Linear algebra review

1. Linear regression
2. Principal component analysis

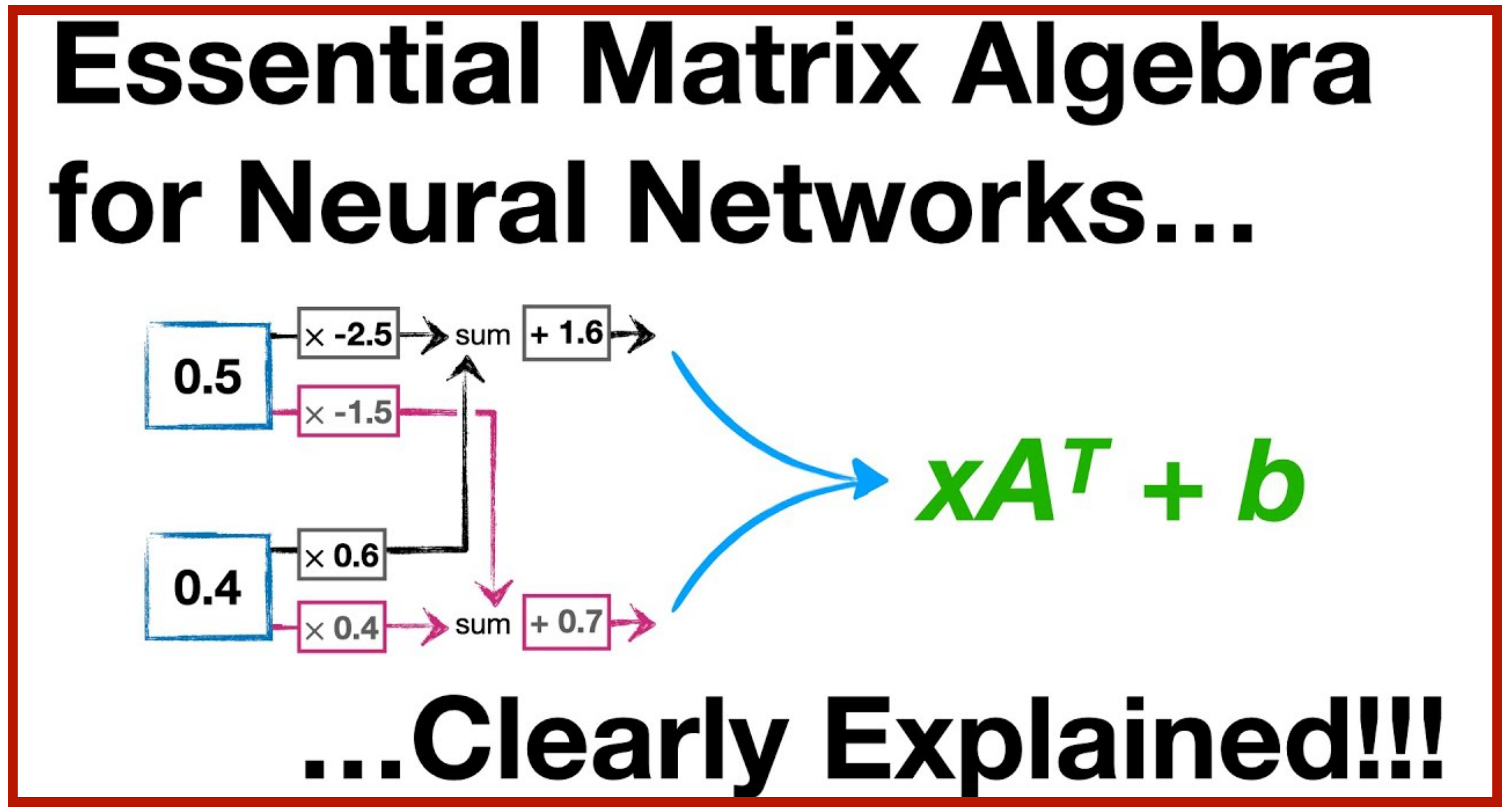
Soumik Purkayastha (soumikp@umich.edu)

Linear algebra review (mostly matrices)

1. Linear regression [transpose and invert matrices]
2. Principal component analysis [eigen-things of matrices]

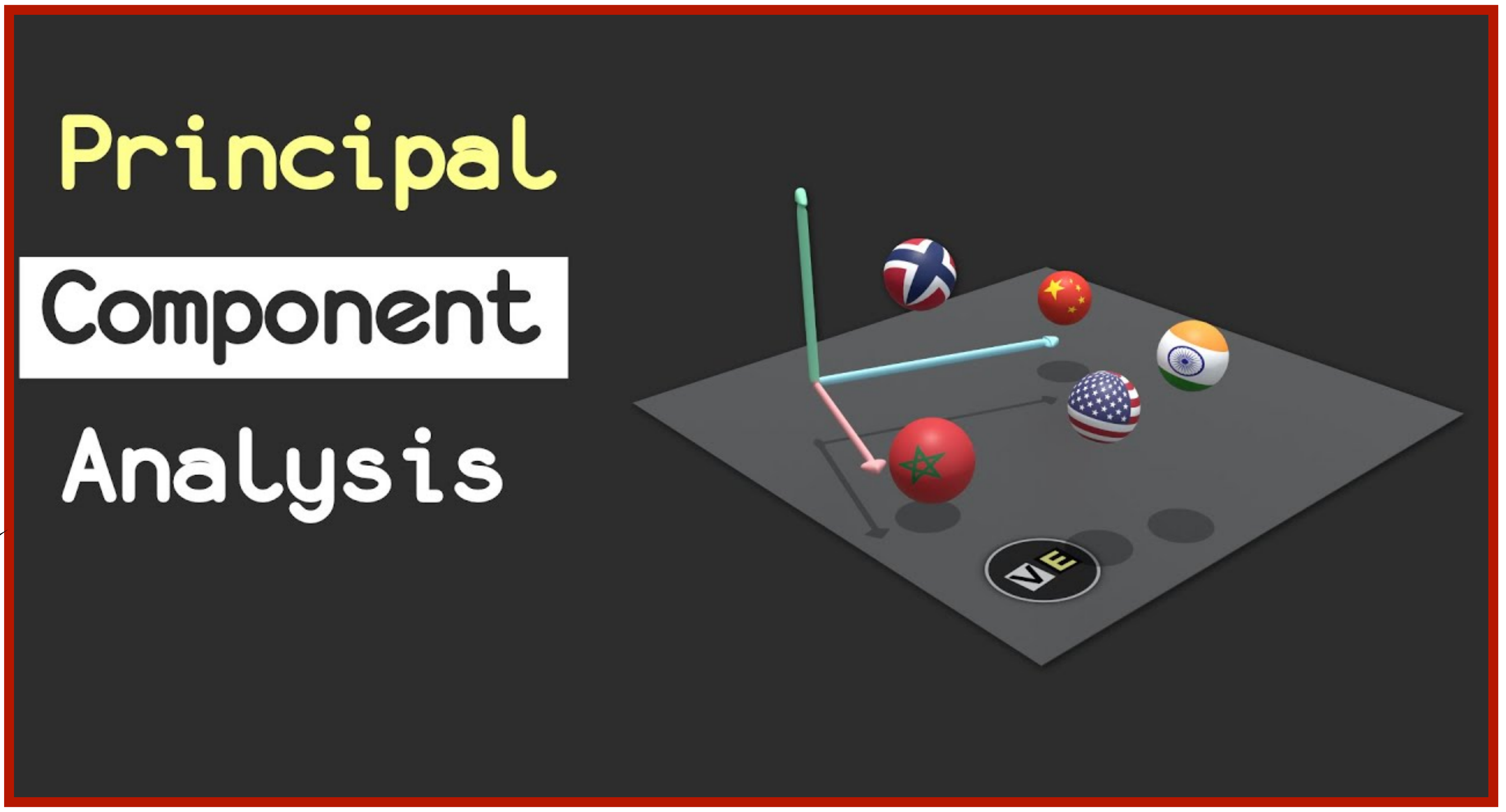
2024 Big Data Summer Institute, Ann Arbor.

To prepare for the class, I watched these two videos [~ 25 minutes]



Start here and watch until 18:47!

Key things to look out for:
- linear transformations
- matrix multiplication
- matrices and linear equations



Please watch the whole thing!

Key things to look out for:
- PCA involves projections
- eigenproblem in PCA
- dimension reduction

Agenda

On June 25 we'll be talking about…

1. **Dataset:** Diagnostic Wisconsin Breast Cancer Database

2. **Linear regression**
    1. Writing data in matrix form
    2. Estimation and inference using matrix algebra

3. **Principal component analysis**
    1. Why bother with PCA?
    2. Implementation using matrix algebra

Dataset: Breast Cancer Wisconsin (Diagnostic)

Biomedical dataset with 569 patients and 30 features

1. **Dataset Composition:** 569 patients with digitized images of breast mass.
2. **Features:** radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension of the cell nuclei.
3. **Classes:** either **malignant** (212 cases) or **benign** (357 cases).

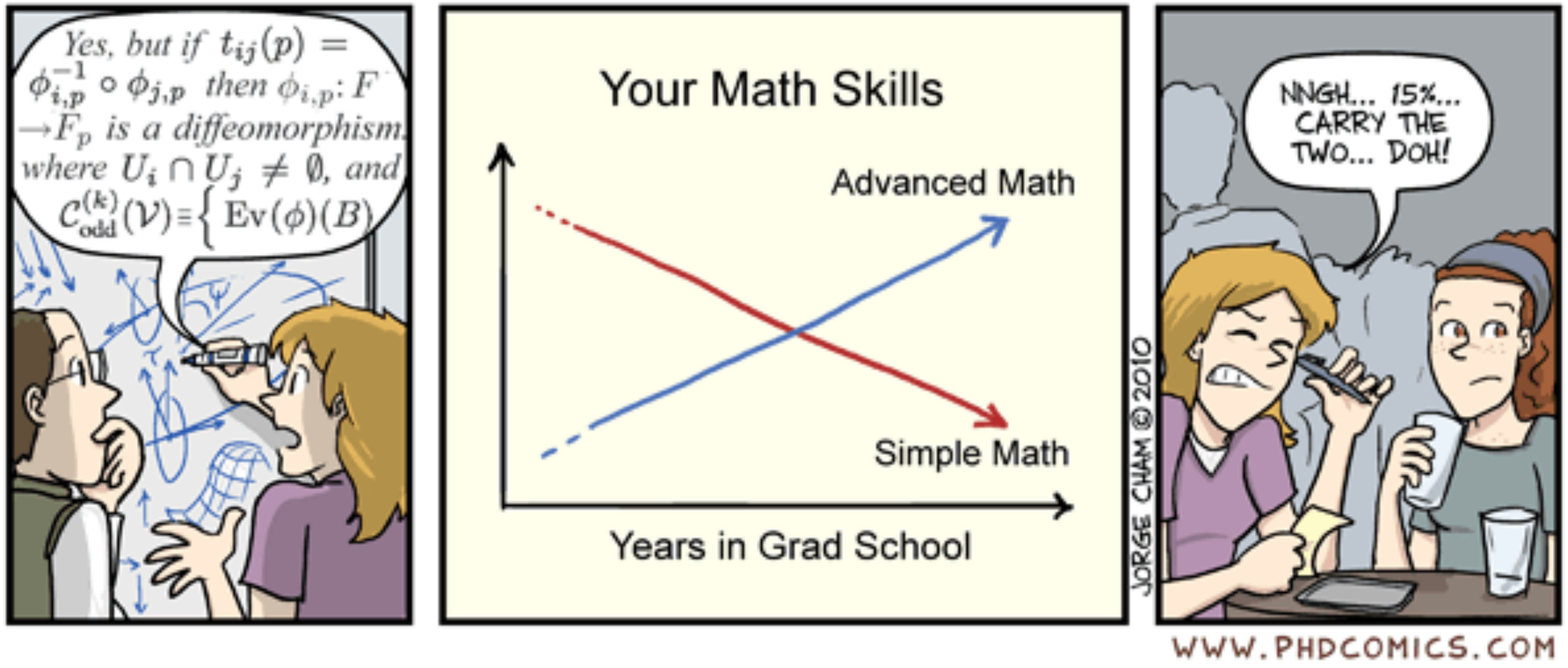
Aim: distinguish between malignant and benign breast cancer cases based on features.

```r
# Load required libraries
library(tidyverse)
library(caret)
library(ggplot2)

# Load the dataset
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data"
columns <- c('ID', 'Diagnosis', paste0('feature_', 1:30))
bc_data <- read.csv(url, header = FALSE, col.names = columns)

view(bc_data)
```

```r
# Load required libraries
library(tidyverse)
library(caret)
library(ggplot2)

# Load the dataset
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data"
columns <- c('ID', 'Diagnosis', paste0('feature_', 1:30))
bc_data <- read.csv(url, header = FALSE, col.names = columns)

view(bc_data)
```

You **S**hould **K**now:

1. How many rows, how many columns?
2. What does each row signify?
3. What does each column signify?


Yes, but if $t_{ij}(p) = \phi_{i,p}^{-1} \circ \phi_{j,p}$ then $\phi_{i,p}: F \to F_p$ is a diffeomorphism, where $U_i \cap U_j \neq \emptyset$, and $\mathcal{C}_{odd}^{(k)}(\mathcal{V}) \equiv \{ \mathrm{Ev}(\phi)(B)$
Your Math Skills
Advanced Math
Simple Math
Years in Grad School
JORGE CHAM © 2010
NNGH... 15%... CARRY THE TWO... DOH!
WWW.PHDCOMICS.COM

"linear regression is simple math"

Linear regression

Data in matrix form

$$Y = \mathbf{X}\beta + \epsilon$$

1. $\mathbf{Y}$ is the vector of target values (dependent variable)
2. $\mathbf{X}$ is the matrix of input features (covariates)
3. β is the vector of coefficients (weights)
4. ϵ is the vector of errors (residuals)

$$\begin{aligned} \boxed{Y_1} &= \beta_0 + \beta_1 X_{11} + \ldots + \beta_p X_{p1} + \epsilon_1 \\ Y_2 &= \beta_0 + \beta_1 X_{12} + \ldots + \beta_p X_{p2} + \epsilon_2 \\ &\vdots \\ Y_n &= \beta_0 + \beta_1 X_{1n} + \ldots + \beta_p X_{pn} + \epsilon_n \end{aligned}$$

n: number of patients

p: number of features

Linear regression

Data in matrix form

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon$$

1. $\mathbf{Y}$ is the vector of target values (dependent variable)
2. $\mathbf{X}$ is the matrix of input features (covariates)
3. **β is the vector of coefficients (weights)**
4. ϵ is the vector of errors (residuals)

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}, \quad \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

```
# # create a 4*4 matrix
x <- matrix(rpois(16, 5), ncol = 4)
view(x)

# transpose of matrix
t(x)

# inverse of matrix
solve(x)
```

You Should Know:

1. Transpose of a matrix
2. Inverse of a matrix

Linear regression

Estimation

$$Y = X\beta + \epsilon, \qquad \mathbb{V}(\epsilon) = \sigma^2.$$

Step 1: estimation and inference for β

$$\hat{\beta} = (X^tX)^{-1}(X^tY)$$

Step 2: predictions $\hat{Y} = X(X^tX)^{-1}(X^tY) = P_XY$

$P_X = X(X^tX)^{-1}X^t$ is the **projection** matrix.

Step 3: residuals $e = \hat{\epsilon} = Y - \hat{Y} = (I - P_X)Y$

$I - P_X$ is the **annihilator** matrix.

Linear regression

Estimation

$$Y = X\beta + \epsilon, \qquad \mathbb{V}(\epsilon) = \sigma^2.$$

Step 1: estimation and inference for β

$$\hat{\beta} = (X^tX)^{-1}(X^tY)$$

Step 2: predictions $\hat{Y} = X(X^tX)^{-1}(X^tY) = P_XY$

$P_X = X(X^tX)^{-1}X^t$ is the **projection** matrix.

Step 3: residuals $e = \hat{\epsilon} = Y - \hat{Y} = (I - P_X)Y$

$I - P_X$ is the **annihilator** matrix.

```
                    STEP 1

# Load required libraries
library(tidyverse)

# Set seed for reproducibility
set.seed(48103)

# Generate synthetic data of size 100
# true Y = 3 + 2*X
# epsilon variance = 4

n <- 100
x <- runif(n, min = 0, max = 10)
y <- 3 + 2 * x + rnorm(n, mean = 0, sd = 2)
```

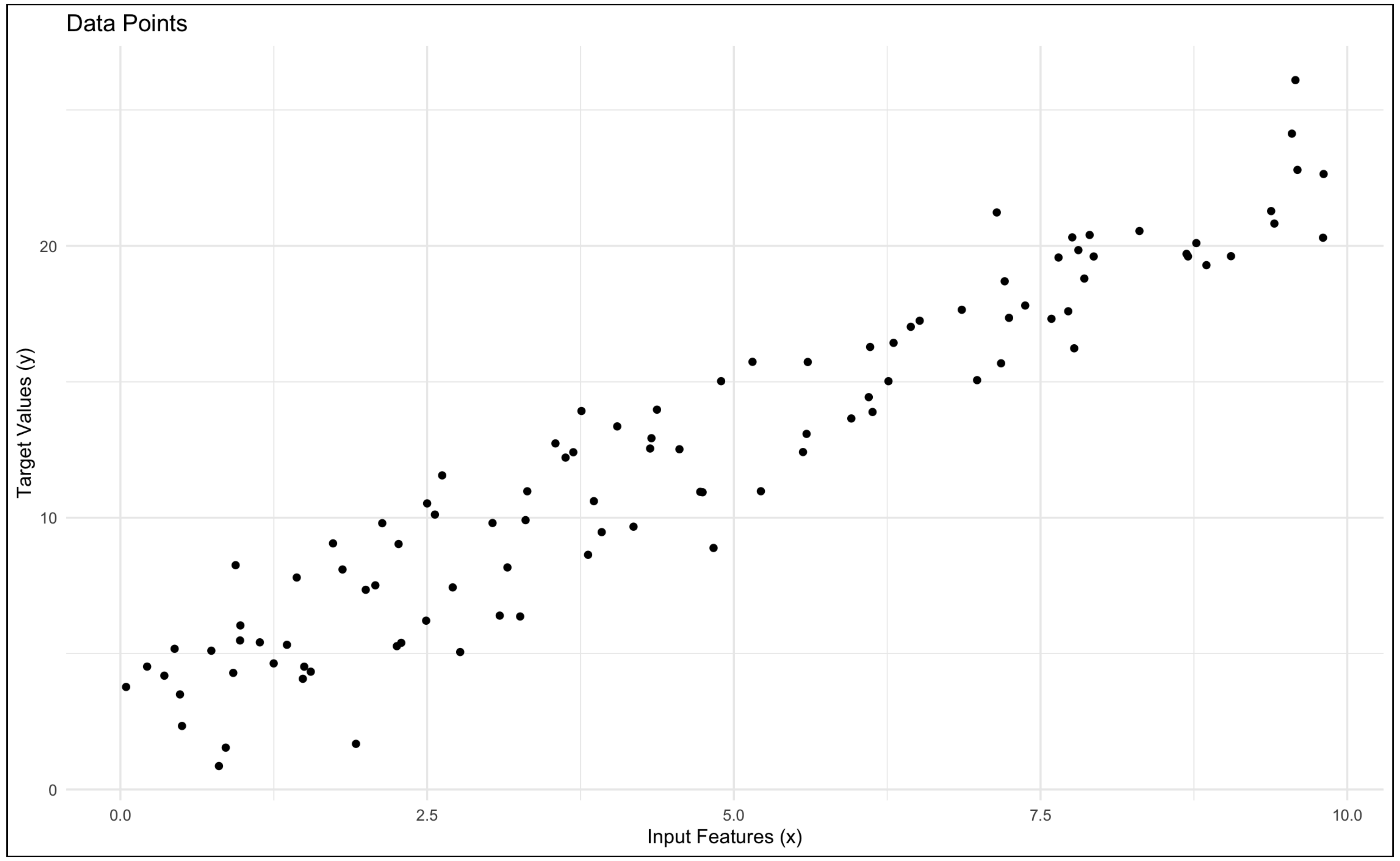
```
                    STEP 2

# Create a data frame
data <- tibble(x = x, y = y)

# Plot the data points
ggplot(data, aes(x = x, y = y)) +
  geom_point() +
  labs(
    title = "Data Points",
    x = "Input Features (x)",
    y = "Target Values (y)"
  ) +
  theme_minimal()
```

