code_laong

2025-10-08

Categorical Inputs Practice

```r
legos <- read.csv('https://vinnys-classes.github.io/data/legos_data.csv')
head(legos)
```

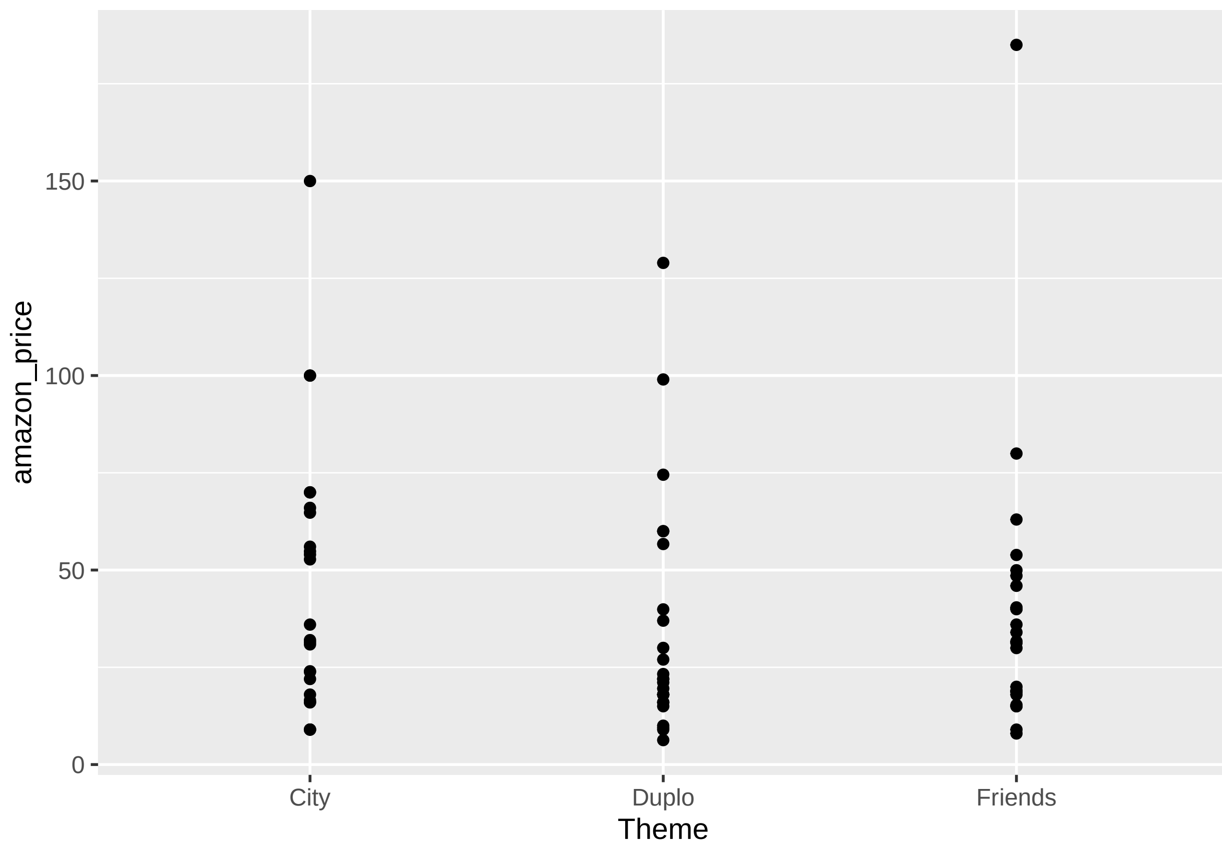
```
##   Item_Number                   Set_Name Theme Pieces Year Pages Minifigures
## 1       10859         My First Ladybird Duplo      6 2018     9          NA
## 2       10860         My First Race Car Duplo      6 2018     9          NA
## 3       10862      My First Celebration Duplo     41 2018     9          NA
## 4       10864 Large Playground Brick Box Duplo     71 2018    32           2
## 5       10867            Farmers' Market Duplo     26 2018     9           3
## 6       10870               Farm Animals Duplo     16 2018     8          NA
##     Packaging Unique_Pieces  Size amazon_price age
## 1         Box             5 Large        16.00   1
## 2         Box             6 Large         9.45   1
## 3         Box            18 Large        39.89   1
## 4 Plastic box            49 Large        56.69   2
## 5         Box            18 Large        36.99   2
## 6         Box            13 Large         9.99   2
```

