
Data Exploration

Understand the data

- Make sure the data is appropriate and you have a feel for it **before** starting your research project.
  - This will help with “sanity checks” and intuition.
- Be careful though: exploration may suggest hypotheses, but those hypotheses might be best tested with a separate data set.

Role of variable

- Predictor or target?

Data type

- Strings? Dates? Numbers?

Categorical or Continuous?

- Healthy vs sick? Systolic 110?

3 dimensions of variable classification

```
[61] diabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows x 9 columns

Look at the data

Looking at the data helps identify variables, their types, and any potential issues.

Does the data have the potential to answer your question?

- e.g. are all the relevant cases present in high enough amounts?
  - Even simpler: how many data points do you have?

- are variables like age, gender, etc that were supposed to be controlled for still balanced after accounting for dropouts, nonresponse, etc

Is the data structured correctly?

- Are all the variables of the right data type?
  - Does any conversion need to happen?
- Are the ranges valid?
  - e.g. two digit birth years instead of four, or often mixed together
- Are all the values present?
  - e.g. people with unusually low or high income may be less likely to share an amount

Tell pandas about types as needed

```
diabetes['Outcome'] = (
  diabetes['Outcome'].astype('category'))
```

```
patients['dob'] = (
  pd.to_datetime(patients['dob']))
```

September 2019

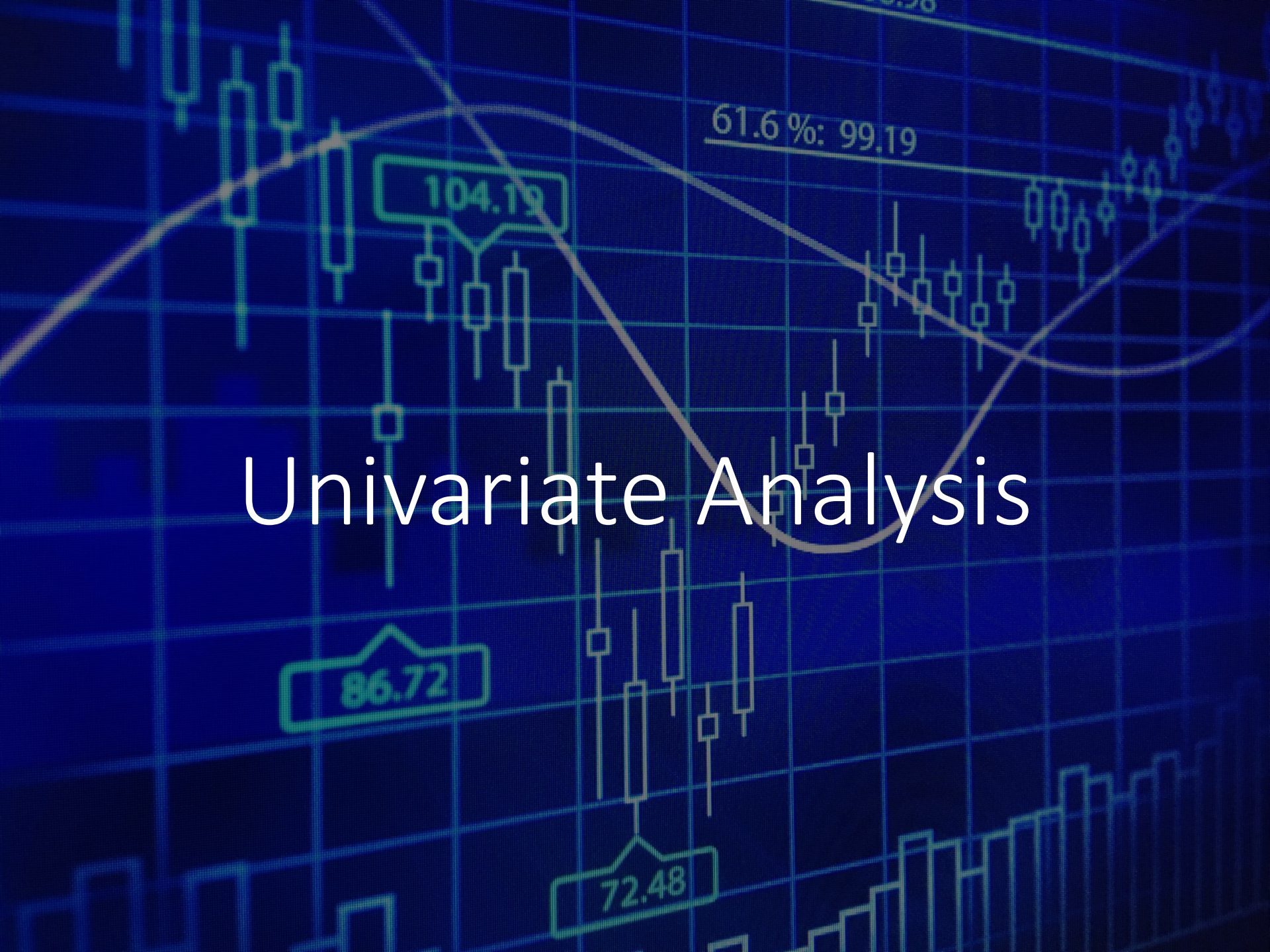
Su	M	Tu	W	Th	F	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

Sometimes the data you may want might not be explicitly present in your dataset but might be derivable from your dataset.

20190911	W
20190919	Th
20190923	M
20190926	Th

Feature engineering