STEP 1

```
# Load required libraries
library(tidyverse)

# Set seed for reproducibility
set.seed(48103)

# Generate synthetic data of size 100
# true Y = 3 + 2*X
# epsilon variance = 4

n <- 100
x <- runif(n, min = 0, max = 10)
y <- 3 + 2 * x + rnorm(n, mean = 0, sd = 2)
```

STEP 2

```
# Create a data frame
data <- tibble(x = x, y = y)

# Plot the data points
ggplot(data, aes(x = x, y = y)) +
  geom_point() +
  labs(
    title = "Data Points",
    x = "Input Features (x)",
    y = "Target Values (y)"
  ) +
  theme_minimal()
```



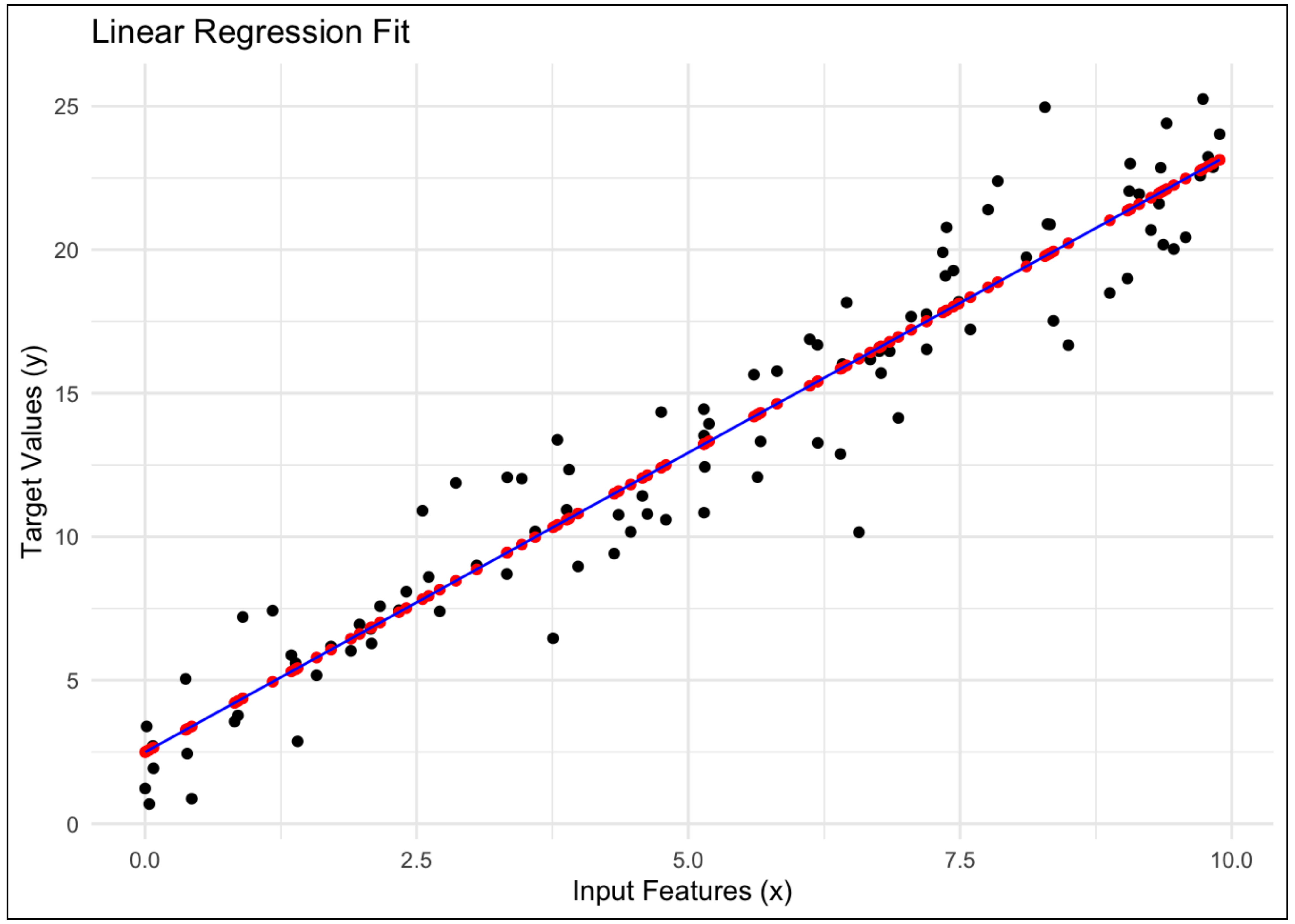
Obtaining estimated coefficients
Plotting fitted values
Plotting regression line

STEP 3

```
# Matrix X, add column of ones for intercept
X <- cbind(1, data$x)

# Create the vector Y
Y <- data$y

# Estimate the coefficient beta
B <- solve(t(X) %*% X) %*% t(X) %*% Y

# Print the coefficients
B
```

STEP 4

```
# Predicted values
data <- data %>% mutate(y_pred = X %*% B)

# Plot the data points and the regression line
ggplot(data, aes(x = x)) +
  geom_point(aes(y = y)) +
  geom_point(aes(y = y_pred), color = "red") +
  geom_line(aes(y = y_pred), color = "blue")
```



STEP 3

```
# Matrix X, add column of ones for intercept
X <- cbind(1, data$x)

# Create the vector Y
Y <- data$y

# Estimate the coefficient beta
B <- solve(t(X) %*% X) %*% t(X) %*% Y

# Print the coefficients
B
```

STEP 4

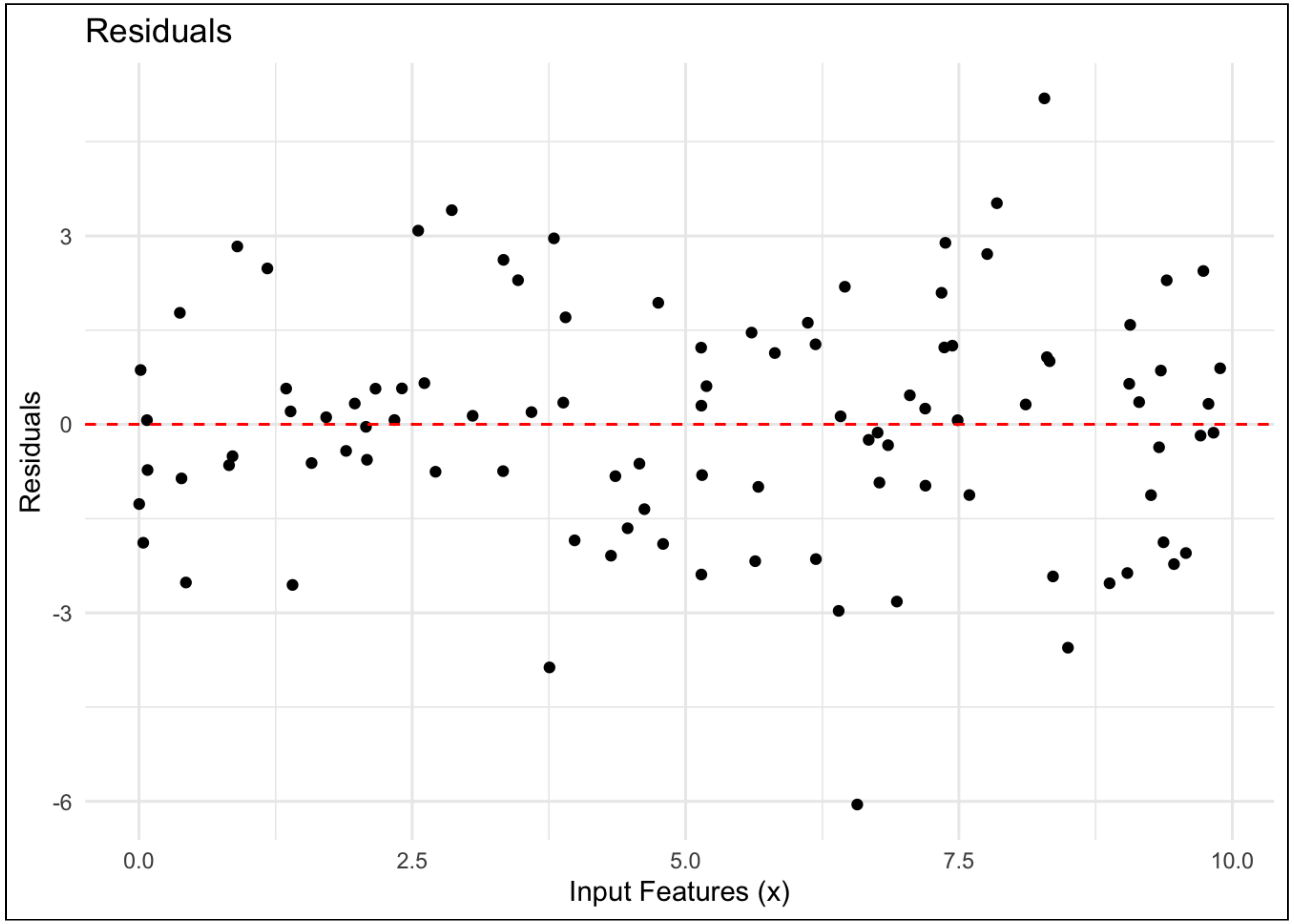
```
# Predicted values
data <- data %>% mutate(y_pred = X %*% B)

# Plot the data points and the regression line
ggplot(data, aes(x = x)) +
  geom_point(aes(y = y)) +
  geom_point(aes(y = y_pred), color = "red") +
  geom_line(aes(y = y_pred), color = "blue")
```

```
                STEP 5

# Calculate residuals
data <- data %>%
mutate(residual = y - y_pred)

# Plot residuals
ggplot(data,
aes(x = x, y = residual)) +
  geom_point() +
  geom_hline(yintercept = 0,
linetype = "dashed", color =
"red") +
  labs(
    title = "Residuals",
    x = "Input Features (x)",
    y = "Residuals"
  )
```

Obtaining and plotting the residuals

STEP 5

```
# Calculate residuals
data <- data %>%
mutate(residual = y - y_pred)

# Plot residuals
ggplot(data,
aes(x = x, y = residual)) +
  geom_point() +
  geom_hline(yintercept = 0,
linetype = "dashed", color =
"red") +
  labs(
    title = "Residuals",
    x = "Input Features (x)",
    y = "Residuals"
  )
```



Everything is simpler in `R`!

Use `lm` instead of matrix algebra

$$Y = X\beta + \epsilon, \qquad \mathbb{V}(\epsilon) = \sigma^2.$$

Step 1: estimation and inference for β

$$\hat{\beta} = (X^tX)^{-1}(X^tY)$$

```
lm(y ~ x, data = data)

Call:
lm(formula = y ~ x, data = data)

Coefficients:
(Intercept)            x
      2.896        2.027
```

Step 2: predictions $\hat{Y} = X(X^tX)^{-1}(X^tY) = P_XY$

$P_X = X(X^tX)^{-1}X^t$ is the **projection** matrix.

```
lm(y ~ x, data = data)$fitted
```

Step 3: residuals $e = \hat{\epsilon} = Y - \hat{Y} = (I - P_X)Y$

$I - P_X$ is the **annihilator** matrix.

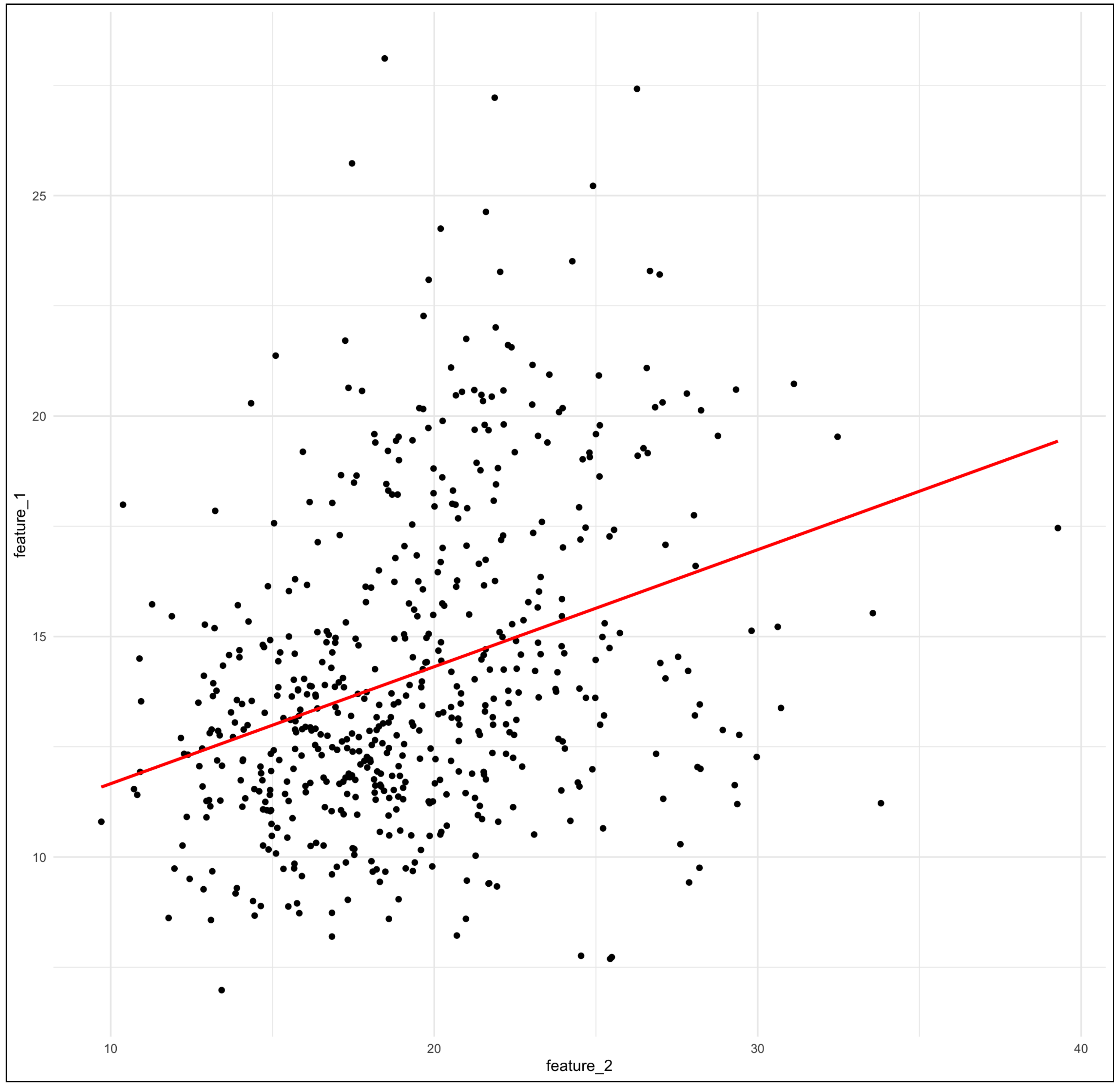
```
lm(y ~ x, data = data)$residuals
```

```
## regression of feature_1 using feature_2
lm(feature_1 ~ feature_2, data = bc_data)

## plot of fitted regression line
bc_data %>%
  ggplot(aes(x = feature_2, y = feature_1)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE,
color = "red")
```

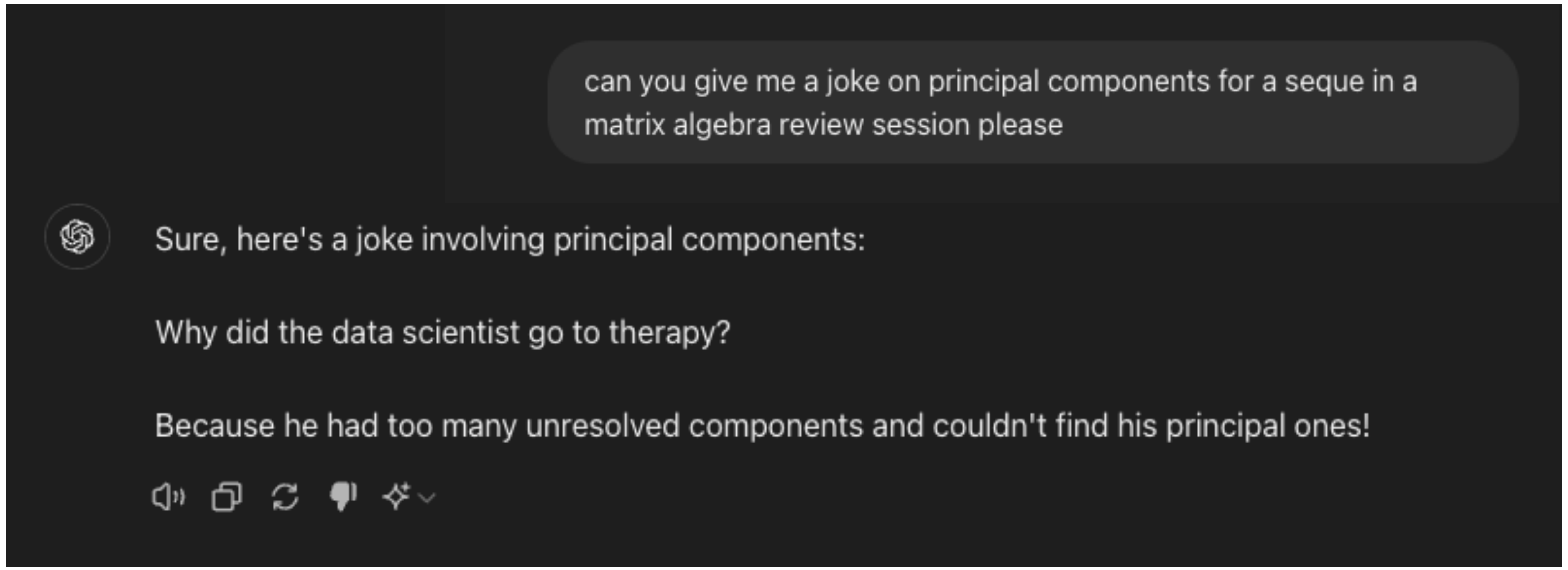
$$\text{feature}_1 = \beta_0 + \beta_1 \text{feature}_2 + \epsilon$$

```
## regression of feature_1 using feature_2
lm(feature_1 ~ feature_2, data = bc_data)

## plot of fitted regression line
bc_data %>%
  ggplot(aes(x = feature_2, y = feature_1)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE,
color = "red")
```

```
> lm(feature_1 ~ feature_2, data = bc_data)

Call:
lm(formula = feature_1 ~ feature_2, data = bc_data)

Coefficients:
(Intercept)    feature_2
     9.0099       0.2653
```



