titansoft

Question 1: After a seven-day test, which model will you choose, Model A or Model B?

Cleaning the data

```
library(readr)
data = read_csv("C:/Users/Jun Jie Choo/Dropbox/My PC (DESKTOP-IRUDREG)/Desktop/model_compare.csv")
```

```
## Parsed with column specification:
## cols(
##   Date = col_date(format = ""),
##   By = col_character(),
##   `Margin%` = col_character(),
##   Customer_Count = col_double(),
##   Sales_Amount = col_double(),
##   Gross_amount = col_double()
## )
```

```
#rearranging the dataset for clearer view
new_data = data[1:7, 1]
new_data["margin_A"] = data[1:7, 3]
new_data["margin_B"] = data[8:14, 3]
new_data["customer_count_A"] = data[1:7, 4]
new_data["customer_count_B"] = data[8:14, 4]
new_data["sales_amount_A"] = data[1:7, 5]
new_data["sales_amount_B"] = data[8:14, 5]
new_data["gross_amount_A"] = data[1:7, 6]
new_data["gross_amount_B"] = data[8:14, 6]

#creating new variables
new_data["sales_per_customer_A"] = data[1:7, 5]/data[1:7,4 ]
new_data["sales_per_customer_B"] = data[8:14, 5]/data[8:14,4 ]
new_data["gross_per_customer_A"] = data[1:7, 6]/data[1:7,4 ]
new_data["gross_per_customer_B"] = data[8:14, 6]/data[8:14,4 ]
```

Preliminary analysis

First, I am going to analyse the average margins over 7 days for both models A and B. The day to day margin is more volatile and easily affected by random factors, therefore the average margin over 7 days is more indicative of the true margin. Let average margin be defined as:

$$avg\ margin = \frac{total\ gross\ amount\ over\ 7\ days}{total\ sales\ amount\ over\ 7\ days}$$

```
avg_margin_A = sum(new_data["gross_amount_A"])/sum(new_data["sales_amount_A"])
avg_margin_B = sum(new_data["gross_amount_B"])/sum(new_data["sales_amount_B"])

paste("avg margin for model A is", avg_margin_A)
```

```
## [1] "avg margin for model A is 0.0111305265240183"
```

```
paste("avg margin for model B is", avg_margin_B)
```

```
## [1] "avg margin for model B is 0.0119599830914471"
```

It would seem that the margins for model A and model B are very similar with a 0.08% difference. Therefore a finer analysis is needed to determine the better model. We need to analyse the components that make up avg margin, namely: total sales over 7 days and total gross over 7 days. By the same logic of avoiding day to day volatility, we analyse the respective total over 7 days.

```
total_gross_A = sum(new_data["gross_amount_A"])
total_gross_B = sum(new_data["gross_amount_B"])
total_sales_A = sum(new_data["sales_amount_A"])
total_sales_B = sum(new_data["sales_amount_B"])

paste("Total gross for model A is", total_gross_A)
```

```
## [1] "Total gross for model A is 337.6"
```

```
paste("Total gross for model B is", total_gross_B)
```

```
## [1] "Total gross for model B is 84.88"
```

```
paste("Total sales for model A is", total_sales_A)
```

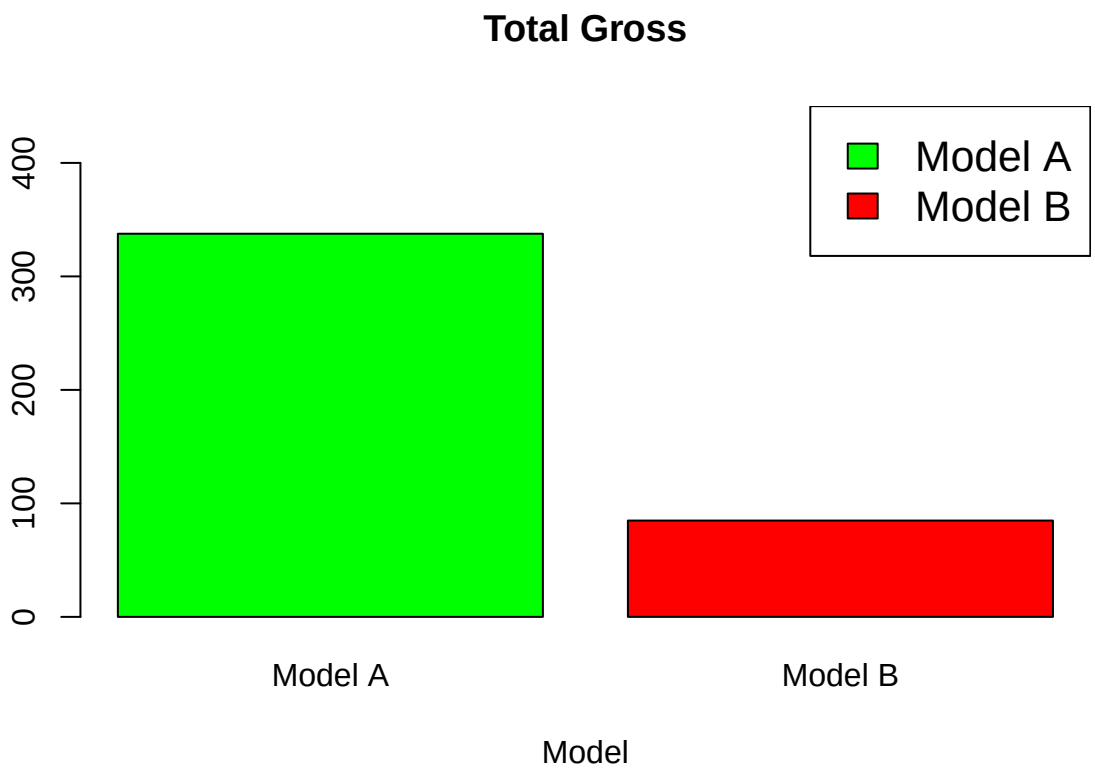
```
## [1] "Total sales for model A is 30331"
```

```
paste("Total sales for model B is", total_sales_B)
```