```
pd.to_datetime('June 10, 2020').weekday()
```



Univariate Analysis

Basic statistics

Mean

- `series.mean()`

Median

- `series.median()`

Mode

- `series.mode()`

Standard Deviation

- `series.std()`

Min

- `series.min()`

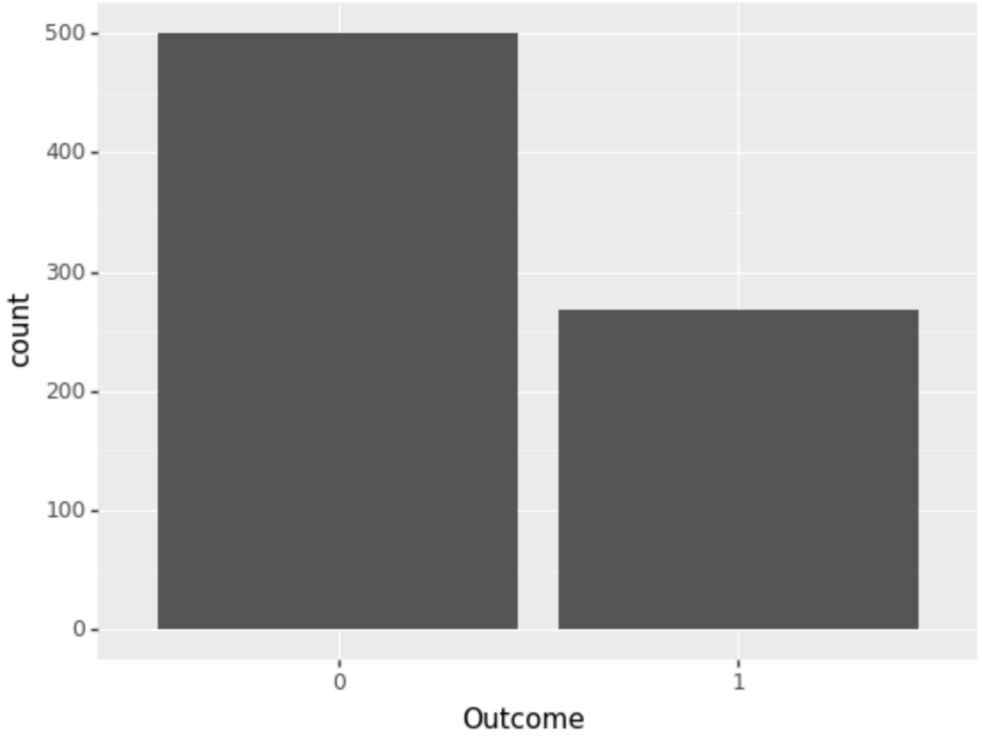
Max

- `series.max()`

Compare counts of categorical variables

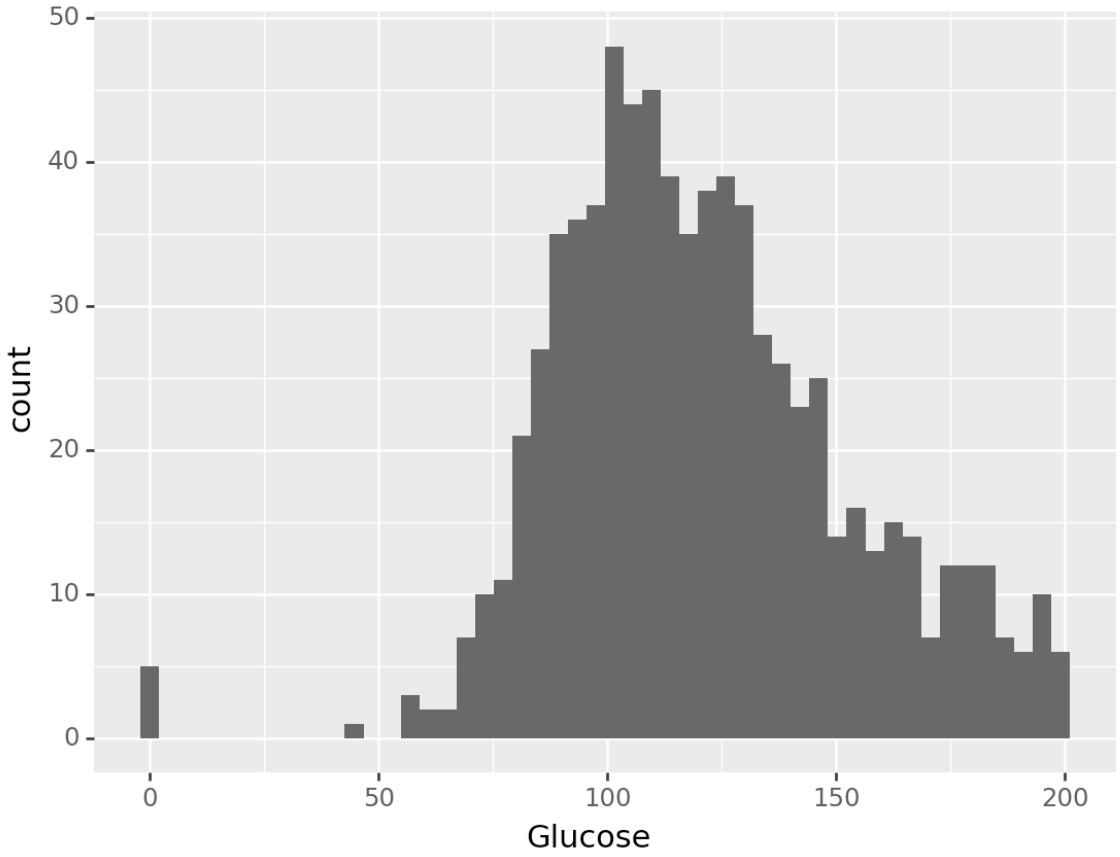
```
diabetes['Outcome'].value_counts()
```

Compare counts of categorical variables



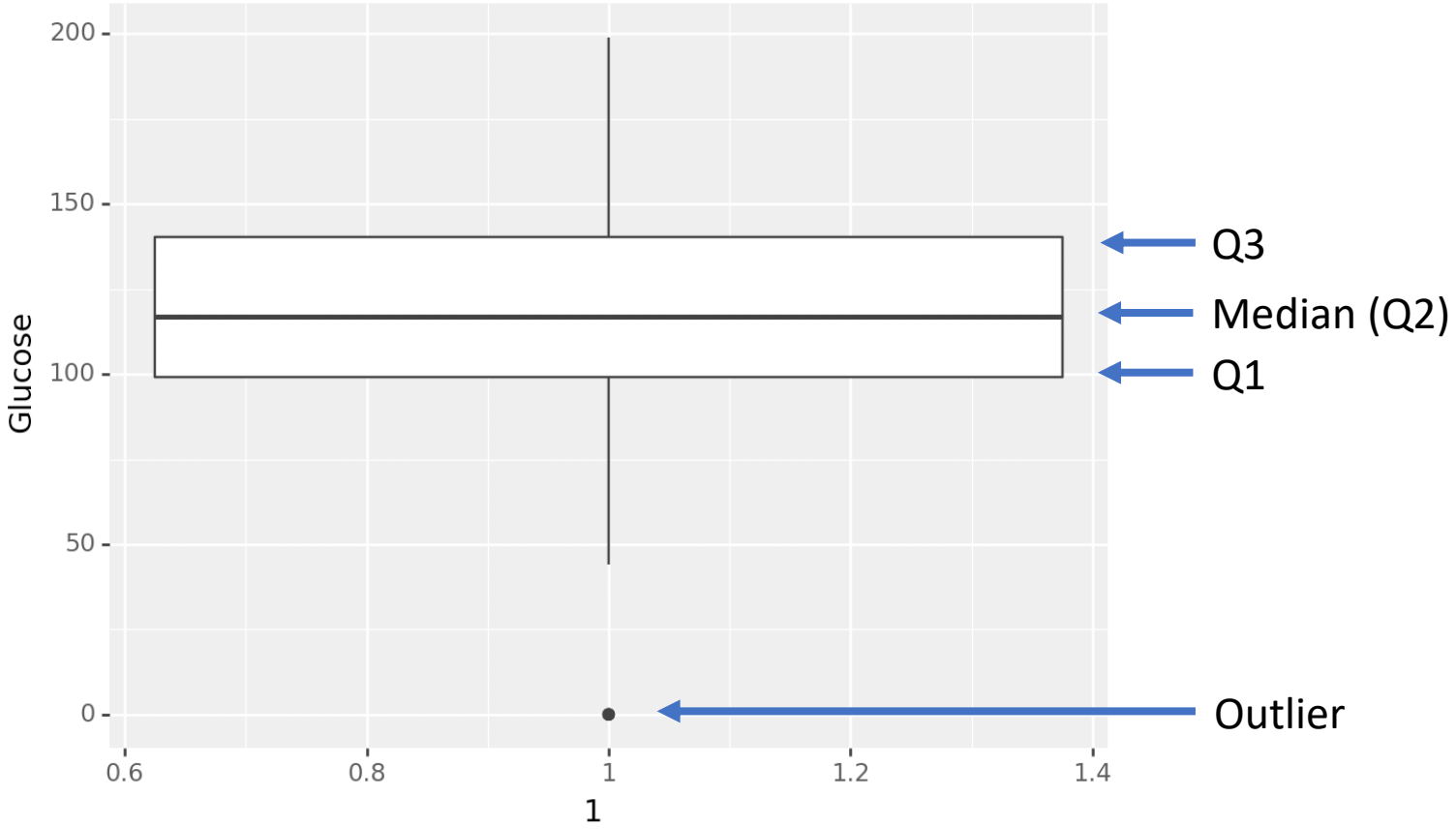

```
(
  p9.ggplot(
    diabetes,
    p9.aes(
      x='Outcome'))
  + p9.geom_bar()
)
```

Distributions: Histograms



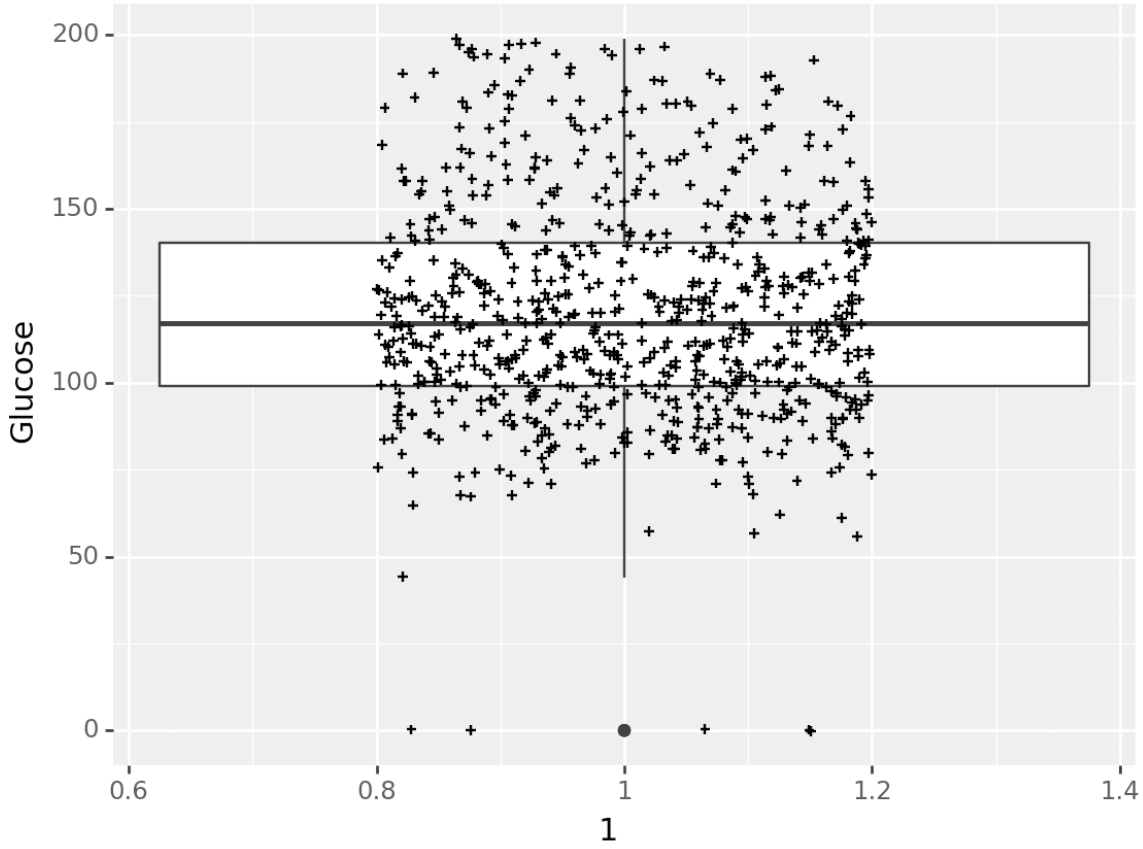