A nice segue for principal components analysis…



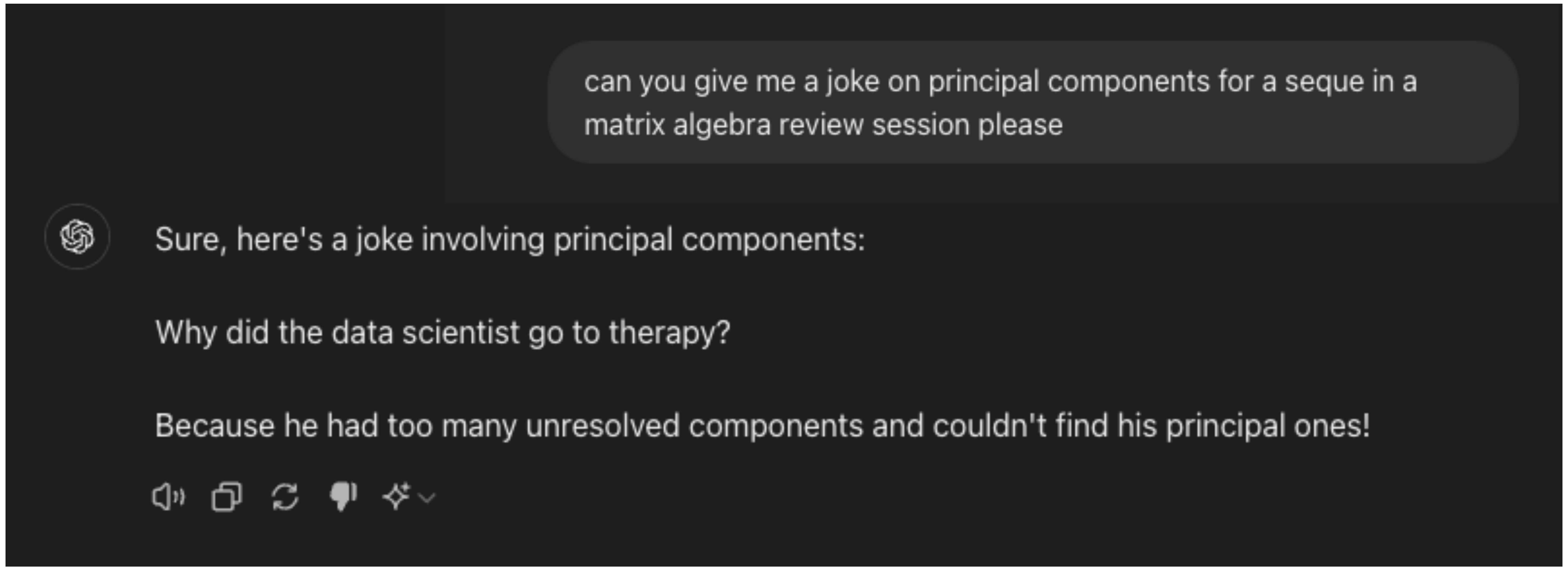
can you give me a joke on principal components for a seque in a matrix algebra review session please

Sure, here's a joke involving principal components:

Why did the data scientist go to therapy?

Because he had too many unresolved components and couldn't find his principal ones!

A ~~nice~~ segue for principal components analysis…

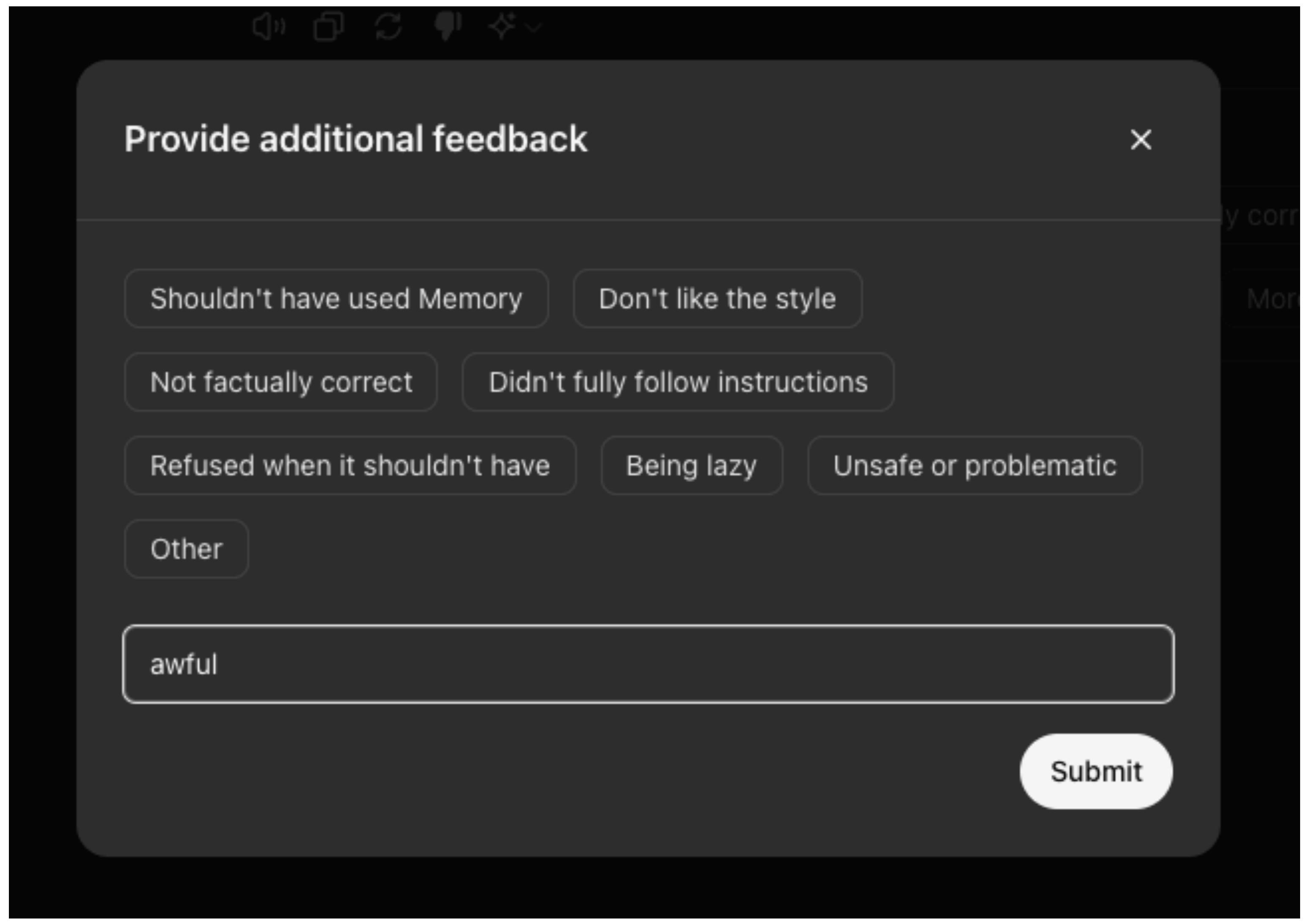

can you give me a joke on principal components for a seque in a matrix algebra review session please

Sure, here's a joke involving principal components:

Why did the data scientist go to therapy?

Because he had too many unresolved components and couldn't find his principal ones!

A ~~nice~~ segue for principal components analysis…


Provide additional feedback

Shouldn't have used Memory
Don't like the style
Not factually correct
Didn't fully follow instructions
Refused when it shouldn't have
Being lazy
Unsafe or problematic
Other
Submit

awful

Helping OpenAI train their models better :)

Introduction to PCA

1. Statistical technique used for dimensionality reduction.

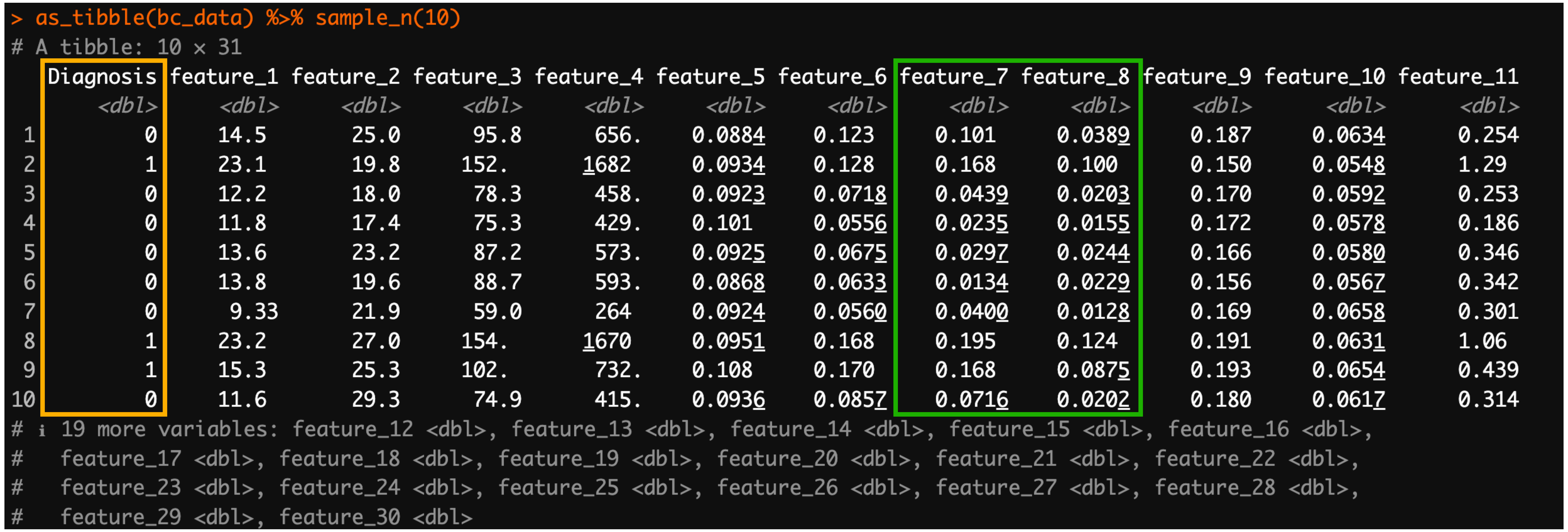
2. Transforms the data into a new and “better” coordinate system.
   A. Are there emerging patterns in the data?
   B. What variables are “important” in the new system?

3. How “good” is this new coordinate system anyway?

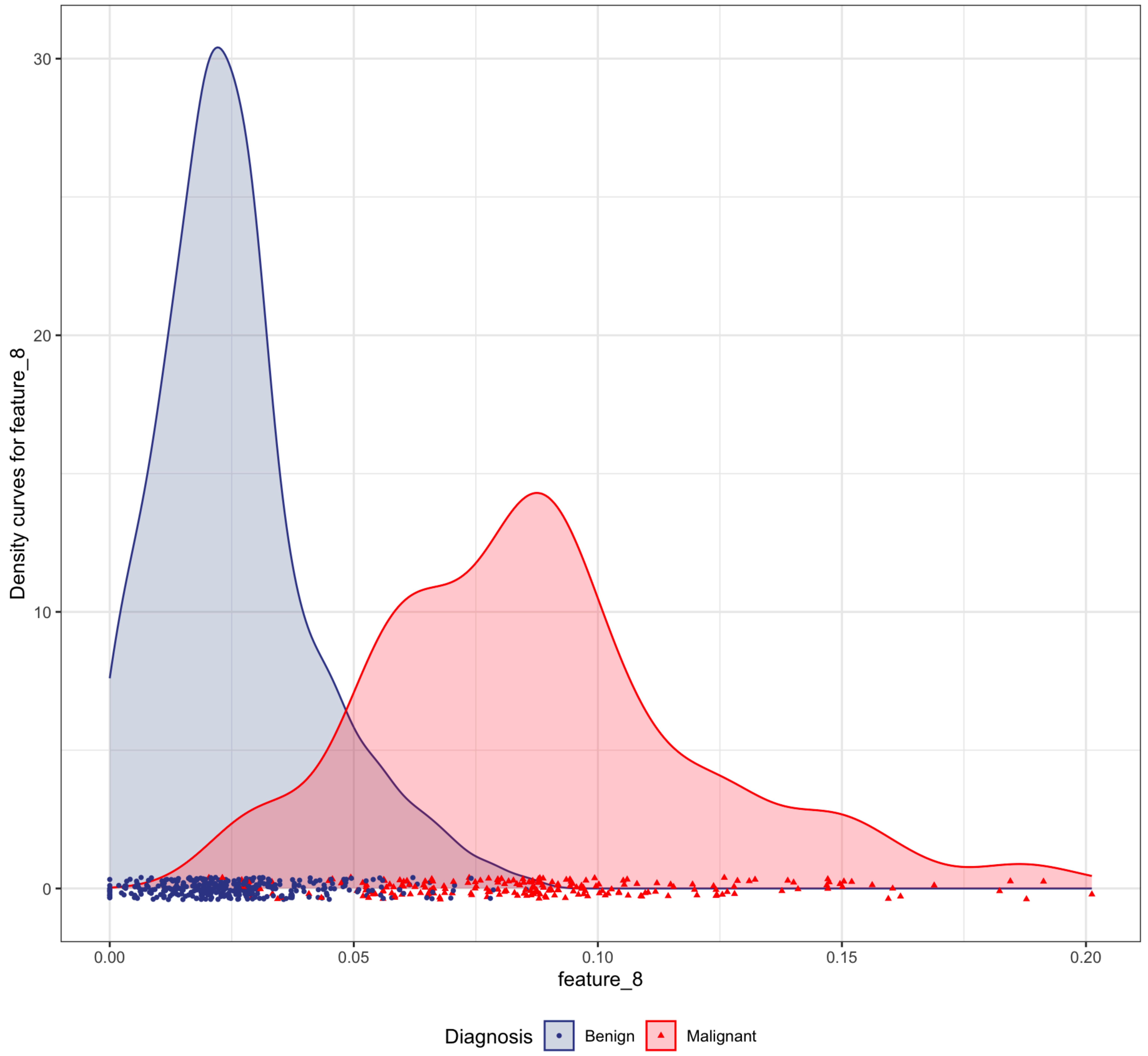
Dataset: Breast Cancer Diagnostics

Biomedical dataset with 569 patients and 30 features

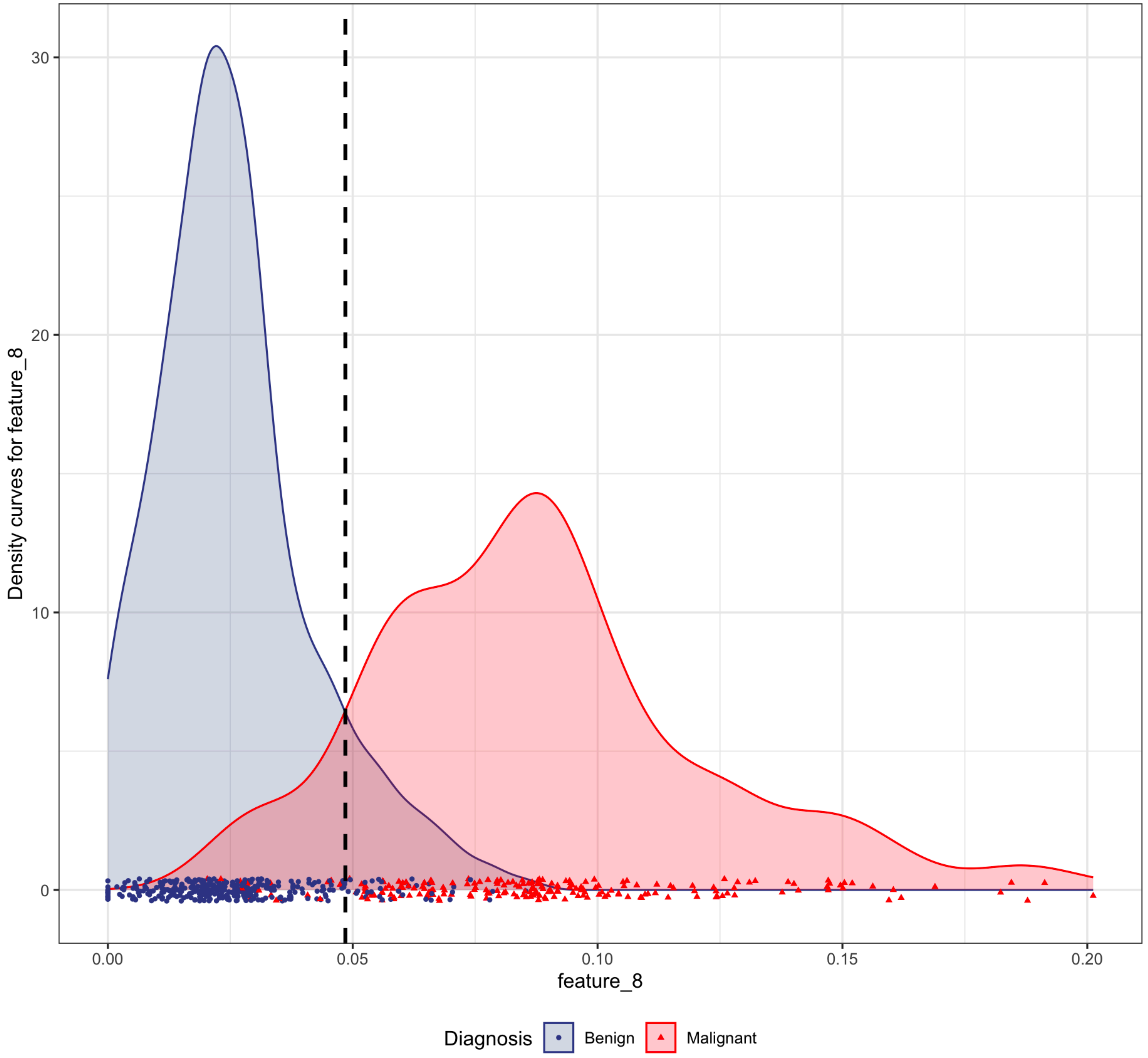
Using **30 features** classify into malignant/benign **class**



```
> as_tibble(bc_data) %>% sample_n(10)
# A tibble: 10 × 31
   Diagnosis feature_1 feature_2 feature_3 feature_4 feature_5 feature_6 feature_7 feature_8 feature_9 feature_10 feature_11
       <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>      <dbl>      <dbl>
 1         0     14.5      25.0       95.8      656.    0.0884    0.123     0.101     0.0389     0.187     0.0634      0.254
 2         1     23.1      19.8      152.      1682     0.0934    0.128     0.168     0.100      0.150     0.0548      1.29
 3         0     12.2      18.0       78.3      458.    0.0923    0.0718    0.0439    0.0203     0.170     0.0592      0.253
 4         0     11.8      17.4       75.3      429.    0.101     0.0556    0.0235    0.0155     0.172     0.0578      0.186
 5         0     13.6      23.2       87.2      573.    0.0925    0.0675    0.0297    0.0244     0.166     0.0580      0.346
 6         0     13.8      19.6       88.7      593.    0.0868    0.0633    0.0134    0.0229     0.156     0.0567      0.342
 7         0      9.33     21.9       59.0      264     0.0924    0.0560    0.0400    0.0128     0.169     0.0658      0.301
 8         1     23.2      27.0      154.      1670     0.0951    0.168     0.195     0.124      0.191     0.0631      1.06
 9         1     15.3      25.3      102.       732.    0.108     0.170     0.168     0.0875     0.193     0.0654      0.439
10         0     11.6      29.3       74.9      415.    0.0936    0.0857    0.0716    0.0202     0.180     0.0617      0.314
# ℹ 19 more variables: feature_12 <dbl>, feature_13 <dbl>, feature_14 <dbl>, feature_15 <dbl>, feature_16 <dbl>,
#   feature_17 <dbl>, feature_18 <dbl>, feature_19 <dbl>, feature_20 <dbl>, feature_21 <dbl>, feature_22 <dbl>,
#   feature_23 <dbl>, feature_24 <dbl>, feature_25 <dbl>, feature_26 <dbl>, feature_27 <dbl>, feature_28 <dbl>,
#   feature_29 <dbl>, feature_30 <dbl>
```

