Can a deep learning model read residual plots?

Shuofan Zhang [*]
Research Assistant, Monash University
and
Dianne Cook
Professor of Business Analytics, Monash University

December 11, 2018

Abstract

Residuals plots are a primary means to diagnose statistical models. It requires human evaluation to determine if the structure in the plot is consistent with a 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.

Keywords: Model Diagnostics, Computer Vision, Statistical Graphics, Data Science, Exploratory Data Analysis

[*]The authors gratefully acknowledge . . .

1 Introduction

The derivation of hypothesis tests and their asymptotic distributions constitute a considerable part of the statistics literature. However, the derivation is often complex and the resulting test may lack power. For example, in time series analysis, the commonly used unit root tests all suffer from low power in distinguishing the unit root null from stationary alternatives. In addition, data plots show a lot more information than a single statistic. A good illustration would be Anscombe's Quartet. *"It is a set of four distinct data sets 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 as shown in figure 1 (Matejka & Fitzmaurice 2017). Instead of using single statistics, if the information in the plots can be utilized in performing hypothesis tests, the power (1-β) may be improved.

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 a valid hypothesis test. One example of these protocols that provides inferential validity is called "lineup" which was introduced by Wickham et al. (2010). *"The protocol consists of generating 19 null plots (could be other numbers), 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 likely to be different from the null hypothesis, so we reject the null hypothesis with 5% chance to be wrong (Type I error). Because if all 20 plots are generated from the null distribution, the chance of one plot being picked is 1/20. With the assistance of "lineup", we avoid falling into the trap of apophenia where we see patterns in random noise. This protocol has been proved to be valid and powerful theoretically as well as practically through human experiments (Majumder et al. 2013). The human factors that may influence visual statistical inference were also investigated by Majumder et al. (2014). The experiments in Majumder et al. (2014) suggest that *"individual skills vary substantially, but demographics do not have a huge effect on performance."* Although there

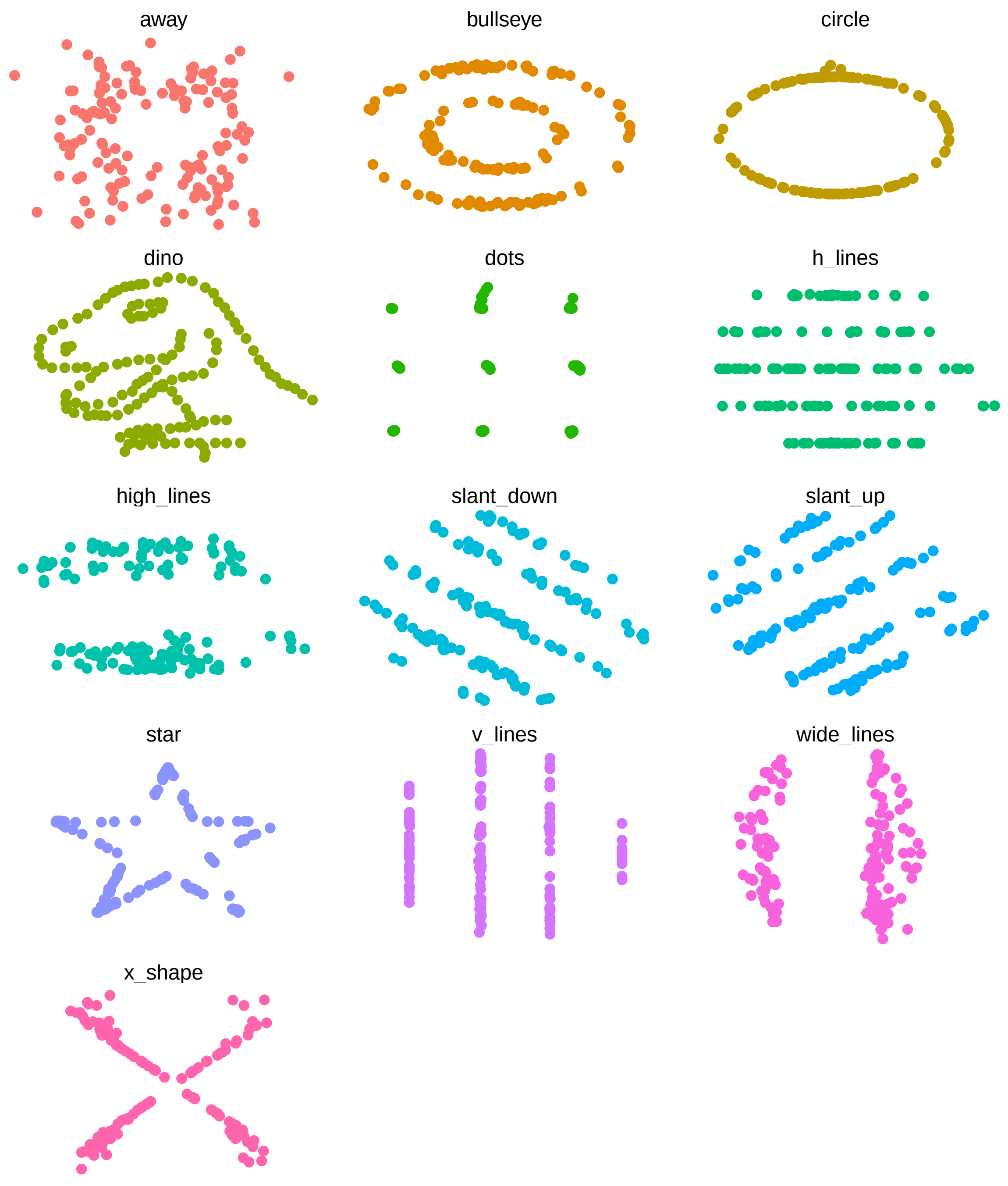


Figure 1: Each dataset has the same summary statistics to two decimal places: (E(x)=54.26, E(y)=47.83, Pearson's r=, sd(x)=16.76, sd(y)=26.93

are some statistically significant factors such as “having a graduate degree” and “living country”, the effects of these factors are minimal. These results demonstrate the robustness of the test against different human factors. Figure 2 is an example of the lineup. Which plot do you think is the most different? If you choose plot one, we are 95% confident to reject the no-relationship assumption between the two variables, “hp” and “disp” (Simchoni 2018). The lineup protocol can also be used for other types of testing by choosing different types of plot. For example, normality can be tested using QQ plot; the difference in mean can be tested using box plot.

The question that arises today is whether the human evaluation can be aided and supplemented by computer. If it is feasible, we can have a computer process a lot more data than a human can manage. Thus, the cost of rendering visual inference will become much lower. Therefore, in this study, we introduce the computer vision technique to test the null of no structure against linear patterns in a scatter plot, as an alternative to the conventional t-test. To be more specific, the hypothesis we are trying to test is:

H_0: There are no relationships between the two variables in the scatter plot.

H_1: There is a linear relationship between the two variables in the scatter plot where all Gauss-Markov assumptions are met.

The accuracy of the computer model is compared to both a database of human evaluations from Majumder et al. (2013) and the t-test. The 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 the 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 failed to pick up a strong non-linear relationship even though the Pearson’r is as high as -0.84 (Simchoni 2018). 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. A strong correlation does not necessarily mean a linear relationship. We should always refer to the plot before making any statement. In addition, if we require the model to be more flexible, it can be realised by adjusting the training design accordingly.