```
p9.ggplot(diabetes, p9.aes(x="Glucose")) + p9.geom_histogram(bins=50)
```

Distributions: Box Plots



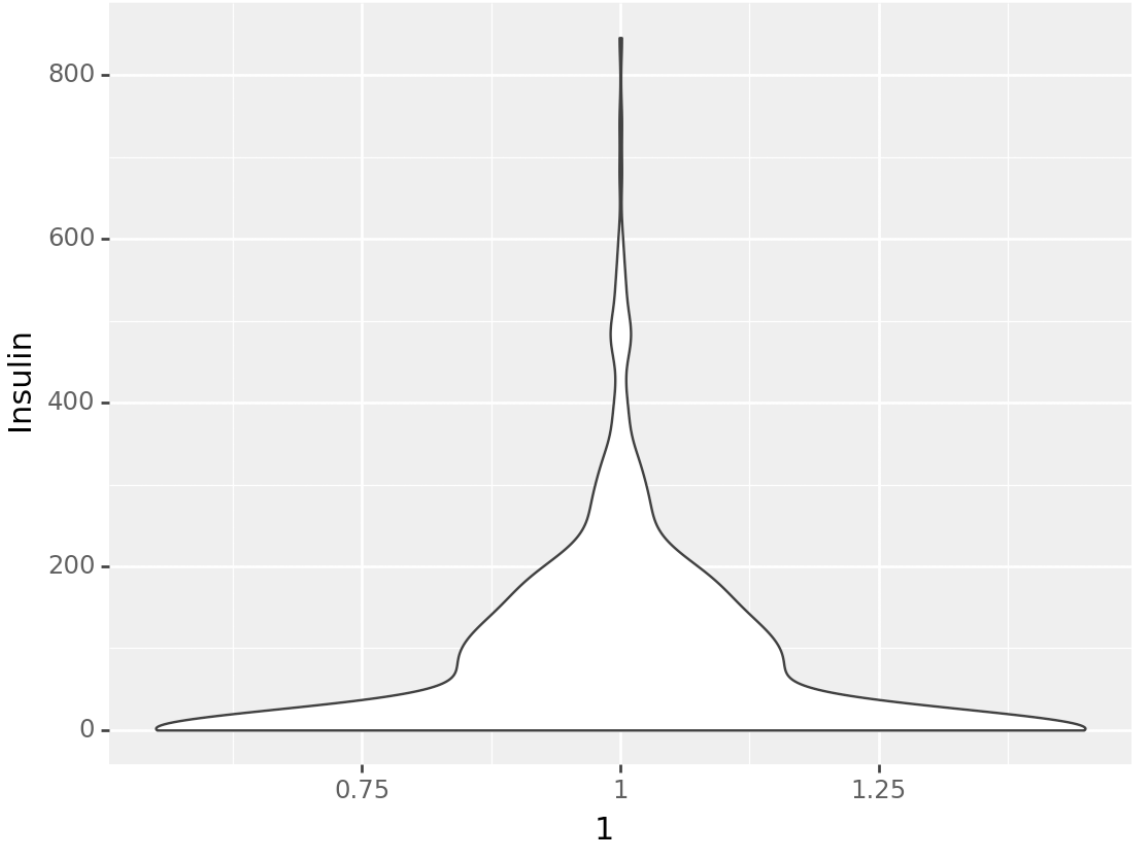

```
p9.ggplot(diabetes, p9.aes(x=1, y="Glucose")) + p9.geom_boxplot()
```

Distributions: Box Plots



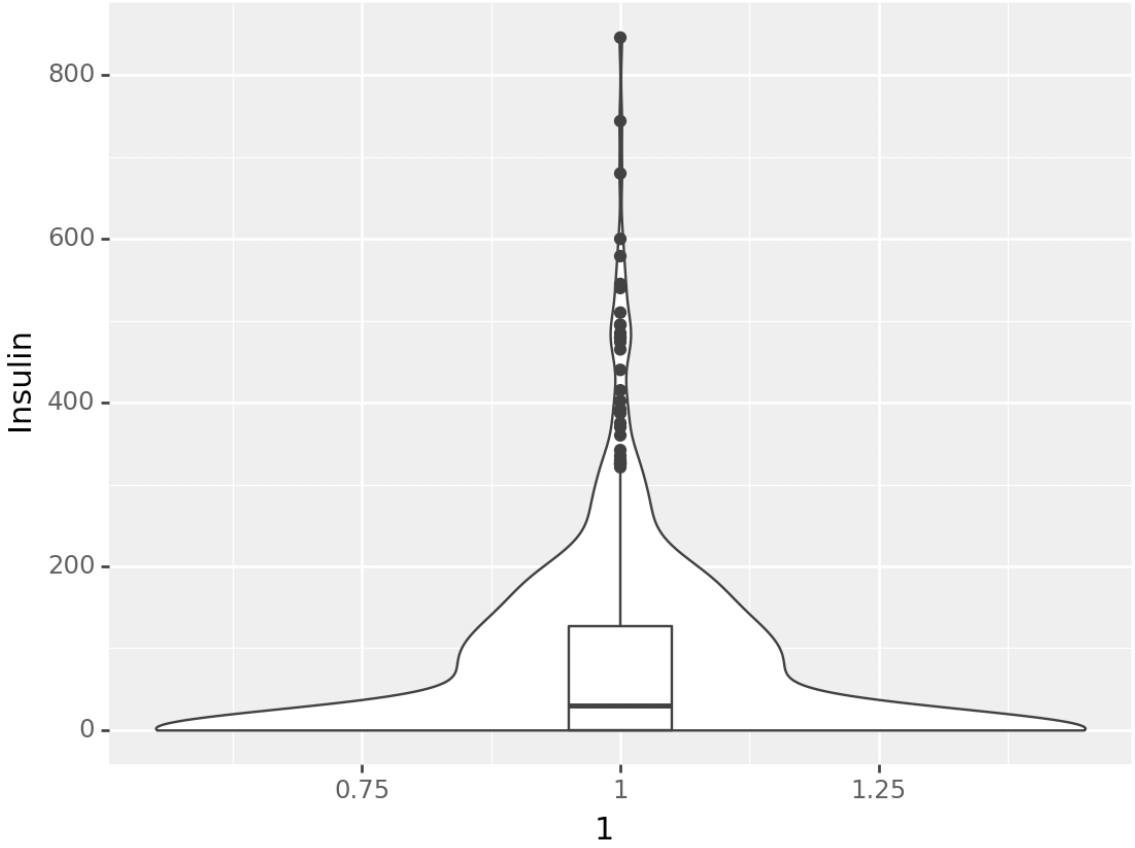

```
(p9.ggplot(diabetes, p9.aes(x=1, y="Glucose"))
 + p9.geom_boxplot()
 + p9.geom_jitter(shape='+', position=p9.position_jitter(0.2)))
```

Distributions: Violin Plots




```
p9.ggplot(diabetes, p9.aes(x=1, y="Insulin")) + p9.geom_violin()
```

Distributions: Violin Plots



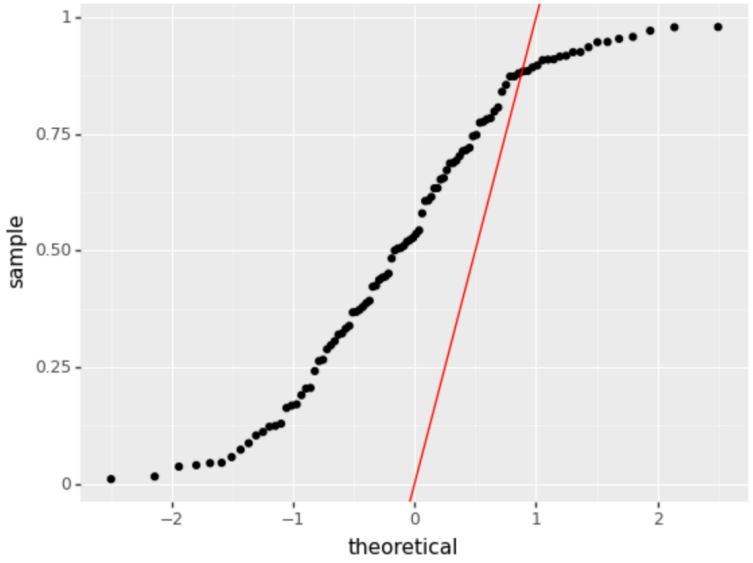