#Let's remake the plot from the slide deck

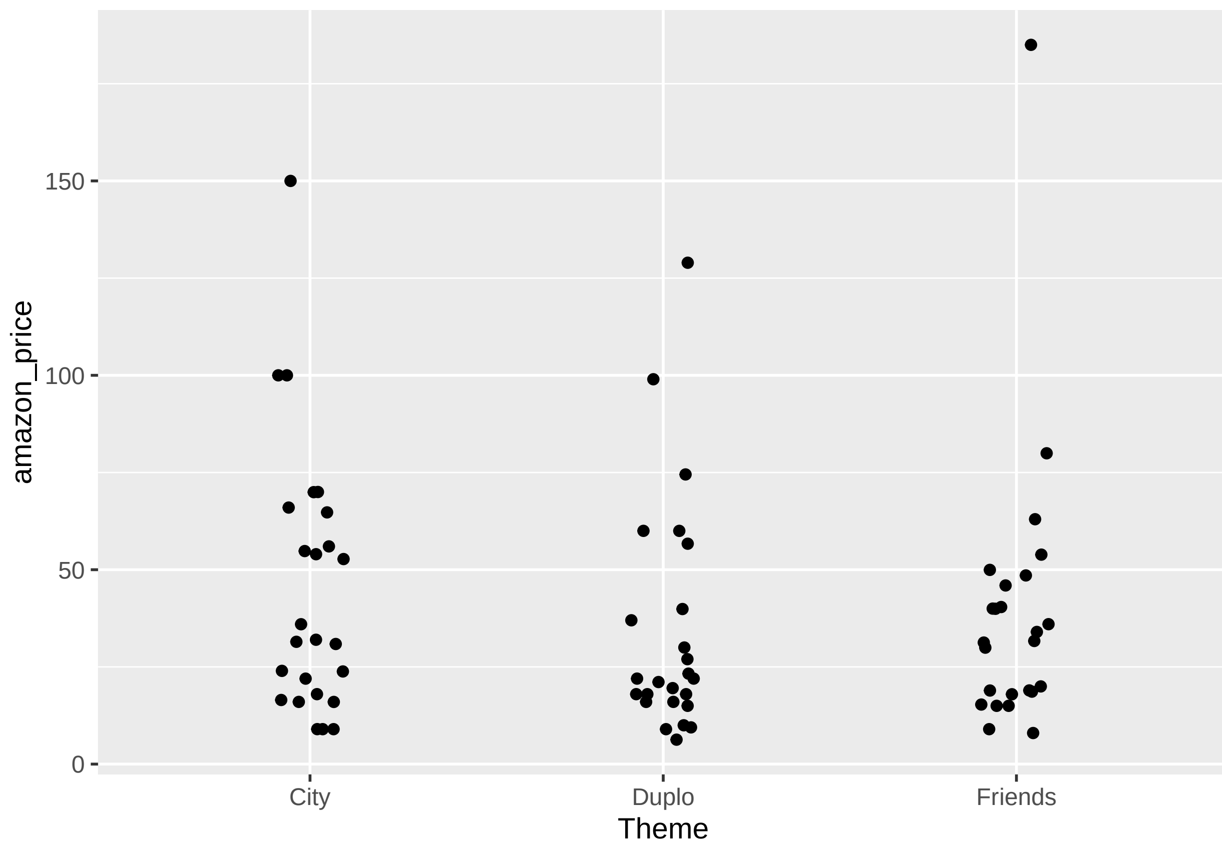
Always good to get some extra ggplot2 practice in. This also showcases the times we want geom_jitter() over geom_point()

```r
library(ggplot2)

#Without geom_jitter the points seem to lie on
#top of each other which is hard to read
ggplot(data = legos,
       aes(x = Theme,
           y = amazon_price)) +
    geom_point()
```

