# Principal Component Analysis

# Week 6: Principal components

- Covariation between variables
- Eigendecomposition
- Principal components analysis

# Principal components

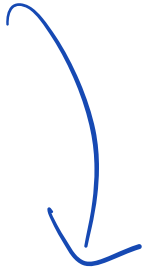
**Objective**: introduce an exploratory technique for exploring covariation among several variables.

- **Covariation between variables**
  - What is covariation and how is it measured quantitatively?
  - Covariance and correlation
  - Correlation matrix
  - Heatmap visualization

---

- **Eigendecomposition**
  - The eigenvalue problem
  - Geometric interpretation
  - Computations
- **Principal components analysis**
  - What is PCA exactly?

Summarizing correlations



# Covariation between variables

- What is covariation and how is it measured quantitatively?
- Covariance and correlation matrices
- Visualizations

# City sustainability data

We'll continue to use the following dataset on the sustainability of U.S. cities:

```
1 print(city_sust)
```

```
# A tibble: 933 × 6
   GEOID_MSA      Name          Econ_Domain Social_Domain Env_Domain
Sustain_Index
   <chr>          <chr>               <dbl>         <dbl>      <dbl>
<dbl>
 1 310M300US10100 Aberdeen, …       0.565         0.591      0.444
1.60
 2 310M300US10140 Aberdeen, …       0.428         0.521      0.429
1.38
 3 310M300US10180 Abilene, T…       0.481         0.497      0.454
1.43
 4 310M300US10220 Ada, OK Mi…       0.467         0.526      0.426
1.42
 5 310M300US10300 Adrian, MI…       0.498         0.602      0.282
```

# City sustainability data

For each **M**etropolitan **S**tatistical **A**rea (MSA), a sustainability index is calculated based on economic, social, and environmental indicators (also indices).

$$\text{sustainability index} = \text{economic} + \text{social} + \text{environmental}$$

The domain indices are computed from a large number of development indicator variables. To keep the application simple, we'll work with this derived data instead of the underlying variables.

***If you're interested,*** you can dig deeper on the Sustainable Development Report website, which provides detailed data reports related to the U.N.'s 2030 sustainable development goals.

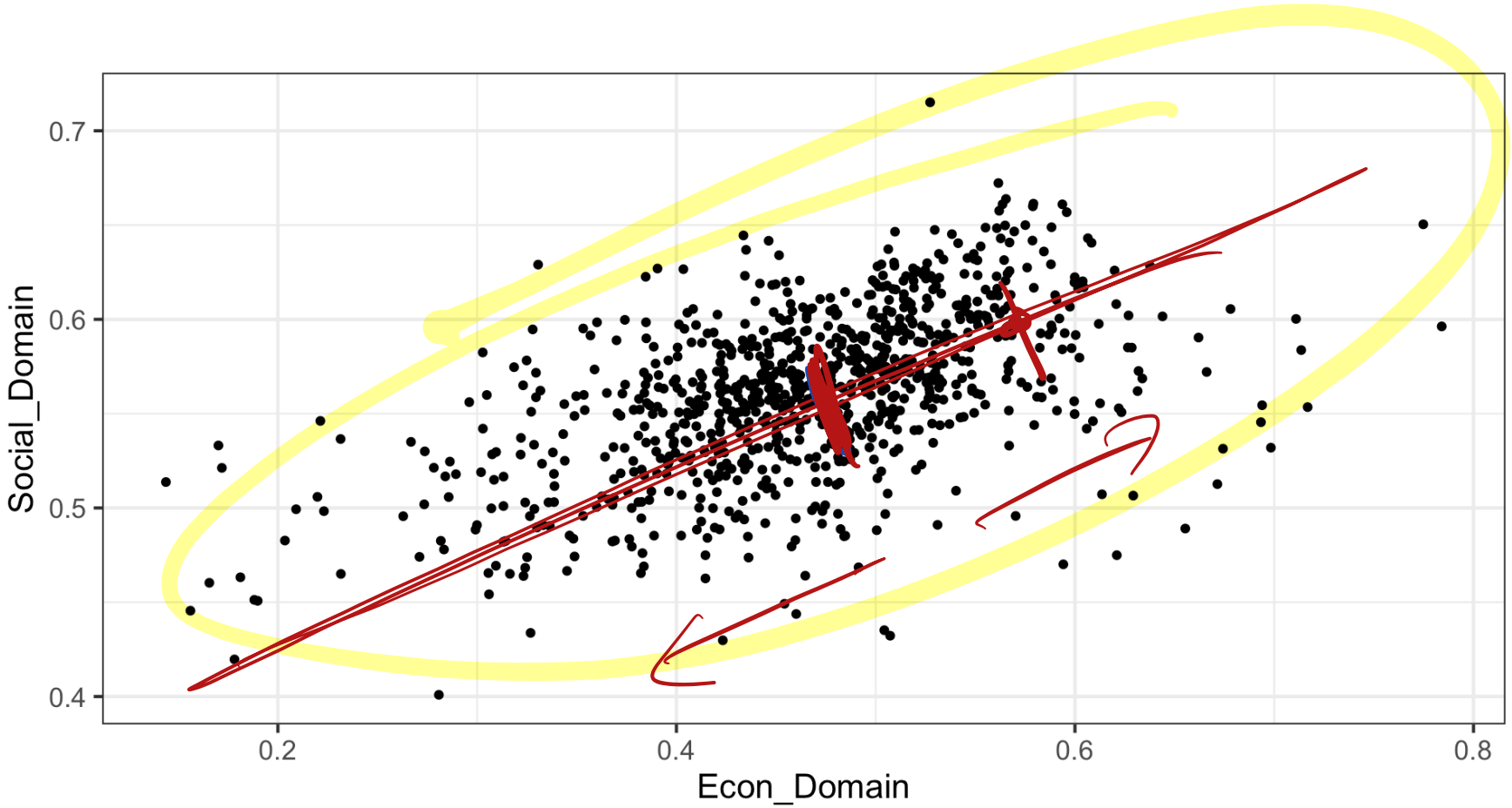
# What is covariation?

