing. The comparison between computer and human is made on a very restricted and controlled set of residual plot structures. A new small human subject study is also conducted to compare human vs. computer in reading heteroscedasticity.

Chapter 1

Introduction and literature review

"The multiple regression model for cross-sectional data is still the most widely used vehicle for empirical analysis in economics and other social sciences" (Wooldridge, 2015). Detecting possible violations of the Gauss-Markov assumptions is crucial to interpret the data properly, especially in the early stage of analysis. There are several distribution tests that are commonly used, for instance, the Pearson correlation test for detecting linear relationship; the Breusch-Pagan test and White test for investigating heteroskedasticity. But primarily residual plots are the main diagnostic tool and these rely on human evaluation. Because data plots show a lot more information than a single statistic. A good example here would be Anscome's Quartet. "It is a set of four distinct datasets each consisting of 11 (x,y) pairs where each dataset produces the same summary statistics (mean, standard deviation, and correlation) while producing vastly different plots" (Anscombe, 1973). Matejka and Fitzmaurice also did an interesting study on this issue, they used 'datasaurus' data from Cairo (2016) and generated a series of data with same statistics but very different plots (Matejka and Fitzmaurice, 2017).

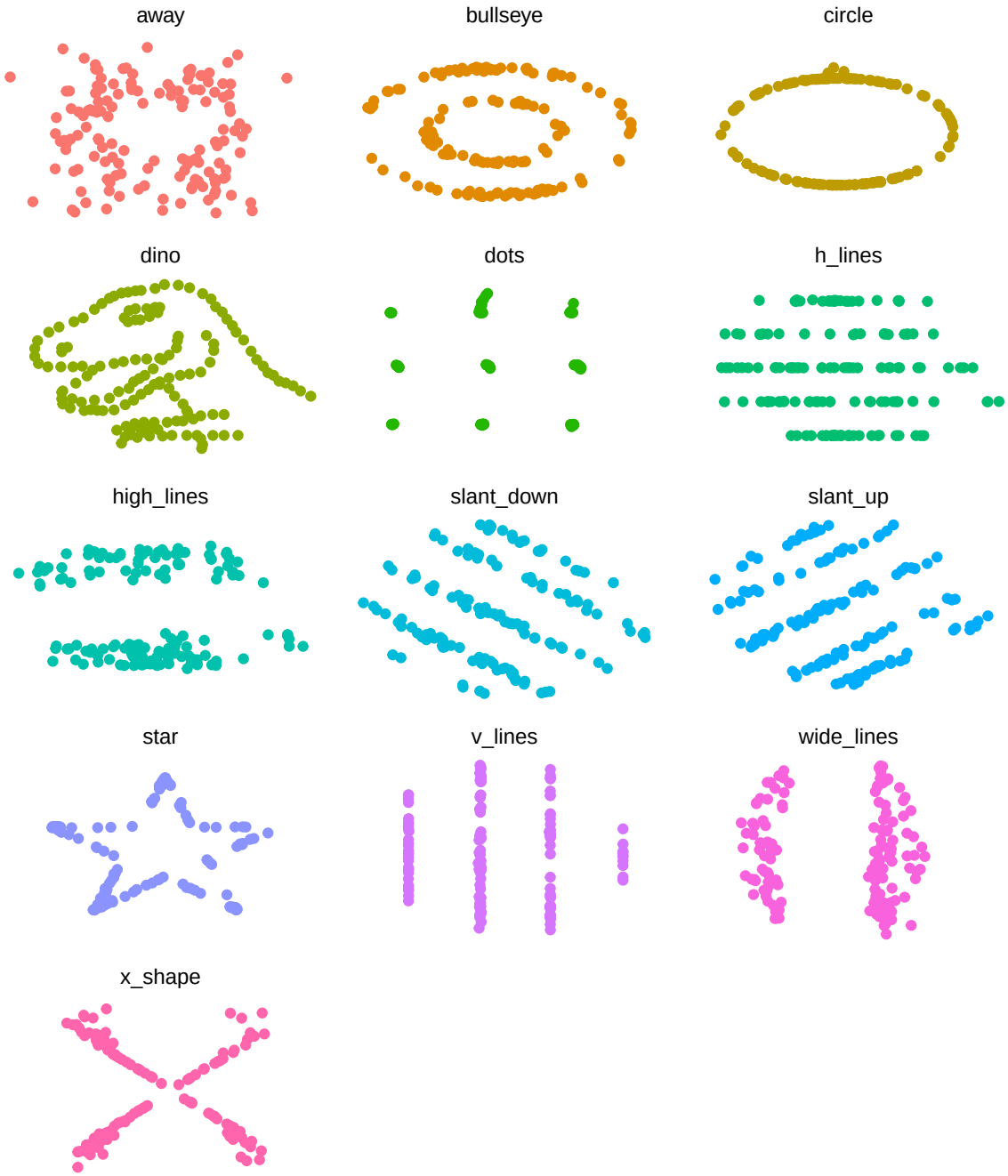

Figure 1.1: *Each dataset has the same summary statistics to two decimal places: (E(x)=54.26, E(y)=47.83, Pearson's r=, sd(x)=16.76, sd(y)=26.93*

1.1 Lineup protocol

Former studies have shown that human eyes are sensitive to the systematic patterns in data plots. With proper manipulation, visualized plots can be used as test statistics and perform valid hypothesis test. One example of these protocols that provides inferential validity is lineup which is introduced by Wickham et al. (2010). "The protocol consists of generating 19 null plots (could be other number), inserting the plot of the real data in a random location among the null plots and asking the human viewer to single out one of the 20 plots as most different from the others" (Wickham et al., 2010). If the real plot is chosen, it means the real data is different from the null hypothesis, so we reject the null hypothesis with 5% chance to be wrong (Type I error). Figure 1.2 is an example of lineup. Which plot do you think is the most different? If you choose one, we are 95% confident to reject the no-relationship assumption between the two variables, hp and disp (Simchoni, 2018). This protocol has proved valid and powerful theoretically as well as practically through human experiments, especially when the assumptions for doing conventional tests are violated (Majumder, Hofmann, and Cook, 2013).

The question that arises today is whether we can train a computer to read scatter plots, particularly with a computer vision approach such as deep learning.

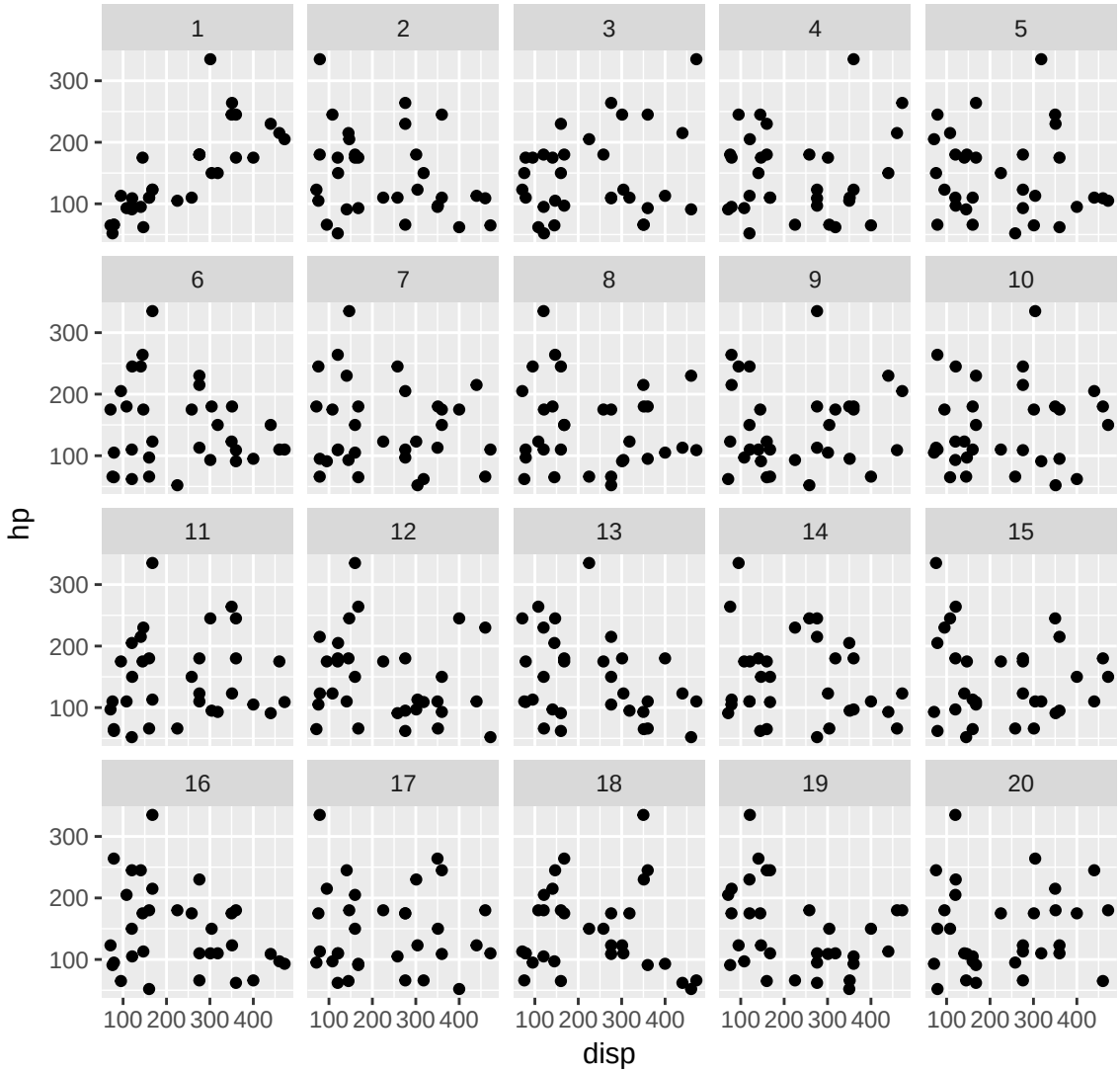

Figure 1.2: *Scatterplot lineup example: one plot is the data, the rest are generated from a null model assuming no relationship between the two variables. In this lineup it is easy to see that plot 1, which is the data plot, is different from the rest.*

1.2 Computer vision

Motivation for the task is provided in a blog post by Giora Simchoni (Simchoni, 2018). He has designed a deep learning model to test the significance of linear relationship between two variables for samples of size 50. The model reached over 93% accuracy on unseen test data. He also mentioned that the computer fails to pick up a strong non-linear relationship even though the Pearson'r is as high as -0.84 (Simchoni, 2018). So the short conclusion is the computer vision is not perfect, in that it is not as flexible as human vision. As Simchoni explained in his article, the model can only distinguish linear relationship

from no-relationship as trained. However, we think this fact is just another example reflecting the importance of visualization as we discussed above. Strong correlation does not necessarily mean linear relationship. We should always refer to the plot before making any statement. What's more, if we want the model to be more flexible, we could simply adjust our design of training accordingly. Therefore, in this article, we are trying to further Simchoni's study. More specifically, we will build a computer model to perform two hypothesis tests as following.

$$H_0$$

: There are no relationships between the two variables.

$$H_1$$

: There is linear relationship between the two variables where all Gauss-Markov assumptions are met.

$$H_0$$

: There is linear relationship between the two variables where all Gauss-Markov assumptions are met.

$$H_1$$

: There is linear relationship between the two variables where the variance of the error term is not a constant while all other Gauss-Markov assumptions are met.