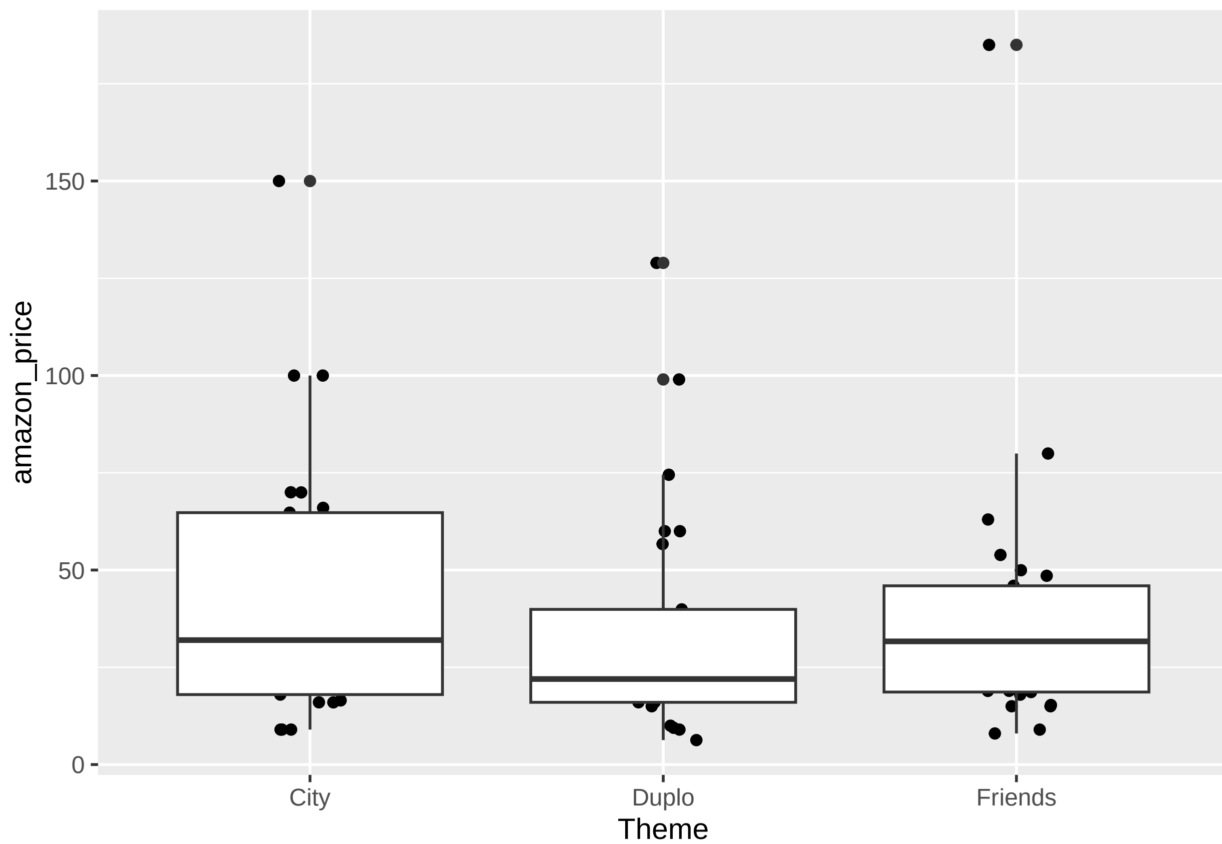



```
#One in the notes
ggplot(data = legos,
       aes(x = Theme,
           y = amazon_price)) +
    geom_jitter(width = .1)  #width let's us define how much
```