```
(p9.ggplot(diabetes, p9.aes(x=1, y="Insulin"))
+ p9.geom_violin() + p9.geom_boxplot(width=0.1))
```

Does your data look normal? Should it?

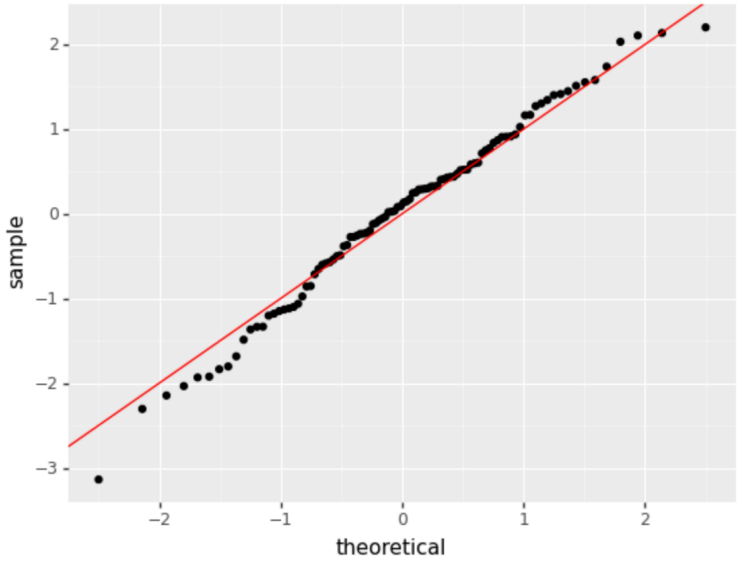
Many statistical procedures assume normality. Can visually test for this with e.g. a Q-Q plot.

```
(p9.ggplot(
  pd.DataFrame(
   {'x': data}),
  p9.aes(sample='x'))
 + p9.geom_qq()
 + p9.geom_abline(
    slope=1,
    intercept=0,
    color='red'))
```

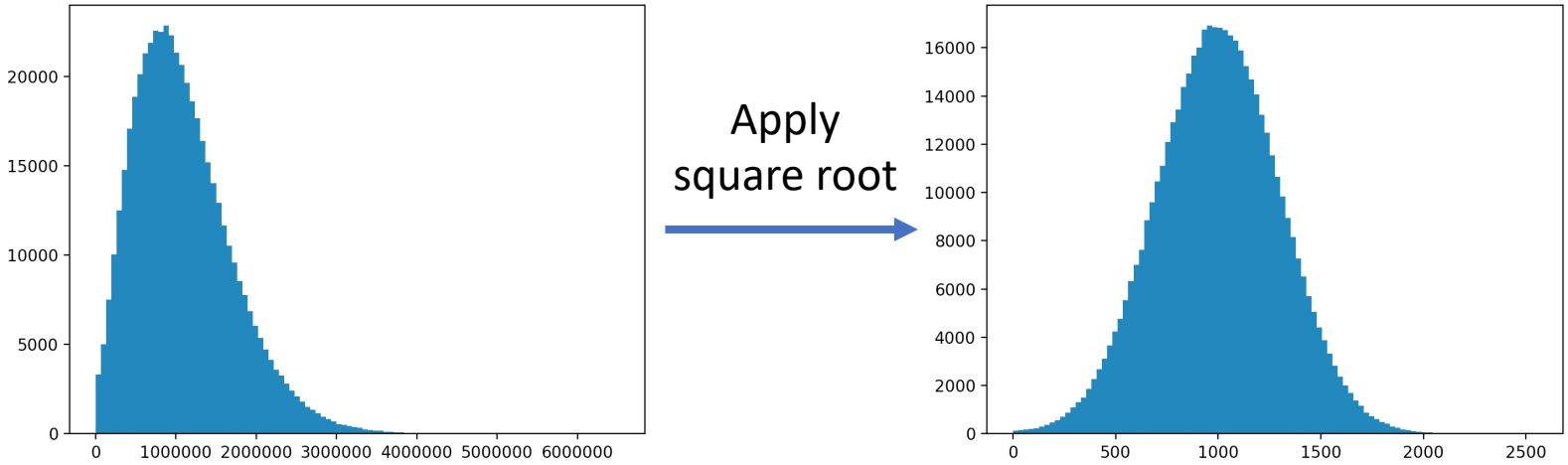
Uniform distribution

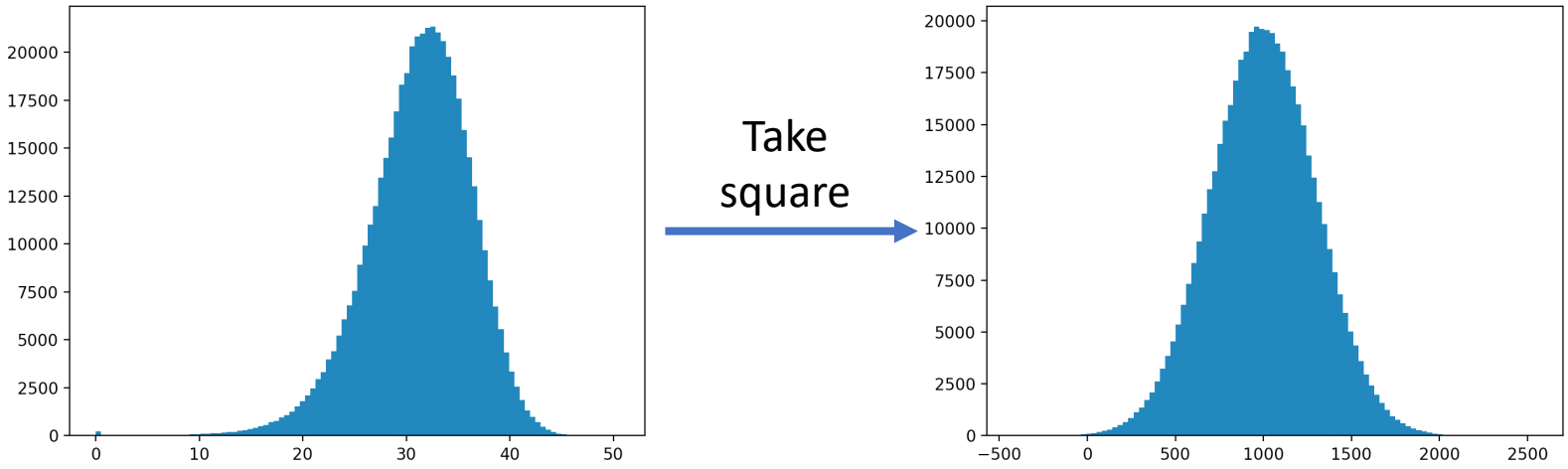


Normal distribution



Making data look more normal





Standardizing Data

Recall that for $x_1, x_2, \ldots, x_n$ data points with sample mean $\bar{x}$ and sample standard deviation s , if

$$z_i = \frac{x_i - \bar{x}}{s}$$

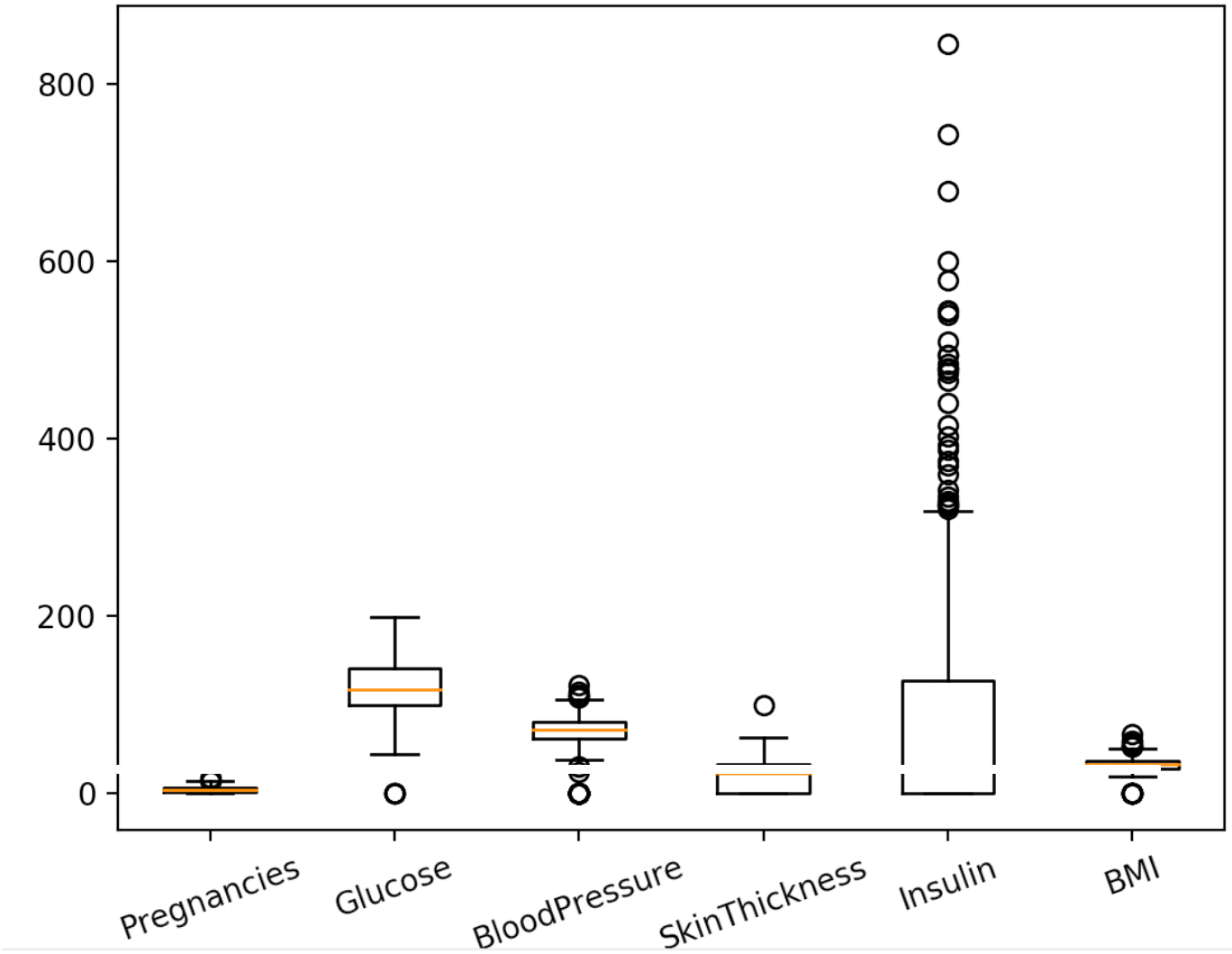
then $z_1, z_2, \ldots, z_n$ will have mean 0 and standard deviation 1.

To study the relation of variables to each other, it is often a good idea to standardize them in this way.