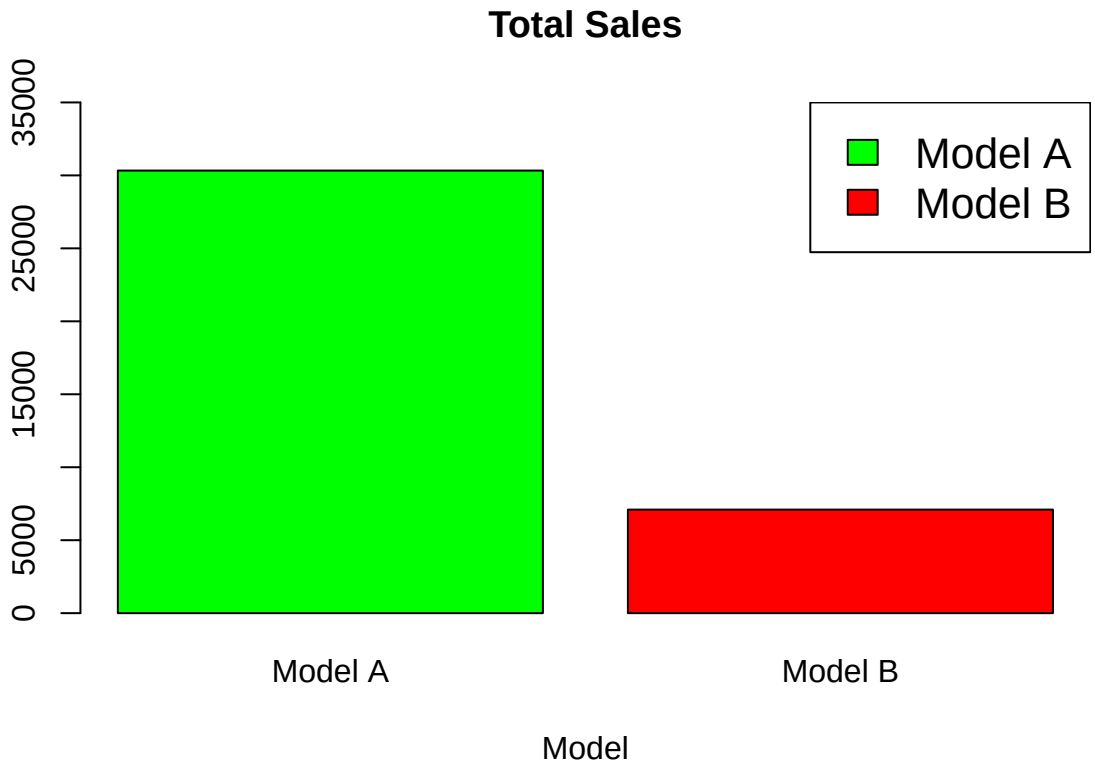
```
## [1] "Total sales for model B is 7097"
```

```
# Create the input vectors.
colours = c("green","red")
metric <- c("Total gross")
model <- c("Model A", "Model B")

# Create the matrix of the values.
Values1 <- c(total_gross_A, total_gross_B)
Values2 <- c(total_sales_A, total_sales_B)
```

```r
# Create the bar chart
barplot(Values1, main = "Total Gross", names.arg = model, xlab = "Model", col = c("green", "red"), widt

# Add the legend to the chart
legend("topright", model, cex = 1.3, fill = colours)
```




```r
barplot(Values2, main = "Total Sales", names.arg = model, xlab = "Model", col = c("green", "red"), widt
legend("topright", model, cex = 1.3, fill = colours)
```



Preliminary analysis conclusion

From this preliminary analysis, we can see that model A sells better and makes more profit in the short term. But under certain conditions, it is possible for model B to do better, and it is mainly due to the slightly higher margin 1.19% vs 1.11%. If model B is able to sell as much as model A, the total gross for model B will be: $\frac{84.88}{7097} * 30331 = 362.75$. And this represents a $\frac{362.75}{337.6} = 1.0745$: 7.45% increase in total gross for the company, which is a sizeable increase.