Last week we talked about exploring variation among individual variables: the tendency of values to change between observations.

**Covariation** refers to ***the tendency of two variables to change together between observations***. Covariation is about relationships.

```
1 # scatterplot of social vs economic indices
2 city_sust |> ggplot() + geom_point(aes(x=Econ_Domain, y=Social_Domain)) + t
```

# What is covariation?



# How is covariation measured?

Let $(x_1, y_1) \ldots, (x_n, y_n)$ denote $n$ values of two variables, $X$ and $Y$.

If $X$ and $Y$ tend to vary together, then whenever $X$ is far from its mean, so is $Y$.

- In other words, *their deviations from typical values coincide*.

This coincidence (or lack thereof) can be captured quantiatively by the (sample) **covariance**:

empirical

$$\operatorname{cov}(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

# How is covariation measured?

```
xy <- city_sust |> select(2:4) # x = econ, y = social
xy_ctr <- xy |> mutate(across(Econ_Domain:Social_Domain,
                              \(x) x - mean(x)))
xy_ctr
```

```
# A tibble: 933 × 3
   Name                                  Econ_Domain Social_Domain
   <chr>                                       <dbl>         <dbl>
 1 Aberdeen, SD Micro Area                    0.0989        0.0297
 2 Aberdeen, WA Micro Area                   -0.0387       -0.0408
 3 Abilene, TX Metro Area                     0.0147       -0.0647
 4 Ada, OK Micro Area                         0.000195     -0.0354
 5 Adrian, MI Micro Area                      0.0315        0.0409
 6 Akron, OH Metro Area                       0.0247        0.0174
 7 Alamogordo, NM Micro Area                 -0.0676       -0.0263
 8 Albany, GA Metro Area                     -0.142        -0.0933
 9 Albany, OR Metro Area                     -0.00561       0.0271
10 Albany-Schenectady-Troy, NY Metro Area     0.0992        0.00672
# ℹ 923 more rows
```

```
xy_cov = mean(xy_ctr$Econ_Domain * xy_ctr$Social_Domain)
xy_cov
```

0.002

# Correlation: standardized covariance

Covariance is a little tricky to interpret. Is 0.00199 large or small?

It is a little more useful to compute the (sample) **correlation**:

$$\text{corr}(\mathbf{x}, \mathbf{y}) = \frac{\text{cov}(\mathbf{x}, \mathbf{y})}{S_x S_y}$$

SD(Econ) SD(Social)

This is simply ***covariance standardized to [-1, 1]***.