```
glucose = diabetes['Glucose']

diabetes['Glucose_standardized'] = (
                                (glucose - glucose.mean()) / glucose.std())
```

Standardizing data (the problem)



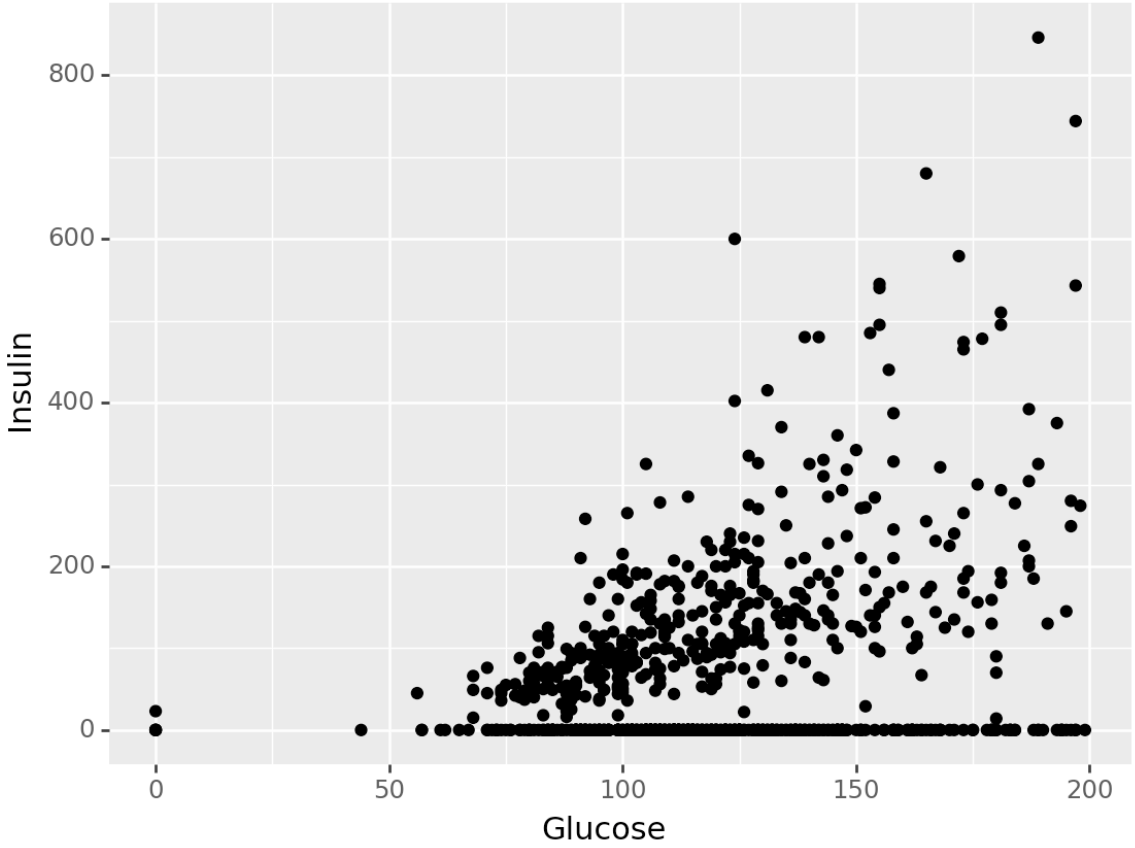
Sometimes other statistical transformations are useful

eg Scaling variables to be between 0 and 1

- Especially if there is a fundamental max and min independent of your data.

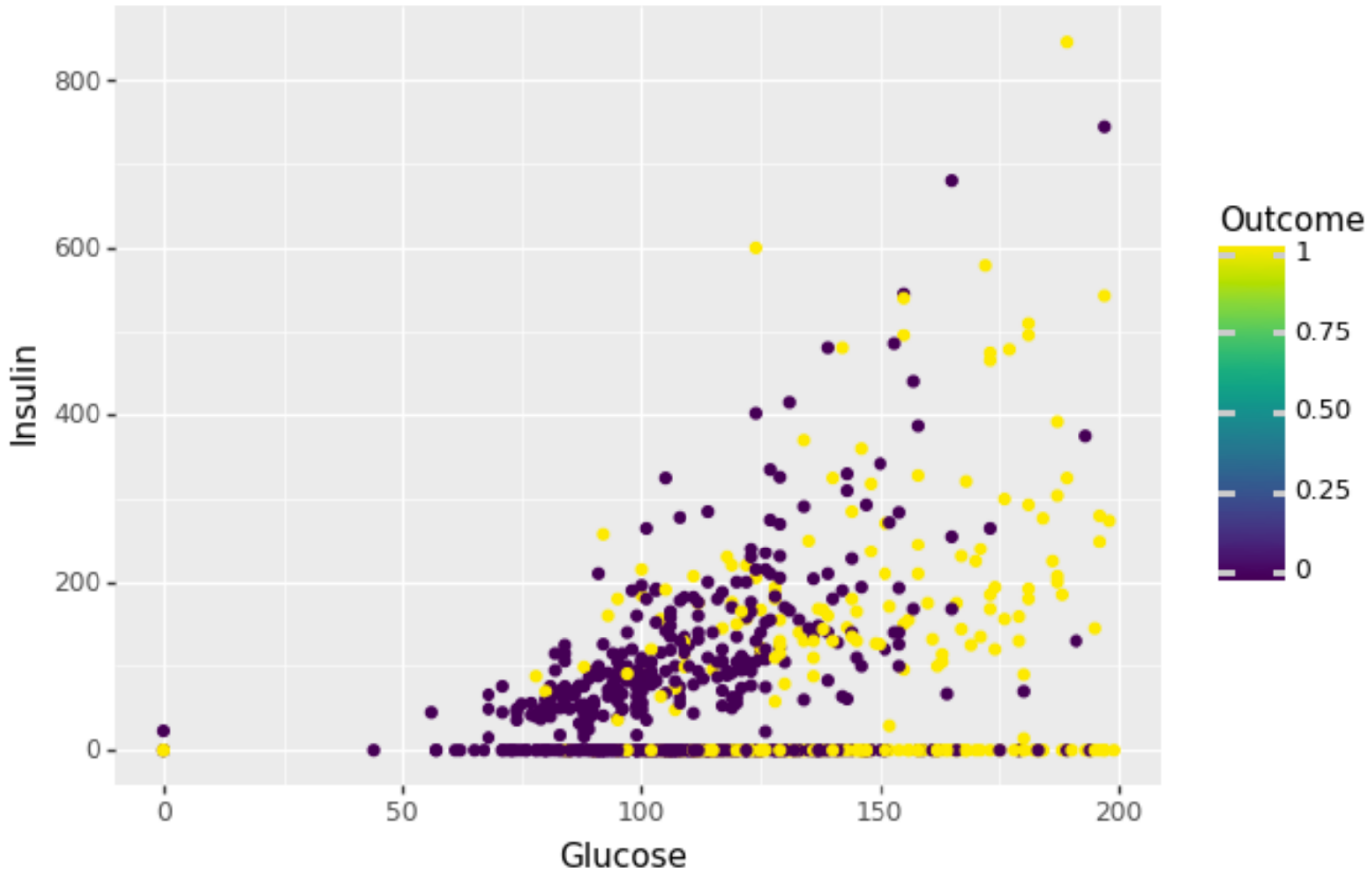
Bivariate Analysis

Scatter Plots