Possible scenario where Model B will be better

Given the question of "After a seven-day test, which model will you choose, Model A or Model B?", I will assume that the company does not plan to continue making both products simultaneously and only one or the other will be chosen.

With the above assumption, I can think of one scenario where model B can achieve the same amount of sales as model A. If both model A and B have low "rebuyability", like being a one-off purchase and the product is being sold to a niche industry with a small customer pool and therefore run a risk of running out of customers. Also given that the product life cycle is long enough and that alternative products such as from competitors either do not exist or are not as attractive as model B, it is possible for model B to sell as much as model A and therefore make a considerably larger profit (7.45%).

Also, in the previous paragraphs, I have been using total sales to have the same meaning as total customers, and that is because both model A and model B have very similar sales per customers, with up to a 2% difference. That is:

```r
sales_per_customer_A = total_sales_A/sum(new_data["customer_count_A"])
sales_per_customer_B = total_sales_B/sum(new_data["customer_count_B"])

paste("Sales per customer for model A is", sales_per_customer_A)
```

```
## [1] "Sales per customer for model A is 33.0043525571273"
```

```r
paste("Sales per customer for model B is", sales_per_customer_B)
```

```
## [1] "Sales per customer for model B is 32.4063926940639"
```

Therefore if model A and model B have the same total sales, they will have very similar amount of total customers.

Final conclusion

The recommendation of model A and model B depends on many variables. To reiterate, they are, product life cycle, existence of alternative products and how attractive they are (either from self or competitors), risk of running out of customers, "rebuyability" of product, etc. There is also the possibility that the average margin difference of 0.8% is due to random noise, rendering the previous discussion meaningless. Given all this, it might be wiser to pick model A due to its popularity and ease of sale, generating faster profits and faster recuperation of R&D costs for model A and model B. In general, model A will be my recommendation due to it being a safer, less risky option. But as the analysis above has shown, model B can give higher profits under certain scenarios, so it really depends.

Note: A statistical test like the t-test was not performed because a t-test with 7 data points will result in a test with very low power. That is to say, if there is indeed a difference of margins between model A and model B, it will be very hard for the t-test to detect this difference.

```r
set.seed(1)
margin_A_list = c(2.81,1.11,1.86,2.2,1.61,0.64,0.28)
margin_B_list = c(3.31,1.36,1.27,1.14,2.23,0.55,0.13)


t.power = function(nsamp=c(7,7),nsim=1000,means=c(mean(margin_A_list),
    mean(margin_B_list)),sds=c(sd(margin_A_list),margin_B_list)){
    lower = qt(.025,df=sum(nsamp) - 2)
    upper = qt(.975,df=sum(nsamp) - 2)
    ts = replicate(nsim,
    t.test(rnorm(nsamp[1],mean=means[1],sd=sds[1]),
            rnorm(nsamp[2],mean=means[2],sd=sds[2]))$statistic)

    sum(ts < lower | ts > upper) / nsim
}

paste("The power of the t-test for 7 samples is", t.power())
```

```
## [1] "The power of the t-test for 7 samples is 0.071"
```

This means that if there is a difference between margin A and margin B, we will have a 7.1% chance of detecting it, which is very small. Therefore, a t-test was not performed.

Q2: Based on the 10 user data in our system, should we get this new customer with the following information below?

Cleaning the data

```
buyer = read_csv("buyer.csv")
```

```
## Parsed with column specification:
## cols(
##   User = col_double(),
##   Age = col_double(),
##   Gender = col_character(),
##   `Annual income` = col_character(),
##   Married = col_logical(),
##   Buy = col_logical()
## )
```

```
#replacing k in annual income with thousand
data_clean <- function(x) sapply (strsplit(x , '[k]' ), `[` , 1)
buyer["Income"] = as.numeric(data_clean(buyer["Annual income"][[1]]))
```

Upon initial inspection, there are a few things to note about the dataset. No buy decision was issued to anyone below 300k annual income. Amongst people who received a buy decision, 3 out of 4 are married, which indicates to me that married individuals are preferred. The buy decision for user 2 and 5 are very peculiar to me, they are both married, have the same annual income, and are of similar age, but they received different buy decisions. I can only speculate it is because of their difference in gender, in which females are preferred. There does not seem to be an obvious pattern for age preference when issuing buy decisions.

To decide if we should get this new customer, I will perform logistic regression, as the dataset has a binary response variable. i.e: Buy decision is True/False.