For ease of exposition, only regression model with one explanatory variable will be considered in this paper, but many of the results can be generalized to other cases including multiple regression model. Because the "statistics" we will use is scatter plot, in terms of teaching the computer of reading the plot, one variable is enough to generate different patterns in that plot for convnets to learn. And this makes the design process much simpler.

The model we will use is the convolutional neural networks, also known as convnets, a type of deep-learning model "almost universally used in computer vision applications" (Chollet and Allaire, 2018). Unlike the classical programming where human input rules, in deep learning paradigm, we provide data and the answers associated with the data. Deep learning algorithm will output the rules, and these rules can then be used on new data to make predictions. We can also think of the deep learning neural network as a complex nonlinear model which could estimate millions of parameters ($\mathbf{w}$) with big enough dataset. As usual regression problem, to get the estimates of unknown parameters ($\mathbf{w}$), we need to provide the model with dependent variable (y_i) and independent variables ($\mathbf{x}_i$). In this case, the independent variable will be the images of data plots (in forms of matrices) simulated from the null distribution and the alternative distribution, and dependent variable will be the labels of that plot indicating the true relationship of the original data. Once we have the estimated parameters ($\hat{\mathbf{w}}$), we then can use them to classify unseen data plots, eg. to perform hypothesis tests. The estimation method for deep-learning model is called Backpropagation algorithm which "is a way to train chains of parametric operations using gradient-descent optimization". (Chollet and Allaire, 2018) The gradient-descent optimizer is meant to find the set of parameters such that the cost function reaches its minimum. The form of the cost functions or loss function, should be determined according to each question. In both of the two experiments conducted in this paper, the deep learning model is expected to complete binary classification task, eg. tell "linearly correlated" variables from "independent" variables for the first experiment, tell "heteroskedasticity errors" from "normal errors" for the second experiment. "Crossentropy is usually the best choice (as the loss function) when you're dealing with models that output probabilities" as introduced by Chollet and Allaire (2018). Originated from Information Theory, Crossentropy is a quantity measuring the distance between probability distributions. In deep learning world, it measures the distance between the true distribution and the predictions. Therefore, in this paper, the binary crossentropy loss function will be used. The associated cost function is of the form,

$$J(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \, log\hat{y}_i + (1-y_i)\, log(1-\hat{y}_i)\right)$$

where $\hat{y}_i = g(\mathbf{w} \times \mathbf{x}_i) = \frac{1}{1+e^{-\mathbf{w}\times\mathbf{x}_i}}$ and $g(z)$ is the logistic function.

1.3 Comparing human vs. computer

Chapter 2 will compare computer performanace against database of human evaluation in reading linear relationship. Steps of constructing computer experiment will be discussed, Turk's study will be explained, the comparison results will be given. Chapter 3 will compare computer performance against the results from the new human subject study. Details of this new human subject study will be provided. The results will also be presented.

Chapter 2

Comparing computer performance against database of human evaluation

A database of human evaluations of scatterplots of residuals against fitted is available from prior studies. This is used to compare the performance of the computer model. The computer model is trained on a broader parameter simulation framework, and tested on the same data as the human evaluations.

2.1 Amazon Mechanical Turk study explanation

A large database of results from a human subjects was collected examine the performance of the lineup protocol relative to a classical tests. The work is published in Majumder, Hofmann, and Cook (2013). This database forms the basis of the test set used to examine the computer model performance.

In Majumder, Hofmann, and Cook (2013), "three experiments were conducted to evaluate the effectiveness of the lineup protocol relative to the equivalent test statistic used in the regression setting." In each experiment, they simulated data from a controlled setting and then generated associated lineup for human to evaluate. The human subjects were hired

Table 2.1: *Combination of parameter values used for simulation in Turk's study.*

Sample size (n)	Error SD(sigma)	Experiment 1 beta2	Experiment 2 beta1
100	5	0,1,3,5,8	0.25, 0.75, 1.25, 1.75, 2.75
100	12	1,3,8,10,16	0.5, 1.5, 3.5, 4.5, 6
300	5	0,1,2,3,5	0.1, 0.4, 0.7, 1, 1.5
300	12	1,3,5,7,10	0, 0.8, 1.75, 2.3, 3.5

from Amazon Mechanical Turk where is a a marketplace for work that requires human intelligence.

The controlled model in their first experiment is

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \epsilon_i$$

where $\beta_0 = 5, \beta_1 = 15, X_1 \sim Poisson(\lambda = 30), \epsilon_i \sim N(0, \sigma^2), i = 1, 2, ..., n$. While in the null model $\beta_2 = 0$, and the null data was generated by simulating from $N(0, \hat{\sigma}^2)$. This experiment was aimed to test the ability of human on detecting the effect of X_2.

Their second experiment is very similar to the first one, but there is only one continuous variable X_1 on the right hand side. The actual data model is

$$Y_i = \beta_0 + \beta_1 X_{i1} + \epsilon_i$$

where $\beta_0 = 6, X_1 \sim N(0, 1)$, and the null data was generated from $N(X\hat{\beta}, \hat{\sigma}^2)$.

The third experiment in their paper contains contaminated data where the actual data were in fact generated from two different specifications.

$$Y_i = \begin{cases} \alpha + \beta X_i + \epsilon_i & X_i \sim N(0, 1) \;\; i = 1, ..., n \\ \lambda + \eta_i & X_i \sim N(\mu, 1/3) \;\; i = 1, ..., n_c \end{cases}$$

where $\epsilon_i \sim N(0, \sigma), \eta_i \sim N(0, \sigma/3), \; \mu = -1.75, \; \beta \in (0.1, 0.4, 0.75, 1.25, 1.5, 2.25)$. And $n = 100, n_c = 15, alpha = 0, \lambda = 10, \sigma = 3.5$. The null plots were generated from $N(0, \hat{\sigma}^2)$.

Other parameters in the "actual data sets" of experiment one and two are shown in the table below.

Their experiment 2 examined the performance of humans in recognising linear association between two variables, in direct comparison to conducting a t-test of $H_o : \beta_k = 0$ vs $H_a : \beta_k \neq 0$ assessing the importance of including variable k in the linear model. An example lineup is shown in Figure 2.1. For this lineup, 63 of the 65 people who examined it selected the data plot (position 20) from the null plots. There is clear evidence that the data displayed in plot 20 is not from $H_o : \beta_k = 0$.

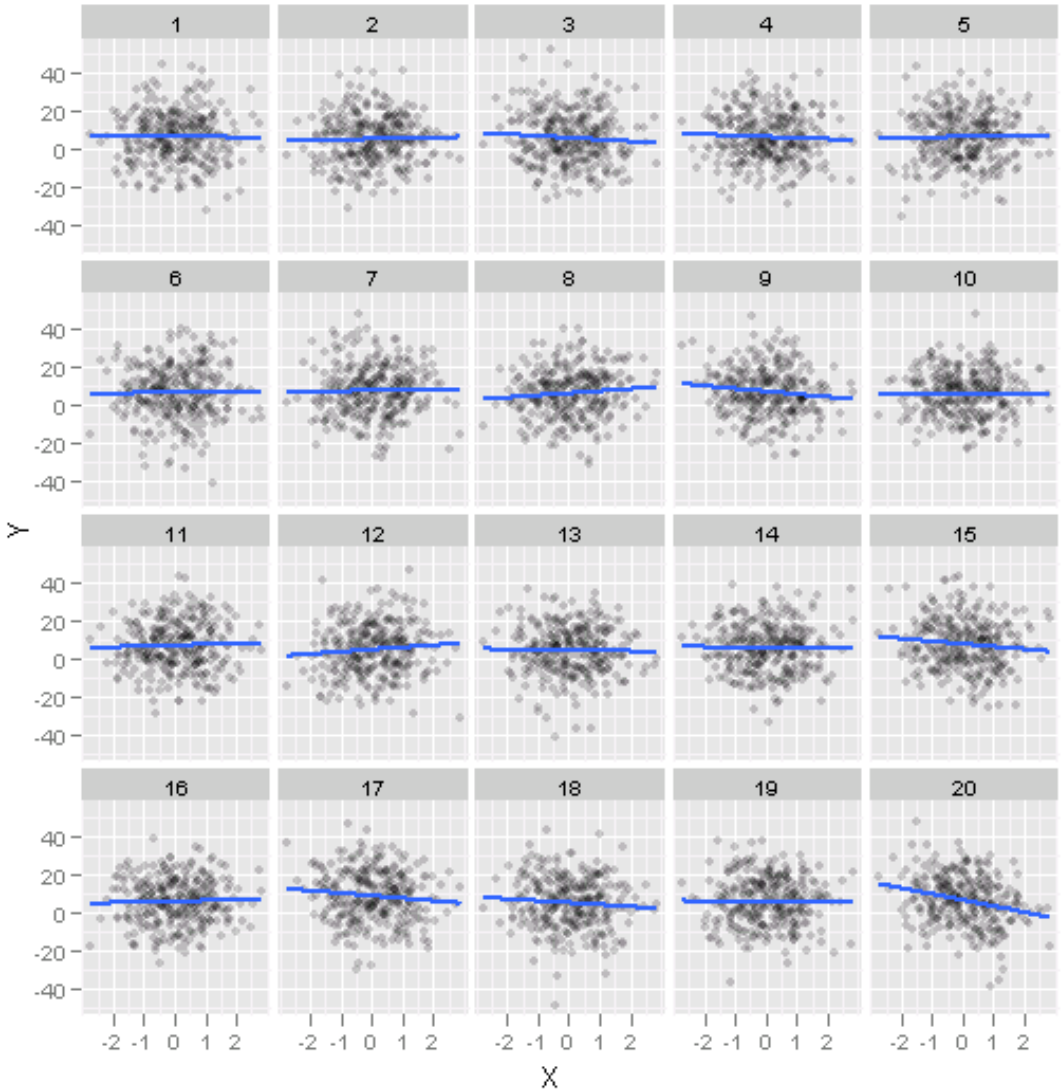

Figure 2.1: *One of 70 lineups used in experiment 2 Majumder et al (2012). Of the 65 people who examined the lineup, 63 selected the data plot, which is in position 20.*

This experiment 2 utilised 70 lineups of size 20 plot, with varying degrees of departure from the $H_o : \beta_k = 0$. There were 351 evaluations by human subjects. These results will be used for comparison with the deep learning model.