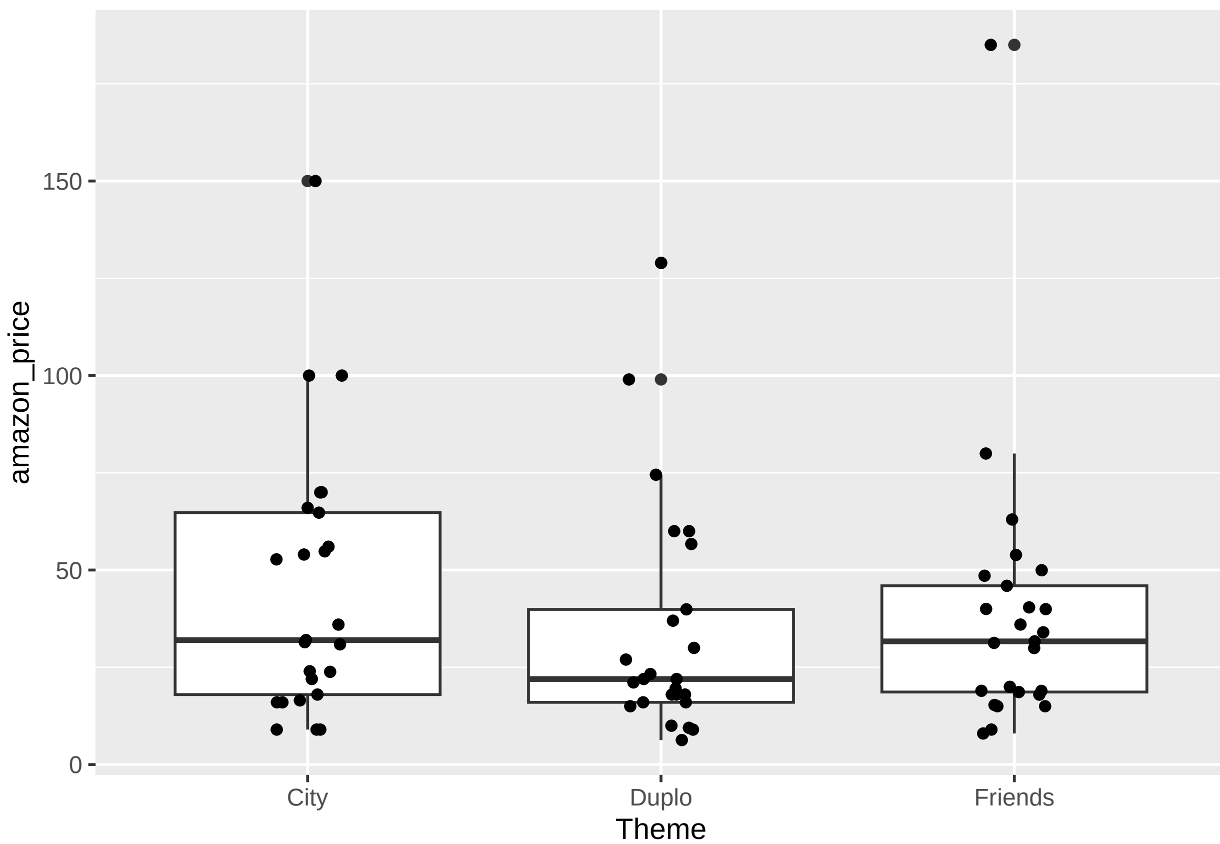



```
                                  #white noise we want to add in the x-direction
                                  #Try running this with width = 2 for example

#we can slap on a boxplot
ggplot(data = legos,
       aes(x = Theme,
           y = amazon_price)) +
    geom_jitter(width = .1)  +
  geom_boxplot()
```




```r
#or slap the points on top of the boxplot
ggplot(data = legos,
       aes(x = Theme,
           y = amazon_price)) +
  geom_boxplot() +
    geom_jitter(width = .1)
```