Initial attempt

We first try the main effects model without any interactions:

```
fit1 = glm(Buy ~ Age + Income + factor(Married) + factor(Gender), family=binomial, data=buyer)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

A warning message "glm.fit: fitted probabilities numerically 0 or 1 occurred" appeared. This indicates to me that there could be either complete separation or quasi-complete separation, but not necessarily. My intuition says there is quasi-complete separation because a hyperplane at Income = 300k can completely separate Buy decisions, where income above 300k gets a Buy decision and income below 300k does not get a Buy decision. The reason it is quasi-complete is because at Income = 300k, there are Buy and no Buy decisions, which belong to User 2 and 5 respectively. Let us run some diagnostics to confirm our suspicion.

Diagnostics

```r
summary(fit1)
```

```
##
## Call:
## glm(formula = Buy ~ Age + Income + factor(Married) + factor(Gender),
##     family = binomial, data = buyer)
##
## Deviance Residuals:
##          1           2           3           4           5           6
## -2.110e-08   1.995e-06  -2.110e-08   4.435e-06  -9.664e-06   9.664e-06
##          7           8           9          10
## -2.110e-08  -2.110e-08  -7.608e-06   2.110e-08
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.395e+02  7.411e+05       0        1
## Age                  3.408e+00  1.006e+04       0        1
## Income               5.043e-01  1.202e+03       0        1
## factor(Married)TRUE -4.505e+01  1.891e+05       0        1
## factor(Gender)Male  -4.391e+01  4.367e+05       0        1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.3460e+01  on 9  degrees of freedom
## Residual deviance: 2.6832e-10  on 5  degrees of freedom
## AIC: 10
##
## Number of Fisher Scoring iterations: 25
```

```r
paste("Log likelihood of the model is", logLik(fit1))
```

```
## [1] "Log likelihood of the model is -1.34157573938865e-10"
```

Large standard errors on the parameters indicate complete/quasi-complete separation and the negative log likelihood confirms that it is quasi-complete separation. The reason standard errors are so high is because the estimates are actually at positive/negative infinity. Because the log-likelihood approaches a limiting value as the parameter value grows unboundedly, the log-likelihood actually looks flat. And because the variance of the parameter comes from its curvature as described by the negative inverse of the matrix of second partial derivatives (which is very small since log-likelihood is flat as parameters approach infinity), the parameter variance reported is huge.

The issue with quasi-complete separation is that the dispersion matrix is unbounded meaning that estimates for parameters that are affected by the quasi-separation will not have a maximum likelihood solution and their magnitude will grow indefinitely towards positive or negative infinity until the estimation algorithm stops iterating. However, we are not interested in doing inference and therefore the maximum likelihood estimates of the parameters and their standard errors are not relevant to us. We are in fact interested in doing prediction for the new data, so quasi-complete separation will not pose a problem.

We can run some further diagnostics to show the importance of Income for a prediction model:

```r
fit2 = glm(Buy ~ Age + factor(Married) + factor(Gender), family=binomial, data=buyer)
summary(fit2)
```

```
##
## Call:
## glm(formula = Buy ~ Age + factor(Married) + factor(Gender), family = binomial,
##     data = buyer)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1101  -0.8964  -0.5876   0.8115   2.1167
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.97124    2.68834  -0.733    0.463
## Age                  0.09051    0.10620   0.852    0.394
## factor(Married)TRUE -0.27073    2.61769  -0.103    0.918
## factor(Gender)Male  -2.32859    2.86725  -0.812    0.417
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13.460  on 9  degrees of freedom
## Residual deviance: 10.785  on 6  degrees of freedom
## AIC: 18.785
##
## Number of Fisher Scoring iterations: 4
```

```
#correlation measure between predicted probabilities of the model and the actual buy decision
paste("Correlation measure for model with Income is", cor(buyer$Buy,fit1$fitted.values))
```

```
## [1] "Correlation measure for model with Income is 1"
```

```
paste("Correlation measure for model without Income is", cor(buyer$Buy,fit2$fitted.values))
```

```
## [1] "Correlation measure for model without Income is 0.55831331105255"
```

The above analysis shows a few things. Without Income as a predictor, the model does not achieve quasi-complete separation anymore. i.e: It is not able to perfectly predict response values without the Income predictor. The correlation measure analysis supports this: we note the huge drop in correlation after Income is removed.

However, Income is not the only predictor that is required to perfectly predict the response values.

```
fit3 = glm(Buy ~ Income, data=buyer, family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
paste("Correlation measure for model with only Income is", cor(buyer$Buy,fit3$fitted.values))
```

```
## [1] "Correlation measure for model with only Income is 0.849836585598797"
```

The correlation measure of 0.85 indicates the importance of Income as a predictor, but also shows that without the other predictors like Marriage, Gender, Age, our model doesn't predict as well.

Formal justification for the inclusion of Income as predictor

Even though we already know from our previous tests about the importance of Income as a predictor, we can formalise this further. If we do an ANOVA to compare the two nested models with and without income as a predictor. Then the difference of deviances is the likelihood-ratio test. Under the null hypothesis of Income having no effect, this difference has an approximate chi-squared distribution with d.f $= p_1 - p_0$, where p_1 and p_0 are the number of parameters for the nested models.