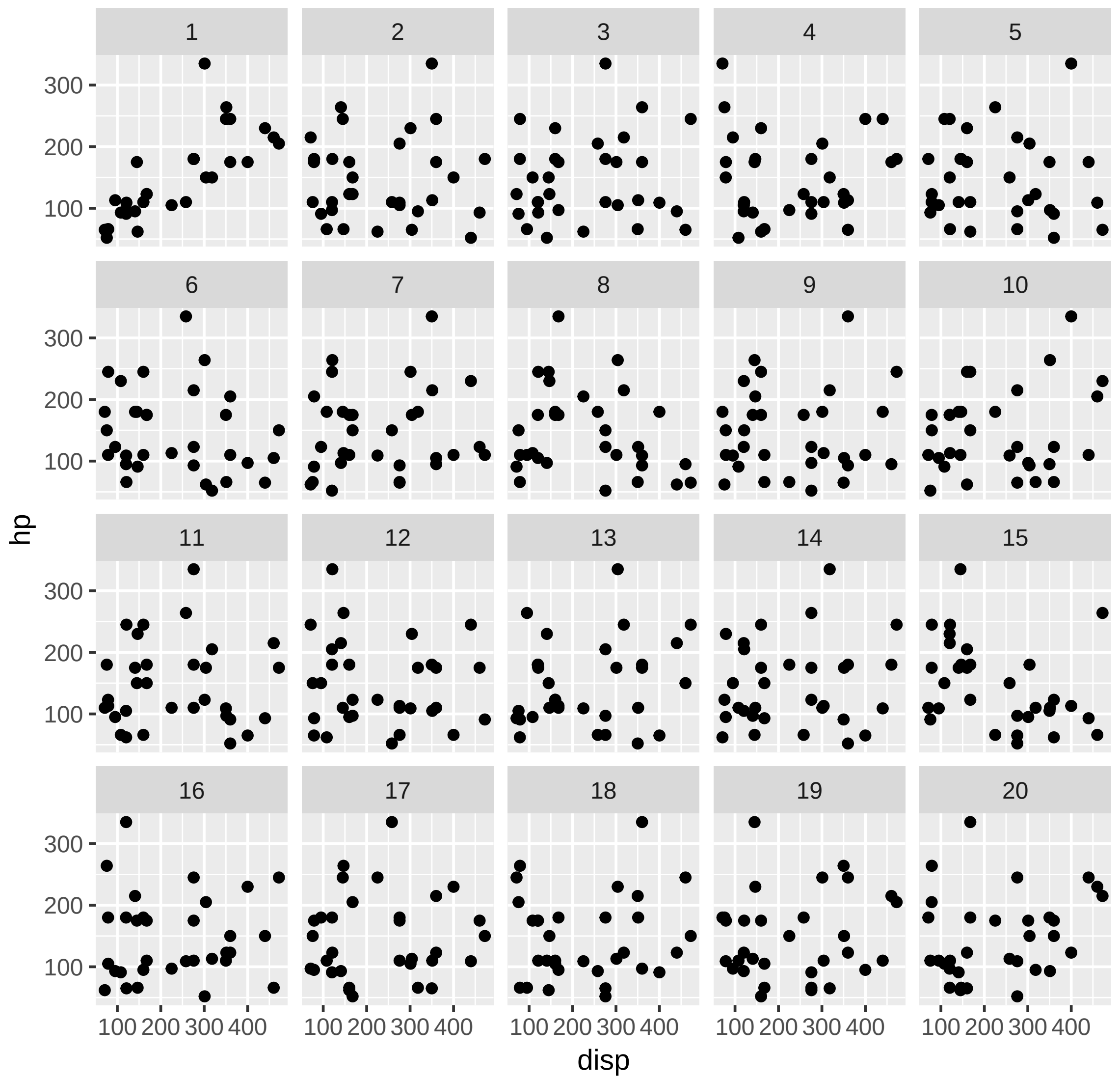


Figure 2: Scatterplot lineup example: one plot is the data, the rest are generated from a null model assuming no relationship between the two variables. In this lineup it is easy to see that plot 1, which is the data plot, is different from the rest.

2 Experimental Methods

2.1 The Human Evaluation

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

In Majumder et al. (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 the human to evaluate. The human subjects were hired from Amazon Mechanical Turk, a marketplace for work that requires human intelligence. (Buhrmester et al. 2011)

The controlled model in their first experiment is

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

where $\beta_0 = 5, \beta_1 = 15, X_1 \sim Poisson(\lambda = 30), \epsilon_i \sim N(0, \sigma^2), i = 1, 2, ..., n,$ β_2 used in generating real data is specified in table 2.1. 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. It 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. 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 $N(0, \hat{\sigma}^2)$.

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

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

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 Turk experiment one and Turk experiment two are shown in table 1.

Table 1: Parameter values for simulation in Turk's study.

Sample size (n)	Error SD(sigma)	Exp 1-beta2	Exp 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

Their second experiment utilized 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 in this study for comparison with the deep learning model. An example lineup question in Turk experiment 2 is shown in Figure 3. 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 $H_o : \beta_k = 0$. The main procedures of the human evaluating lineup are given in figure 4. "Real data" and "null data" stand for datasets simulated under the alternative hypothesis and the null hypothesis respectively.

2.2 The Computer Model

The model we use is the convolutional neural networks, also known as convnets, a type of deep-learning model *"almost universally used in computer vision applications"* (Chollet & Allaire 2018). The very first convolutional neural networks, called the "LeNet5" which was born in 1994, propelled the field of deep learning. However, this technique was in incubation from 1998 to 2010. In recent years, with the increasing data availability and more advanced technology, the design of the neural network architecture became more and more successful. Many types of neural network architectures have been developed since then, such as the "Dan Ciresan Net" which enabled the implementation of GPU for the first time and the "AlexNet", which used the so-called "ReLU" function as the activation function and started a small revolution in the deep learning world. (Culurciello 2017) Basically convolutional neural networks has two interesting properties: "the patterns they learn are translation invariant", and "they can learn spatial hierarchies of patterns" (Chollet & Allaire 2018). The first property implies that once the model learns how to recognize the linear patterns, it can detect these patterns regardless of their direction, thus handling negative/positive relationship automatically in our case. The architecture used in this study is a fundamental one.

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. One can think of the deep learning neural network as a complex nonlinear model which could estimate millions of parameters ($\mathbf{w}$) with a big enough dataset. As usual regression problem, to get the estimates of unknown parameters ($\mathbf{w}$), we need to provide the model with the dependent variable (y_i) and the 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. The estimation method for the deep-learning model is called "backpropagation" algorithm which is a way to train chains of parametric operations using gradient-descent optimization. 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 is determined per each question. In this paper, the deep learning model is expected to complete binary classification task, detect "linearly correlated variables" from "unrelated variables". As introduced by Chollet & Allaire (2018), *"crossentropy is usually the best choice (as the loss function) when you're dealing with* *models that output probabilities"*. 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,

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

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. Once we have the estimated parameters ($\hat{\mathbf{w}}$), we then can use them to classify unseen data plots, e.g. to perform a hypothesis test.

The main procedures involved in constructing and selecting a convnets model are shown in figure 5. The convnets is trained on "train" and "validation" set. A certain number of iterations over all samples are done, the fitted convnets given by each iteration are saved, one best model is chosen as the representative for the computer according to the overall accuracy on the unseen "test" set.

All convolutional neural network related work is done by the Keras (Chollet et al. 2015) package in R (R Core Team 2013), which interfaces to the python software. The R package ggplot2 (Wickham 2009) is used to generate the plots. All plots are resized to 150 × 150 pixel and saved as png. This size is similar to the plot size used in the lineup for human evaluation.