2.2 Linear relationship simulation

The design for this model is similar to what Simchoni (2018) did in his blog but is tailored to compare the computer performance with the Turk study results.

The model is designed as:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i, \;\; i = 1, \ldots, n$$

which is the same with the data generating model of Turk's experiment 2. And all the parameters in our model will be designed to cover the range used in the second experiment in Majumder, Hofmann, and Cook (2013). The relevant parameters are generated using the following specification.

- $X \sim N[0,\ 1]$

  Distributions of X has impact on the shape of the scatters. For instance, if X is generated from a uniform distribution, then the plots will look like a square especially when the sample size is large; while more like a circle if X follows normal distribution.

- $\beta_0 = 0$

  Intercept is set to be zero, because it will not change the patterns in the data plots.

- $\beta_1 \sim U[-10, -0.1] \bigcup [0.1, 10]$

  β_1 is designed to be uniformly generated from -10 to 10 (excluding -0.1 to 0.1).

- $\varepsilon \sim N(0, \sigma^2)$ *where* $\sigma \sim U[1, 12]$

  ε is designed to be uniformly generated from 1 to 12.

- $n = U[50, 500]$

  The sample sizes of each data set vary from 50 to 500.

Figure 2.2 shows four example plots generated using the specifications above. Under this controlled structure, a total number of 240,000 data sets are simulated. The histograms of the simulated n, β, σ, the estimated p value and the scatter plots of β against n, σ against n in figure 2.3 show good coverage over all the values.

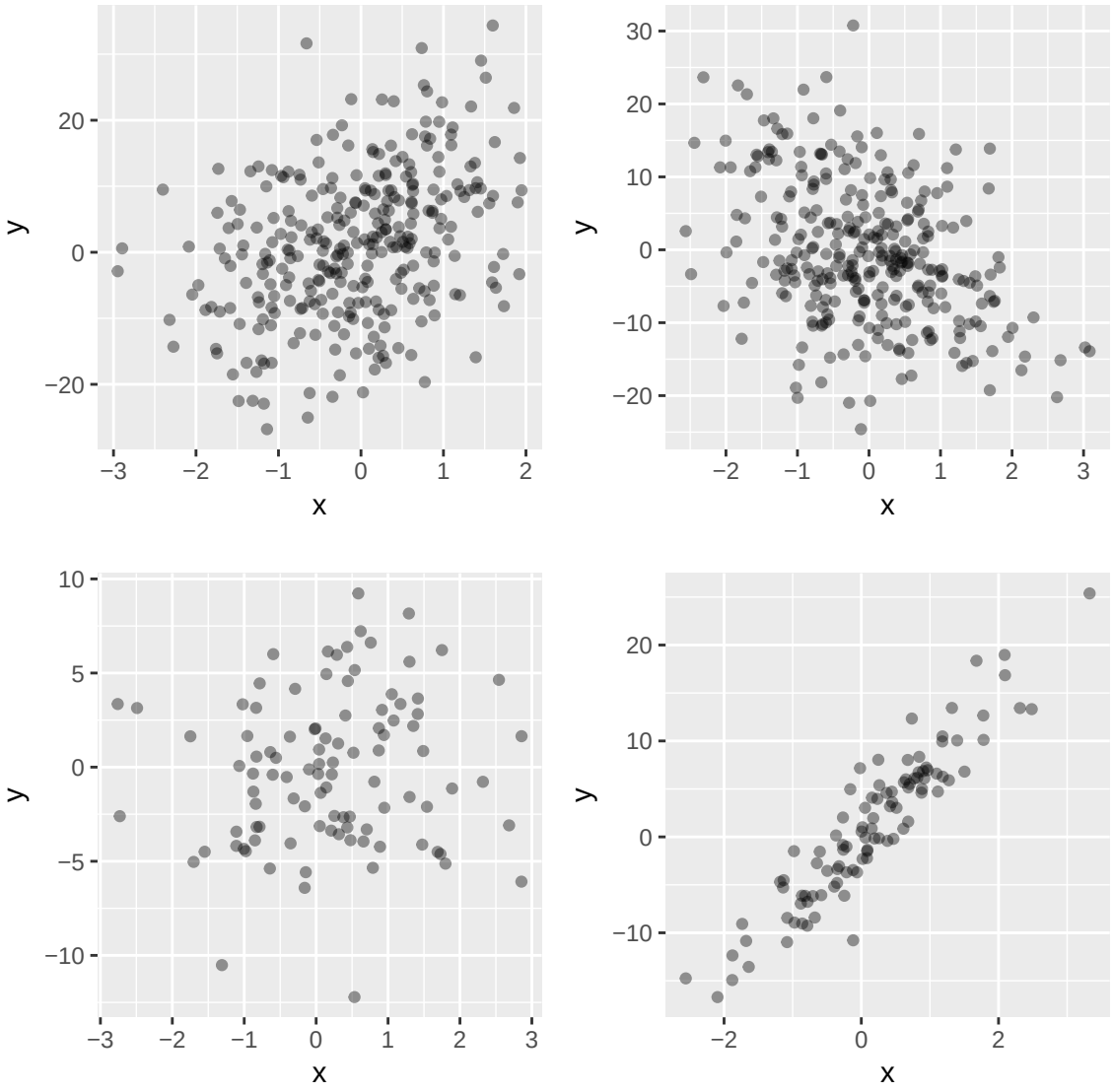

Figure 2.2: *Four examples of data plots generated from the classic linear model.*

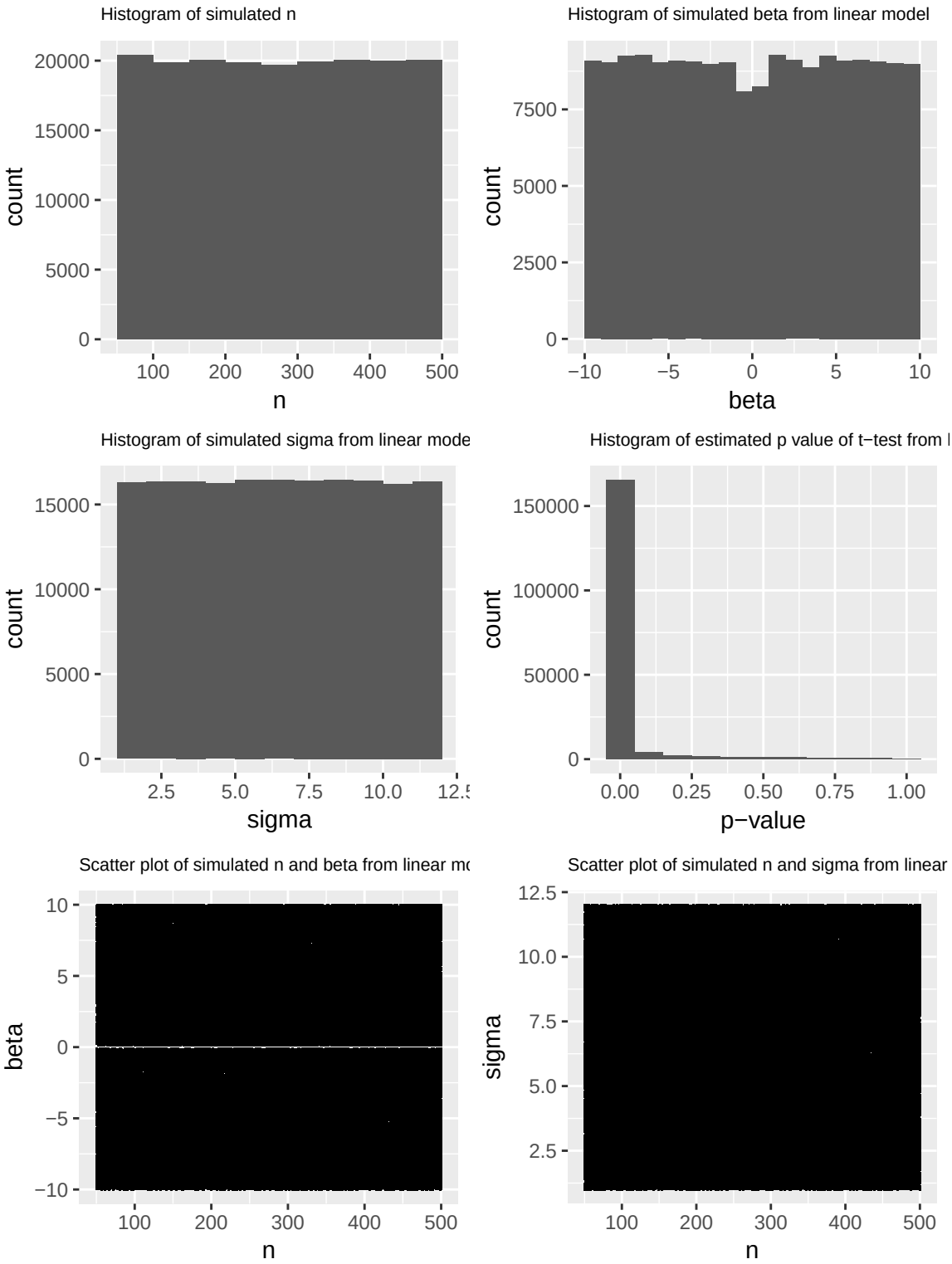

Figure 2.3: *Overview of parameter values used in the linear class simulation, for computer model training. Good coverage is obtained across the parameter space.*

2.3 Null plot simulation

This is the null scenario in our first experiment, eg. the two variables under tested are independent of each other. If the data arises from this situation, then the data plots will not show any systematic patterns theoretically.

The model is designed the same as the linear model:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i, \;\; i = 1, \ldots, n$$

with elements of the model generated using the same specification as the linear model, except

- $\beta_1 = 0$

The coefficient of X_i is always zero. So X_i and Y_i are uncorrelated of each other.