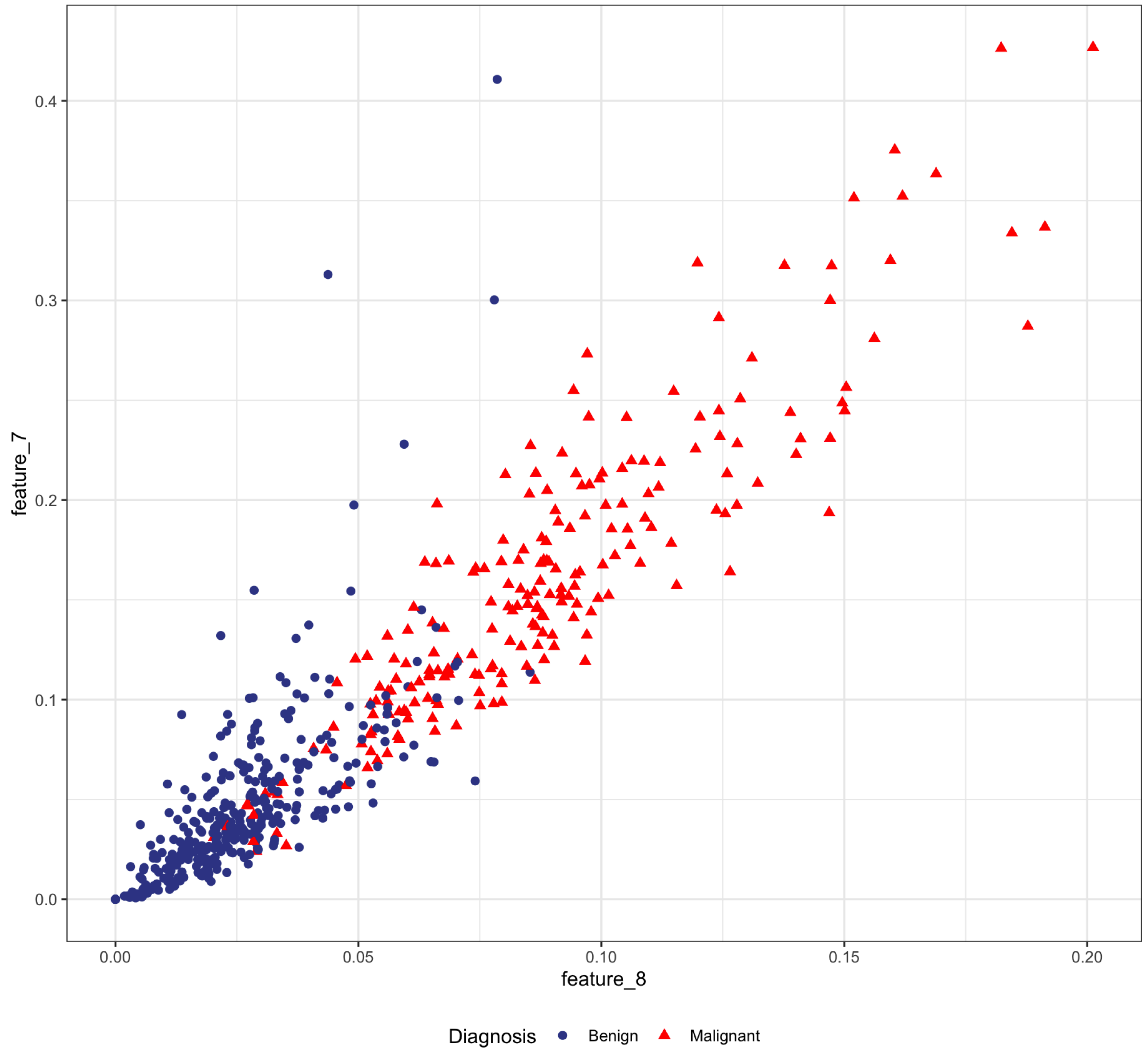


```
set.seed(48103)

as_tibble(bc_data) %>%
  mutate(Diagnosis = factor(case_when(Diagnosis == 0 ~ “B”,
                                      Diagnosis == 1 ~
“M"),
         levels = c("B", “M"),
         labels = c("Benign", "Malignant"))) %>%
  ggplot(aes(x = feature_8, color = Diagnosis)) +
  geom_density(aes(fill = Diagnosis), alpha = 0.2) +
  geom_jitter(aes(y = 0, pch = Diagnosis), size = 1) +
  scale_color_aaas() +
  scale_fill_aaas() +
  theme_bw() +
  theme(legend.position = "bottom") +
  labs(y = "Density curves for feature_8")
```

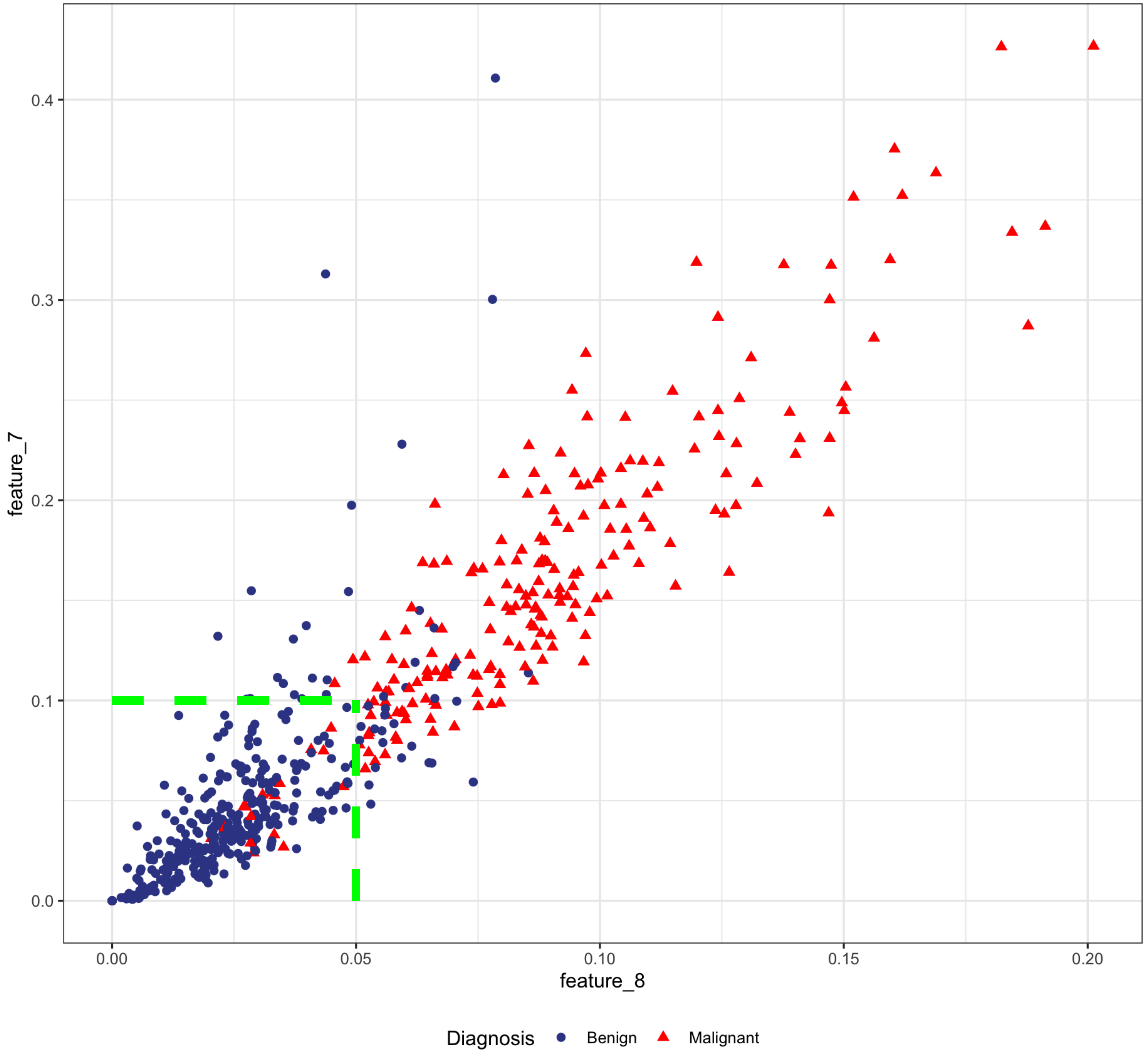


```
set.seed(48103)

as_tibble(bc_data) %>%
  mutate(Diagnosis = factor(case_when(Diagnosis == 0 ~ "B",
                                      Diagnosis == 1 ~
"M"),
         levels = c("B", "M"),
         labels = c("Benign", "Malignant"))) %>%
  ggplot(aes(x = feature_8, color = Diagnosis)) +
  geom_density(aes(fill = Diagnosis), alpha = 0.2) +
  geom_jitter(aes(y = 0, pch = Diagnosis), size = 1) +
  scale_color_aaas() +
  scale_fill_aaas() +
  theme_bw() +
  theme(legend.position = "bottom") +
  labs(y = "Density curves for feature_8") +
  geom_vline(xintercept = 0.0485, linetype = "dashed",
             size = 1, color = "black")
```

```
as_tibble(bc_data) %>%
  mutate(Diagnosis = factor(case_when(Diagnosis == 0 ~ “B”,
                                      Diagnosis == 1 ~
“M"),
         levels = c("B", “M"),
         labels = c("Benign", "Malignant"))) %>%
  ggplot(aes(x = feature_8, y = feature_7,
             color = Diagnosis)) +
  geom_point(aes(pch = Diagnosis), size = 2) +
  scale_color_aaas() +
  scale_fill_aaas() +
  theme_bw() +
  theme(legend.position = "bottom") +
  labs(y = "feature_7", x = "feature_8")
```

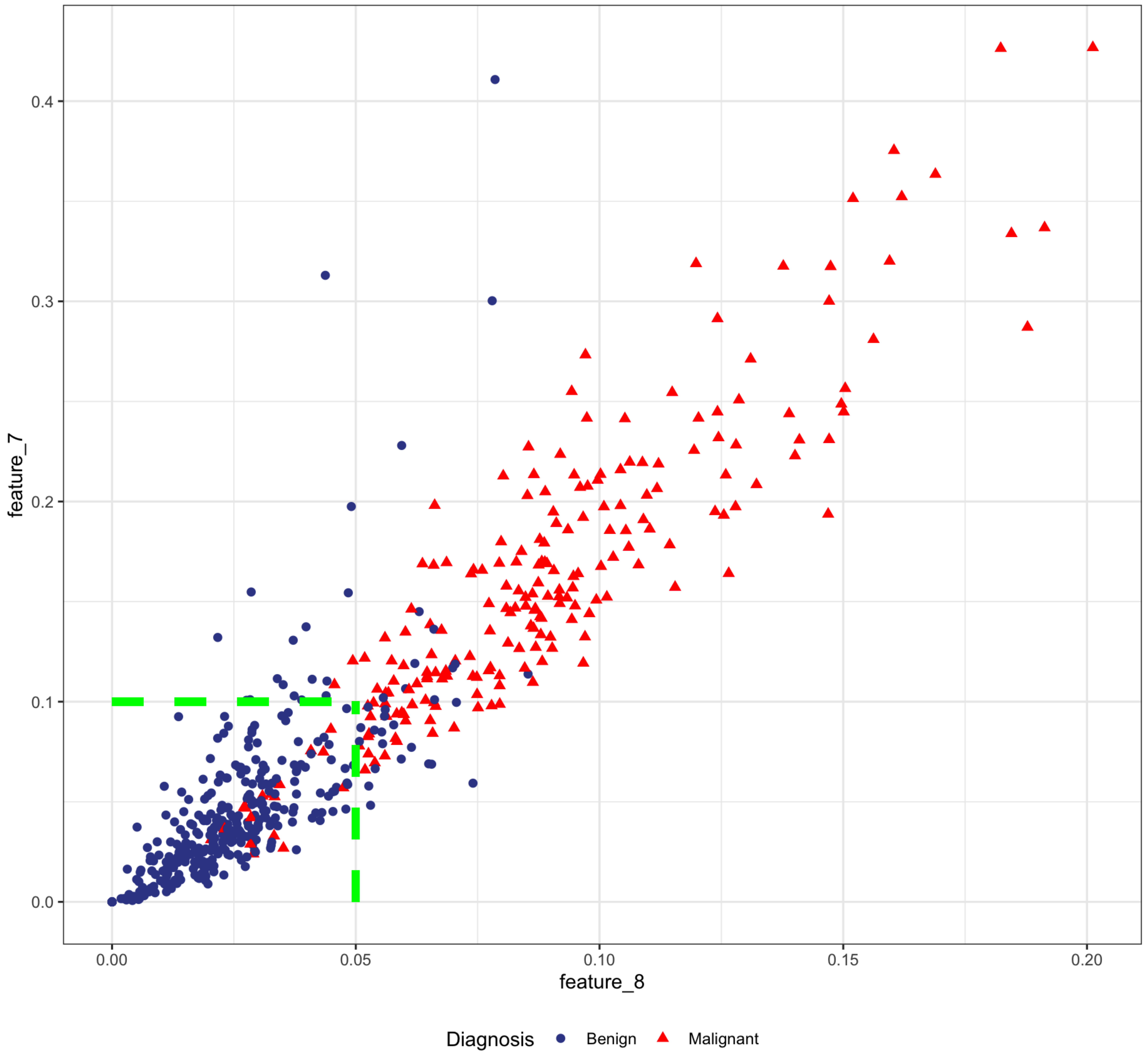


```
as_tibble(bc_data) %>%
  mutate(Diagnosis = factor(case_when(Diagnosis == 0 ~ "B",
                                      Diagnosis == 1 ~
"M"),
         levels = c("B", "M"),
         labels = c("Benign", "Malignant"))) %>%
  ggplot(aes(x = feature_8, y = feature_7,
             color = Diagnosis)) +
  geom_point(aes(pch = Diagnosis), size = 2) +
  scale_color_aaas() +
  scale_fill_aaas() +
  theme_bw() +
  theme(legend.position = "bottom") +
  labs(y = "feature_7", x = "feature_8") +
 geom_segment(aes(x = 0, xend = 0.05, y = 0.1, yend = 0.1),
              color = "green", size = 2,
              linetype = "dashed") +
 geom_segment(aes(x = 0.05, xend = 0.05, y = 0, yend =
0.1),
              color = "green", size = 2,
              linetype = "dashed")
```



```
as_tibble(bc_data) %>%
  mutate(Diagnosis = factor(case_when(Diagnosis == 0 ~ "B",
                                       Diagnosis == 1 ~
"M"),
         levels = c("B", "M"),
         labels = c("Benign", "Malignant"))) %>%
  ggplot(aes(x = feature_8, y = feature_7,
             color = Diagnosis)) +
  geom_point(aes(pch = Diagnosis), size = 2) +
  scale_color_aaas() +
  scale_fill_aaas() +
  theme_bw() +
  theme(legend.position = "bottom") +
  labs(y = "feature_7", x = "feature_8") +
 geom_segment(aes(x = 0, xend = 0.05, y = 0.1, yend = 0.1),
              color = "green", size = 2,
              linetype = "dashed") +
 geom_segment(aes(x = 0.05, xend = 0.05, y = 0, yend =
0.1),
              color = "green", size = 2,
              linetype = "dashed")
```