# Correlation: standardized covariance

Properties:

- $\mathrm{corr}(\mathbf{x}, \mathbf{x}) = 1$, since any variable's deviations coincide *perfectly* with themselves
- the sign indicates whether $X$ and $Y$ vary together or in opposition
- $\mathrm{corr}(\mathbf{x}, \mathbf{y}) = 1, -1$ are the *strongest possible* correlations
- $\mathrm{corr}(\mathbf{x}, \mathbf{y}) = 0$ is the weakest possible correlation
- moderate to large correlations indicate that $X$ and $Y$ covary
- small correlations *do not indicate* that $X$ and $Y$ don't covary

# Correlation of social and economic indices

Standardizing the covariance makes it more interpretable:

```
1 xy_sd <- xy |> summarize(across(2:3, sd))
2 xy_cov / prod(xy_sd) # cov(x, y)/(sx*sy)
```

```
[1] 0.530921
```

$$\frac{Cov(X,Y)}{SD(X)\,SD(Y)}$$

The correlation indicates that the social and economic indices vary together (positive) weakly (moderate magnitude on [0, 1] scale).

***This is just a number that quantifies what you already knew from the graphic***: there is a positive relationship.

# What is the use of a quantitative measure of covariation?

It helps to have a number so that we can talk precisely about the *strength* of a relationship, and compare relationships quantitatvely.

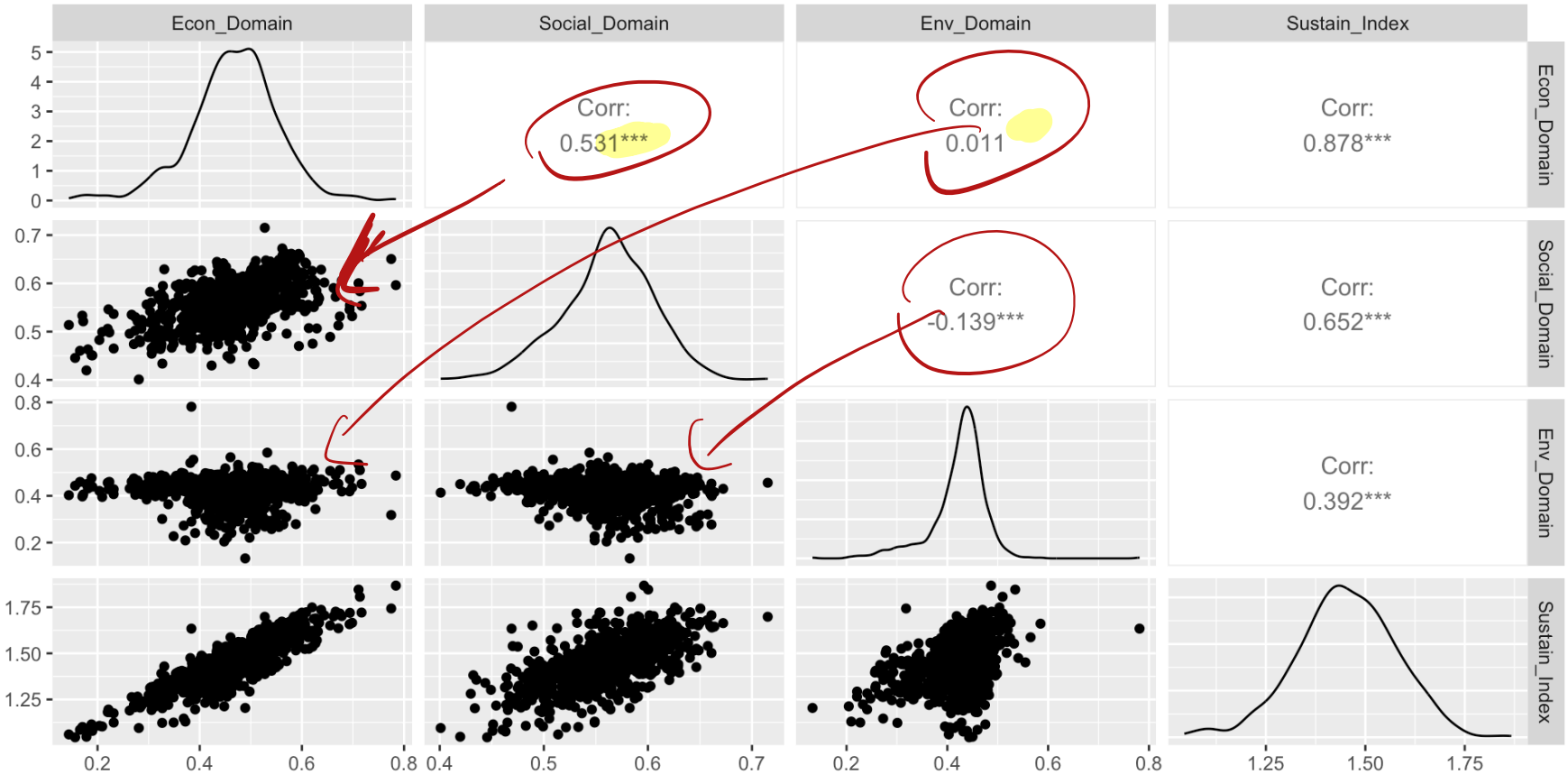
Consider looking at all pairwise relationships for patterns of covariation:

```
# extract social and economic indices
x_mx = city_sust |> select(2:5)

GGally::ggpairs(city_sust |> select(3:6))
```