Figure 2.4 are four example plots generated using the specifications above. Same as the linear model simulation, a total number of 240,000 data sets are simulated under this structure. The histograms of the simulated n, β, σ, the estimated p value and the scatter plots of β against n, σ against n in figure 2.5 show good coverage over all the values.

All simulated data and associated parameters including estimated sample p-values of t-test are saved and are used later on for calculating the performance of conventional t-test.

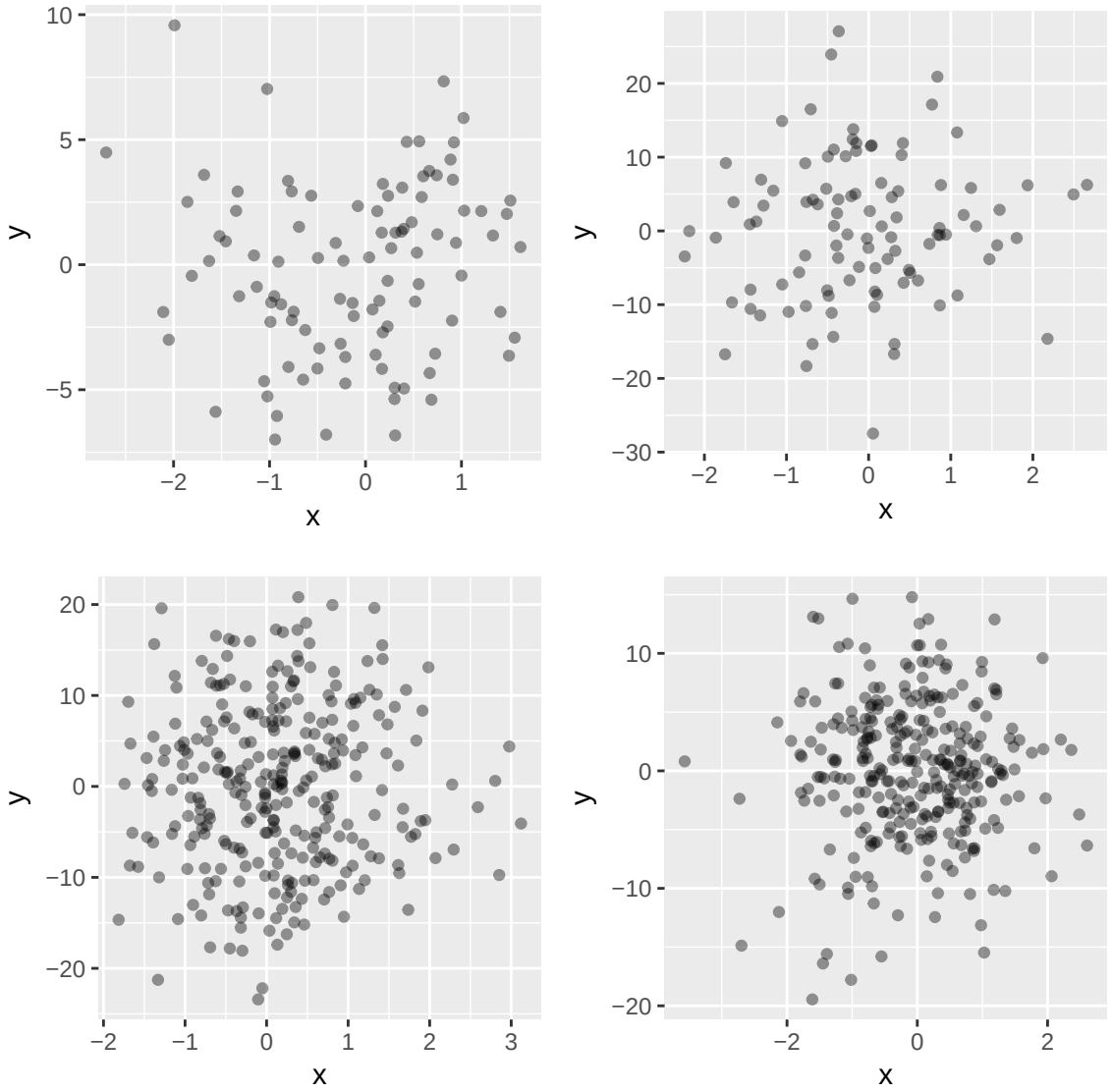

Figure 2.4: *Four examples of data plots generated with two independent variables*

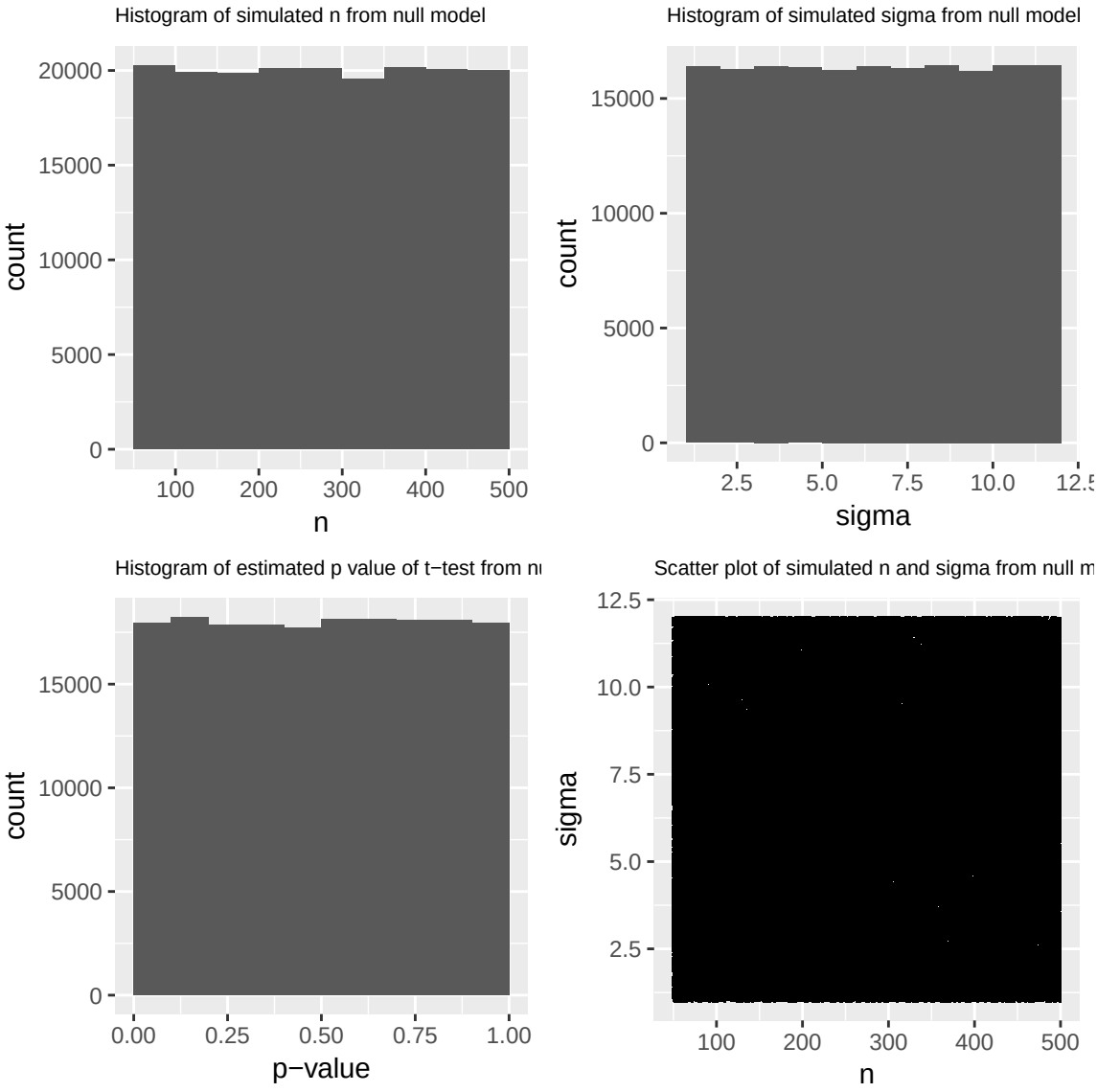

Figure 2.5: *Overview of parameter values used in the null class simulation, for computer model training. Good coverage is obtained across the parameter space.*

2.4 Computer model

Convolutional neural networks have been developed primarily for classifying images. It is the primary clasification model used in computer vision. It has two interesting properties: "the patterns they learn are translation invariant", and "they can learn spatial hierarchies of patterns" (Chollet and Allaire, 2018). The first one implies that once the model learns how to recognize linear patterns, it can be detected regardless of the direction, thus handling different slopes.

All convolutional neural network modeling is done by the Keras (Chollet et al., 2015) package in R, which interfaces to the python software. The plots used for training and testing in this section is the scatter plot between the dependent variable Y and the independent variable X. The R package Wickham (2009) is used to generate the plots. All plots are saved as png and will be resized to have width and height both equal to 150 pixels. This size is similar to the plot size used in the lineup for human evaluation. As for the labels given to each image, we use the true population as the samples' identification directly. It is true that there will be some undesired patterns formed out of randomness, especially when the sample size is small. Unlike what Simchoni did in his post, no conventional tests will be used to sort out the "significant" observations. Because the answer to the question that if the deep learning model can distinguish from patterns formed by chance and by nature is also interested.

As mentioned above, 240,000 data sets are generated for each of the two groups in the first experiment. 100,000 of them are set apart for training. Another 40,000 of the data sets are set apart as validation set in order to monitor during training the accuracy of the model on data it has never seen before. And the leftover (100,000 data sets) become the unseen test set. We make the test set so large that we can compare the performance of the convnet with the conventional t-test properly.

"A convnet takes as input tensors of shape (image height, image width, image channels)."(Chollet and Allaire, 2018) The channels are normally equal to three for RGB. In our case, the input tensors are of shape $150 \times 150 \times 1$ because they are grayscale images.