More than three dimensions? :(

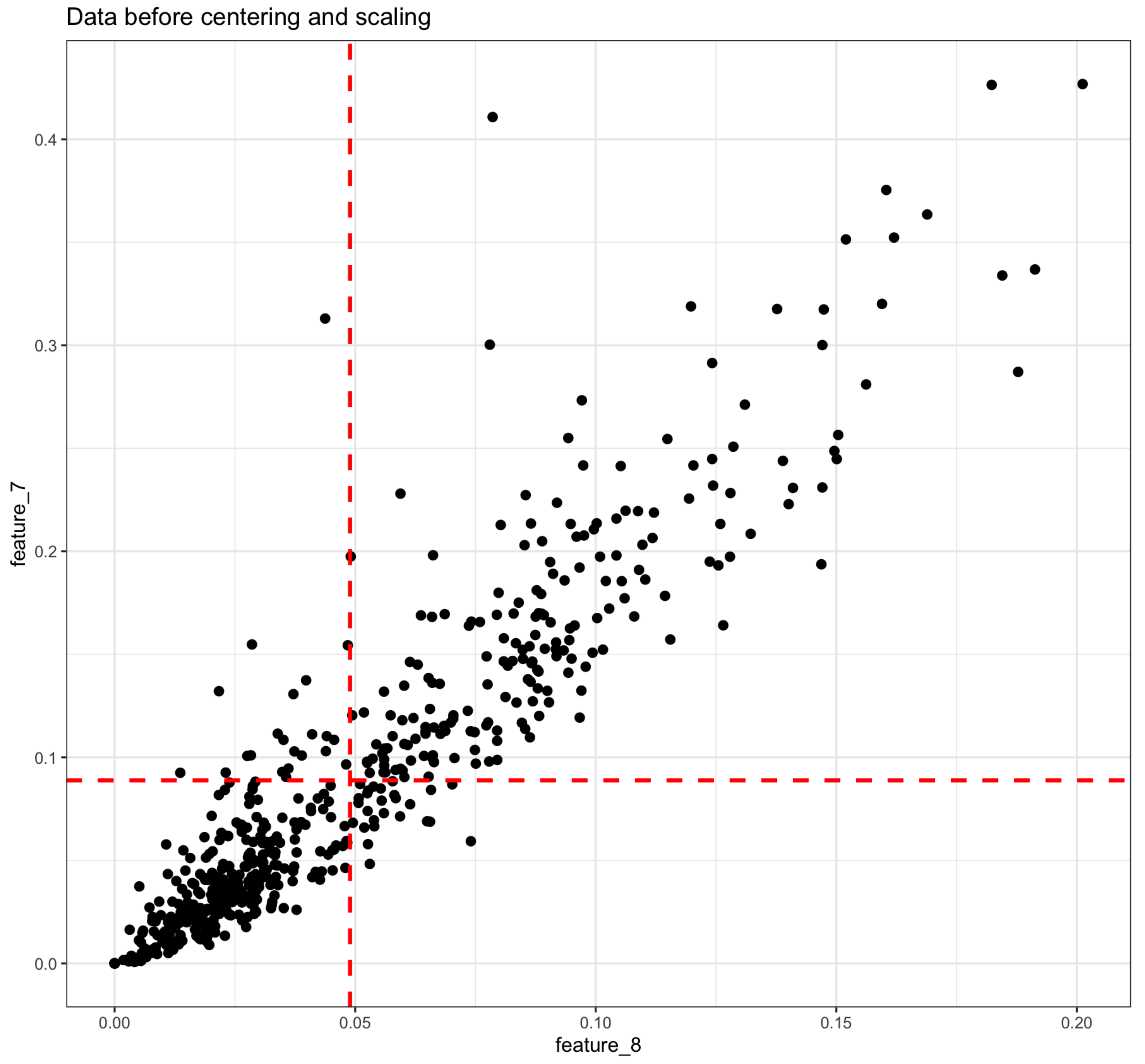


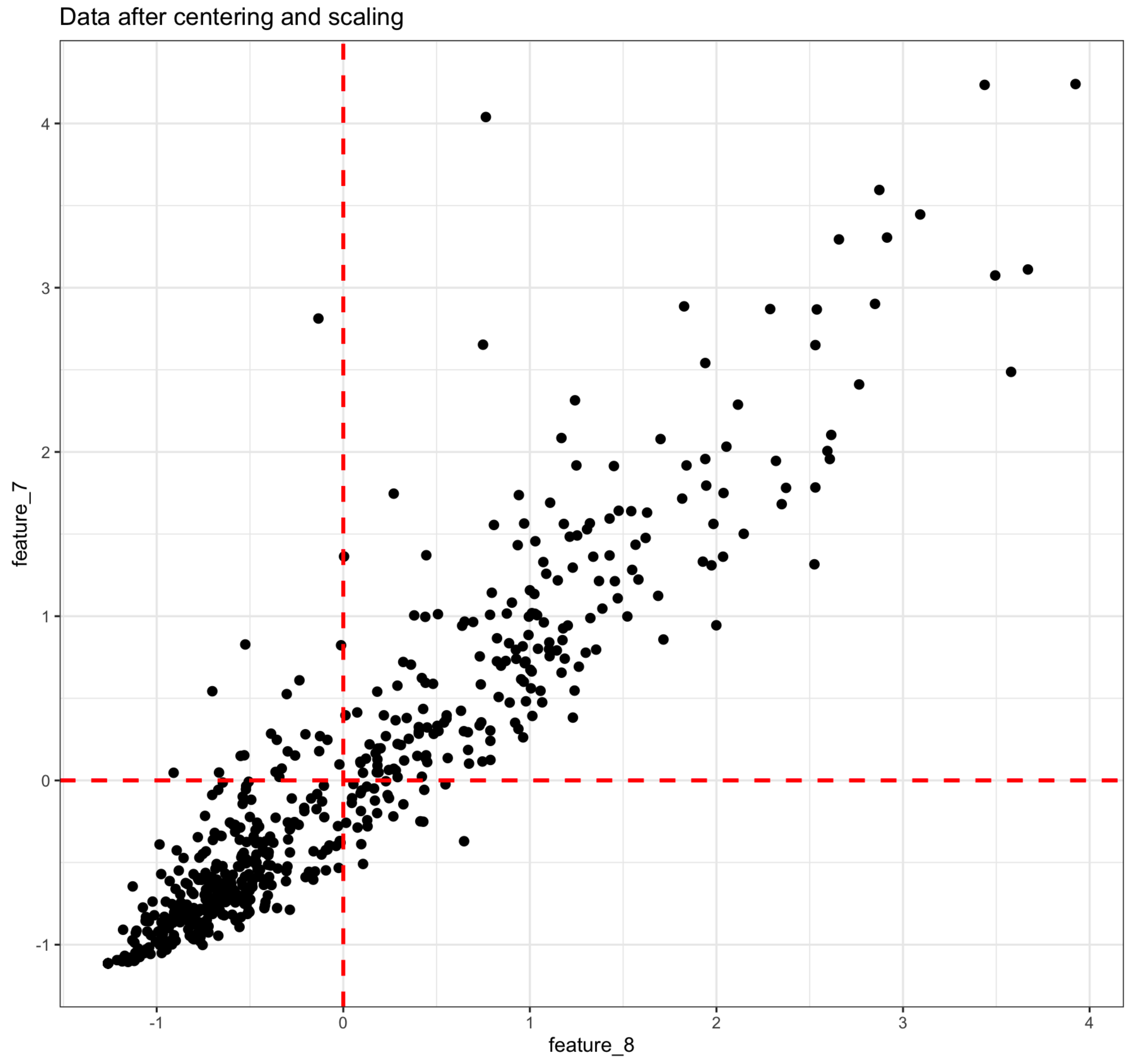
PCA will help us “reduce dimensionality”

1. **Do “similar” patients cluster together? (Benign/malignant)**
2. **Which original variable(s) are most useful when forming clusters?**
3. **How reliable is this new PCA approach?**

PCA in two variables: step 1

Make life easier: center and scale





Intuition: Data points are in the same "relative" position as before, should not hurt clustering

Returning to…

Introduction to PCA

1. Statistical technique used for dimensionality reduction.

2. **Transforms the data into a new and “better” coordinate system.**
   A. Are there emerging patterns in the data?
   B. What variables are “important” in the new system?

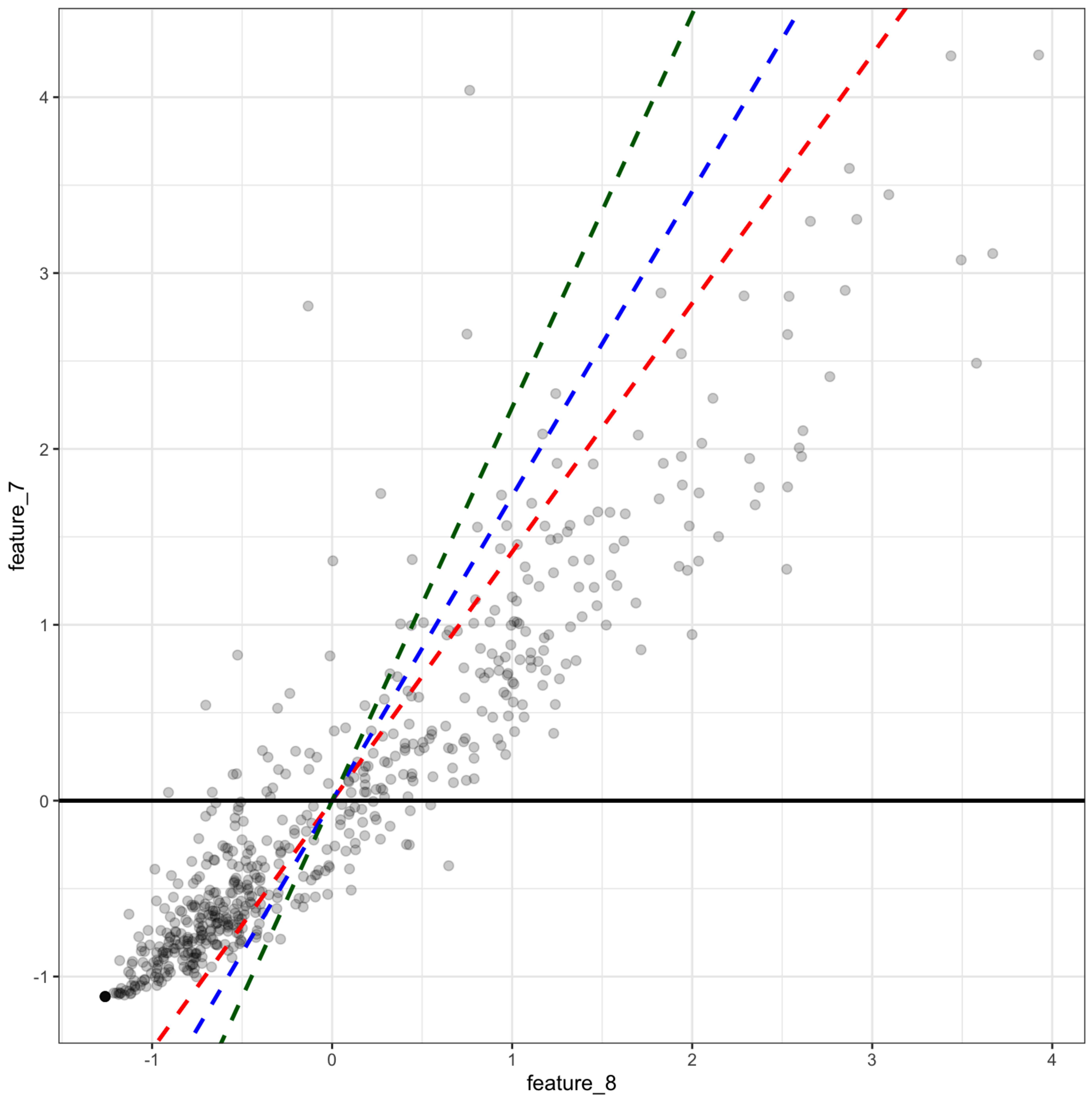
3. How “good” is this new coordinate system anyway?

PCA in two variables: step 2

Create “new coordinate system”

Must create a new pair of axes.
Which axes to pick?

Which coordinate system is the best?



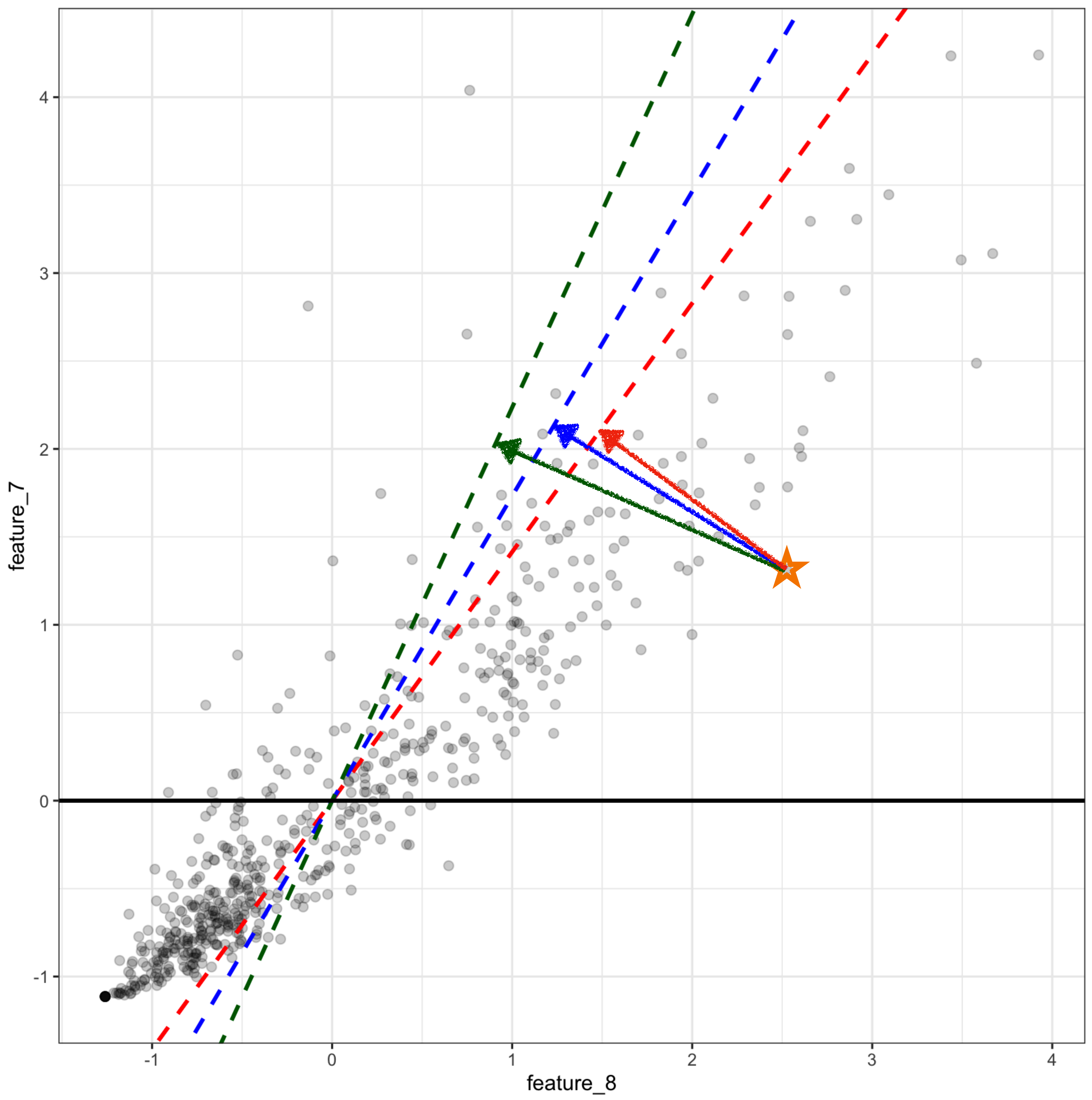
PCA in two variables: step 2

Create “new coordinate system”

Must create a new pair of axes.
Which axes to pick?

Fix a point ★ and calculate perpendicular distance of point from a given line

Which coordinate system is the best?



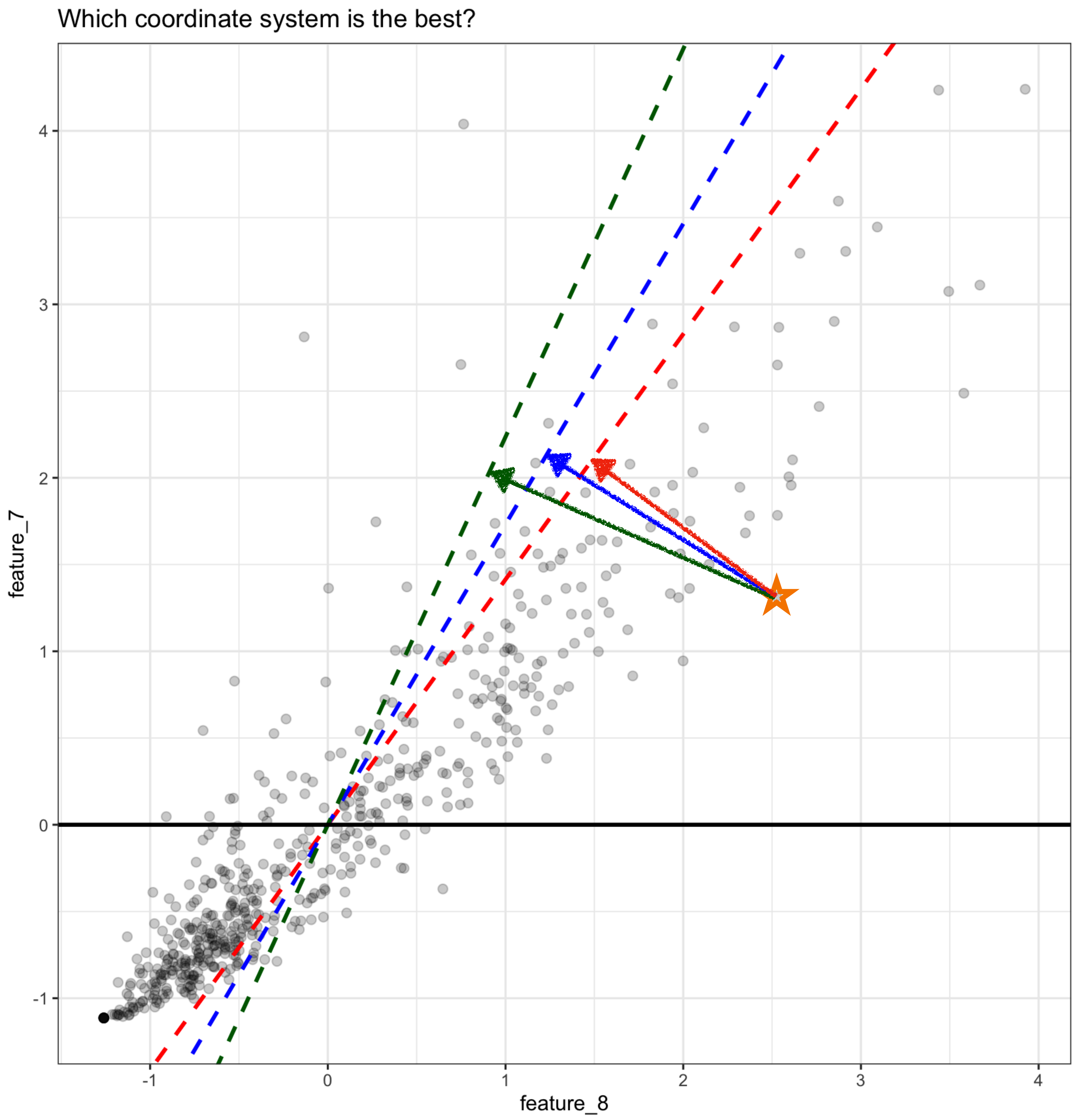
PCA in two variables: step 2

Create “new coordinate system”

Must create a new pair of axes.
Which axes to pick?

Fix a point ★ and calculate perpendicular distance of point from a given line

Pick that line which has the **least distance** from all points (not just ★) to the line.