Is the economic index *more* related to the environmental index or the social index?

# What is the use of a quantitative measure of covariation?



# Correlation matrix (example)

The pairwise correlations among all three variables can be represented in a simple square matrix:

```
1 corr_mx <- cor(city_sust |> select(3:last_col()))
2 corr_mx
```

```
              Econ_Domain Social_Domain  Env_Domain Sustain_Index
Econ_Domain     1.00000000     0.5314906  0.01120601     0.8775728
Social_Domain   0.53149064     1.0000000 -0.13867434     0.6523087
Env_Domain      0.01120601    -0.1386743  1.00000000     0.3923506
Sustain_Index   0.87757283     0.6523087  0.39235065     1.0000000
```

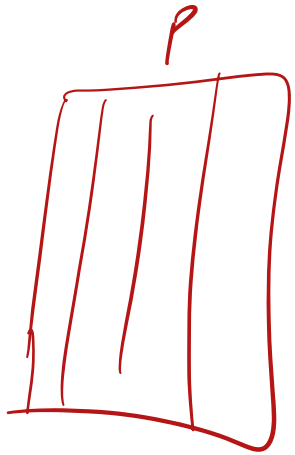
The strongest relationship is between social and economic indices; the weakest is between environmental and economic indices.

***Notice*** the correspondence between this matrix and the scatterplot panel – there is one number per plot that describes

# Correlation matrix (definition)

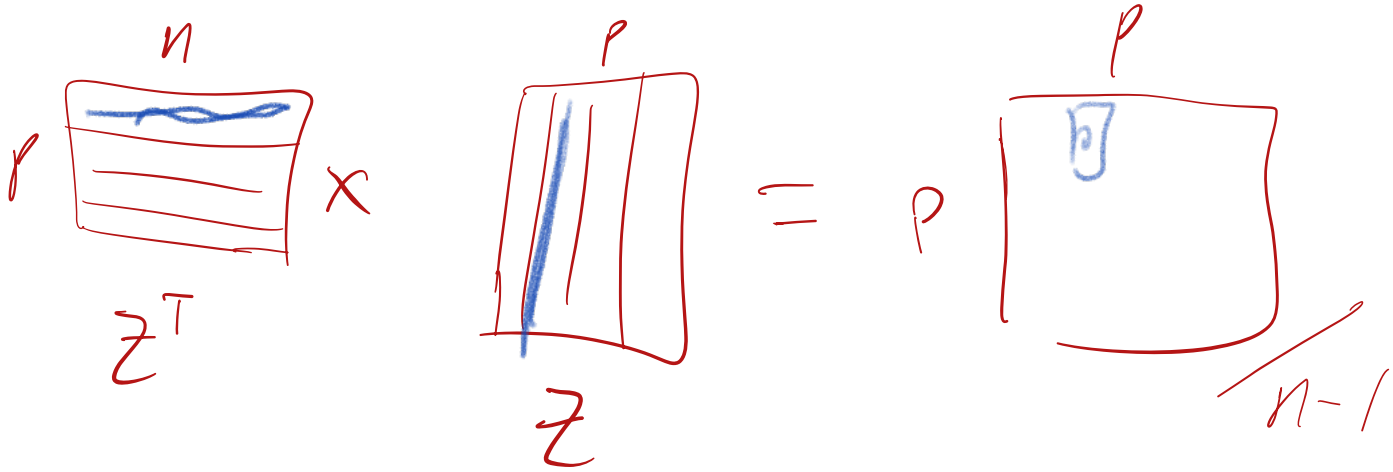
The (sample) correlation matrix can be expressed as a simple matrix product; let $\mathbf{X} \in \mathbb{R}^{n \times p}$ denote the data values for $n$ observations of $p$ variables.