```
anova(fit2,fit1)
```

```
## Analysis of Deviance Table
##
## Model 1: Buy ~ Age + factor(Married) + factor(Gender)
## Model 2: Buy ~ Age + Income + factor(Married) + factor(Gender)
##   Resid. Df Resid. Dev Df Deviance
## 1         6     10.785
## 2         5      0.000  1   10.785
```

The likelihood ratio statistic is 10.785 with degree of freedom = 1.

Testing this with the chi-square statistic at level 0.95, we have:

```
10.785 > qchisq(0.95,1)
```

```
## [1] TRUE
```

This shows that Income is a significant predictor, and we reject the null of Income having no effect at significance level alpha = 0.05.

We can see that income contributes significantly to our model, and that it was shown that we need other predictors as well. We will use the full model with all predictors from the dataset except for the "User" variable.

Now we are ready to predict for the new data.

Prediction of new data

```
#new test data
test = data.frame(User=11, Age=40, Gender="Female",Married=(TRUE), Income=310)
predict.glm(fit1, newdata= test, type="response")
```

```
##         1
## 0.9997057
```

A prediction of probability ~ 1 indicates a Buy decision for the new test data, as expected, due to the quasi-complete separation achieved at Income = 300k. We should get this new customer.

Q3: Based on this data, how would you predict the sales amount of 500th day and 1000th day?

Cleaning the data

```
library(readr)
sale_amount = read_csv("sale_amount.csv")
```

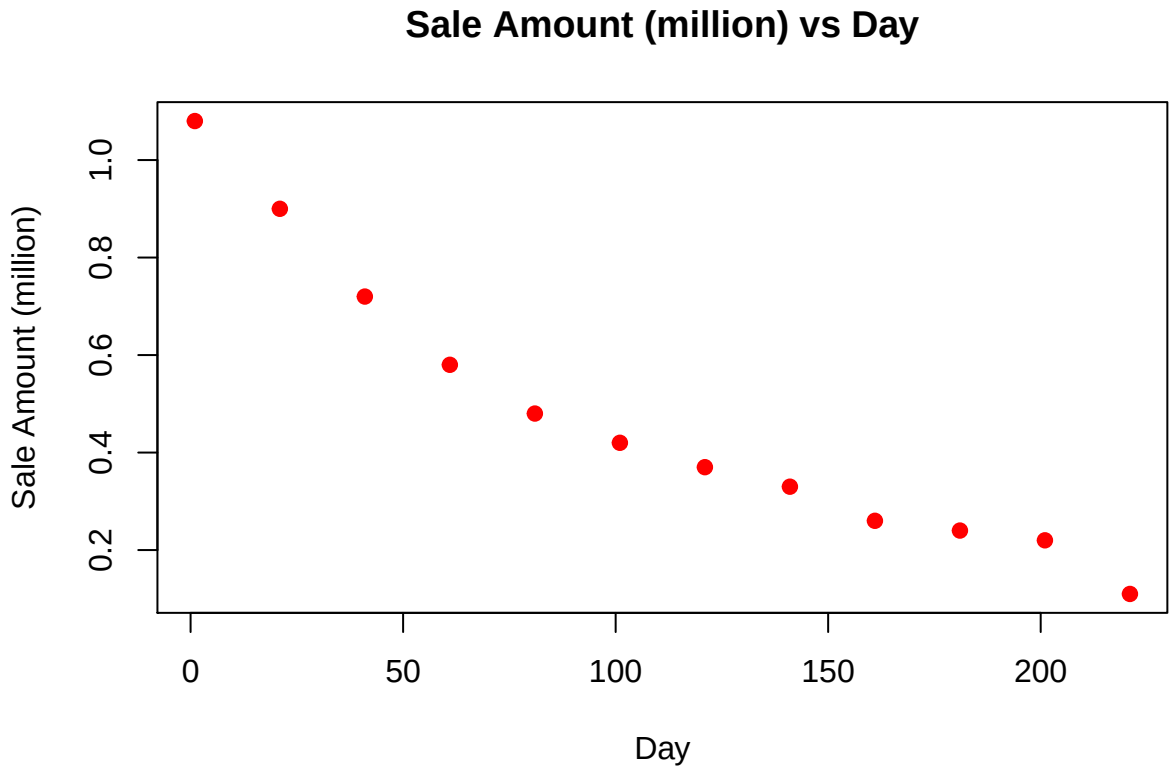
```
## Parsed with column specification:
## cols(
##   Day = col_double(),
##   `Sale amount (million)` = col_double()
## )
```

```
names(sale_amount)[names(sale_amount) == "Sale amount (million)"] <- "Amount"
```

Initial plot

First, we get an initial plot of the data to better visualise the trend.