```
(p9.ggplot(diabetes, p9.aes(x="Glucose", y="Insulin"))
    + p9.geom_point())
```

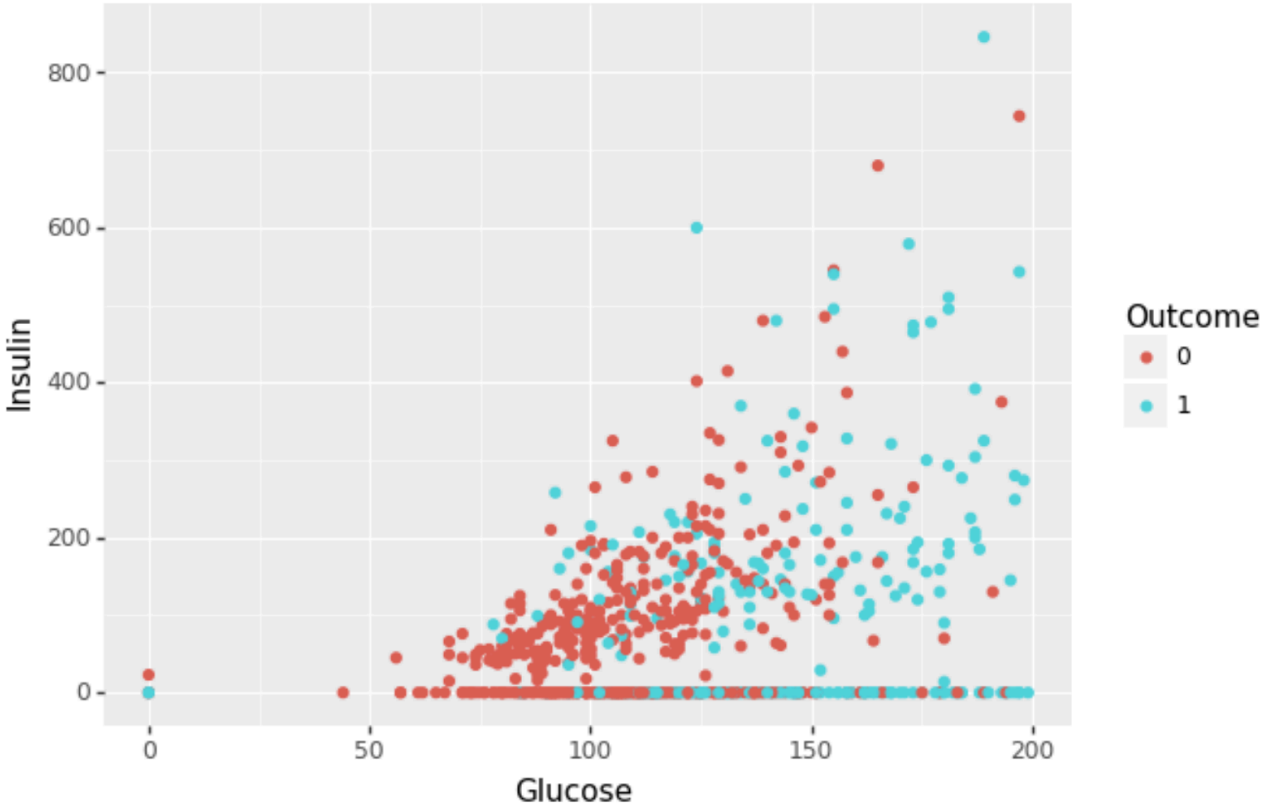
Scatter Plots




```
(
    p9.ggplot(diabetes, p9.aes(x='Glucose', y='Insulin', color='Outcome'))
    + p9.geom_point()
)
```

Scatter Plots

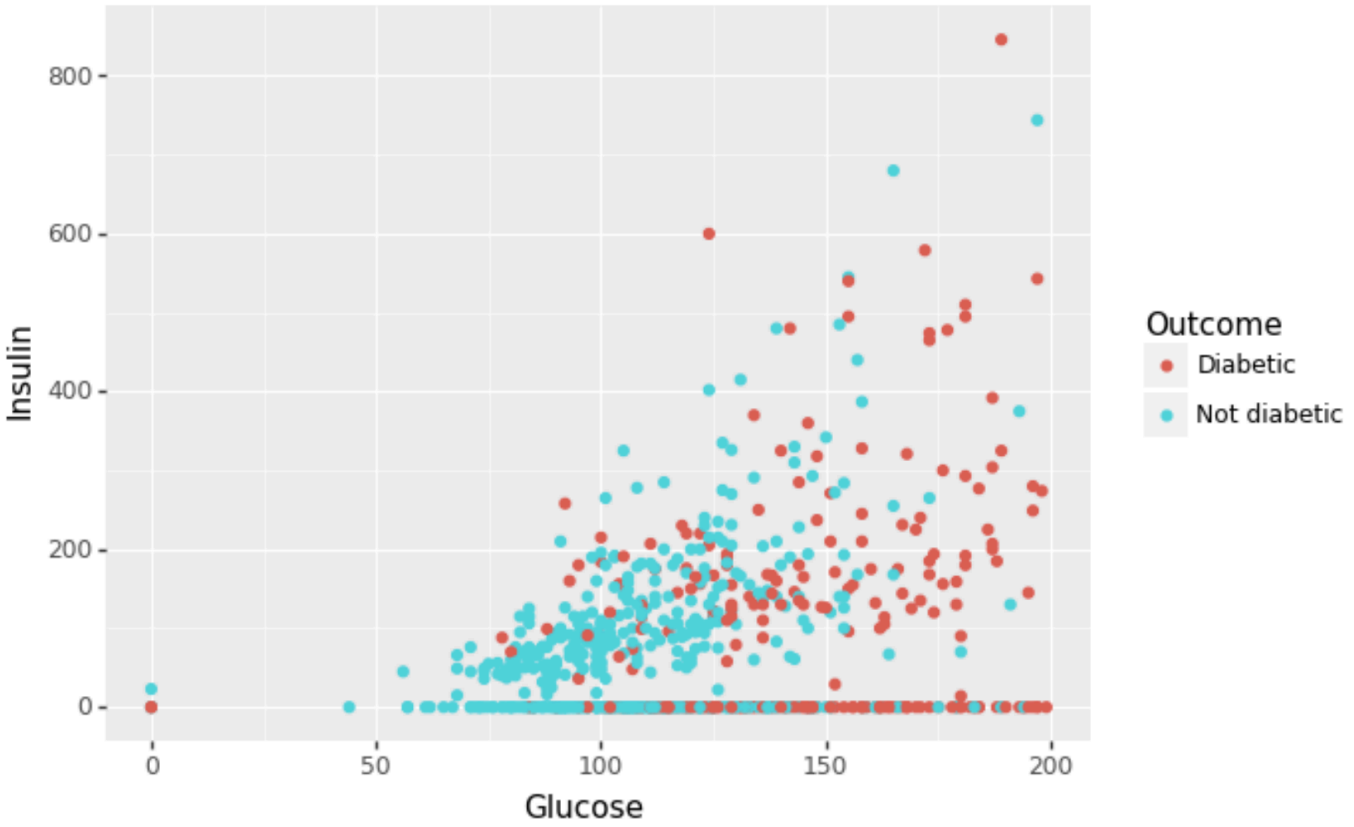
```
diabetes['Outcome'] = diabetes['Outcome'].astype('category')
```



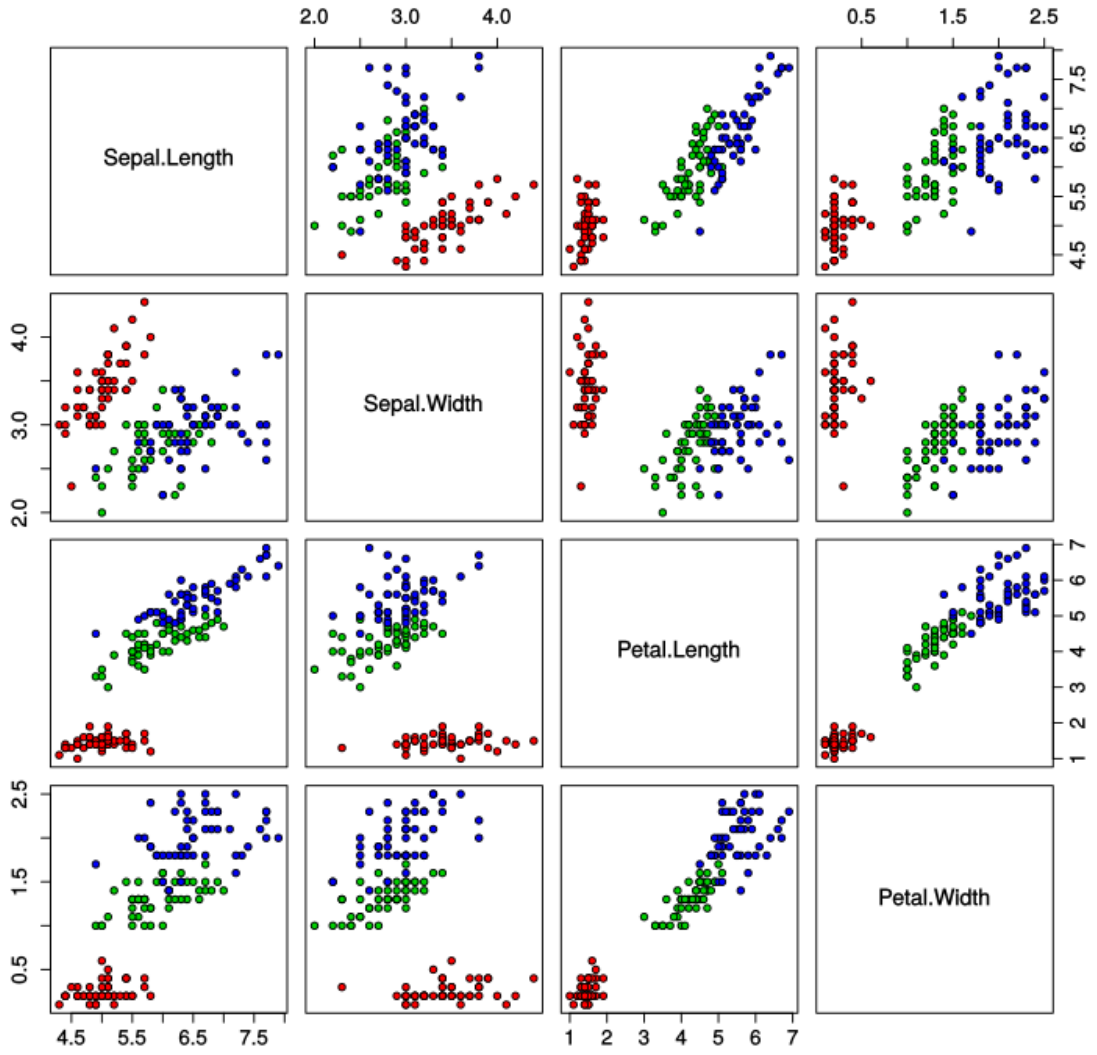