Modeling Amazon Price given Theme

The code to run the function is actually pretty straight forward but interpreting it and write out the equations seems to be a sticking point for some.

```
#the three means we are shooting for
#by using the aggregate function. I want you
#to use the lm() function but aggregate() is
#a nice sanity check to make sure things are
#working how you think
aggregate(amazon_price ~ Theme,
          data = legos,
          FUN = mean)
```

```
##     Theme amazon_price
## 1    City      45.2696
## 2   Duplo      34.2624
## 3 Friends      38.6492
```

```
#Let's run the actual linear model
lego_mod <- lm(amazon_price ~ Theme,
               data = legos)
```

```r
#let's look at the summary of our model
summary(lego_mod)
```

```
##
## Call:
## lm(formula = amazon_price ~ Theme, data = legos)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -36.28 -20.98 -10.99  10.31 146.34
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    45.270      6.689   6.768 2.95e-09 ***
## ThemeDuplo    -11.007      9.459  -1.164    0.248
## ThemeFriends   -6.620      9.459  -0.700    0.486
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 33.44 on 72 degrees of freedom
## Multiple R-squared:  0.01871,	Adjusted R-squared:  -0.00855
## F-statistic: 0.6863 on 2 and 72 DF,  p-value: 0.5067
```

There are a few things we can notice but the point of interest for us at this stage is the ''Coefficents'' table which gives the estimates of our beta's. Those estimates are the first column under the heading Estimate.

Estimated Equation: predicted amazon price = $\hat{\beta}_0 + \hat{\beta}_1 * 1_{DUPLO} + \hat{\beta}_2 * 1_{Friends}$

Estimated Equation: predicted amazon price = 45.27 - 11.007 * 1_{DUPLO} -6.62 * $1_{Friends}$

45.27 is the mean amazon price of our reference group (City Lego Sets). Alternatively, -11.007 is the difference in the mean cost of DUPLO vs the mean cost of our reference category, City. Finally, -6.62 is the difference in the mean cost of Friends vs the mean cost of our reference category, City.

A healthy way to make sure those align with your thinking is to compare those with your mean estimates from the aggregate() function.