"A convnets takes as input tensors of shape (image height, image width, image channels)" (Chollet & Allaire 2018). The channels are normally equal to three for RGB. In our case, the input tensors are of shape 150 × 150 × 1. The channel is equal to one because the input data is 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(150, 150, 1) to the first layer. The R codes below are used to build the convnets in R. From figure 6, the output shape changes after every layer of "conv" and "pooling" operations. The original 150 × 150 × 1 image is finally sliced into a 7 × 7 × 128 object (3D tensor). The figure 7 describes how "convolution" and "max pooling" operation works. By "convolution" the image matrix is multiplied by a filter, different filter gives different output matrix which extracting different features from the image. Max pooling select the max number within a

certain area. 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. $S(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{x}}}$. From this model structure, we can also see that a total number of 3,452,545 parameters need to be estimated, this is done by gradient descent. 10 epochs (1 epoch = 1 iteration over all samples) are done for training in our first experiment. The model specifications given by each epoch are saved, the one gives the overall highest accuracy is chosen to represent the computer.

```r
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")
```

As for the time needed for computer training, since a large number of images are employed in this study, and we rely only on the CPU, 10-20 hours is required for generating and saving all images, another 10-20 hours will be necessary for the convnets model to be trained and tested. With an NVIDIA GPU, this duration will be shortened significantly.

2.3 Simulation

In total 240,000 data sets are simulated under each the null and the alternative hypothesis. 100,000 of them are set apart for training. Another 40,000 of the data sets are set apart as the validation set in order to monitor during training the accuracy of the model on data it has never seen before. And the rest 100,000 data sets become the unseen test set. We make the test set relatively large that we can compare the performance of the convnet with the t-test with small variations. As for the labels given to each image, we use the true population as the samples' identification directly.

2.3.1 Under the Alternative Hypothesis

For ease of exposition, the plots used for training and testing is the scatterplot between the dependent variable Y and the independent variable X. It can also be considered as the residual plot of such data fitting to a constant model. Although only the simple regression model is considered in this paper, but many of the results can be generalized to other cases including multiple regression models. Because the "statistics" we use is the scatterplot, in terms of teaching the computer to recognize linear patterns from others, one variable is enough to generate different patterns. And this makes the design process much simpler.

The design for our model under the alternative hypothesis is similar to what Simchoni (2018) did in his blog. But the parameters are tailored to compare the computer performance with the Turk study results.

The model under the alternative is designed as:

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

And all the parameters in our model were designed to cover the range used in the second experiment in Turk study (Majumder et al. 2013). Therefore, the relevant parameters in our model are generated using the following specification.

- $X \sim N[0,\ 1]$
  Distributions of X has an 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 when the sample size is large; while looking like a circle if X follows a normal distribution.

- $\beta_0 = 0$

  Intercept is arbitrarily set to be zero because it has no impact on 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 distributed from 1 to 12.

- $n = U[50, 500]$

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

Figure **??** shows four example plots generated using the specifications above. To facilitate the computer vision, all texts, ticks, and titles of X and Y axes are removed, so does the background grid. Under this controlled structure, a total number of 240,000 datasets are simulated. Figure **??** contains a histogram of the simulated n, a histogram of the simulated β, a histogram of the simulated σ, a histogram of the estimated sample p-value, a scatter plot of β against n and a scatter plot of σ against n. These plots show good coverage over the alternative parameter space.

2.3.2 Under the Null Hypothesis

When the two variables under tested are independent of each other, then the data plots will not show any systematic patterns theoretically. It is true that there must 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/insignificant" samples. Because the answer to the question that if the deep learning model can distinguish from patterns formed by chance and by nature is also interesting. The model is designed the same as the linear model:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i, \quad i = 1, \dots, n$$

with elements of the model generated using the same specification as the linear model, except
- $\beta_1 = 0$

Hence, the coefficient of X_i is always zero. So X and Y are uncorrelated of each other.

Figure **??** are four example plots generated using the specifications above. Same as the linear model simulation, a total number of 240,000 datasets are simulated under this structure. Figure **??** contains a histogram of the simulated n, a histogram of the simulated σ, a histogram of the estimated sample p-value and a scatter plot of σ against n. These plots show good coverage over the null parameter space. 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.

3 Results and Discussion

The plot of the training history in figure 12 shows high accuracy achieved in both train and validation set (93%-94%); slight overfitting starts from the fourth epoch; the variation of the values of accuracy and loss in validation set are very small after the fourth epoch. Hence, both our convnets and dataset are large enough and the training of our first experiment can be concluded. Then 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 2. In this table, the "$1 - \alpha$" means the accuracy of each computer model tested on the "null data" in the test set only. α here is an 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 in this table is calculated at 5% significance level.

The 8th model is chosen according to the overall accuracy on the test set. We should note that since the majority of the real data plots in Turk's experiment have been generated from linear relationships (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 the α is approximately 2% for the 8th computer model, the 5% significant t-test and 5% human evaluations may have higher power than the computer model.

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

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: 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.

plots);

- Create a separate test directory for the 70 "real plots" exclusively;
- The computer model's predicted accuracy over the 70 "real plots" is recorded as the model's performance.

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

- Count the 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 lineups;
- Calculate p-value associated with each real plot using the formula introduced in section 2 of Majumder et al. (2013);
- Draw the 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. Because the α estimated from the computer model is close to 2%. Therefore, 2% significant t-test and 2% significant human conclusion are also included to give a complete picture of the comparison. The comparing result given in table 3 is interesting. Human achieves the highest accuracy, and the conclusion from the human evaluation is robust to smaller p-values; 5% significant t-test is the second best, 2% significant t-test and the computer model give the same results.

Table 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

Under the condition specified for simulation, the conventional t-test is known to be the uniformly most powerful (UMP) test in terms of detecting the linear relationship according to the Neyman–Pearson lemma. (J.Neyman & E.S.Pearson 1933) Although human achieved the best performance in the Turk experiment dataset, it does not mean computer does badly since the Turk experiment dataset only contains 70 plots. As we can see from table 2, t-test and convnets behave quite similarly on both the test set and the Turk's experiment data.

To check if the t-test and the convnets do perform similarly, we calculated the accuracy of the 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 coincides with the α chosen by the convnets. And since the α of convnets is from 0.0142 to 0.0347 on the test set, we truncated the t-test data to create figure 13. The upper dots represent overall accuracy achieved by convnets (red) and t-test (green), while the lower dots stand for the estimated power in the test set. The smooth line overlaid aids the eye in seeing patterns. From this graph, we can see the convnets and t-test perform very similarly, while t-test has overall better performance.

Given the size of our test set (200,000 images totally), it is reasonable to assume that the convnets is, in fact, trying to approach the t-test. In other words, the best strategy the convnets learned, in this case, may be close to the t-test, the UMP.

In summary, performance of the convnets on testing the linear vs no structure is comparable to both the human subjects' results and the t-test.

Although constructing the assumptions used for simulation and the implications for violation maybe challenging. This study gives hope to the future utilization of convnets in reading the residual plot. Unlike the conventional distribution tests, convnets does not require convoluted mathematical derivations and can be applied to any visualizable problems. Compared to human, the convnets is more stable in that it always gives consistent prediction once it is well trained while different groups of people may have different opinions on one plot. In addition, the computer training and testing can be done by a single computer which barely costs anything (other than our genuine efforts) while human evaluations could be much more expensive.