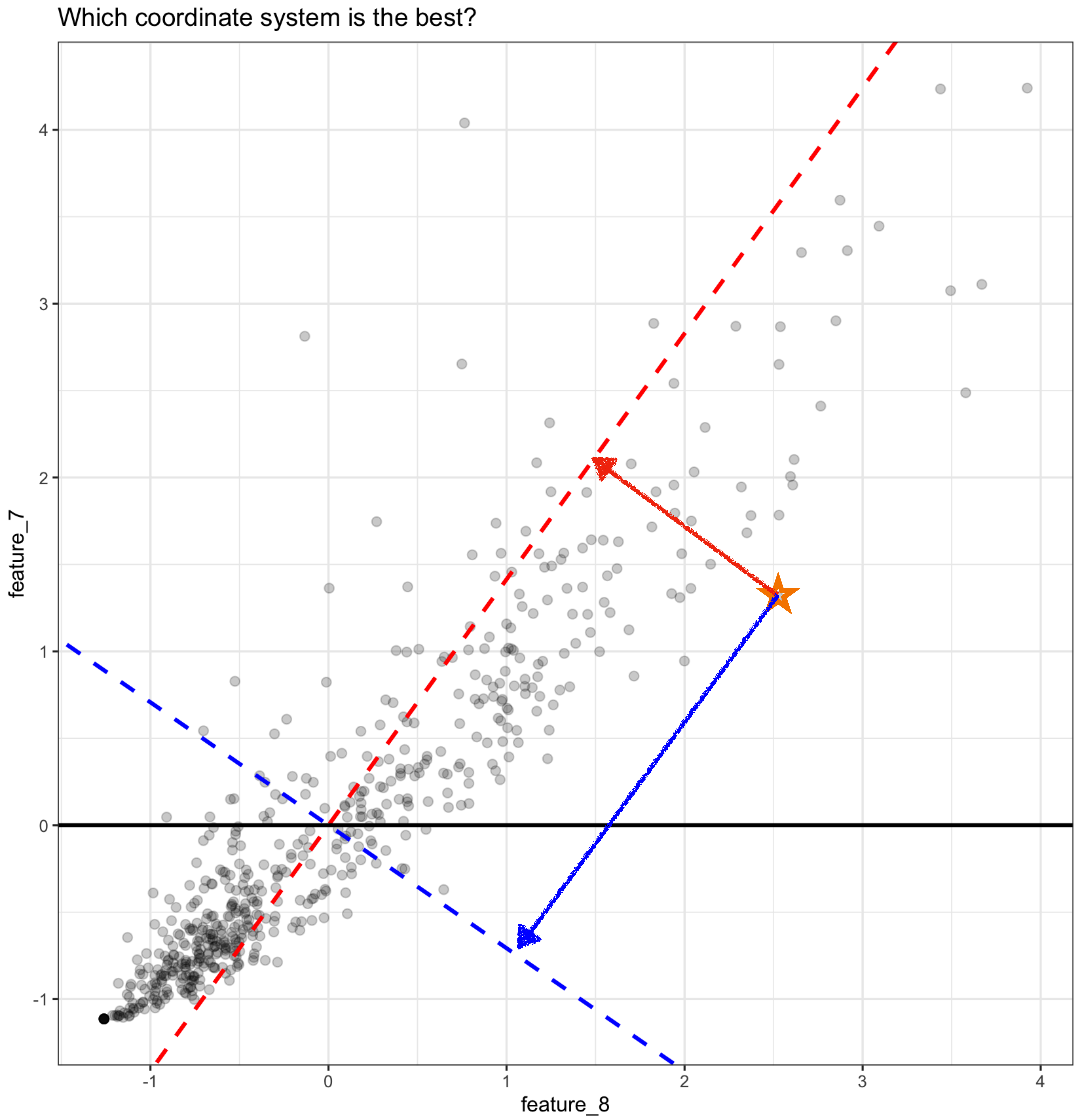
PCA in two variables: step 3

Complete "new coordinate system"

The first "good" line is the first PC, or PC_1.

Repeat the line-finding process again, excluding PC_1 to obtain PC_2.

PC_2 will **<u>ALWAYS</u>** be perpendicular to PC_1



PCA in two variables: step 3

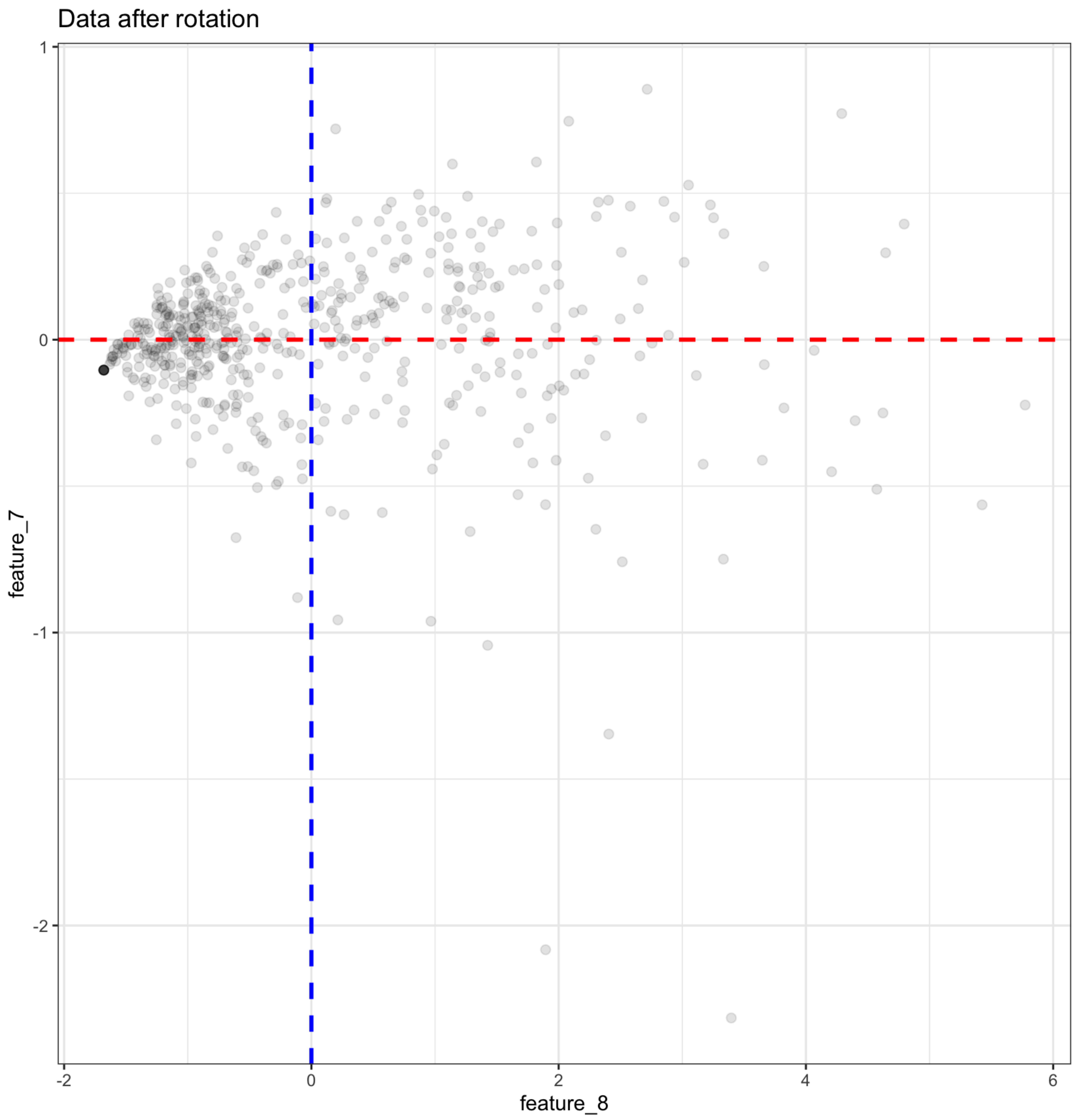
Complete "new coordinate system"

The first "good" line is the first PC, or PC_1.

Repeat the line-finding process again, excluding PC_1 to obtain PC_2.

PC_2 will <u>**ALWAYS**</u> be perpendicular to PC_1

Finally: rotate to avoid neck pain :)



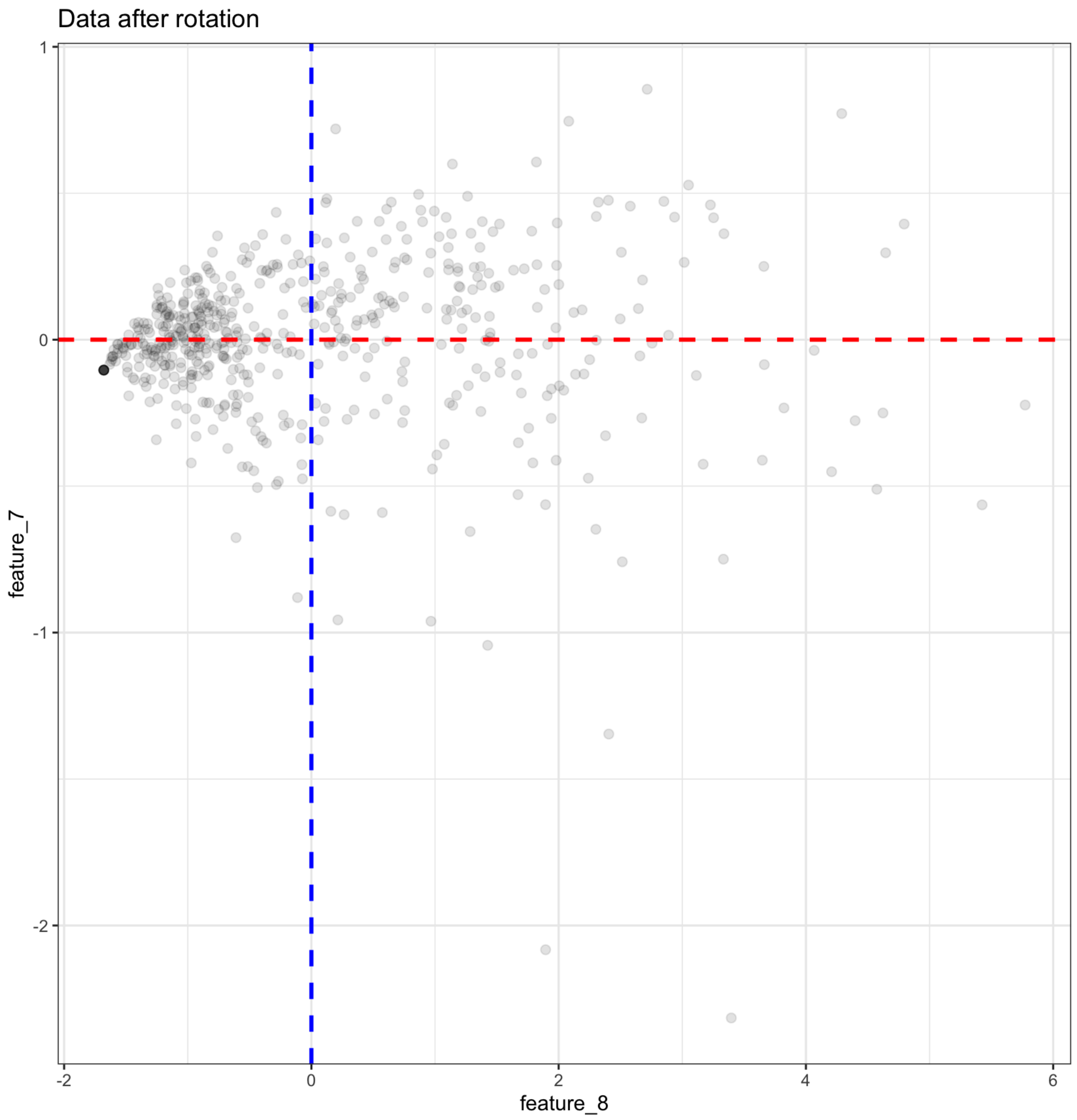
Okay, got the PCs

Now what??

PC_1 and PC_2 identify the directions along which the variation in the data is maximal.

Reduces the dimensionality of the data while retaining most of the variation.

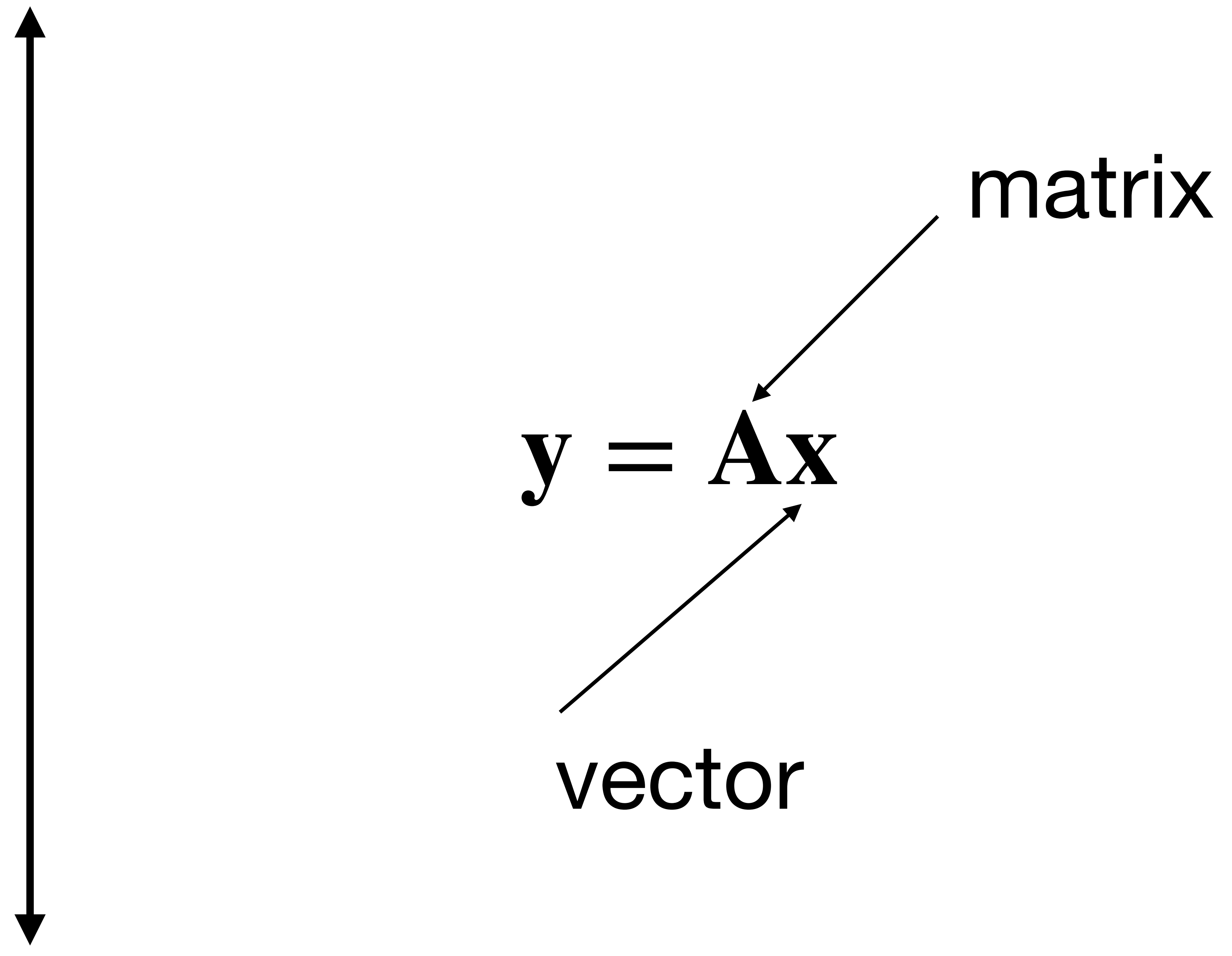
We just used **eigenthings** to find PCs!



Eigenthings of matrices

Eigenvalues and eigenvectors

Multiplying vector $\mathbf{x}$ with matrix $\mathbf{A}$ will give changed vector $\mathbf{y}$



$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

Linear transforming vector $\mathbf{x}$ with matrix $\mathbf{A}$ will give vector $\mathbf{y}$

Eigenthings of matrices

Eigenvalues and eigenvectors

For a square matrix $\mathbf{A}$, and a (non-zero) vector $\mathbf{x}$ where the matrix $\mathbf{A}$ is used to transform $\mathbf{x}$ to $\mathbf{y}$: $\mathbf{y} = \mathbf{A}\mathbf{x}$.

Eigenthings of matrices

Eigenvalues and eigenvectors

For a square matrix $\mathbf{A}$, and a (non-zero) vector $\mathbf{x}$ where the matrix $\mathbf{A}$ is used to transform $\mathbf{x}$ to $\mathbf{y}$: $\mathbf{y} = \mathbf{A}\mathbf{x}$.

If $\mathbf{y}$ is a scaled version of $\mathbf{x}$ ($\mathbf{y} = \lambda\mathbf{x}$ for some scalar λ), then we can say:

"$\mathbf{x}$ is an eigenvector of $\mathbf{A}$ and λ is the corresponding eigenvalue"

For a $p \times p$ square matrix we have p eigenvalue+eigenvector pairs.

Eigenthings of matrices

Eigenvalues and eigenvectors

For a square matrix $\mathbf{A}$, and a (non-zero) vector $\mathbf{x}$ where the matrix $\mathbf{A}$ is used to transform $\mathbf{x}$ to $\mathbf{y}$: $\mathbf{y} = \mathbf{A}\mathbf{x}$.