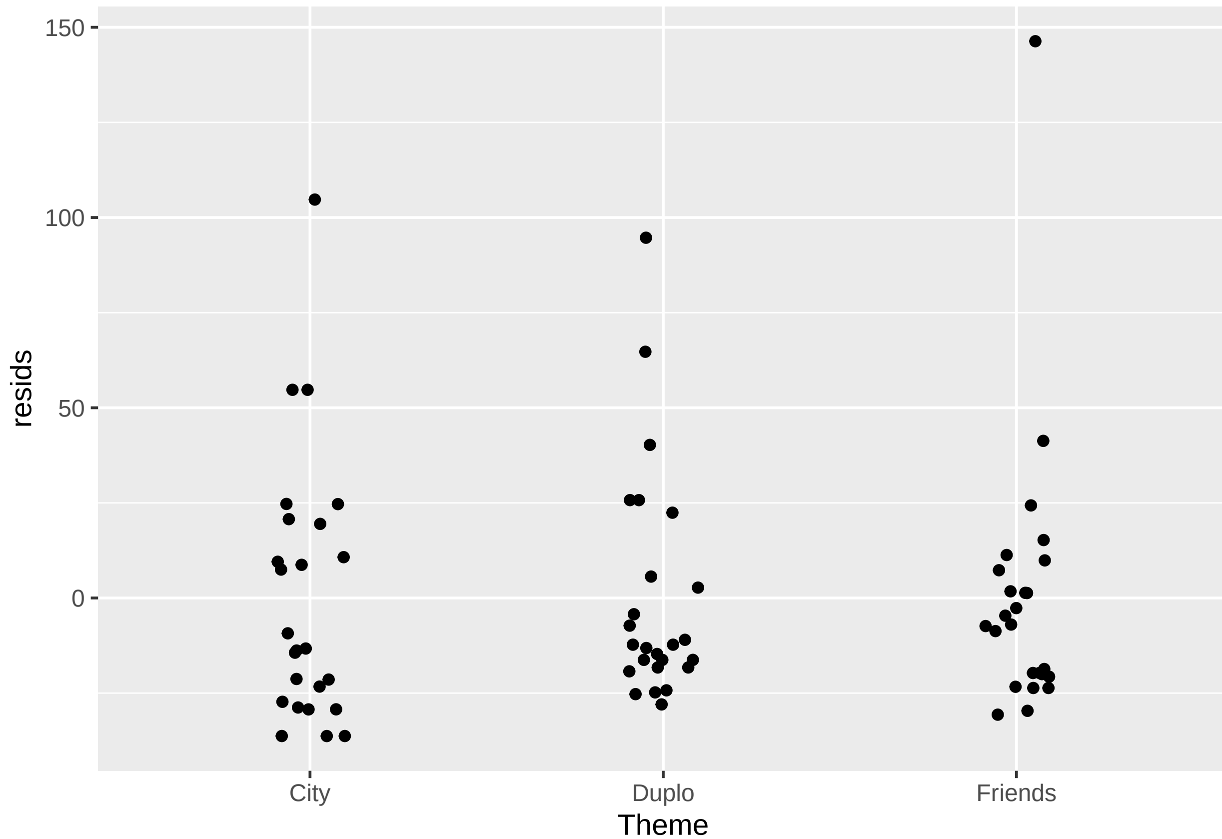
Check Model Assumptions

Still need independence, homoskedasticity, and normality

```r
#save our residuals from the lego model
my_resids <- resid(lego_mod)
legos$resids <- my_resids


#For this residaul graph, the categories are our x-axis and
#the y-axis is the residuals. What we want is an equal spread
#of points above and below the line that's roughly
#symmetric (normality) and each group to have approximately the
#same spread (homoskedasticity). Indpenedence is something
#we have to just think through
ggplot(legos,
```

```
      aes(x = Theme,
          y = resids)) +
  geom_jitter(width = .1)
```



I don't like the residuals as the y-axis is too elongated above 0. In otherwords, it's not symmetric around 0. More so, there seems to be some pretty extreme outliers in the positive y-direction. This residual graph indicates that the log() transform might be useful sooooo let's try it.

Log-Linear Model!

This is the log-linear model in that the response is being logged while the x-axis is being left along (don't take a log of a category as things will go weird).

To do this we can make a new variable that is the log of amazon price or we can just use log() directly inside the lm() function.

```
#the three means we are shooting for albeit they
#are a bit meaningless to me
aggregate(log(amazon_price) ~ Theme,
          data = legos,
          FUN = mean)
```

```
##      Theme log(amazon_price)
## 1     City          3.530074
```

```
## 2   Duplo          3.233602
## 3 Friends          3.397789
```

```
lego_mod_log <- lm(log(amazon_price) ~ Theme,
               data = legos)



summary(lego_mod_log)
```

```
##
## Call:
## lm(formula = log(amazon_price) ~ Theme, data = legos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.39464 -0.46675 -0.06465  0.48984  1.82251
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.5301     0.1513  23.328   <2e-16 ***
## ThemeDuplo     -0.2965     0.2140  -1.385    0.170
## ThemeFriends   -0.1323     0.2140  -0.618    0.538
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7566 on 72 degrees of freedom
## Multiple R-squared:  0.02606,    Adjusted R-squared:  -0.0009918
## F-statistic: 0.9633 on 2 and 72 DF,  p-value: 0.3865
```

```
########
```

So let's fill out the equation for this one. . . .

Estimated Equation: predicted log amazon price $= \hat{\beta_0} + \hat{\beta_1} * 1_{DUPLO} + \hat{\beta_2} * 1_{Friends}$

Estimated Equation: predicted log amazon price = 3.5301 -.2965 1_{DUPLO} - .1323 * $1_{Friends}$

Estimated Equation: predicted amazon price = exp(3.5301 -.2965 1_{DUPLO} - .1323 * $1_{Friends}$)