On the other hand, the computer also has pitfalls. Its ability is limited to the patterns provided by the training. The more patterns we want the convnets to recognize, the more structures we need to feed it. As for the future study, more types of structure could be considered, for example, testing heteroskedasticity, non-linear relationship, outliers, etc. The binary classification can also be extended into a multiclass classification, e.g. one convnets can be trained to recognize several departures from the null. What's more, to provide a more reliable comparison between human and computers, a larger human experiment is required. For an even further step, the convnets can be designed for even more general hypothesis testing purposes, the biggest challenge would be finding the most appropriate plot (the test statistic) which can show the key features for certain tests. For instance, if we are interested in telling a time series data with unit root from a trend stationary one, the most suitable plot may be the time plot.

References

Anscombe, F. (1973), ‘Graphs in statistical analysis’, *The American Statistician* **27**(1), 17–21.

Buhrmester, M., Kwang, T. & Gosling, S. D. (2011), ‘Amazon’s mechanical turk: A new source of inexpensive, yet high-quality, data’, *Perspectives on psychological science* **6**(1), 3–5.

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

Chollet, F. & Allaire, J. (2018), ‘Deep learning with r’.

Chollet, F. et al. (2015), ‘Keras’, `https://keras.io`.

Culurciello, E. (2017), ‘The history of neural networks’.
URL: *http://dataconomy.com/2017/04/history-neural-networks/*

J.Neyman & E.S.Pearson (1933), ‘Ix. on the problem of the most efficient tests of statistical hypotheses’, *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* **231**(694-706), 289–337.

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

Majumder, M., Hofmann, H. & Cook, D. (2014), ‘Human factors influencing visual statistical inference’, *arXiv preprint arXiv:1408.1974* .

Matejka, J. & Fitzmaurice, G. (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.
URL: *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.
URL: *http://ggplot2.org*

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

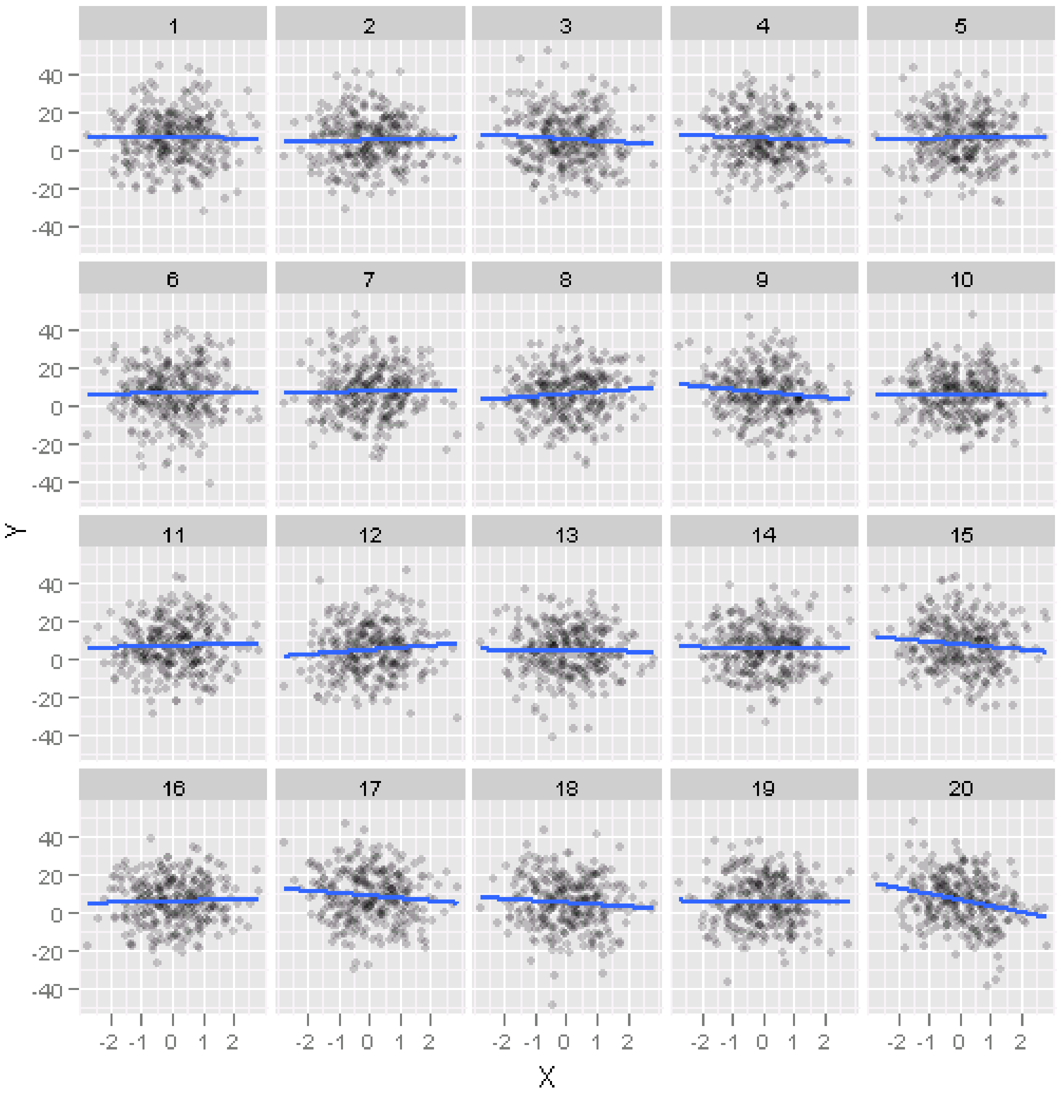


Figure 3: One of 70 lineups used in experiment 2 Majumder et al (2012). Of the 65 people who examined the lineup, 63 selected the data plot, which is in position 20.

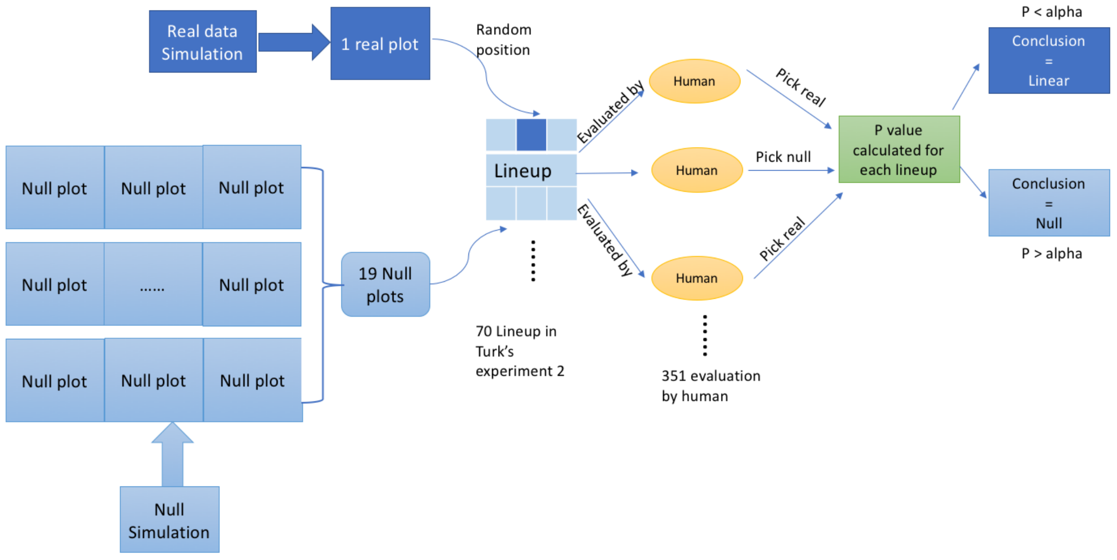


Figure 4: Diagram illustrating the process of human subject evaluation of lineups, and how performance is computed.

