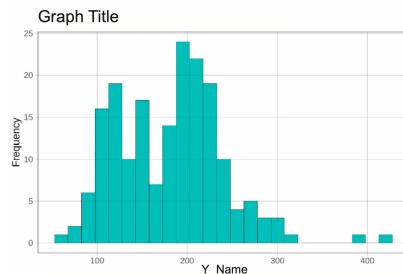


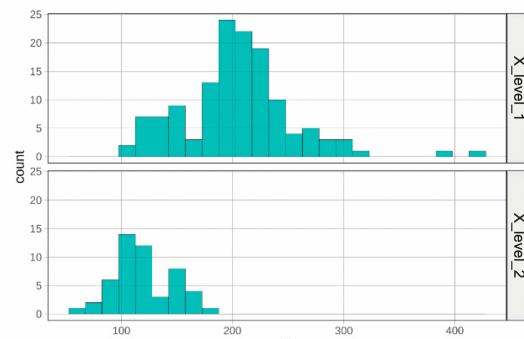
Word Equations	Summary Tables	Simulation	
<pre>outcome = explanatory + other stuff Y = X + other stuff</pre>	<pre># compute five-number summary favstats(~ Y, data = data_set)  # create frequency table tally(data_set\$Y) tally(~ Y, data = data_set)  # tally by condition tally(~ Y &lt; 1900, data = data_set)  # two-way frequency table tally(Y ~ X, data = data_set)</pre>	<pre># sample without replacement sample(data_set, 6) # sample with replacement resample(data_set, 10)  # randomize sampling distribution # of b1s, centered on 0 sdob1 &lt;- do(1000) *   b1(shuffle(Y) ~ X, data = data_set)  # bootstrap sampling distribution # of b1s, centered on sample b1_sdob1_boot &lt;- do(1000) *   b1(Y ~ X, data = resample(data_set))</pre>	<pre># return TRUE for # middle 95% of distribution middle(sdob1\$b1, .95)  # randomize sampling distribution # of PREs sdoPRE &lt;- do(1000) *   PRE(shuffle(Y) ~ X,   data = data_set)  # randomize sampling distribution # of Fs sdoF &lt;- do(1000) *   fVal(shuffle(Y) ~ X, data = data_set)</pre>
<h3>Basics</h3> <pre>print("Hello world!")  # assign value to object myNumber &lt;- 5 # combine values into vector myVector &lt;- c(1, 2, 3) # first element in vector myVector[1]  # arithmetic operations sum(1, 2, 100), +, -, *, / # logical operations &gt;, &lt;, &gt;=, &lt;=, ==, !=,  , &amp;</pre>			
<h3>Data Frame</h3> <pre># view first/last six rows head(data_set) tail(data_set)  # structure of data frame str(data_set) glimpse(data_set)  # select variable (a column) data_set\$Y</pre>	<pre># select multiple variables select(data_set, Y1, Y2)  # select first row data_set[1, ]  # find rows that meet condition data_set[data_set\$Y &gt; 40] filter(data_set, Y &gt; 300)</pre>	<pre># arrange rows by variable arrange(data_set, Y)  # sort in a descending order arrange(data_set, desc(Y))  # get rid of all cases with any # missing values na.omit(data_set)</pre>	<pre># convert quantitative variable # to categorical factor(data_set\$Y)  # convert categorical variable # to quantitative as.numeric(data_set\$Y)</pre>
<h3>Fitting Models to Data</h3> <pre># empty model empty_model &lt;- lm(Y ~ NULL, data = data_set)  # use one explanatory variable one_model &lt;-   lm(Y ~ X, data = data_set)  # extract the best fitting b1 b1(shuffle(Y) ~ X, data = data_set)  # multivariate model multi_model &lt;-   lm(Y ~ X1 + X2, data = data_set)  # model predictions and residuals data_set\$empty_predict &lt;- predict(empty_model) data_set\$empty_resid &lt;- resid(empty_model)</pre>	<h3>Comparing Models</h3> <pre>pre(Y ~ X, data = data_set) f(Y ~ X, data = data_set)  # sample F for X2 f(Y ~ X1 + X2,   data = data_set, predictor = ~X2)  # all the model comparisons that can be # made in relation to the multivariate model generate_models(multi_model)</pre>		<h3>Evaluating Models of DGP</h3> <pre># produce ANOVA table supernova(empty_model) supernova(multi_model)  # t-test, using pooled variance t.test(Y ~ X, data =   data_set, var.equal=TRUE)  # confidence interval confint(lm(Y ~ X,   data = data_set))  # pairwise comparison # corrections: "Bonferroni" or "none" pairwise(one_model, correction = "none")</pre>