Therefore the convnet will be configured to process inputs of size (150, 150, 1). We'll do this by passing the argument input_shape = c(28, 28, 1) to the first layer.

```
library(keras)
model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3, 3),
                activation = "relu",
                input_shape = c(150, 150, 1)) %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 64, kernel_size = c(3, 3),
                activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3),
                activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3),
                activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dense(units = 512, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")
model
```

```
## Model
## _________________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =========================================================================
## conv2d_1 (Conv2D)            (None, 148, 148, 32)      320
## _________________________________________________________________________
## max_pooling2d_1 (MaxPooling2D)  (None, 74, 74, 32)     0
## _________________________________________________________________________
```

```
## conv2d_2 (Conv2D)                (None, 72, 72, 64)           18496
## _________________________________________________________________
## max_pooling2d_2 (MaxPooling2D)   (None, 36, 36, 64)           0
## _________________________________________________________________
## conv2d_3 (Conv2D)                (None, 34, 34, 128)          73856
## _________________________________________________________________
## max_pooling2d_3 (MaxPooling2D)   (None, 17, 17, 128)          0
## _________________________________________________________________
## conv2d_4 (Conv2D)                (None, 15, 15, 128)          147584
## _________________________________________________________________
## max_pooling2d_4 (MaxPooling2D)   (None, 7, 7, 128)            0
## _________________________________________________________________
## flatten_1 (Flatten)              (None, 6272)                 0
## _________________________________________________________________
## dense_1 (Dense)                  (None, 512)                  3211776
## _________________________________________________________________
## dense_2 (Dense)                  (None, 1)                    513
## =================================================================
## Total params: 3,452,545
## Trainable params: 3,452,545
## Non-trainable params: 0
## _________________________________________________________________
```

As shown in the table above, the "conv" and "pooling" working together slided the data from $150 \times 150 \times 1$ to $7 \times 7 \times 128$ (3D output). The figure 2.6 decribes this transformation.

Then we need to flatten these 3D tensor into 1D tensor so that they can be processed by the "sigmoid" function in the end. The "sigmoid" is in fact a special case of logistic function. $S(x) = \frac{1}{1+e^{-x}}$. This function is the same one used to predict $\hat{y}_i$ and to calculate the cost function.

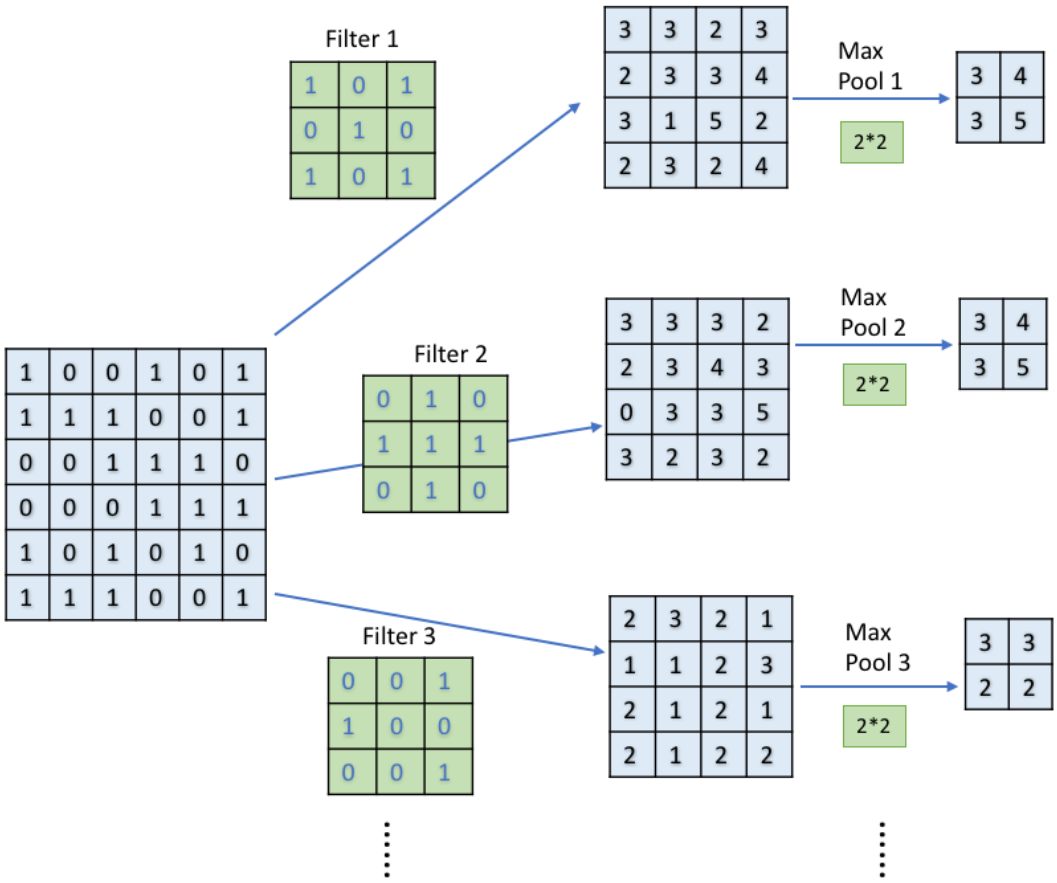

Figure 2.6: *Illustration of convolution and pooling steps on an image. The convolution step applies a fixed number of filters to sliding windows of* 3×3 *cells. Pooling applies a statistic to distinct* 2×2 *tiling of the image. In our model, the statistic used is the maximum of the four values. These transformations are the pre-processing steps done on every image in the training sample, to fit the model, and also to the validation and test images prior to prediction.*

From the model structrue we can see that a total number of 3,452,545 parameters need to be estimated, this is done by gradient descent. 10 epoches (1 epoch = 1 iteration over all samples) are done for training. The model specification given by each epoch is saved, the one gives the overall best performance is chosen to represent the computers. Because the plot of the training history (figure 2.7) shows overfitting starts from the fourth epoch, the values of accuracy and loss from validation set are very close after the fourth epoch.

Hence, we select the fourth, sixth, eighth and the tenth model to have them tested on the unseen test set. And the results are shown in the table below. The "$1 - \alpha$" is the accuracy of each computer model tested on the "null data" in test set only. α here is an

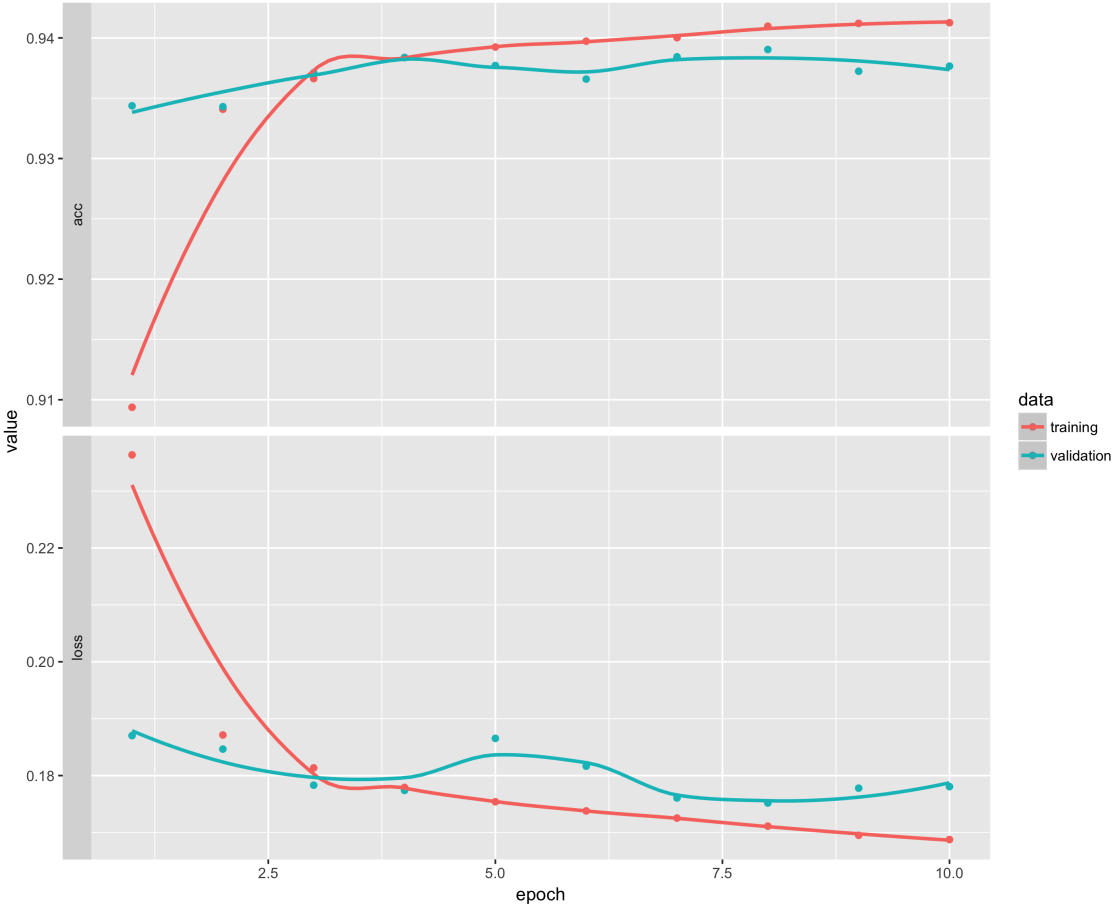

Figure 2.7: *Training and validation metrics of linear vs. null model in our first experiment*

analogy to the Type I error in the conventional hypothesis test. Similarly, the "power" is the accuracy of each computer model tested on the "linear data" in the test set only. The t-test performance is calculated under 5% significance level.

Tests	Linear	Null	Overall
4 epoch	0.892	0.984	0.938
6 epoch	0.889	0.986	0.937
8 epoch	0.896	0.981	0.939
10 epoch	0.904	0.971	0.938
5% *t*-test	0.921	0.949	0.935

Table 2.2: *Performance of four checkpoints from the* convnets *model, and the 5% significant t-test, computed on the test set. Accuracy is reported for each class, and overall. There is a slight improvement as the number of epochs increases, with 10 epochs being reasonably close to the ideal t-test accuracy.*

Since the test set is large enough (200,000 in total) to provide reliable reference. The 8th model is chosen according to the overall accuracy on the test set.

We should note that since the majority of the data plots in Turk's experiment have been generated with linear relationship (when the alternative hypothesis is true), it is a disadvantage for the computer comparing in terms of being tested on the Turk's data. Because of the difference in α ($\alpha \approx 0.02$ for the 8th computer model) the 5% significant t-test and 5% human evaluations may have higher power than the computer model.

2.5 Comparing results

First compare the experiment process for human and computer respectively by two diagrams figure 2.8 and figure 2.9.