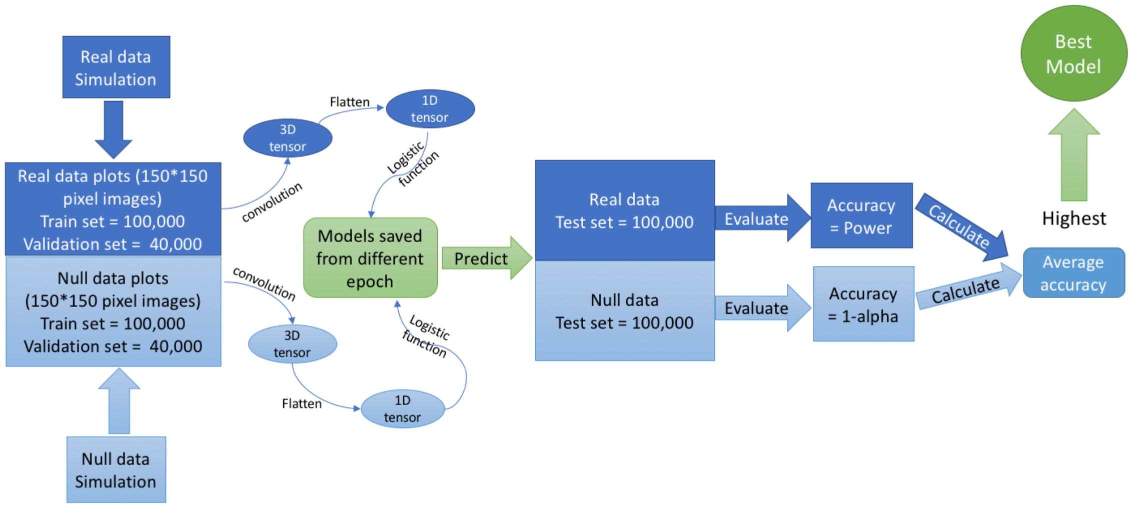
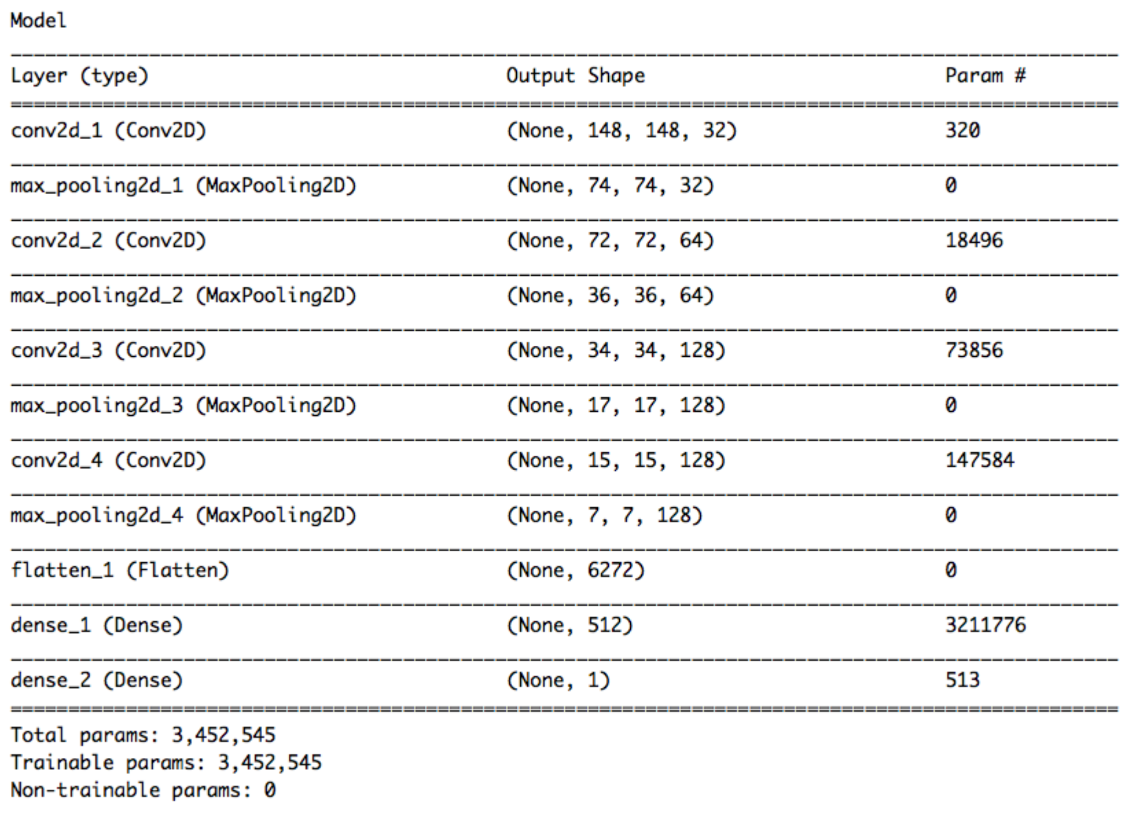


Figure 5: Diagram illustrating the training, diagnosis and choice of the computer model. Based on 480,000 simulated data sets used to create 150×150 pixel images, divided into train, validation and test sets.



```
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
```

Figure 6: The deep learning model structure used for the experiments.

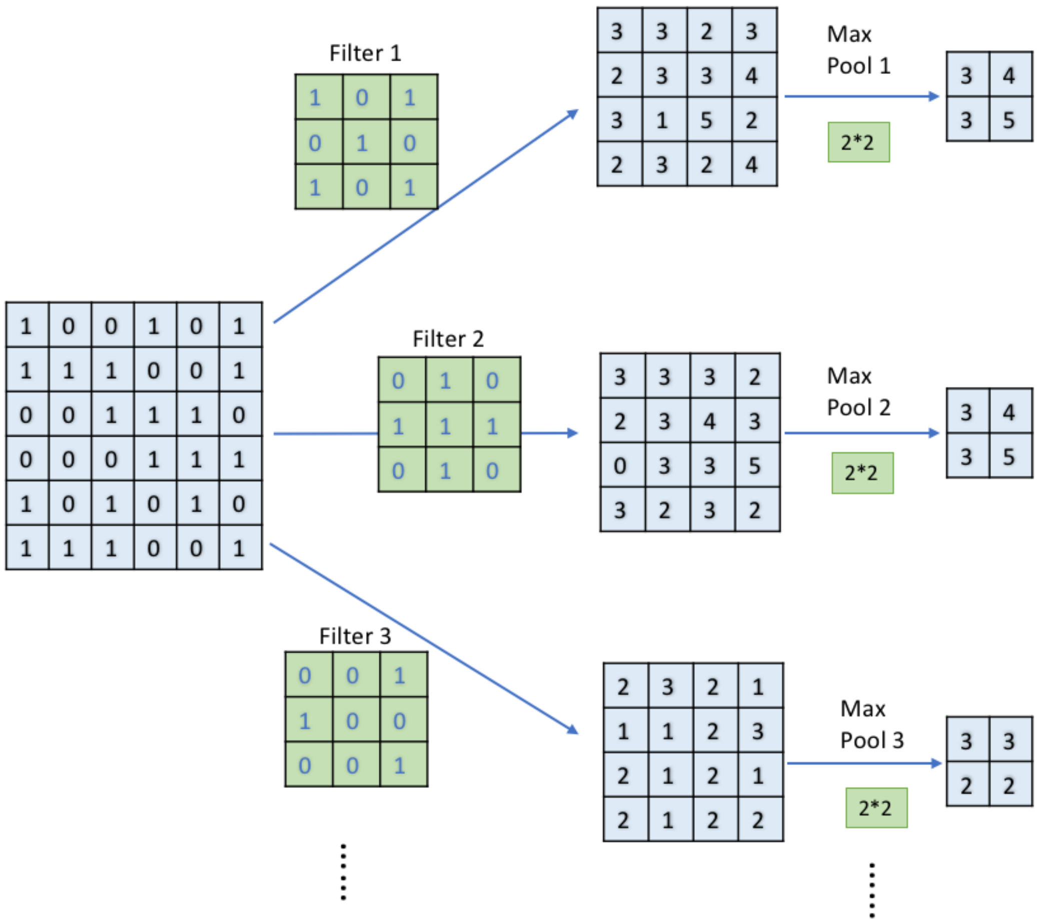


Figure 7: Illustration of convolution and pooling steps on an image. The convolution step applies a fixed number of filters to sliding windows of 3×3 cells. Pooling applies a statistic to distinct 2×2 tiling of the image. In our model, the statistic used is the maximum of the four values. These transformations 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.