```
plot(sale_amount$Day, sale_amount$Amount, main="Sale Amount (million) vs Day", ylab="Sale Amount (millio
```



Preliminary analysis

Deceivingly, a linear model looks like it might fit. However, we start to run into problems when we try to predict for the 500th and 1000th day as a linear model will probably predict negative values for those days. A quick check will show this.

```
fit1 = lm(Amount ~ Day, data=sale_amount)
paste("Linear model predictions for day 500 is", predict(fit1, data.frame(Day=c(500,1000)))[1])
```

```
## [1] "Linear model predictions for day 500 is -1.03732226107226"
```

```
paste("Linear model predictions for day 1000 is", predict(fit1, data.frame(Day=c(500,1000)))[2])
```

```
## [1] "Linear model predictions for day 1000 is -2.98225233100233"
```

Upon closer inspection, we can actually notice that there is an exponential decay in sales amount with sales amount tapering off at around day 200. A simple model that we can use is the log-normal model where we assume the model is:

$$Amount = e^{\beta_0 + \beta_1 Day + \epsilon}$$

Then log-transform the response variable so that the response is linear with the parameters, which will allow us to fit a linear regression.

$$\log(Amount) = \beta_0 + \beta_1 Day + \epsilon$$

However, there is one disadvantage for doing so. Since:

$$\mathbb{E}[\log(Amount)] = \beta_0 + \beta_1 Day$$

The estimates of the log-transformed model does not translate to exact information about the expected sales amount $\mathbb{E}[Amount]$ or the effect of the variable day on $\mathbb{E}[Amount]$, which will result in inaccurate predictions for the sales amount of 500th and 1000th day.

A better alternative would be the following:

$$\log(\mathbb{E}[Amount]) = \beta_0 + \beta_1 Day$$

as we can just exponentiate on both sides to get information about $\mathbb{E}[Amount]$.

A generalised poisson linear model with log-link would achieve this. In this setting, we assume Amount to be independently Poisson distributed with a fixed unknown mean λ. That is: $Amount \sim Poi(\lambda)$ and $Amount_i \perp\!\!\!\perp Amount_j$. Some quick sanity checks can be done to make sure the Poisson specification makes sense.

1) Are the responses non-negative integers with no upper bound.

Not quite, since we have decimal places in Amount. But this can be easily fixed by multiplying by a constant. For example we can turn datapoint 1 with 1.08 to 1080 by multiplying by 1000. Nothing fundamental has been changed, only the units. Therefore, it will have no influence on our analysis. We also assume that there will not be a negative sale amount as the company will not sell at a negative price.

2) Is the distribution of y consistent with the Poisson distribution?

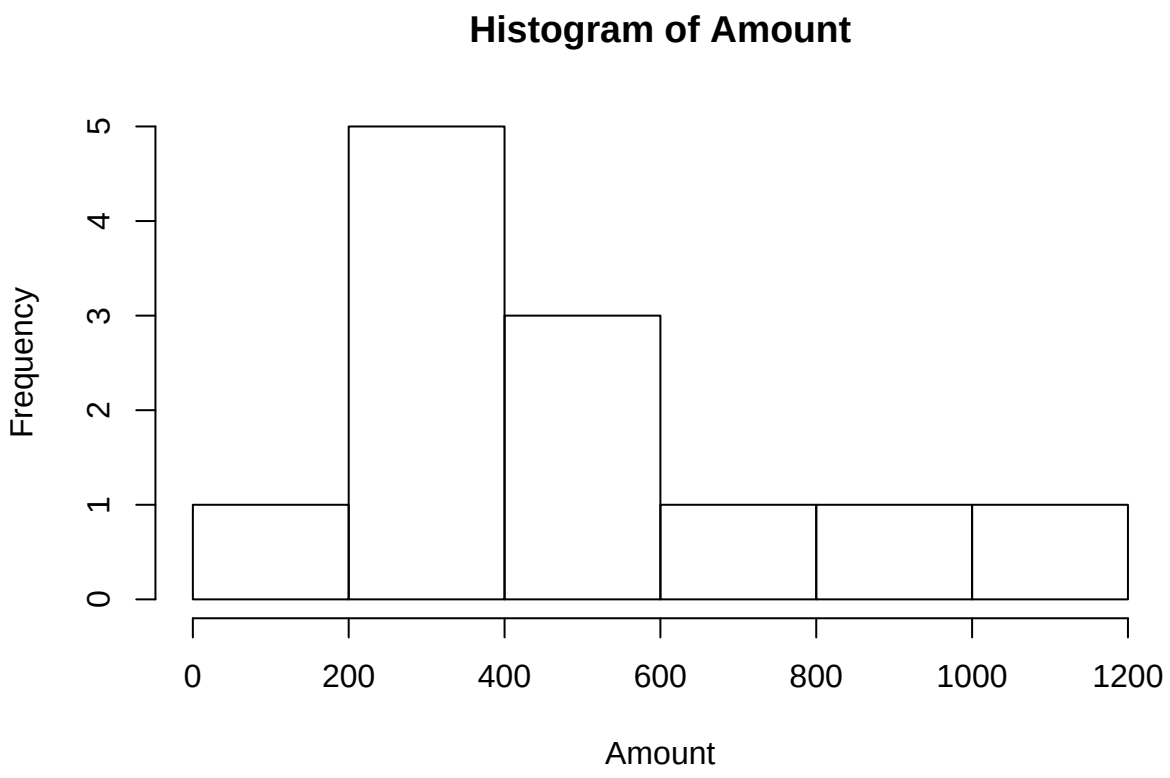
Let us run some diagnostics.

Diagnostics for checking Poisonness

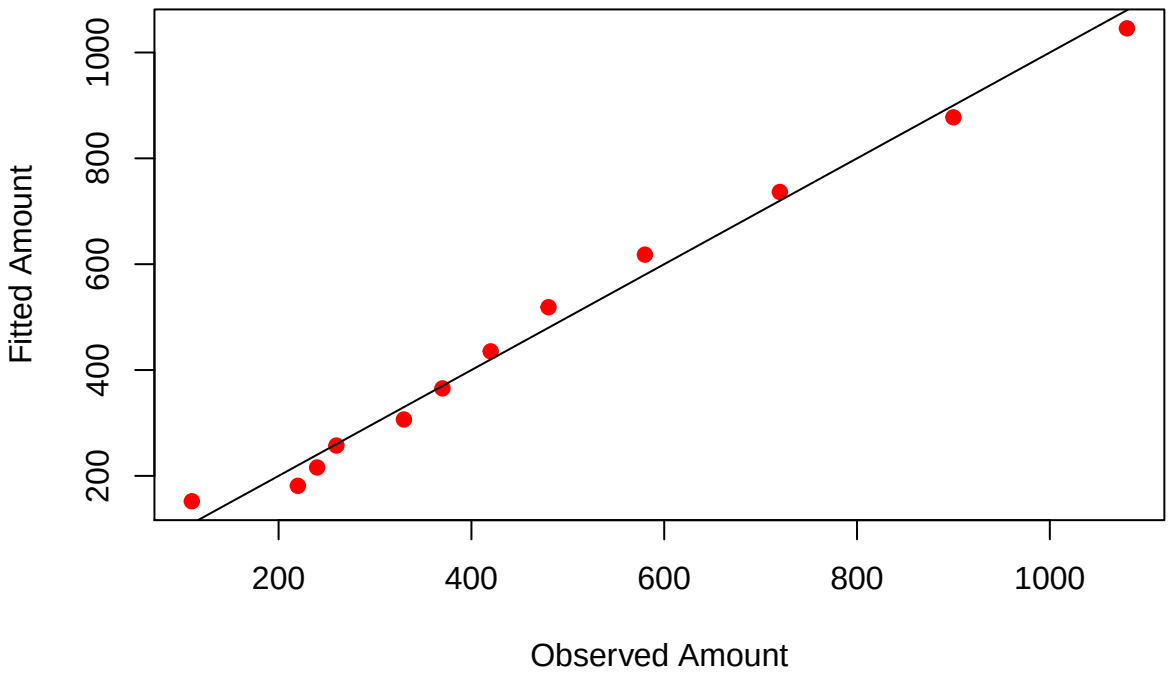
```
#changing Amount to integer
sale_amount = read_csv("sale_amount.csv")
```