```
(
    p9.ggplot(diabetes, p9.aes(x='Glucose', y='Insulin', color='Outcome'))
    + p9.geom_point()
)
```

Scatter Plots

```
diabetes['Outcome'] = diabetes['Outcome'].replace(
                                    {0: 'Not diabetic', 1: 'Diabetic'})
```




```
(
    p9.ggplot(diabetes, p9.aes(x='Glucose', y='Insulin', color='Outcome'))
    + p9.geom_point()
)
```



https://i.stack.imgur.com/eOvhg.png

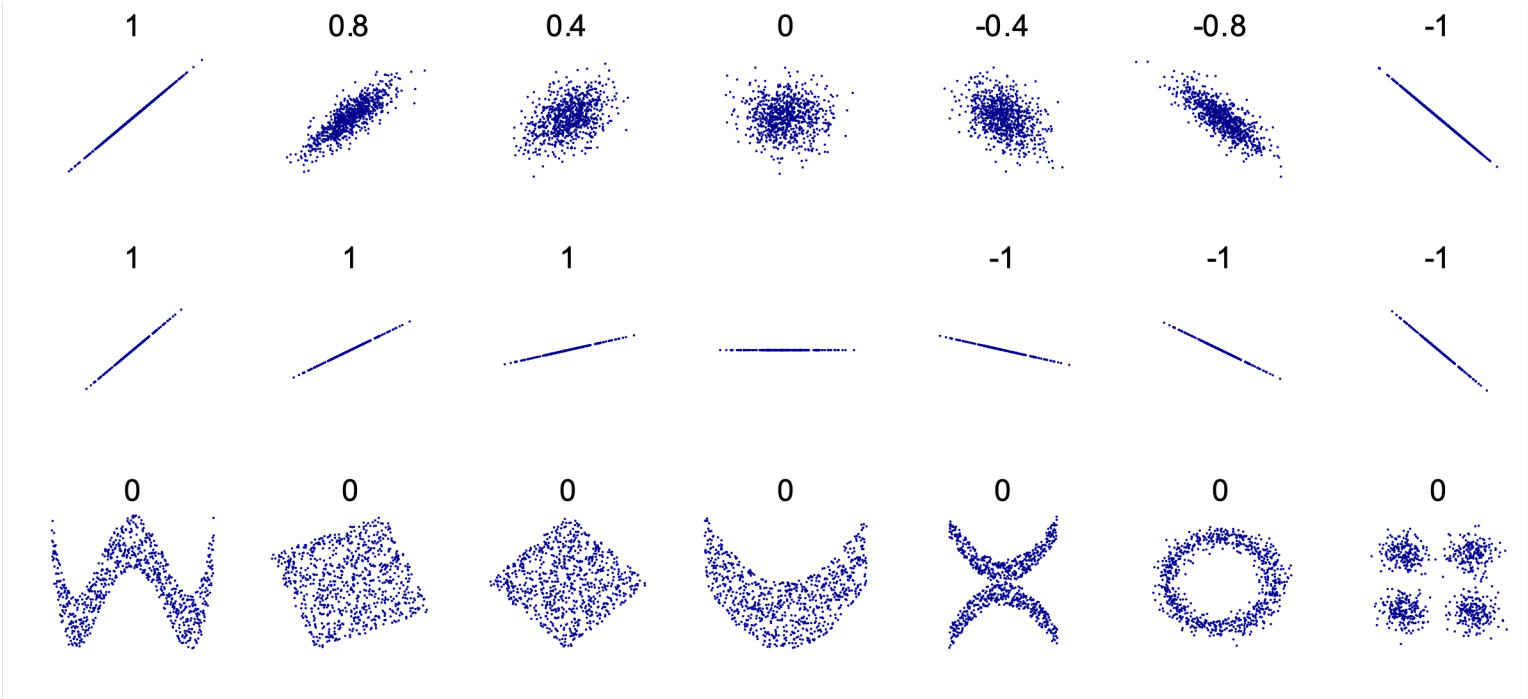


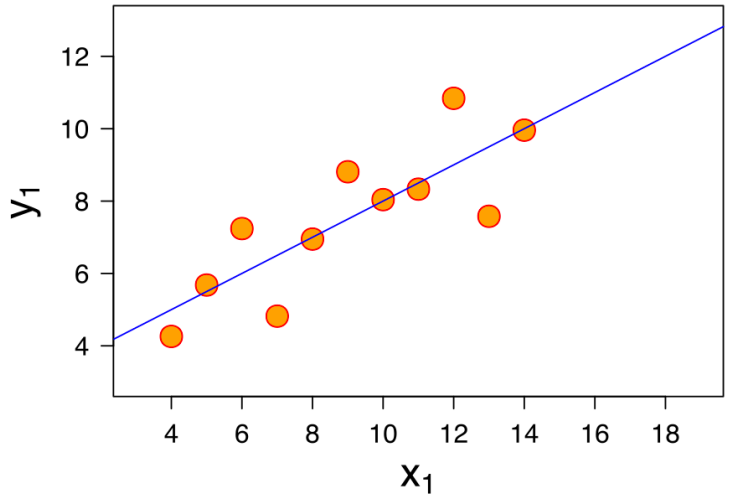
Image CC0 by DenisBoigelot
https://commons.wikimedia.org/wiki/File:Correlation_examples2.svg

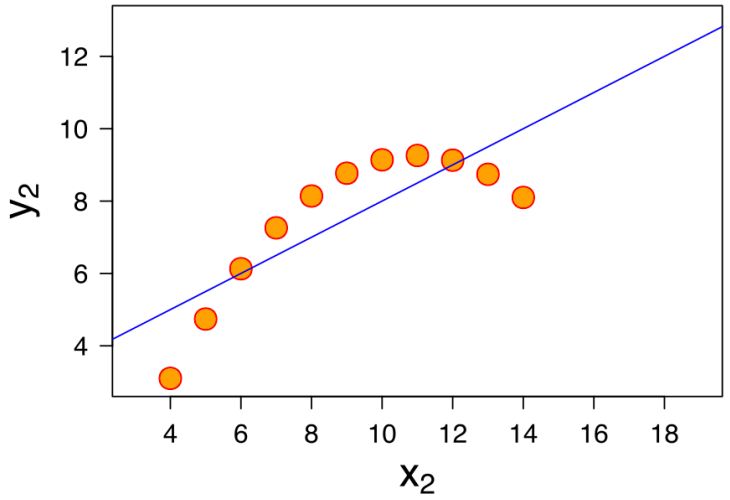
Pearson Correlation

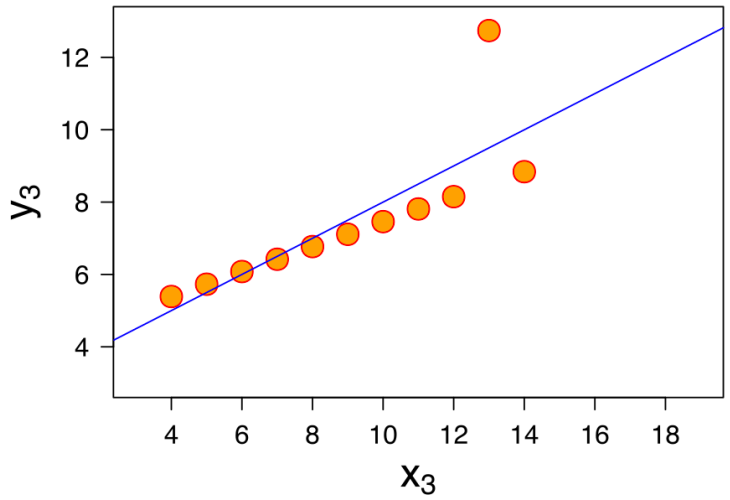
$$r_{xy} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2}\sqrt{\sum(y_i - \bar{y})^2}}$$

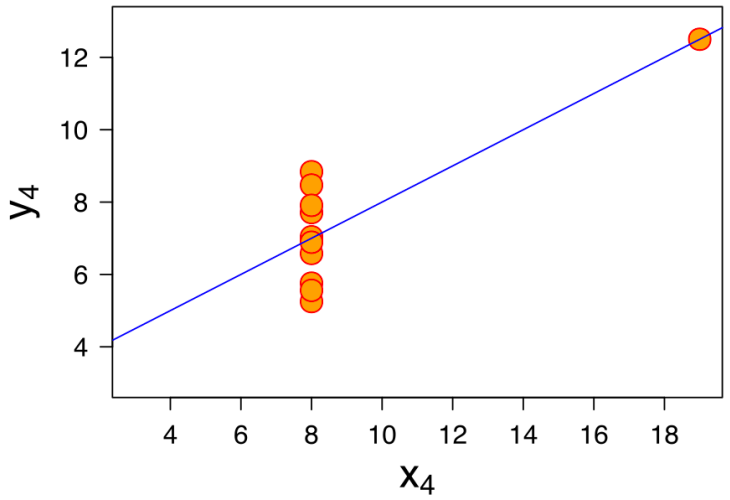
```
a = pd.Series([1, 2, 3, 4])
b = pd.Series([4, 3, 2, 1])
a.corr(b)
```