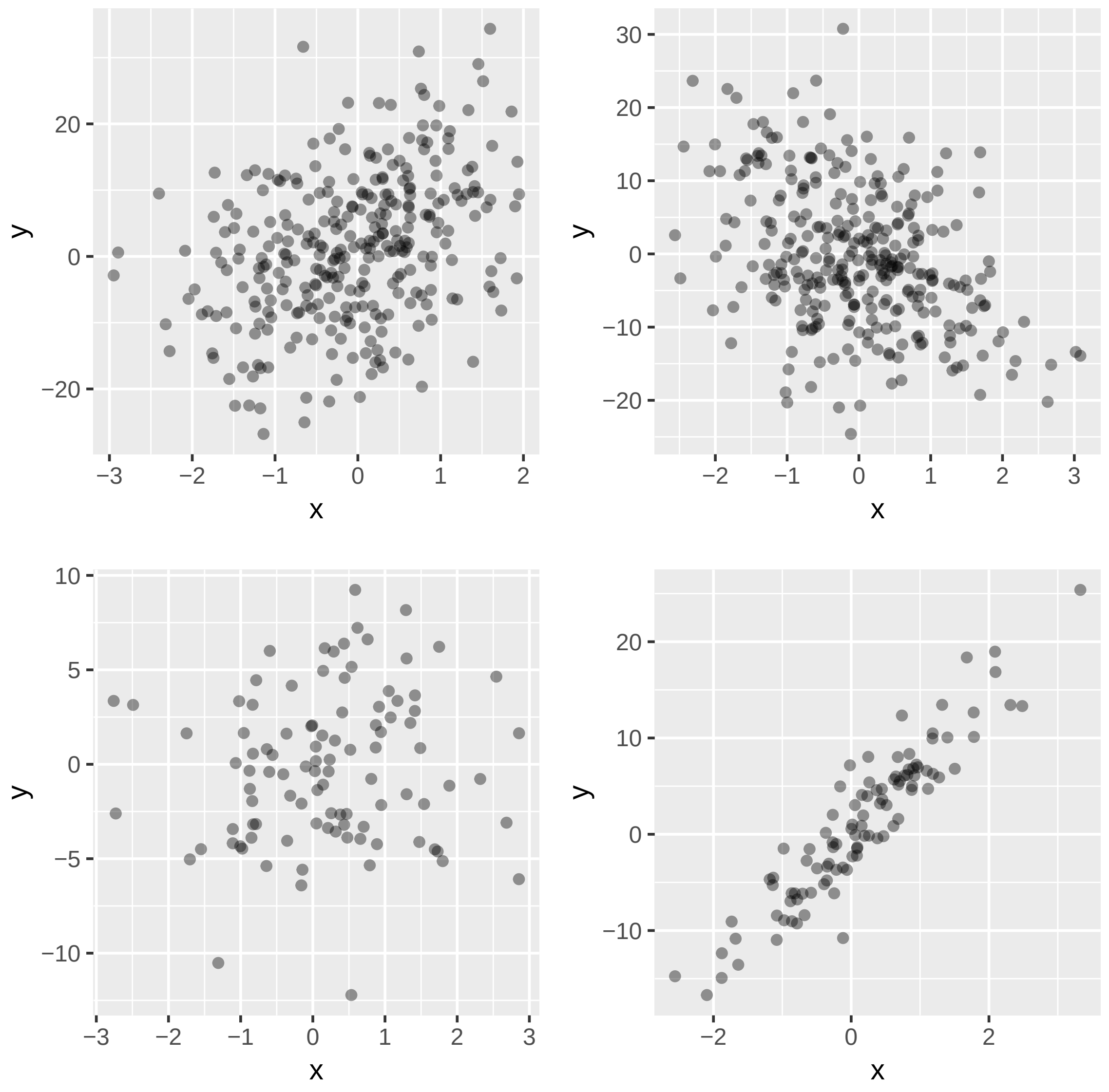


Figure 8: Four examples of data plots generated from the classic linear model.