## Visualizations

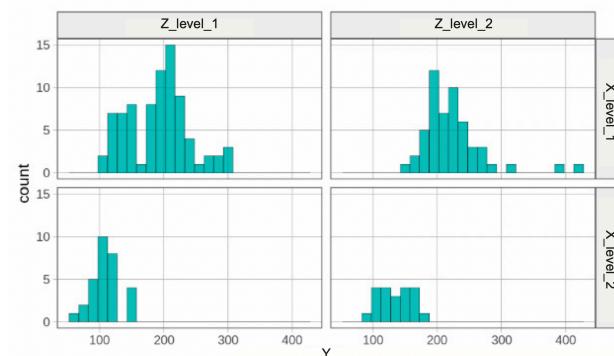
```
gf_histogram(~ Y, data = data_set) %>%
# change labels
  gf_labs(title = "Graph Title", x = "Y_Name",
  y = "Frequency")
```



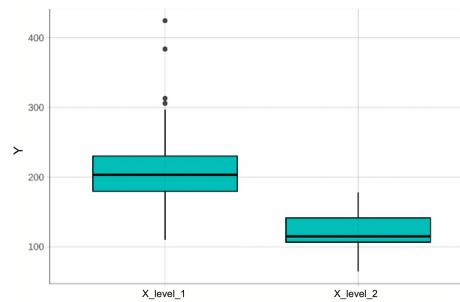
```
# faceted grid of histograms
gf_histogram(~ Y, data = data_set) %>%
  gf_facet_grid(X ~ .)
```



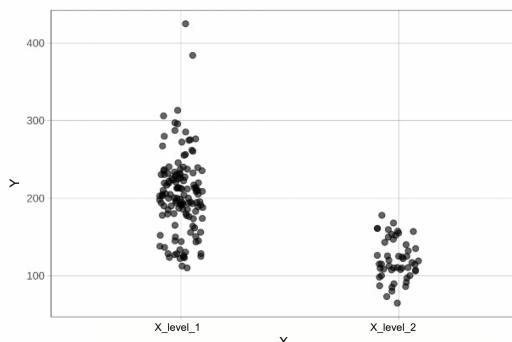
```
gf_histogram(~ Y, data = data_set) %>%
  gf_facet_grid(X ~ Z)
```



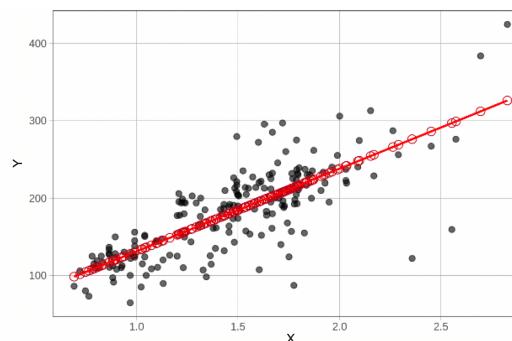
```
gf_boxplot(Y ~ X, data = data_set)
```



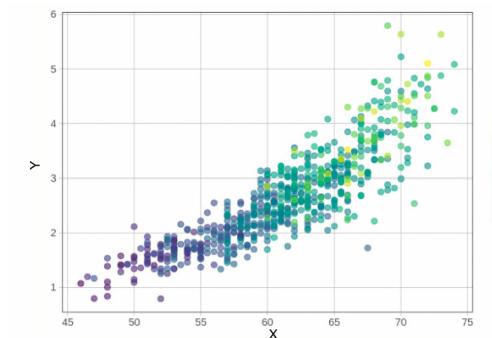
```
gf_jitter(Y ~ X, data = data_set)
```



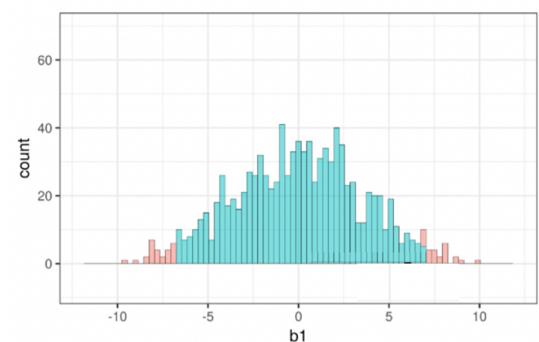
```
gf_point(Y ~ X, data = data_set) %>%
# add model predictions as red points
  gf_point(Y ~ X, shape = 1, size = 3,
  color = "firebrick") %>%
# add best fitting model as a red line
  gf_model(one_model, color = "red")
```



```
gf_point(Y ~ X, color = ~Z,
  data = data_set)
```



```
# sampling distribution of b1
gf_histogram(~b1, data = sdob1,
  fill = ~middle(b1, .95)) %>%
# modify the limits on x- and y-axes
  gf_lims(x = c(-12, 12), y = c(0, 70))
```



```
# F-distribution depicting p-value
sample_F <- fVal(Y ~ X, data = data_set)
xpf(sample_F, df1 = 1, df2 = 42)
```