^ the last equation is overkill but thought I'd show you the backtransform

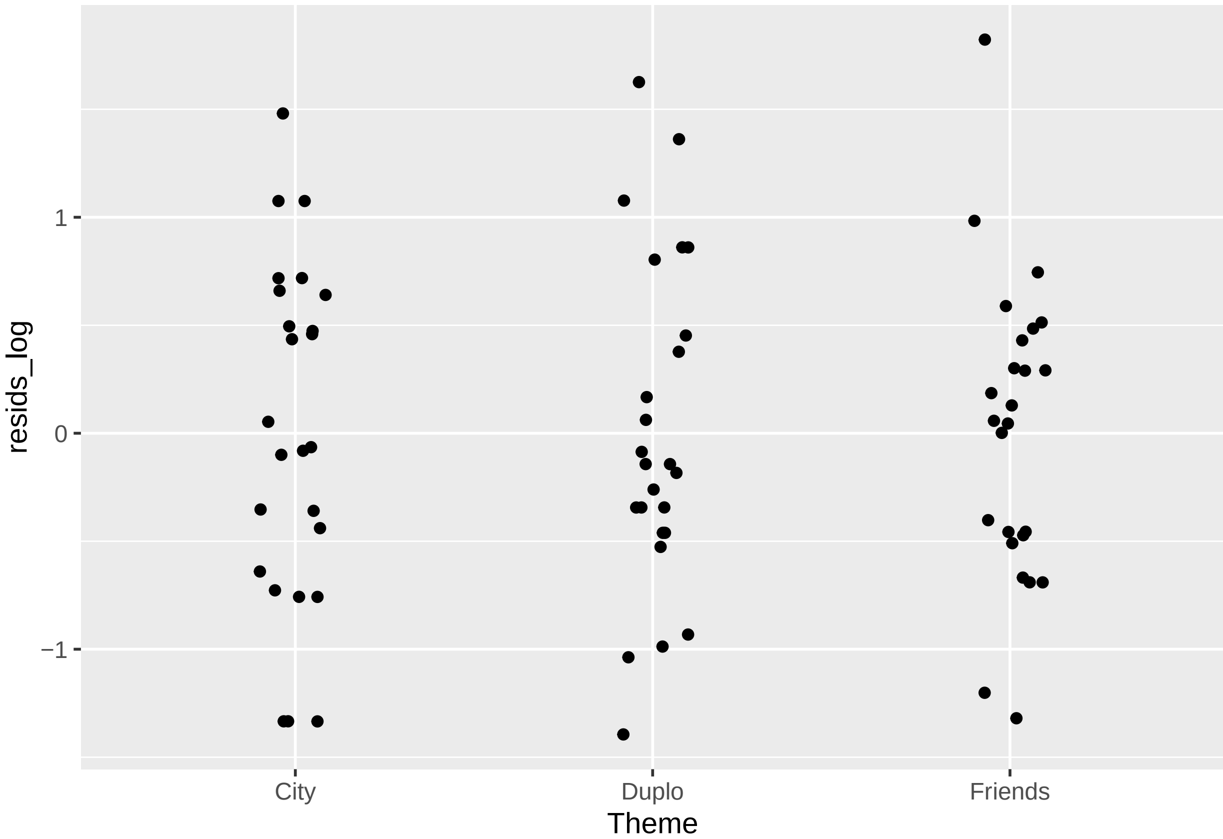
And let's check the assumptions for this one

```
#save the residuals of the log-linear model
my_resids_log <- resid(lego_mod_log)
legos$resids_log <- my_resids_log



#very similar graph as earlier but now the y-axis
#is the residuals for our log-linear model
```

```r
ggplot(legos,
       aes(x = Theme,
           y = resids_log)) +
  geom_jitter(width = .1)
```



Find outlier for Friends

There seems to be an annoying outlier for the Friends category so let's dig into it. First, we use subset() to subset our data so that we have just the friends category of legos.

```r
my_subset <- subset(legos, #data set
                    Theme == 'Friends') #the variable == 'category of interest'

#the below finds the first value for which the input
#(here the amazon price of our lego-friends data set) is
#at it's maximum value. So the fourth row
which.max(my_subset$amazon_price)
```

```
## [1] 4
```

```r
#and we call the fourth row of our lego-friends data set
my_subset[4,]
```

```
##    Item_Number         Set_Name   Theme Pieces Year Pages Minifigures Packaging
## 29       41340 Friendship House Friends    722 2018   164           3       Box
##    Unique_Pieces  Size amazon_price age   resids resids_log
## 29           309 Small       184.99   6 146.3408   1.822513
```