```
## Parsed with column specification:
## cols(
##   Day = col_double(),
##   `Sale amount (million)` = col_double()
## )
```

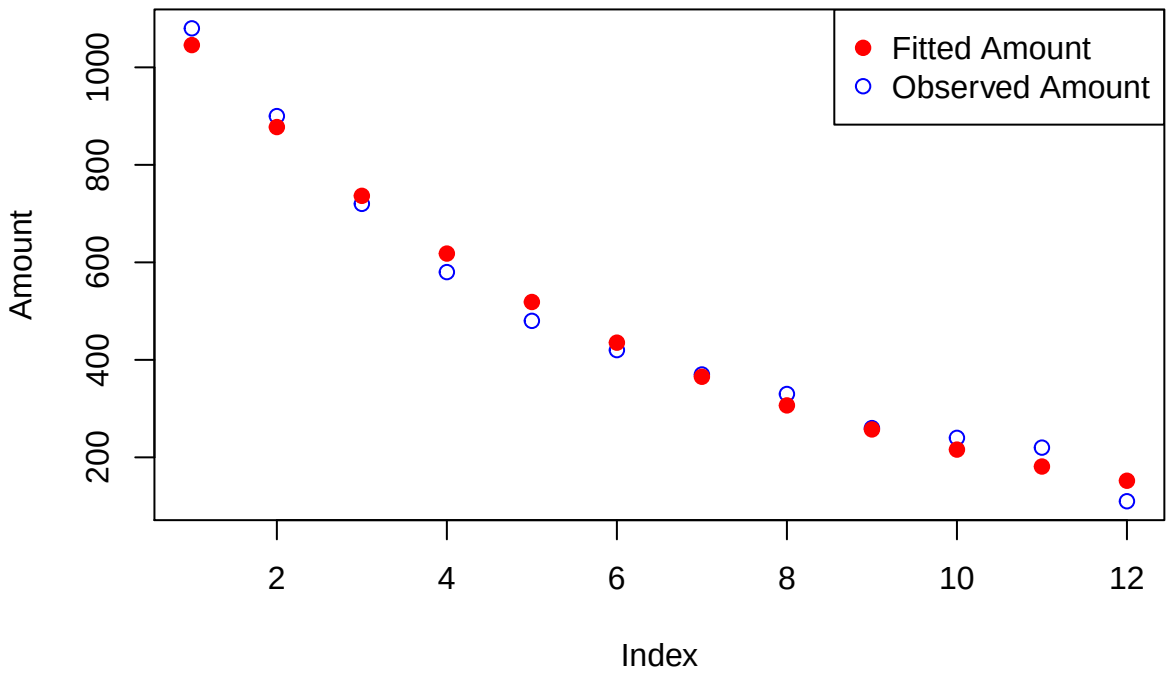
```
names(sale_amount)[names(sale_amount) == "Sale amount (million)"] <- "Amount"
sale_amount["Amount"] = sale_amount["Amount"] * 1000