If $\mathbf{y}$ is a scaled version of $\mathbf{x}$ ($\mathbf{y} = \lambda\mathbf{x}$ for some scalar λ), then we can say:

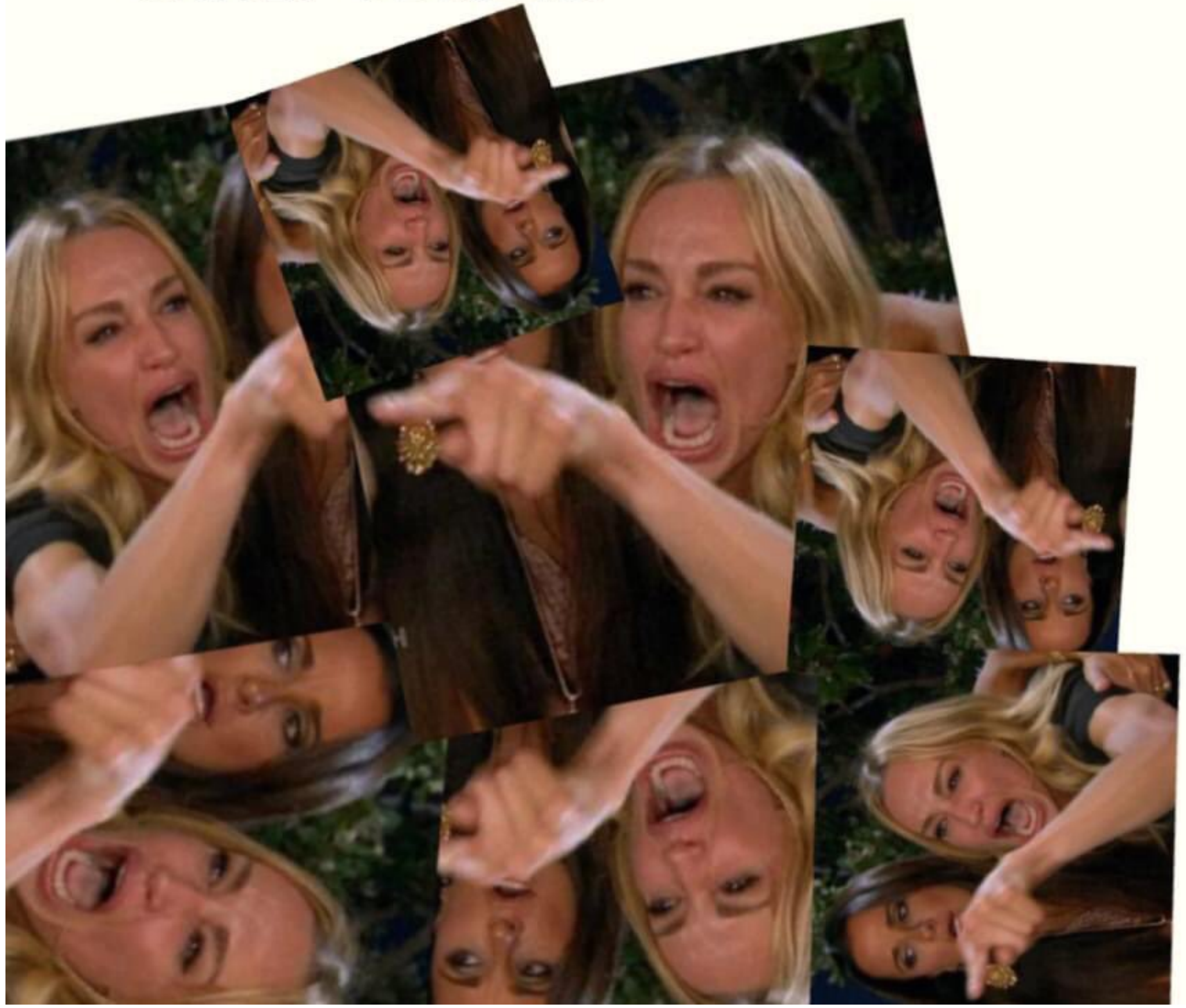
"$\mathbf{x}$ is an eigenvector of $\mathbf{A}$ and λ is the corresponding eigenvalue"

For a $p \times p$ square matrix we have p eigenvalue+eigenvector pairs.

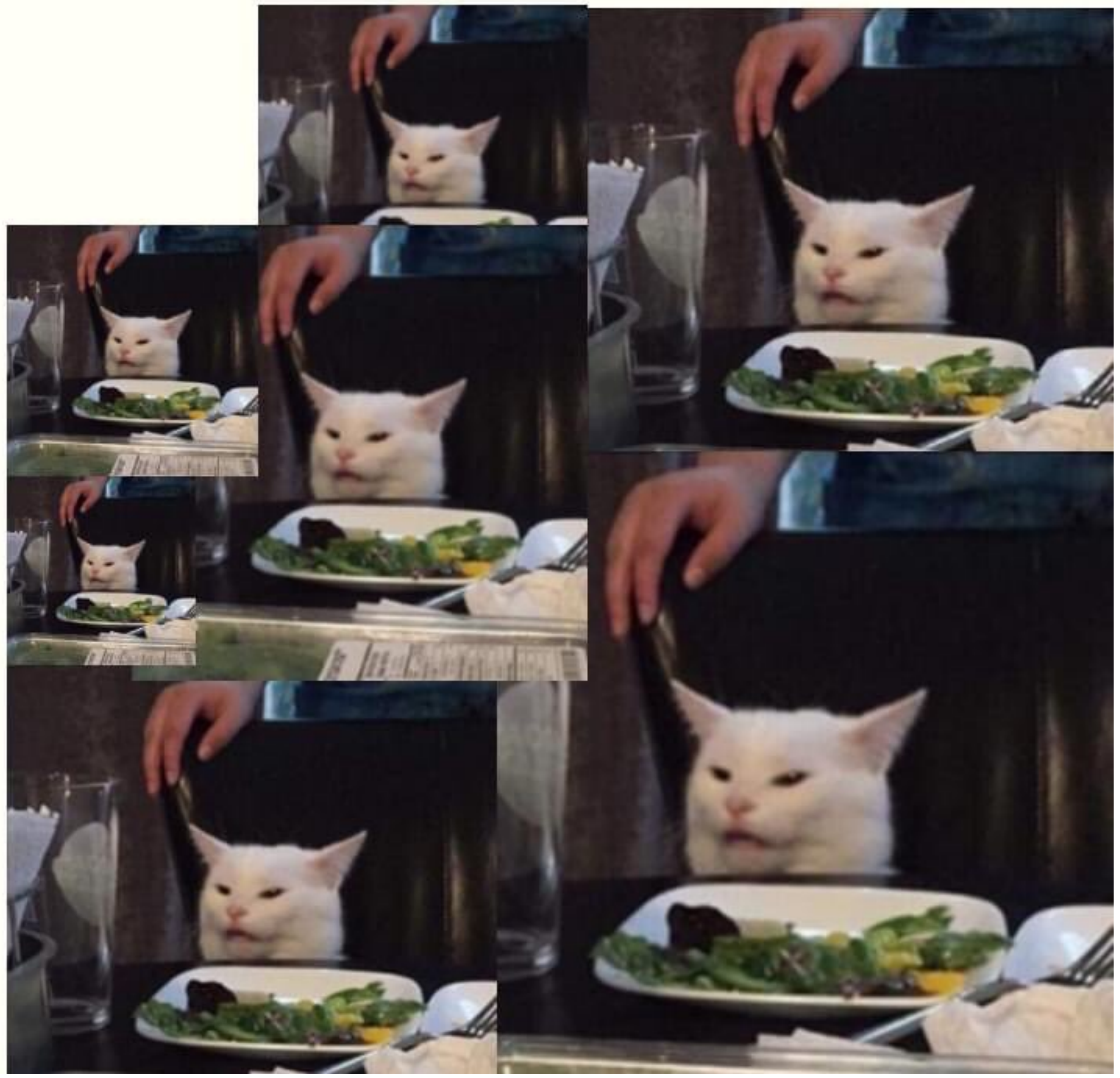
If you give me a 'good' matrix, I can give you it's eigenthings using singular value decomposition (SVD)

A specific linear transformation matrix exists

Other vectors



eigen vectors of that matrix



Okay, got the PCs

Now what??

PCs of the data are related to the
eigenthings of the correlation matrix.

Okay, got the PCs

Now what??

PCs of the data are related to the
eigenthings of the correlation matrix.

Remember we scaled and centered our data?
All means zero, all variances 1. Only correlations remain :)

Okay, got the PCs

Now what??

PCs of the data are related to the
eigenthings of the correlation matrix.

Remember we scaled and centered our data?
All means zero, all variances 1. Only correlations remain :)

PC_1 and PC_2 identify the directions along which the variation in the data is maximal.

Variance of PC_1 = largest eigenvalue λ_1 and direction of PC_1: corresponding eigenvector w_1
Variance of PC_2 = second-largest eigenvalue λ_2 and direction of PC_2: corresponding eigenvector w_2

Okay, got the PCs

Now what??

PCs of the data are related to the
eigenthings of the correlation matrix.

Remember we scaled and centered our data?
All means zero, all variances 1. Only correlations remain :)

PC_1 and PC_2 identify the directions along which the variation in the data is maximal.

Variance of PC_1 = largest eigenvalue λ_1 and direction of PC_1: corresponding eigenvector w_1
Variance of PC_2 = second-largest eigenvalue λ_2 and direction of PC_2: corresponding eigenvector w_2

All the PCs are linear combinations of the original variables.

Variable loadings of PC_1 tell us how the variables are combined linearly to form PC_1
Variable loadings tell us which variables are “more” important.

Dataset: Breast Cancer Diagnostics

Biomedical dataset with 569 patients and 30 features (not 2)

```
# Step 0/a: Drop the ID column
bc_data <- bc_data %>% select(-ID)

# Step 0/b: Encode the diagnosis labels
bc_data <- bc_data %>% mutate(Diagnosis = ifelse(Diagnosis == "M", 1, 0))

# Step 0/c: Separate features and labels
X <- bc_data %>% select(-Diagnosis)
y <- bc_data$Diagnosis
```

```
# Step 1: Standardize the bc_data
scaler <- preProcess(X, method = c("center", "scale"))
X_scaled <- predict(scaler, X)
```

```
# Step 2/a: Apply PCA
pca <- prcomp(X_scaled, center = TRUE, scale. = TRUE)

# Step 2/b: Create a dataFrame with the first two principal components
pca_df <- as_tibble(pca$x[, 1:2]) %>%
  rename(PC1 = PC1, PC2 = PC2) %>%
  mutate(Diagnosis = y)
```

We now have everything we need to answer PCA questions from our dataset :)

Dataset: Breast Cancer Diagnostics

Biomedical dataset with 569 patients and 30 features (not 2)

```
# Step 0/a: Drop the ID column
bc_data <- bc_data %>% select(-ID)

# Step 0/b: Encode the diagnosis labels
bc_data <- bc_data %>% mutate(Diagnosis = ifelse(Diagnosis == "M", 1, 0))

# Step 0/c: Separate features and labels
X <- bc_data %>% select(-Diagnosis)
y <- bc_data$Diagnosis
```

PCA will help us "reduce dimensionality"

```
# Step 1: Standardize the bc_data
scaler <- preProcess(X, method = c("center", "scale"))
X_scaled <- predict(scaler, X)
```

We now have everything we need to answer PCA questions from our dataset :)

```
# Step 2/a: Apply PCA
pca <- prcomp(X_scaled, center = TRUE, scale. = TRUE)

# Step 2/b: Create a dataFrame with the first two principal components
pca_df <- as_tibble(pca$x[ [illegible]
  rename(PC1 = PC1, PC2 = PC2) %>%
  mutate(Diagnosis = y [illegible]
```

1. **Do "similar" patients cluster together? (Benign/malignant)**
2. **Which original variable(s) are most useful when forming clusters?**
3. **How reliable is this new PCA approach?**

Dataset: Breast Cancer Diagnostics

Biomedical dataset with 569 patients and 30 features (not 2)

Big question: can we find clusters of “similar” patients using PCA?

Similar? Diagnosis of breast cancer.

PCA questions:

1. How many features? How many PCs?
2. Which are the “best” PCs?
3. Which are the “important” variables forming the “best” PCs.

Dataset: Breast Cancer Diagnostics

Biomedical dataset with 569 patients and 30 features (not 2)

Big question: can we find clusters of “similar” patients using PCA?

Similar? Diagnosis of breast cancer.

PCA questions:

1. How many features? How many PCs?
2. Which are the “best” PCs? [larger eigenvalues = better PC]
3. Which are the “important” variables forming the “best” PCs. [eigenvectors of a PC give variable loadings]

Dataset: Breast Cancer Diagnostics

Biomedical dataset with 569 patients and 30 features (not 2)

1. Correlation matrix used. Dimension is 30×30, so we have 30 PCs.

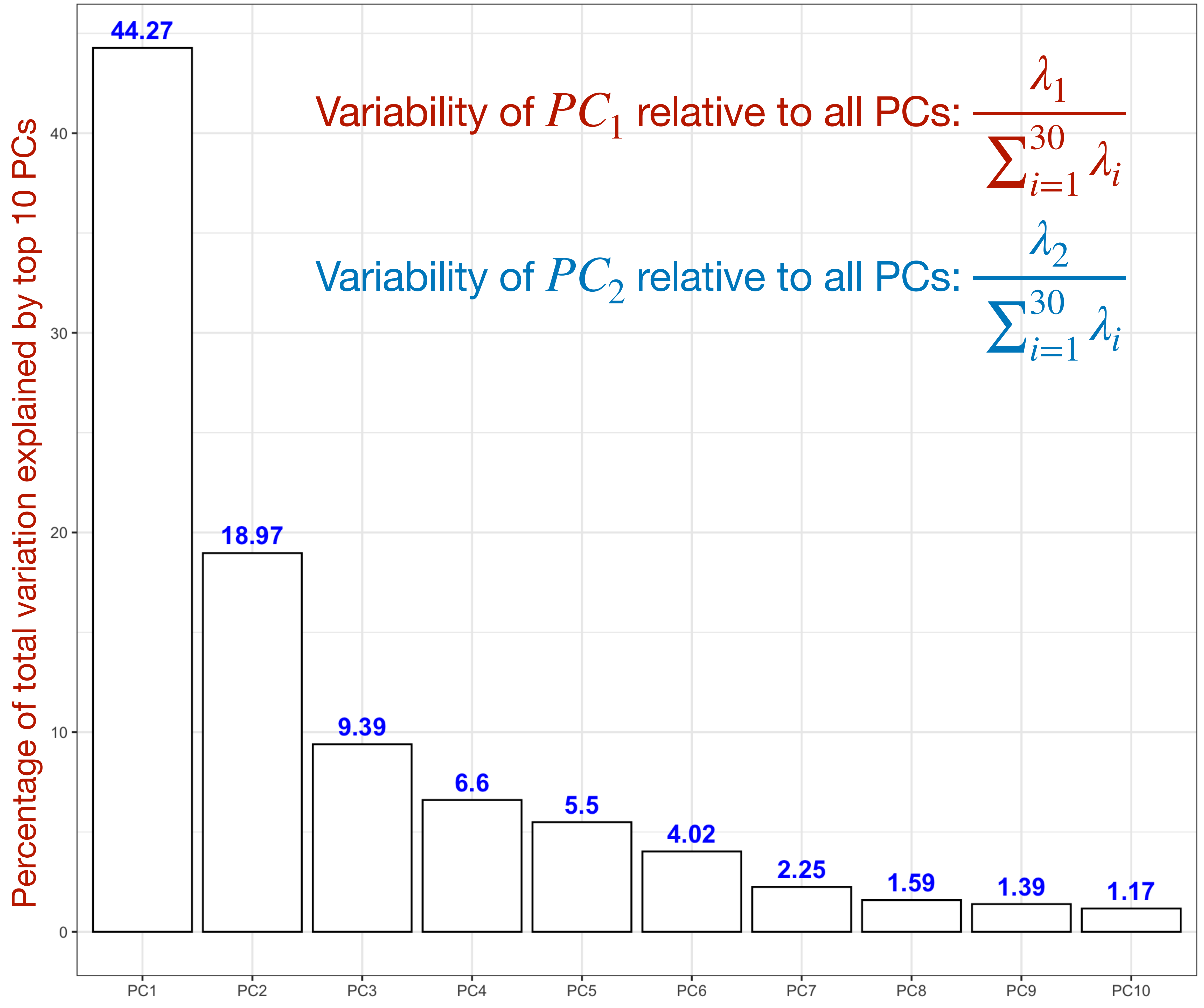
2. We use relative variability to judge PCs. Recall Var$(PC_1) = \lambda_1$.

Variability of PC_1 relative to all PCs: $\dfrac{\lambda_1}{\sum_{i=1}^{30} \lambda_i}$

Dataset: Breast Cancer Diagnostics

Biomedical dataset with 569 patients and 30 features (not 2)

```
vars <- as_tibble(paste0("PC", seq(1:30))) %>%
  mutate(var = 100*(pca$sdev^2)/sum(pca$sdev^2)) %>%
  mutate(value = factor(value,
  levels = paste0("PC", seq(1:30))))

vars %>%
  head(10) %>%
  ggplot(aes(x = value, y = var)) +
  geom_bar(stat = "identity", fill = "white", color = "black") +
  theme_bw() +
  labs(x = "", y = "") +
  geom_text(aes(label = round(var, 2)), vjust = -0.5,
  color = "blue", fontface = "bold", size = 5)
```



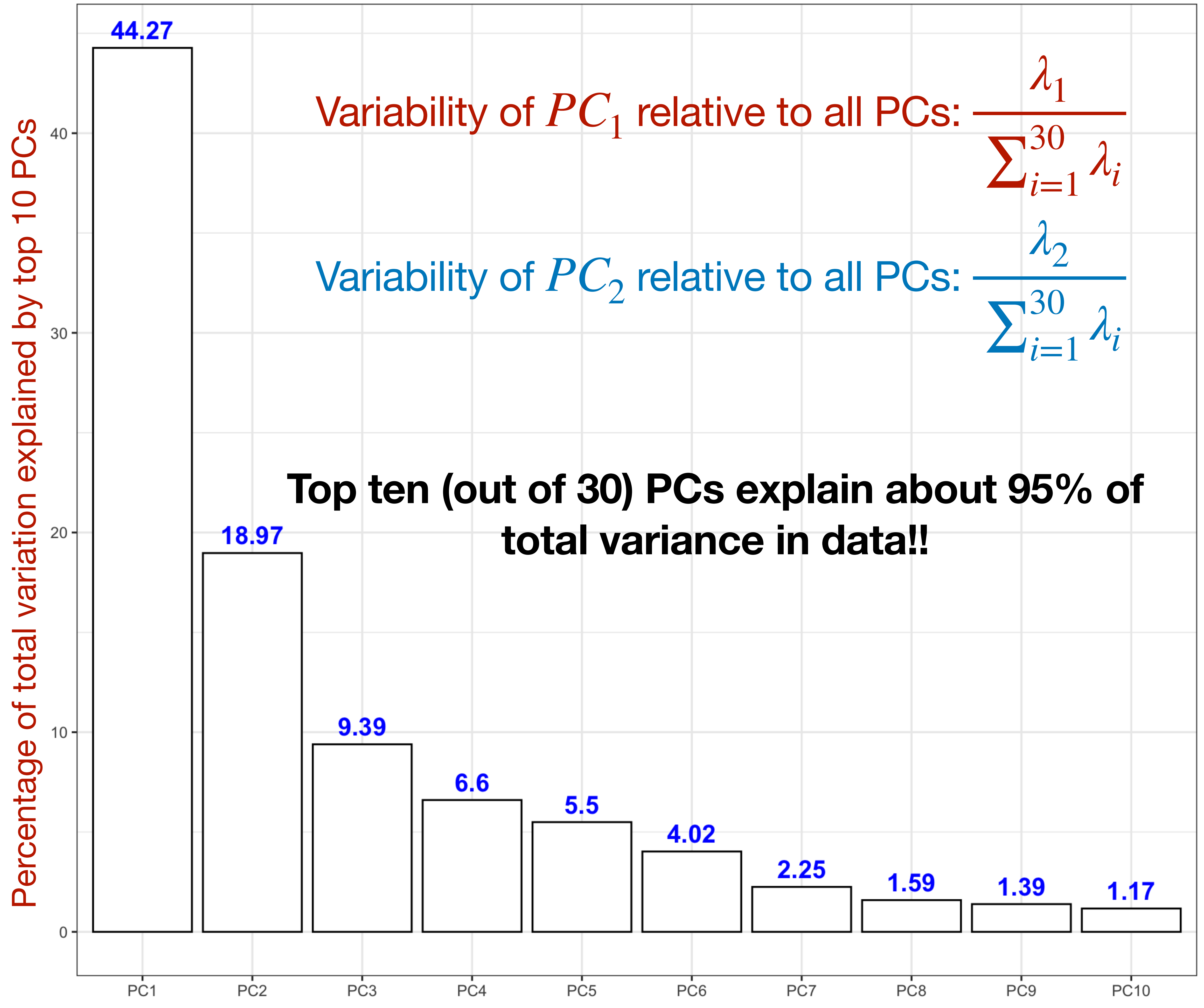
Variability of PC_1 relative to all PCs: $\frac{\lambda_1}{\sum_{i=1}^{30} \lambda_i}$

Variability of PC_2 relative to all PCs: $\frac{\lambda_2}{\sum_{i=1}^{30} \lambda_i}$

Dataset: Breast Cancer Diagnostics

Biomedical dataset with 569 patients and 30 features (not 2)

```
vars <- as_tibble(paste0("PC", seq(1:30))) %>%
  mutate(var = 100*(pca$sdev^2)/sum(pca$sdev^2)) %>%
  mutate(value = factor(value,
  levels = paste0("PC", seq(1:30))))

vars %>%
  head(10) %>%
  ggplot(aes(x = value, y = var)) +
  geom_bar(stat = "identity", fill = "white", color = "black") +
  theme_bw() +
  labs(x = "", y = "") +
  geom_text(aes(label = round(var, 2)), vjust = -0.5,
  color = "blue", fontface = "bold", size = 5)
```



Variability of PC_1 relative to all PCs: $\frac{\lambda_1}{\sum_{i=1}^{30} \lambda_i}$

Variability of PC_2 relative to all PCs: $\frac{\lambda_2}{\sum_{i=1}^{30} \lambda_i}$

Top ten (out of 30) PCs explain about 95% of total variance in data!!