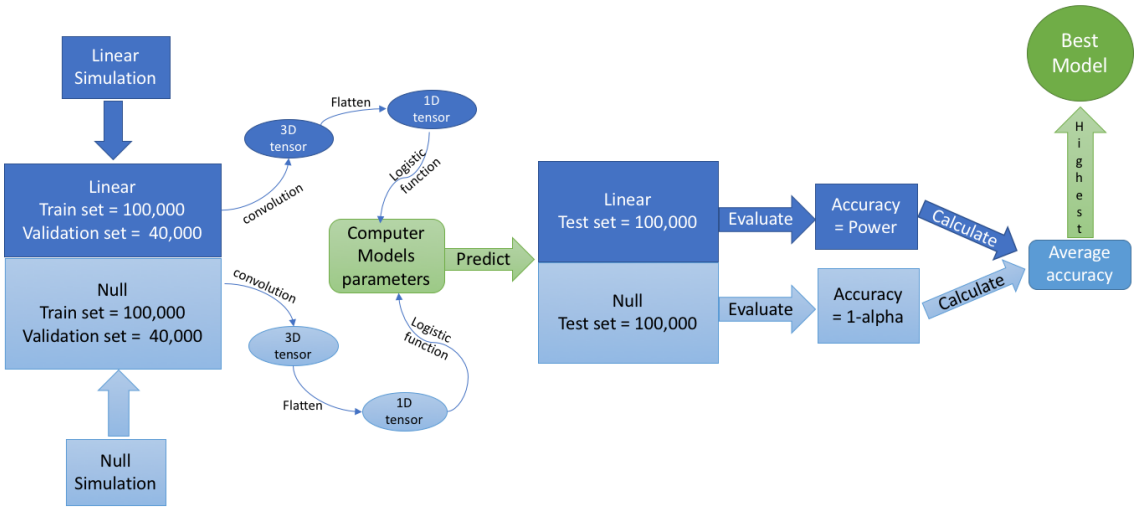

Figure 2.8: *Diagram illustrating the training, diagnosis and choice of the computer model. Based on 480,000 simulated data sets used to create* 150×150 *pixel images, divided into training, validation and test sets.*

The performance of the computer model for the Turk study data is tested in three steps:

- Re-generate the 70 "real plots" using the same data in Turk study (without null plots);
- Create a seperate test directory for the 70 "real plots" exclusively;
- The computer model's predicted accuracy over the 70 "real plots" are recorded as the model's performance.

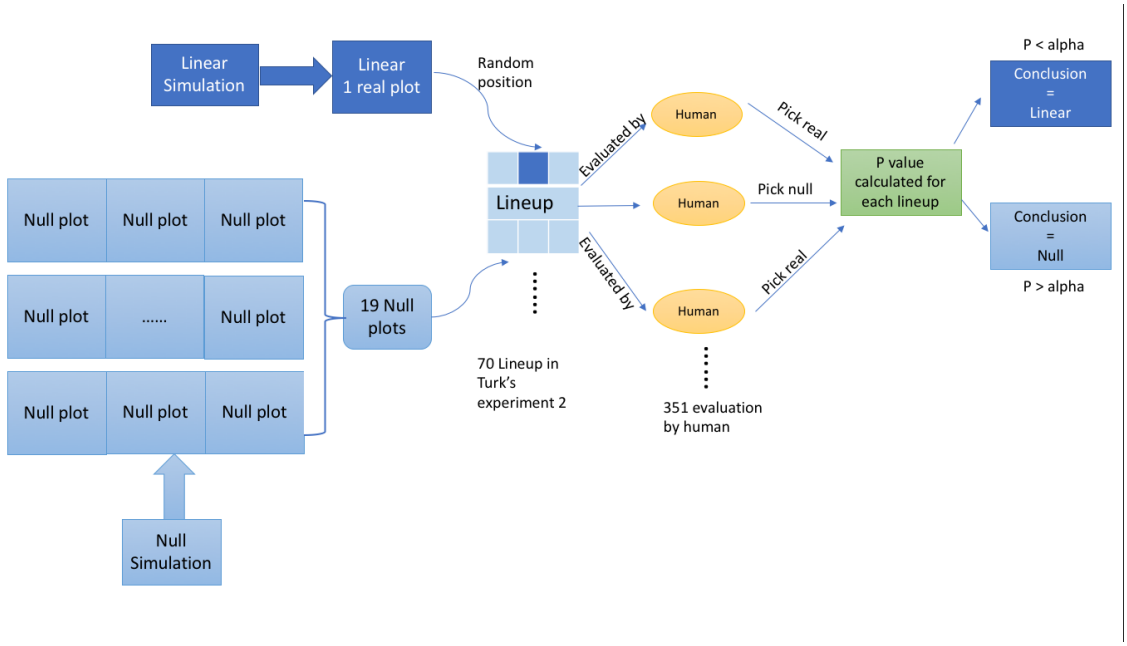

Figure 2.9: *Diagram illustrating the process of human subject evaluation of lineups, and how performance is computed.*

The conclusion of human evaluation is obtained differently from the computer's. Because human evaluated "lineup", not only the "real plots". The performance is tested in five steps:

- Count total number of evaluations made by human for one lineup (N) and the number of correct answers for that lineup (k);
- Obtain N and k for all 70 lineup;
- Calculate p-value associated with each real plot using the formula introduced in section 2 of Majumder, Hofmann, and Cook (2013);
- Draw conclusion: reject the null when the calculated p-value is smaller than α.
- The accuracy of the conclusions the 70 real plots is presenting for the human performance.

For a fair competition, the Type I error (α) should be held the same for all test methods. However, we do not have direct control over the α of the computer model. Therefore, 2% significant t-test and 2% significant human conclusion is also included to give a complete picture of the comparison.

Table 2.3: *Accuracy of testing the 70 data plots evaluated by human computer and the conventional t-test.*

Rank	Tests	No. of correct	Accuracy
1	Human 5%	47	0.6714
1	Human 2%	47	0.6714
2	T-test 5%	43	0.6143
3	Computer 2%	39	0.5571
4	T-test 2%	39	0.5571

The comparing result is interesting. Human achieves the highest accuracy, and the conclusion from human evaluation is robust to smaller p-values; 5% significant t-test is the second best, 2% significant t-test and the computer model perform similarly.

2.6 Aside discussion related to the comparing results

As we can see from the two performance tables above, t-test and convnets behave quite similarly on both the test data set and Turk's experiment data.

It is possible that the convnets is in fact doing the same thing as t-test in this case. Or the strategy it learned in this case turns out to be t-test.

To confirm this idea, we calculated the accuracy of t-test again, with different α (from 0.005 to 0.1 with 0.005 increments) on all 200,000 test sets. The estimated power and overall accuracy were recorded. When $\alpha = 0.015$, the overall accuracy reaches its maximum. This value approximately coincide with the α chosen by the convnets. And since the α of convnets is from 0.0142 to 0.0347 in this case, we truncated the t-test data as well for figure 2.10. From this graph, we can see the two tests perform very similarly, but t-test has overall better performance.

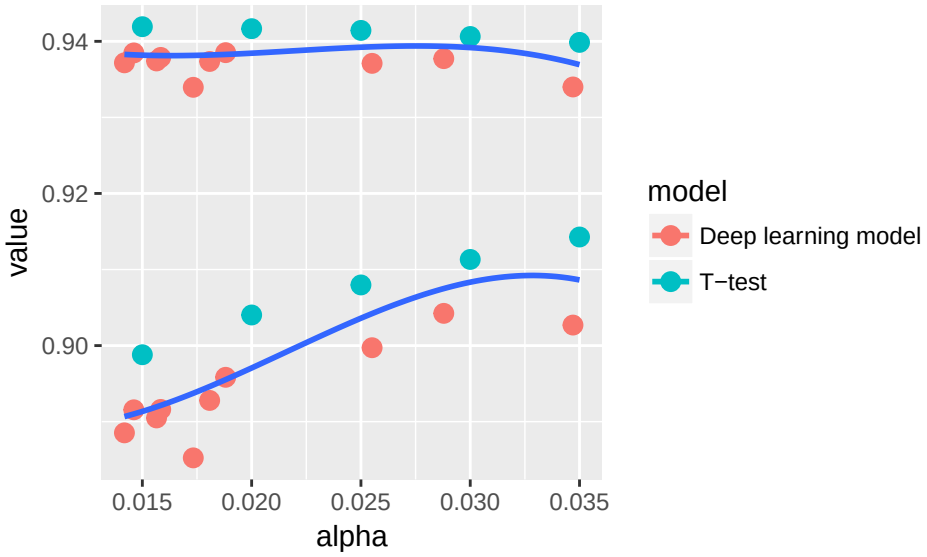

Figure 2.10: *Comparison between computer model and t-test for alpha in (0.01, 0.04), they perform very similarly, but t-test has overall better performance.*

Chapter 3

New experiment comparing human vs. computer on reading heteroskedasticity

Turk experiment mainly considers linear data, in this paper, we extend their study by including heteroscedasticity. A new database of human evaluations is created by a small experiment. This new database is used to compare the performance of the computer model. The computer is trained on the same parameter simulation framework, and tested on the same data as the human evaluations.

3.1 Human experiment explanation

The experiment is to evaluate the human ability of reading heteroskedasticity from residual plots. It is rendered at Monash University, Melbourne Australia. The participants are all students or lecturers in this university.

Four survey are randomly sent to 84 people by email, three of the survey consist of ten lineup questions, and the fourth survey has only four lineup questions. Only one lineup question appears in the survey twice, thus, we have 33 ($10 \times 3 + 4 - 1$) distinct questions in total. A total number of 22 people have participated. Five people evaluated two surveys.

One people selected four plots for each lineup, this person's response is removed from the data. In summary, we have 218 effective evaluations from 21 people.

Figure 3.1 is an example of the lineup used as a question in the survey.