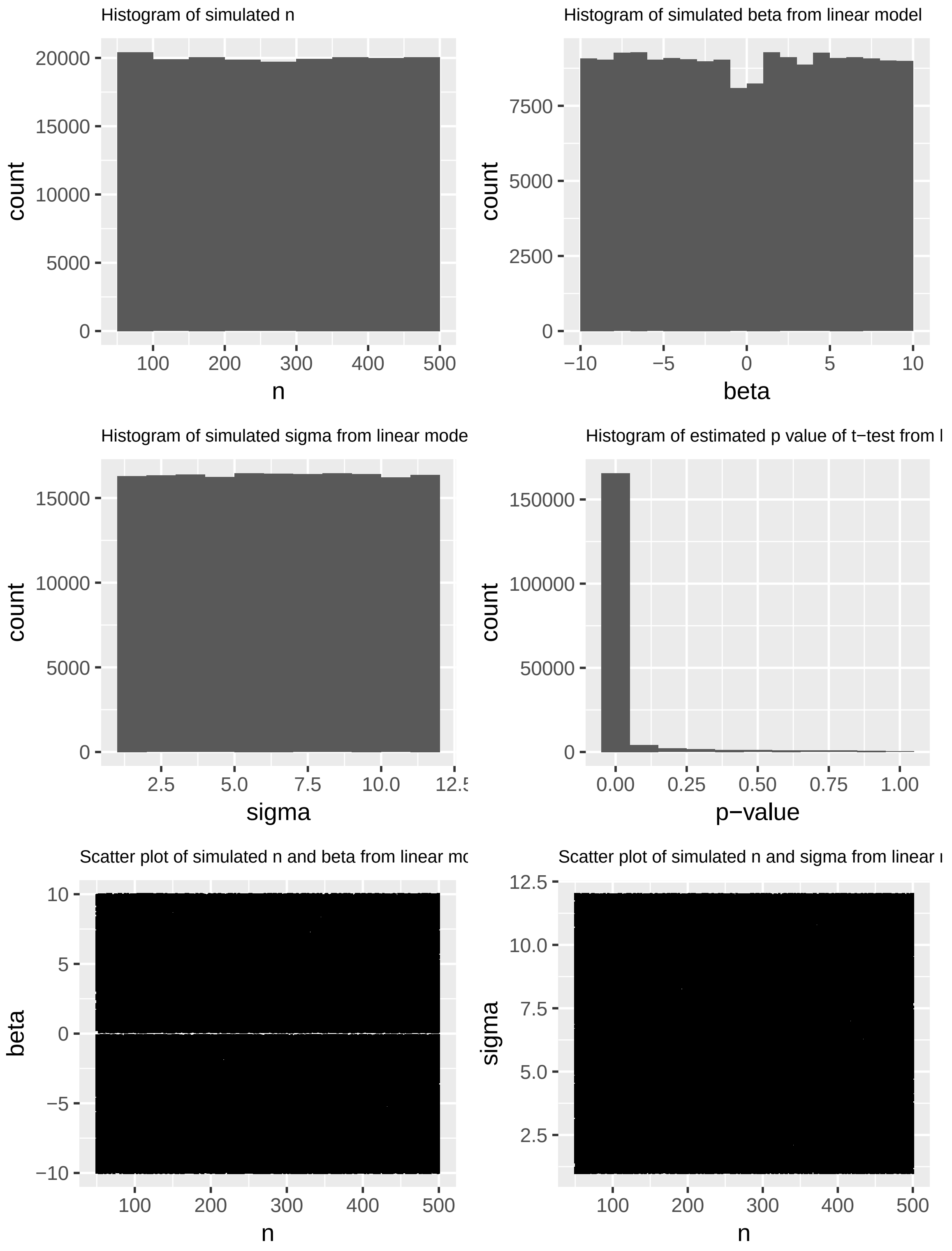


Figure 9: Overview of parameter values used in the linear class simulation, for computer model training. Good coverage is obtained across the parameter space.