```
-1.0
```

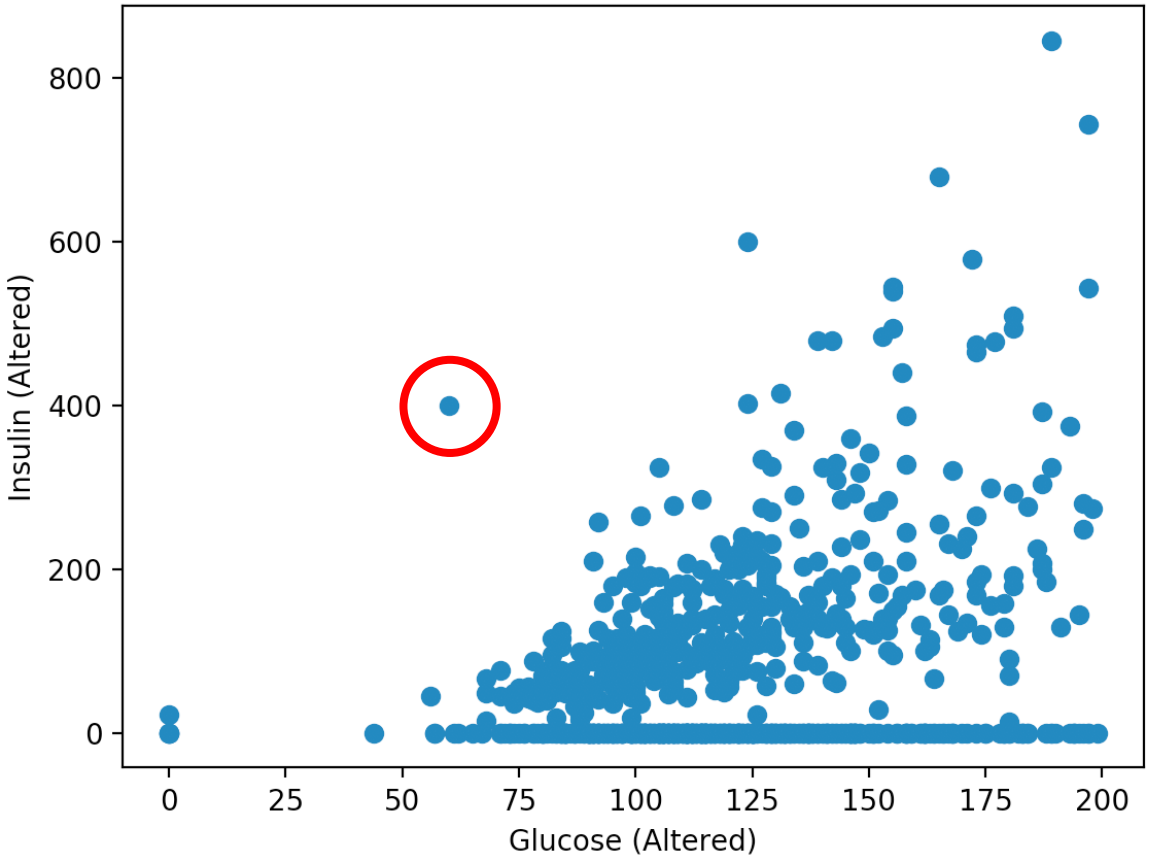




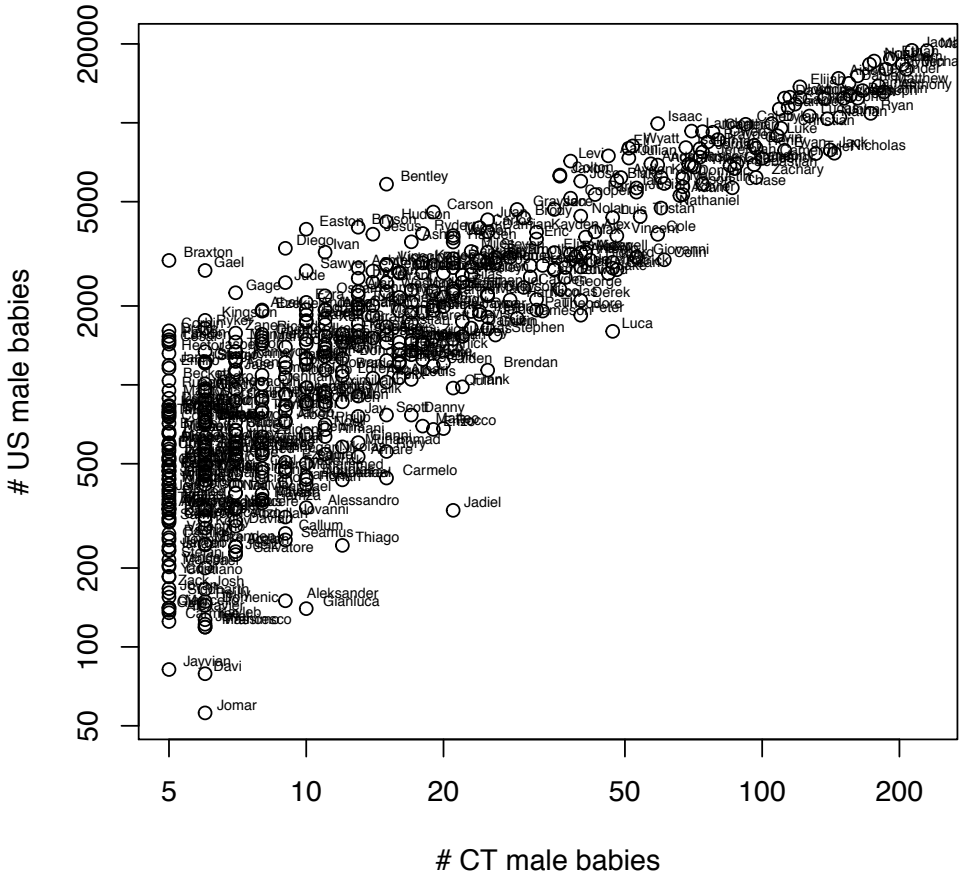




Scatter Plots and outliers



Scatter Plots and Outliers




```
+ p9.scale_x_continuous(trans='log10')
+ p9.scale_y_continuous(trans='log10')
```

2012 name data