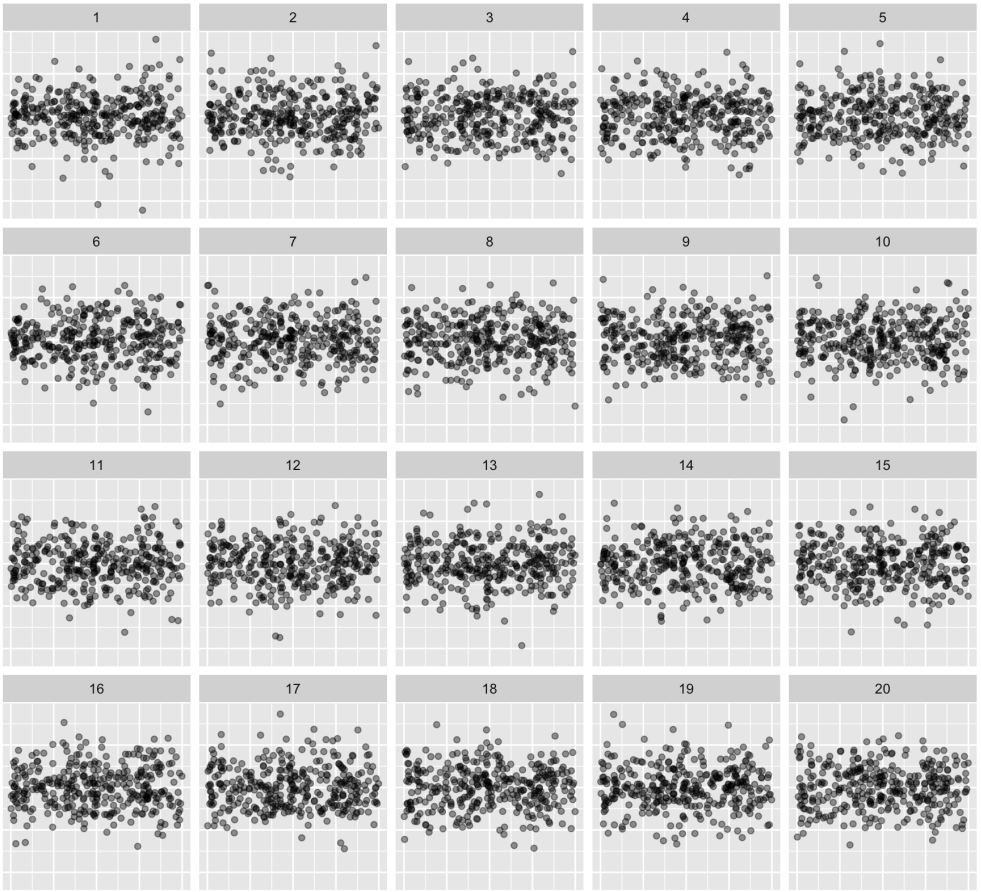

Figure 3.1: *An example question in the survey, 4 out of 6 people picked the real plot, the real plot is the first one.*

The "real plot" and "null plot" data is simulated using the same specifications given in the next two sections respectively.

3.2 Heteroskedasticity simulation

Linear model with heteroskedasticity is the model implied in the alternative hypothesis of the second experiment in this paper, where the constant variance assumption of the linear model is violated while all other conditions are met. By the definition given in Wooldridge (2015), "The homoskedasticity states that the variance of the unobserved error, u, conditional on the explanatory variables, is constant. Homoskedasticity fails whenver

the variance of the unobserved factors changes across different segments of the population, where the segments are determined by the different values of the explanatory variables." There are countless types of heteroskedasticity since the "change of the variance" could be related to "the explanatory variables" in various ways. It is not feasible to list out all kinds of heteroskedasticity by a single function. For simplicity, we will focus on one example of them, a linear correlation between the explanatory variable X and the standard deviation of the error term. Hence the relationship between the explanatory variable X and the variance of the error term will be quadratic. The results from this experiment though can be generalized to more complicated cases.

The model structure is the same with the classic linear model:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i, \;\; i = 1, \ldots, n$$

with elements of the model generated by the following processes

- $X \sim U[-1,\ 1]$ To better present the heteroskedasticity in the data, a uniform distribution of X is used instead of normal distribution. The range is set to be small (from -1 to 1) in order to balance the data with weak heteroskedasticity appearing more frequently.
- $\beta_0 = 0$ Intercept is set to be zero. Because the residual plot but not data plot is used in this experiment. Therefore, the information contained in β_0 will be extracted by the linear regression we fit to the data.
- $\beta_1 \sim U[0.5,\ 1]$ β_1 has little impact in this case as well so it is set to be uniformly generated from 0.5 to 1.
- $\varepsilon \sim N(0,\ (aX + v)^2)$ The variance of the error term is a quadratic function of the explanatory variable which controls the magnitude of heteroskedasticity in the model.
- $a \sim U(-5,\ -0.05) \bigcup (0.05,\ 5)$ The parameter a here, following uniform distribution from -5 to 5 (excluding -0.05 to 0.05), is the correlation coefficient between X and the standard deviation. Larger a gives stronger heteroskedasticity. This range is wide enough for our purpose.

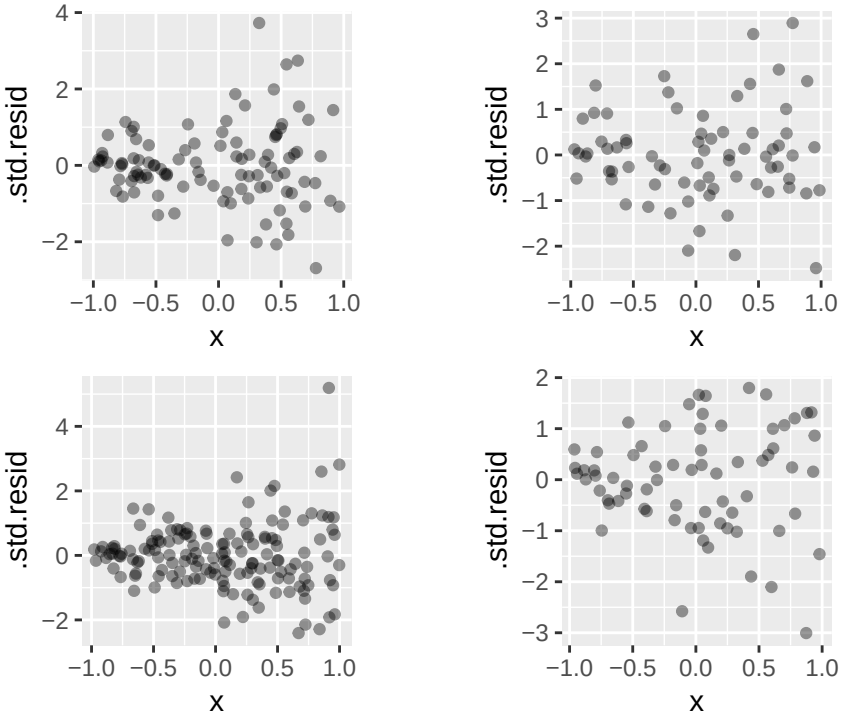

Figure 3.2: *(#fig:heter_plot)Four examples of residual plots generated from linear model with heteroskedasticity*

- $v \sim N(0,\ 1)$ This new error term is added to the variance of ε so the relationship between the data can be more flexible.
- $ax + v - min(ax + v)$ when $min(ax + v) < 0$ To keep the simulated standard deviation positive, and to keep the structure of the relationship between X and the residuals, the $min(ax + v)$ is substracted from $ax + v$ whenever the former is negative.
- $n \sim U[50,\ 500]$ The sample sizes are randomly generated from 50 to 500 to provide reasonable variations.

In general, the choice of the parameters is an empirical work. Primarily, we want the residual plots to show more variation; on the other hand, we need to limit the range of these parameters in order to keep the key features in the data.

3.3 Null plot simulation

The null scenario in this experiment is the classic linear model. The model structure is the same as the heteroskedasticity one. When we simulate this data, we kept most of the

parameters as the same with the alternative data and only changed the key feature of the error term. So the difference in this data set is:

- $\varepsilon \sim N(0,\ c)$
- $c = mean(ax + v)$

c is a constant which equals to the mean of the $ax + v$. All other parameters in the null data are the same with the heteroskedasticity data.

3.4 White test

To provide a reference level of how computer and human perform, a special case of White test is used in this experiment. Every data set simulated from this section has been tested by the White test. The procedure of the White test (Wooldridge, 2015) is:

- Estimate OLS model for the data, obtain residuals ($\hat{u}$) and the fitted values ($\hat{y}$). Computer the squared OLS residuals ($\hat{u}^2$) and the squared fitted values ($\hat{y}^2$).
- Run an auxiliary regression as $\hat{u}^2 = \eta_0 + \eta_1\hat{y} + \eta_2\hat{y}^2 + error$, obtain the R-squared $R^2_{\hat{u}^2}$
- Calculate the LM statistic which follows χ^2_2 distribution
- Conclude based on p-values given certain α

3.5 Computer model

In this experiment, a linear model is fit to the data firstly. Residuals from the fitted model are standardized and extracted. The residual plot is made of standardized residuals against X. The model is still the convnets, and all hyper-parameter in this model are exactly the same as the previous one.

15 epoches are done in this section. All 15 convnets models are saved. The training and validation metrics are shown in figure 3.3.

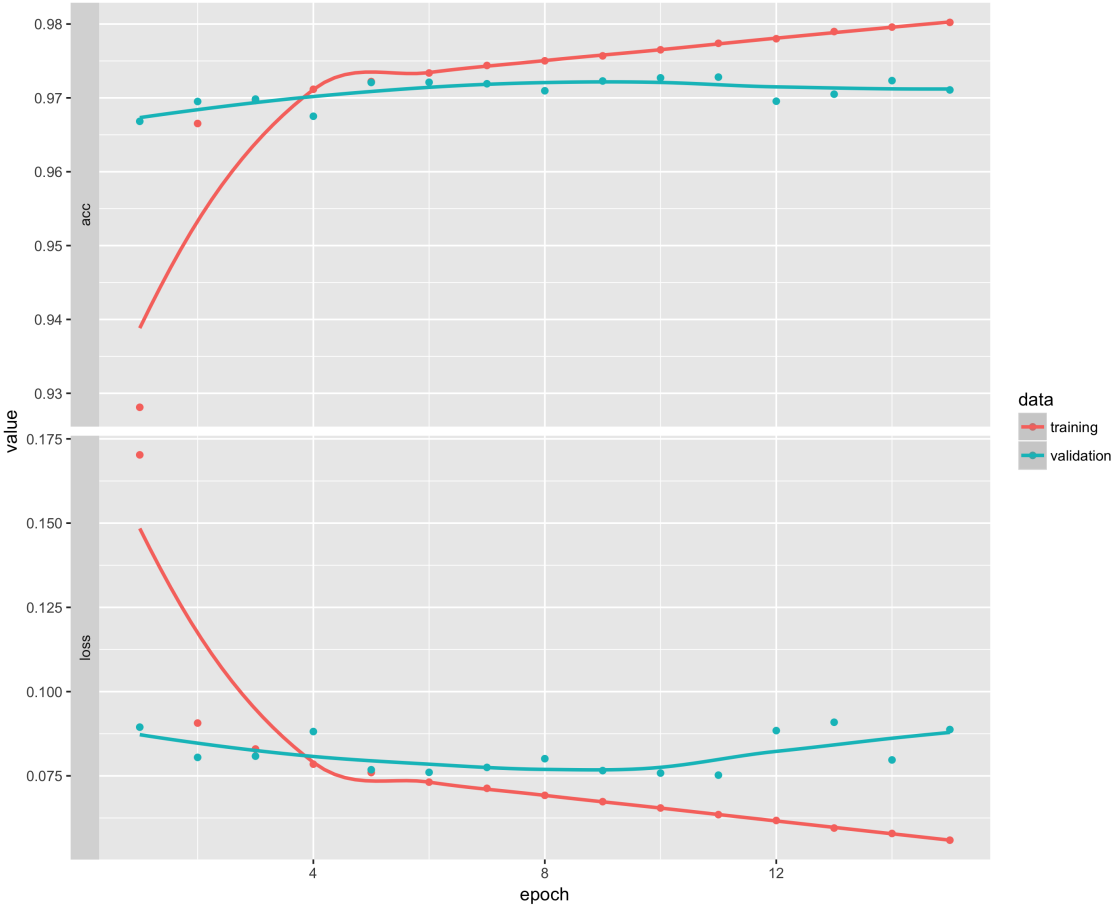

Figure 3.3: *Training and validation metrics of heteroskedasticity vs. null model in our second experiment*