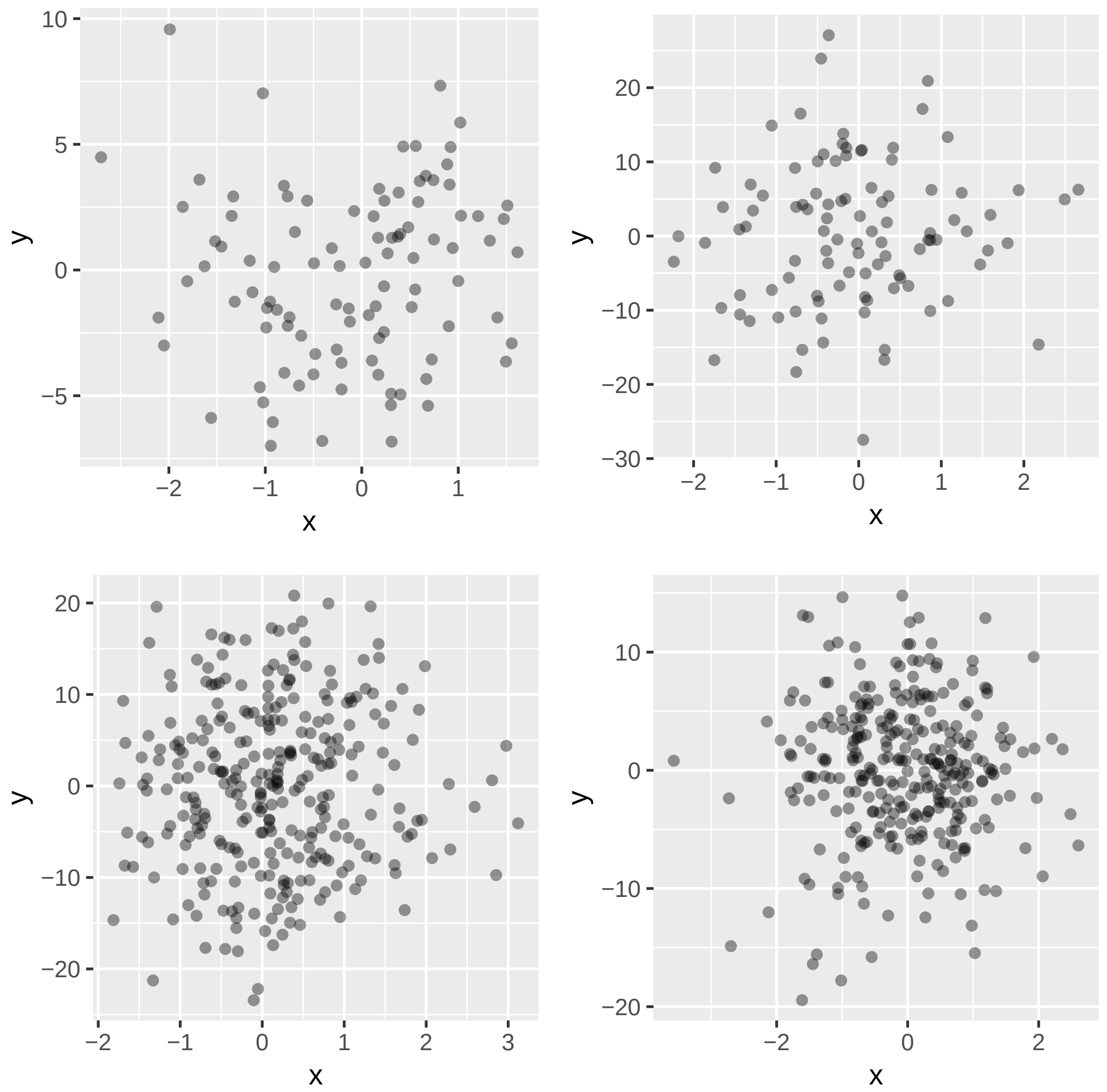


Figure 10: Four examples of data plots generated with two independent variables

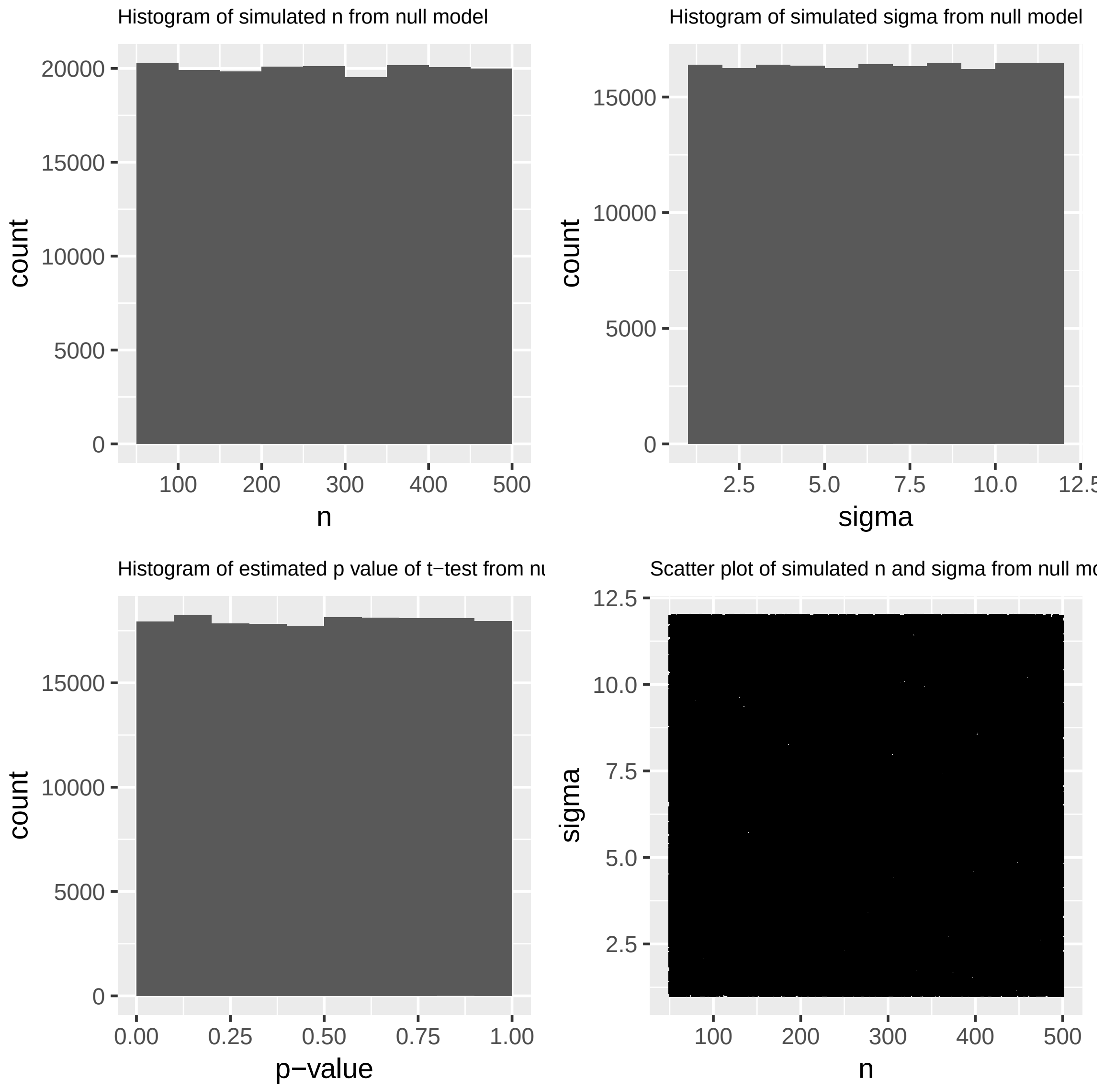


Figure 11: Overview of parameter values used in the null class simulation, for computer model training. Good coverage is obtained across the parameter space.

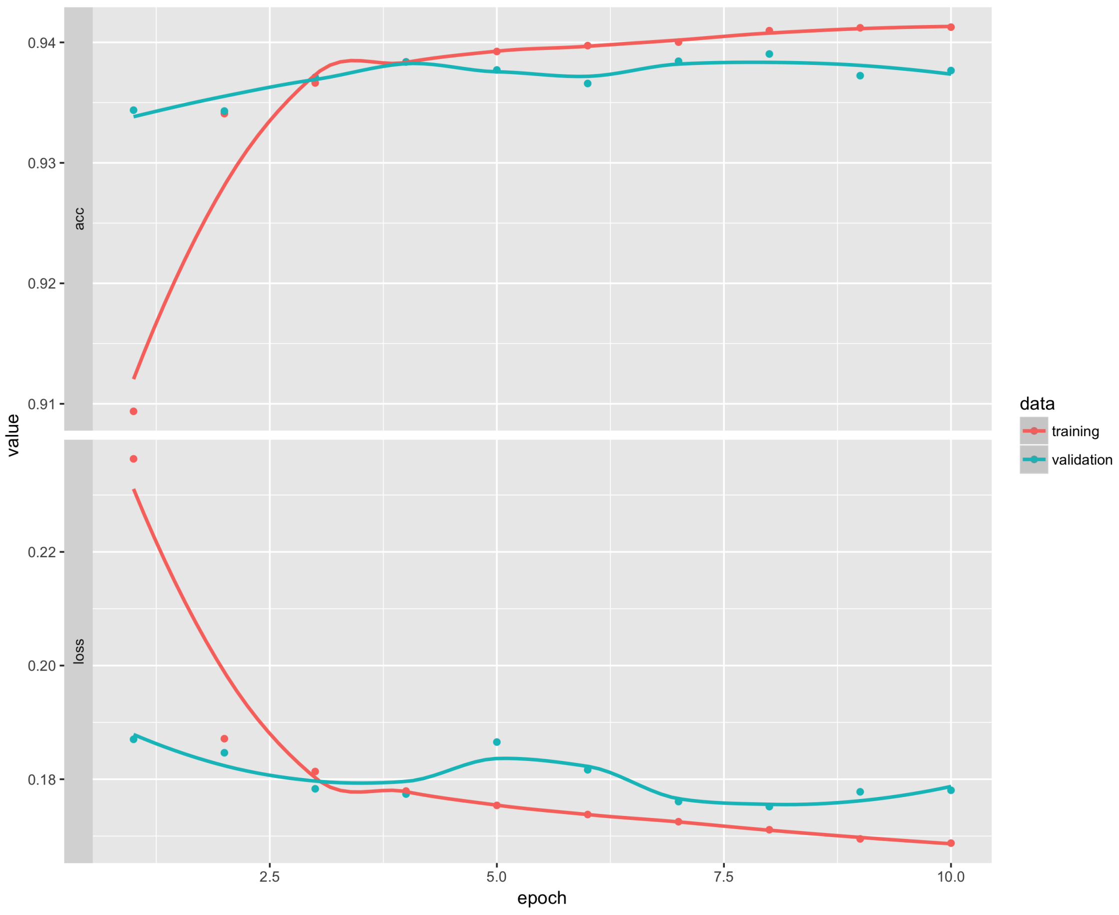


Figure 12: Training and validation metrics of linear vs. null model in our first experiment. The top plot is the accuracy achieved in train and validaton sets, while the bottom plot is the loss. Red presents model performance in train set while green presents validation.

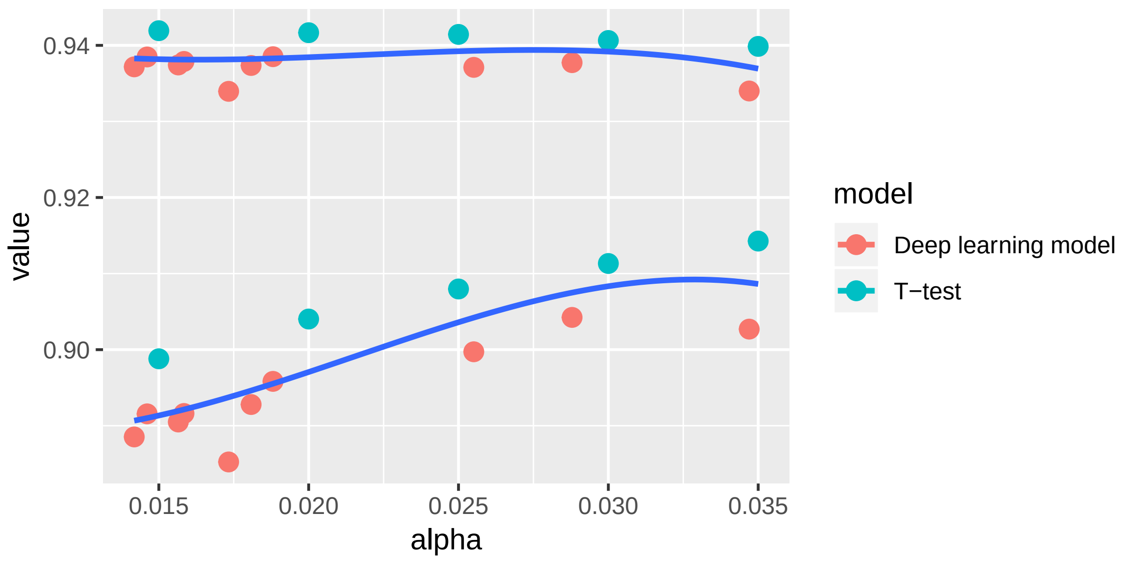


Figure 13: Comparison between computer model and t-test for alpha in (0.01, 0.04), they perform very similarly, but t-test has overall better performance.