Dataset: Breast Cancer Diagnostics

Biomedical dataset with 569 patients and 30 features (not 2)

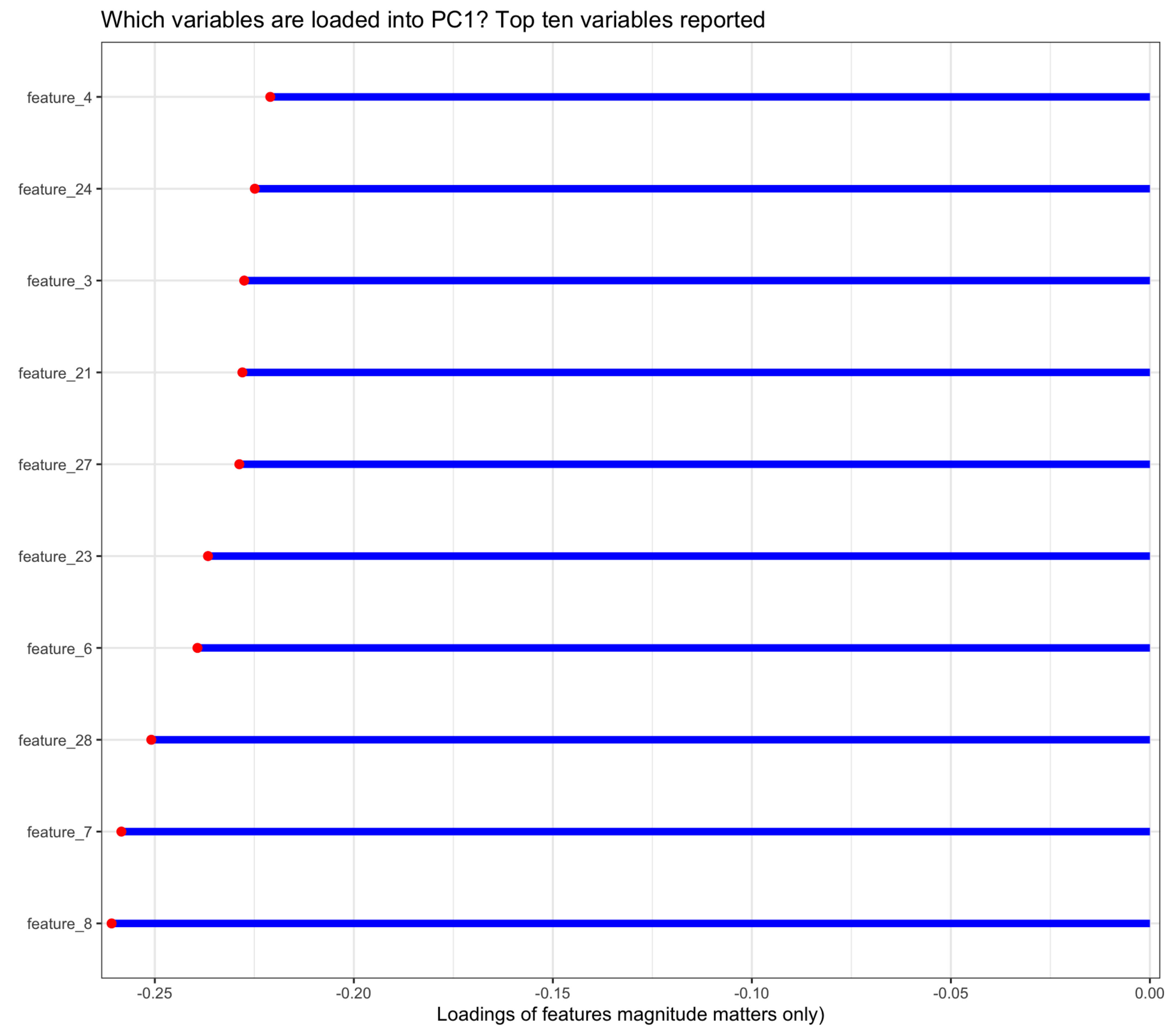
The eigenvector corresponding to PC_1 gives us variable loadings:

$$PC_1 = \boxed{0.33}X_1 + \boxed{0.71}X_2 + \boxed{0.62}X_3$$

X_2 **most important**

Dataset: Breast Cancer Diagnostics

Biomedical dataset with 569 patients and 30 features (not 2)

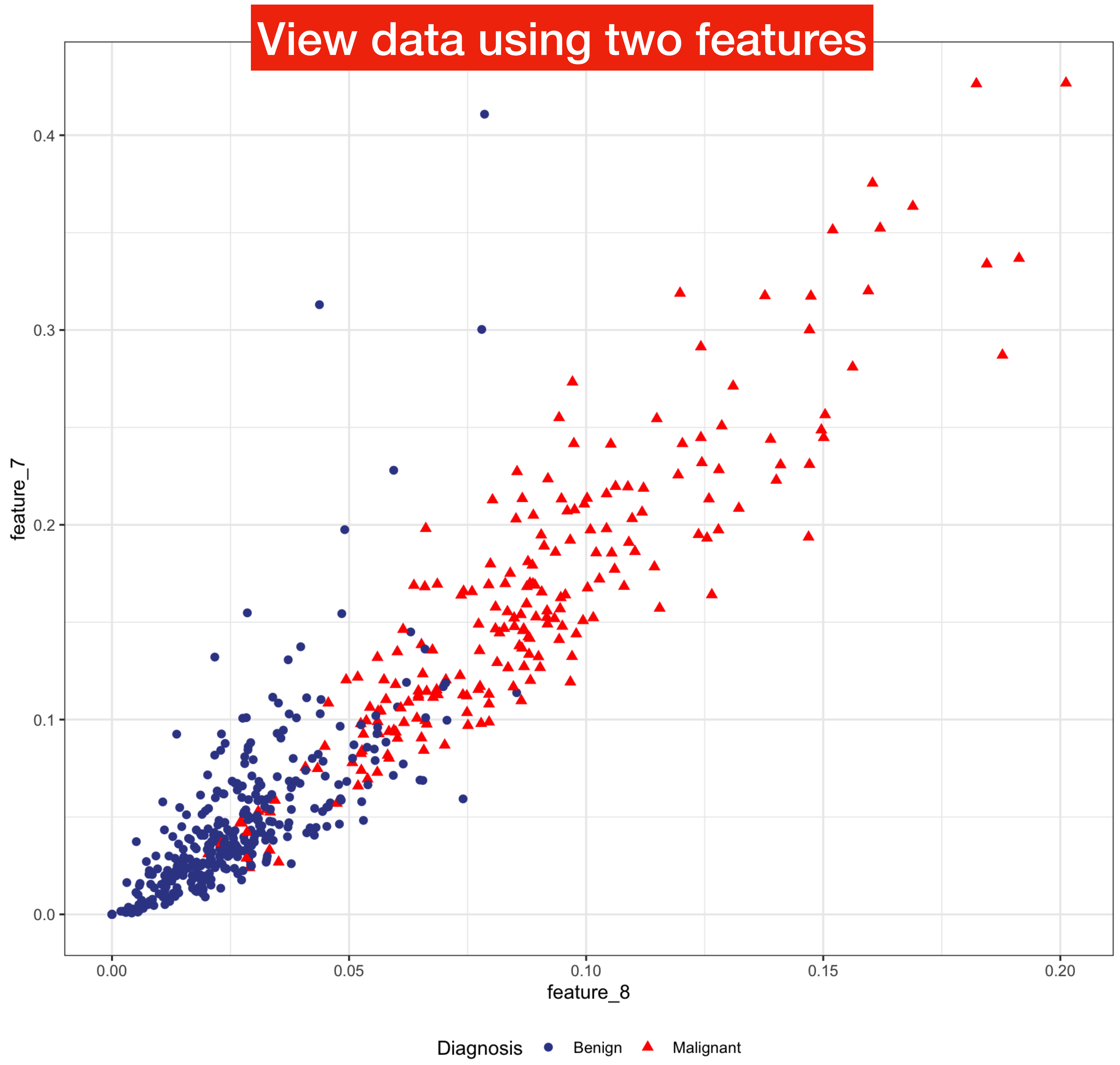


Features 7 and 8 were loaded the most for the PC with maximum variability (44% of total)

```
loadings <- as_tibble(paste0("feature_", seq(1:30))) %>%
  mutate(loadings = pca$rotation[,1]) %>%
  arrange(loadings)

loadings <- loadings %>% mutate(value = factor(value, temp$value))
loadings %>% head(10) %>%
  ggplot() +
  geom_segment(aes(x = value, y = 0, xend = value, yend = loadings),
  color = "blue", size = 2) +
  geom_point(aes(x = value, y = loadings), color = "red", size = 2) +
  theme_bw() +
  labs(x = "", y = "Loadings of features magnitude matters only)",
       title = "Which variables are loaded into PC1? Top ten variables reported") +
  coord_flip() +
  scale_y_continuous(expand = c(0.01, 0))
```

Dataset: Breast Cancer Diagnostics

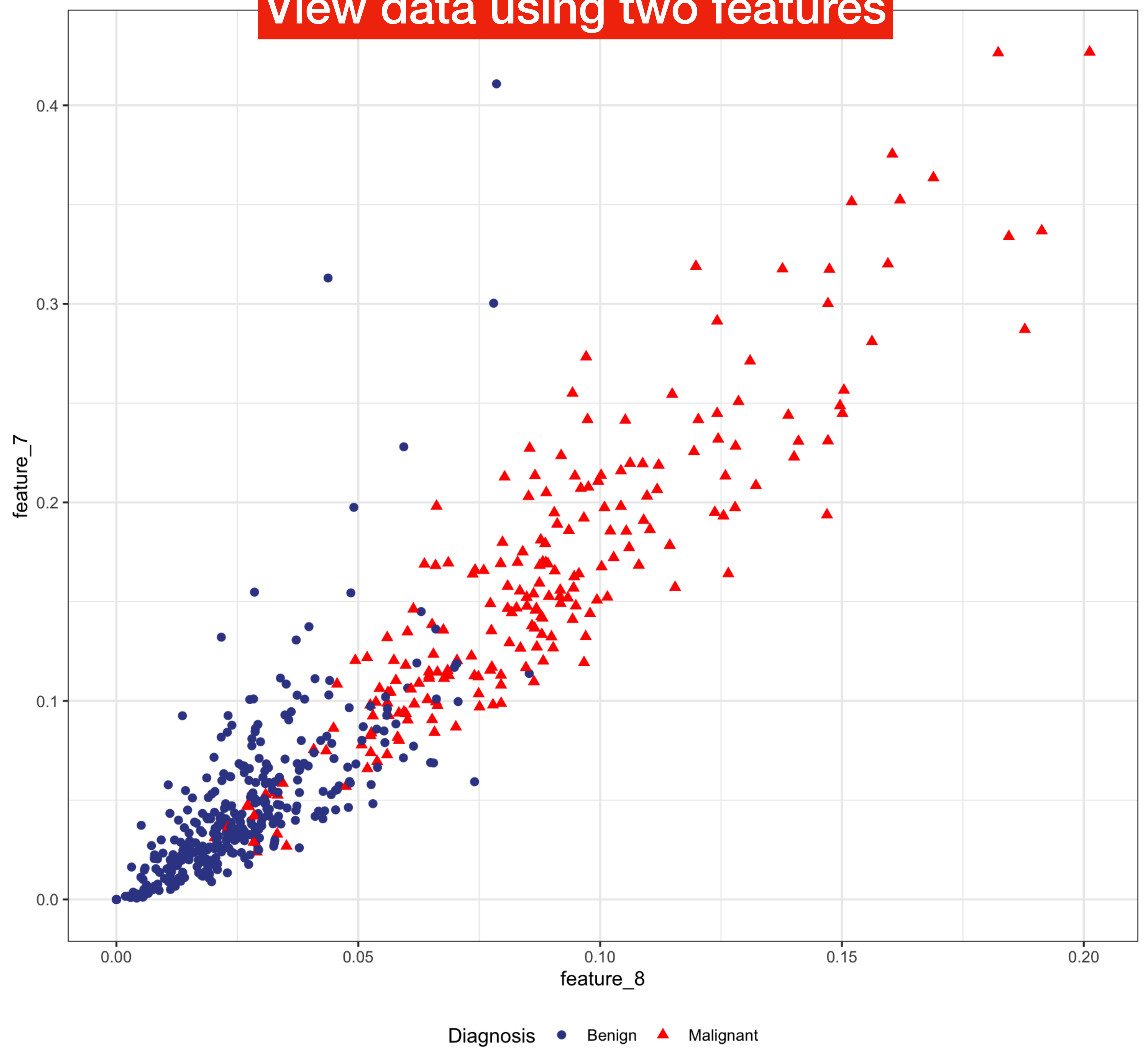
Biomedical dataset with 569 patients and 30 features (not 2)

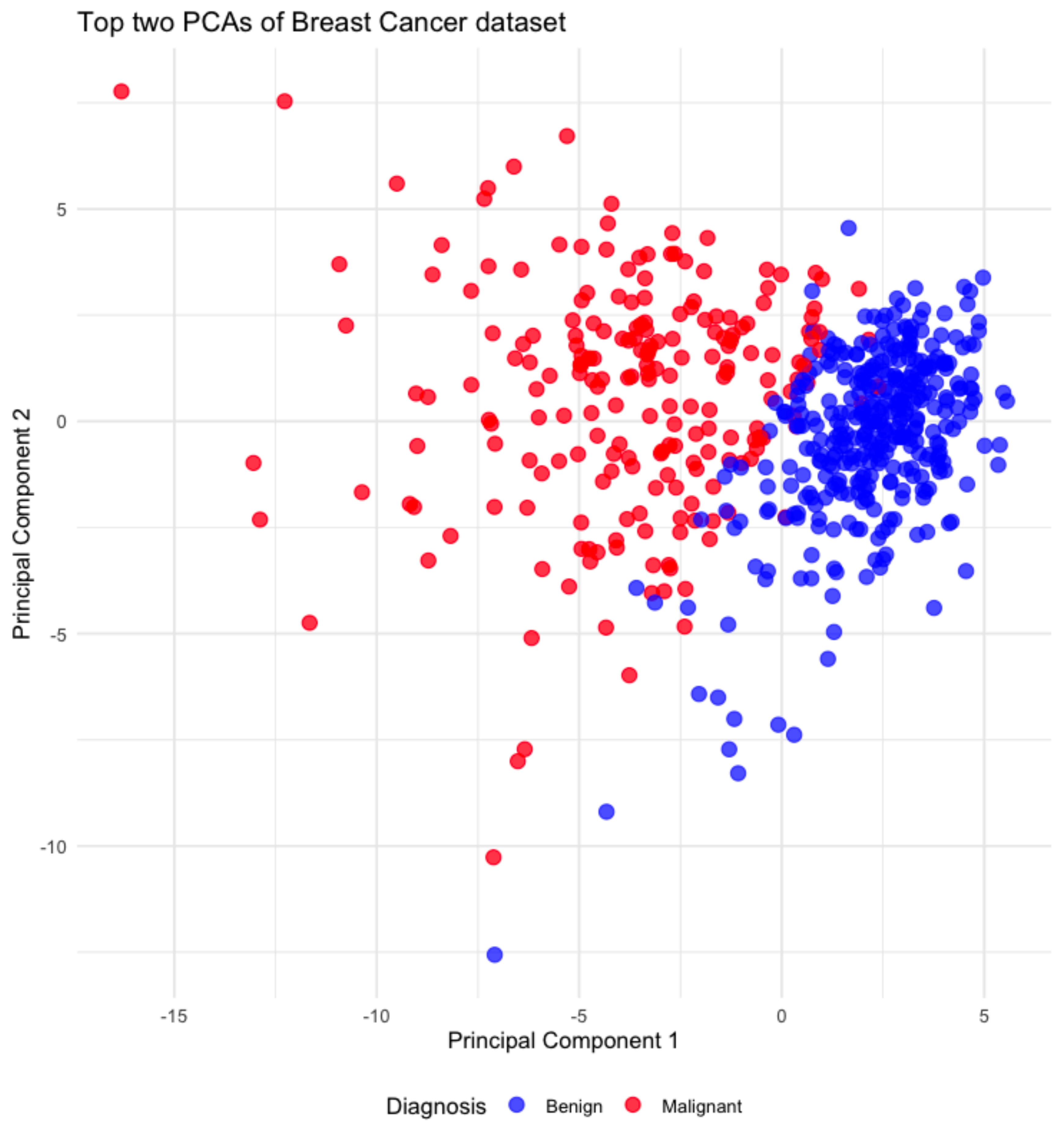


Dataset: Breast Cancer Diagnostics

Biomedical dataset with 569 patients and 30 features (not 2)

View data using two features





View data using two PCs

Matrices in linear regression and PCA

- a *brief* recap

1. Matrix notation and algebra help simplify a lot of math
2. Linear regression can be formulated using matrices
    1. Linear regression = projection = matrix multiplication
    2. `lm` in `R` = matrix algebra
3. Principal components help with dimension reduction
    1. Connected to eigenthings of underlying correlation matrix.
    2. Variance explanation using PCs is very helpful.

Thank you :)

soumikp@umich.edu