3.6 Comparing results

Table 3.1: *Accuracy of testing the 27 data plots evaluated by human computer and the conventional white-test.*

Rank	Tests	No. of correct	Accuracy
1	Computer 2%	25	92.59%
2	Human 5%	17	62.96%
2	White-test 5%	17	62.96%
3	White-test 2%	16	59.26%
4	Human 2%	15	55.56%

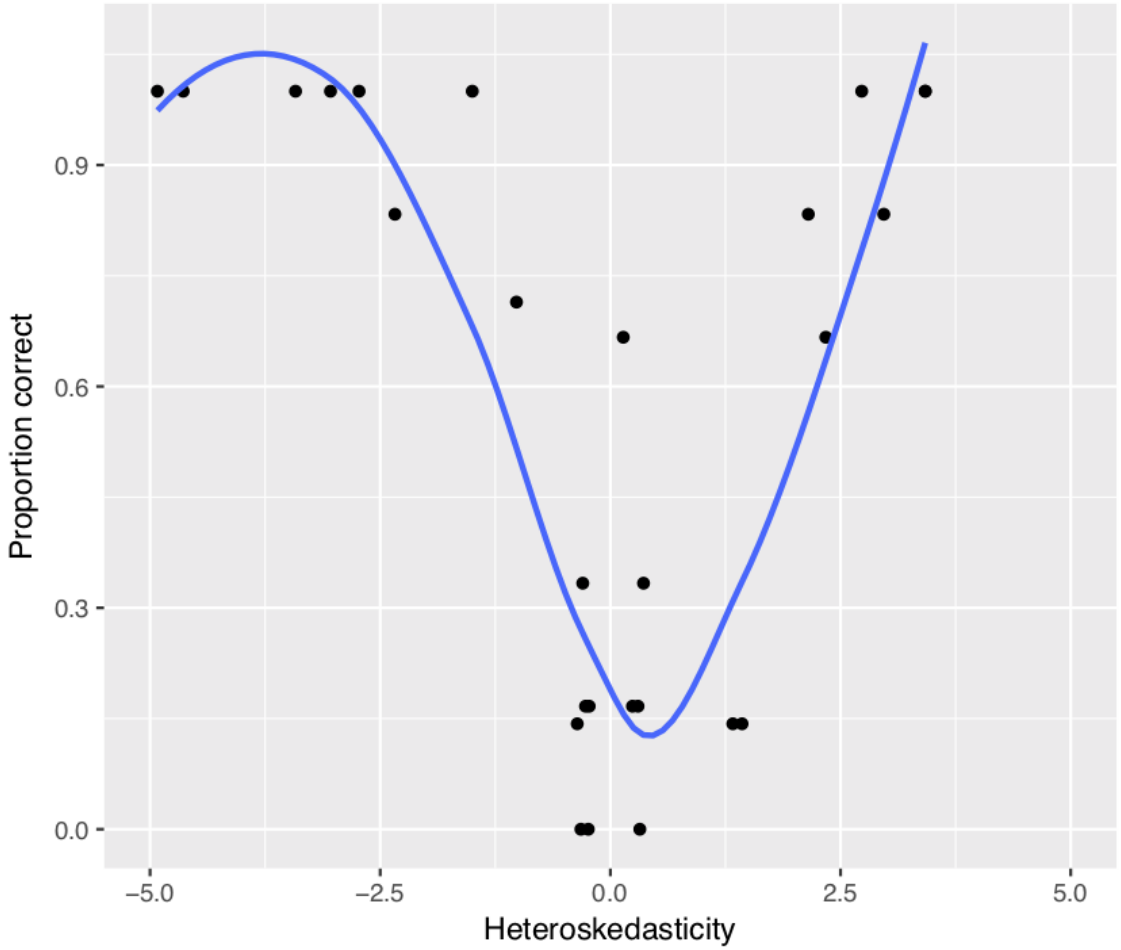

Figure 3.4: *Proportion of correct answers for each lineup question against the simulated correlation "a" from human evaluation*

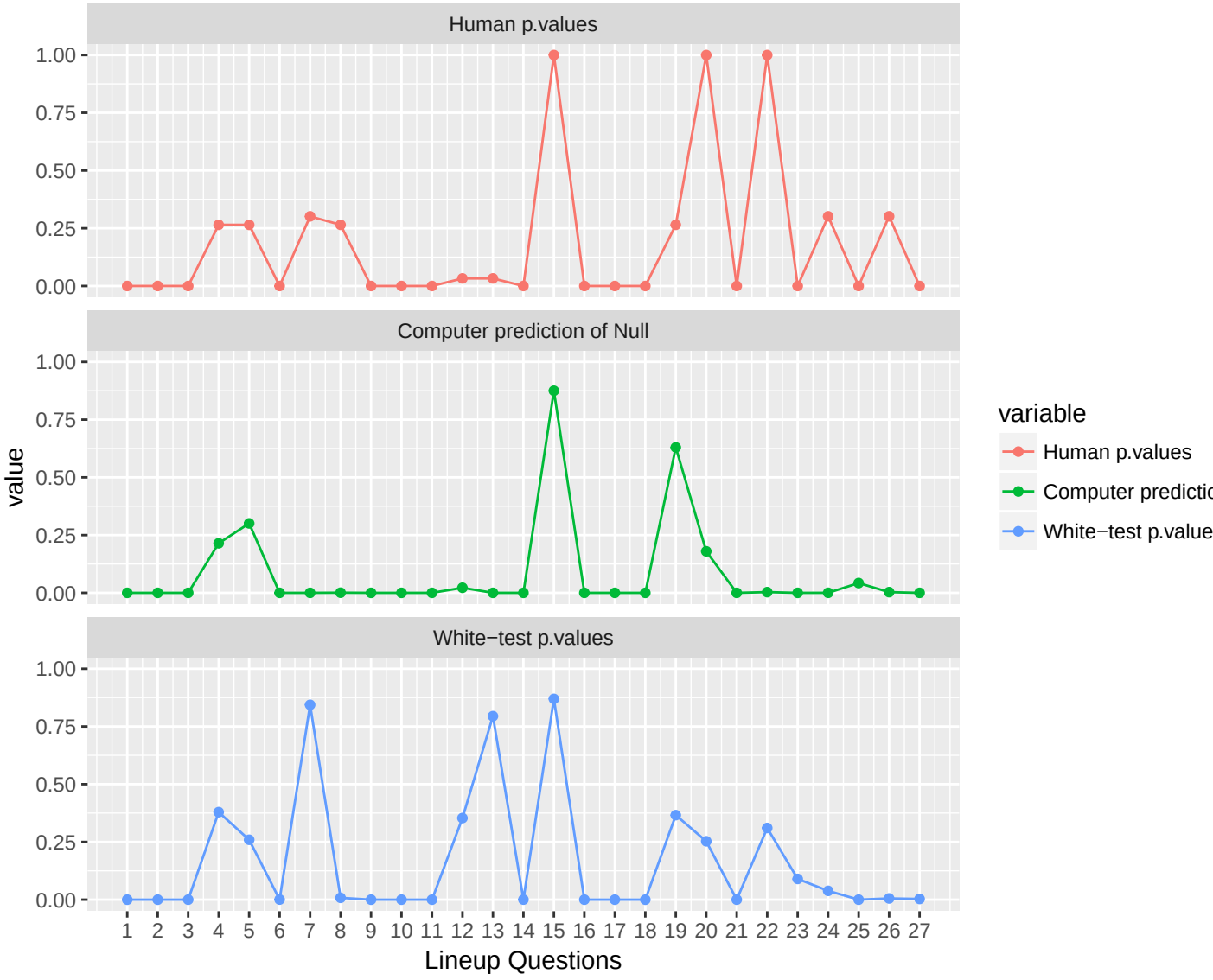

Figure 3.5: *P.values for each real plot given by human, computer and white-test respectively.*

Chapter 4

Conclusion and discussion

SUMMARISE RESULTS BRIEFLY The convnet can be trained to perform similarly to human perception, when the structure in the residual plots is very specific. Performance on the linear vs no structure is comparable to the human subjects results. Performance on detecting heteroskedasticity is also good.

SUMMARISE OTHER FINDINGS BRIEFLY Performance of the convnet exceeds the results obtained by a t-test.

WHAT WOULD YOU DO DIFFERENTLY - Training of the convnet requires many images. Time to train the model was long.

- Other types of structure: could we have many classes of structure?
- How would you expand the experiment to build a model for general residual plot reading?

Bibliography

Anscombe, F (1973). Graphs in Statistical Analysis. *The American Statistician* **27**(1), 17–21.

Cairo, A (2016). "Download the datasaurus: never trust summary statistics alone". Personal blog.

Chollet, F et al. (2015). *Keras*. https://keras.io.

Chollet, F and J Allaire (2018). Deep Learning with R.

Majumder, M, H Hofmann, and D Cook (2013). Validation of visual statistical inference, applied to linear models. *Journal of the American Statistical Association* **108**(503), 942–956.

Matejka, J and G Fitzmaurice (2017). Same stats, different graphs: generating datasets with varied appearance and identical statistics through simulated annealing. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, pp.1290–1294.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. http://www.R-project.org/.

Simchoni, G (2018). "Applying deep learning for the visual inference lineup protocol". Personal blog.

Wickham, H (2009). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. http://ggplot2.org.

Wickham, H, D Cook, H Hofmann, and A Buja (2010). Graphical inference for infovis. *IEEE Transactions on Visualization and Computer Graphics* **16**(6), 973–979.

Wooldridge, JM (2015). *Introductory econometrics: A modern approach*. Nelson Education.