#histogram of Amount
hist(sale_amount$Amount, main="Histogram of Amount", xlab="Amount")
```




```
#fitting a Poisson GLM with log link
fit2 = glm(Amount ~ Day, family=poisson, data=sale_amount) #default link function is log

#comparing fitted amount with observed amount
plot(sale_amount$Amount, fit2$fitted.values, xlab="Observed Amount", ylab="Fitted Amount", pch=19, col=
abline(0,1)
```




```
plot(sale_amount$Amount, ylab="Amount",col="blue",pch=1)
points(fit2$fitted.values, pch=19, col="red")
legend("topright", c("Fitted Amount", "Observed Amount"), pch=c(19,1), col=c("red","blue"))
```



The above diagnostics have shown that Amount does have a Poisson shaped histogram that is unimodal and that the fitted Amount matches quite well with the observed Amount.

Model Improvement

Despite the adequacy of the Poisson GLM model, we can do even better. One drawback that the Poisson GLM suffers from is overdispersion caused by heterogeneity that results from lack of covariates. Since we only have one covariate (Day), it is likely that there is some variation in the Amount that we have not accounted for. A mixture model is a flexible way to acccount for overdispersion.

Suppose the distribution $Amount = y \sim Poi(\lambda)$, but λ varies because of unmeasured covariates. Let $\mu = \mathbb{E}[\lambda]$. Then unconditionally, by tower law and $\mathbb{E}[y|\lambda] = \lambda$

$$\mathbb{E}[y] = \mathbb{E}[\mathbb{E}[y|\lambda]] = \mathbb{E}[\lambda] = \mu$$

.

and by law of total variance and $Var(y|\lambda) = \lambda$,

$$Var(y) = \mathbb{E}[Var(y|\lambda)] + Var(\mathbb{E}[y|\lambda]) = \mathbb{E}(\lambda) + Var(\lambda) = \mu + Var(\lambda) > \mu$$

What I have just shown is that by varying λ, we can increase the variation that the model can account for. If we let λ to have the gamma distribution, the gamma mixture of Poisson distributions yield the negative binomial distribution for y.

Let us fit the negative binomial model and compare it with the Poisson GLM:

Fitting the negative binomial model

```
library(MASS)
#fitting the negative binomial model
fit3 = glm.nb(Amount ~ Day, data=sale_amount)

paste("AIC for Poisson GLM is", fit2$aic)
```

```
## [1] "AIC for Poisson GLM is 130.889886451645"
```

```
paste("AIC for negative binomial model is", fit3$aic)
```

```
## [1] "AIC for negative binomial model is 128.939571730282"
```

```
paste("Log likelihood for Poisson GLM is", logLik(fit2))
```

```
## [1] "Log likelihood for Poisson GLM is -63.4449432258225"
```

```
paste("Log likelihood  for negative binomial model is", logLik(fit3))
```

```
## [1] "Log likelihood  for negative binomial model is -61.4697858651411"
```

As expected, the negative binomial performs better than the Poisson GLM, since the Poisson GLM is merely a special case of the negative binomial model.

We are now ready to predict the Amount for Day 500 and 1000.

Predictions for Day 500 and 1000

```
paste("Predictions for day 500 is", predict(fit3, data.frame(Day=500), type="response")*1000, "dollars.
```

```
## [1] "Predictions for day 500 is 13570.8083458818 dollars."
```

```
paste("Predictions for day 1000 is", predict(fit3, data.frame(Day=1000), type="response")*1000, "dollars
```

```
## [1] "Predictions for day 1000 is 175.611556850551 dollars."
```

We note that the predictions are extrapolations far into the future and the chance of error is high.