Consider the centered and scaled version of the data: $\mathbf{Z} = \left\{ \frac{x_{ij} - \bar{x}_j}{s_{x_j}} \right\}$.




The (sample) **correlation matrix** is defined as:

$$\underset{p \times p}{\operatorname{corr}(\mathbf{X})} = \underbrace{\frac{1}{n-1}\mathbf{Z}'\mathbf{Z}}_{\text{call this } \mathbf{R}}$$


n
p
x
$Z^T$
p
Z
=
p
p
p
n-1

# Correlation matrix

```r
# correlation matrix 'by hand'
x <- city_sust |> select(3:last_col())
n <- nrow(x) # sample size

## center and scale
xscaled <- x |> mutate(across(everything(), \(x) (x - mean(x))/sd(x))) |>
  as.matrix()

t(xscaled) %*% xscaled /(n - 1) # Z'Z/(n - 1)
```

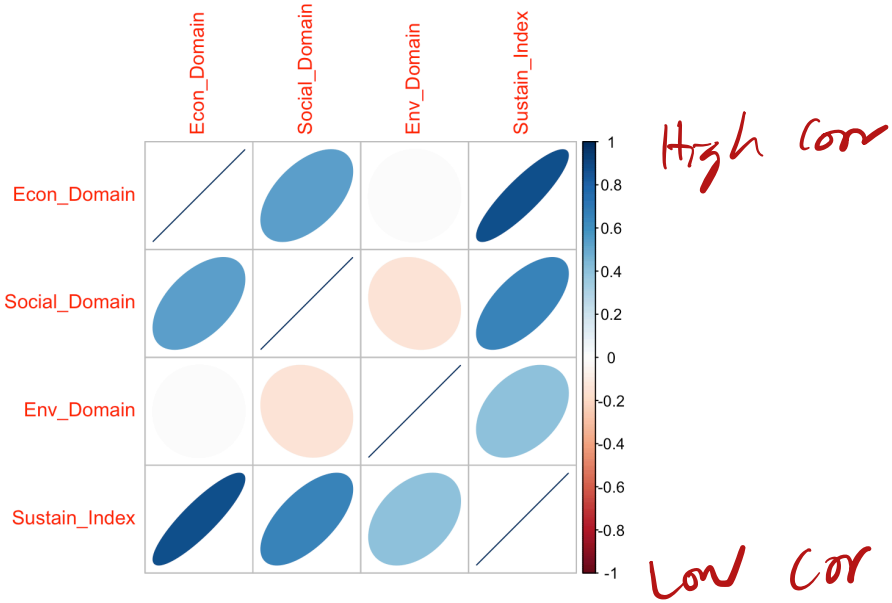
```
              Econ_Domain Social_Domain  Env_Domain Sustain_Index
Econ_Domain    1.00000000     0.5314906  0.01120601     0.8775728
Social_Domain  0.53149064     1.0000000 -0.13867434     0.6523087
Env_Domain     0.01120601    -0.1386743  1.00000000     0.3923506
Sustain_Index  0.87757283     0.6523087  0.39235065     1.0000000
```

Cor

# Visualizing correlation matrices

Often it's useful to visualize the correlation matrix. We can do this with `ggplot` but for quick exploratory plots I prefer to use the `corrplot` function.

```
1 library(corrplot)
2 corrplot(corr_mx, method="ellipse")
```



# Eigendecomposition

- The eigenvalue problem
- Geometric interpretation
- Computations

# Why are we talking about eigendecomposition?

We are going to factor the correlation matrix into components. This is, in essence, PCA.

This is useful for two reasons:

- identifying which variables drive variation and covariation
- reducing correlation structure among many variables to simpler components;
- visualizing multivariate data.

interpretation.

# The eigenvalue problem

Let $\mathbf{A}$ be a square $(n \times n)$ matrix.

The **eigenvalue problem** refers to finding nonzero $\lambda$ and $\mathbf{x}$ that satisfy the equation:

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

For any such solutions:

- $\lambda$ is an **eigenvalue** of $\mathbf{A}$;
- $\mathbf{x}$ is an **eigenvector** of $\mathbf{A}$.

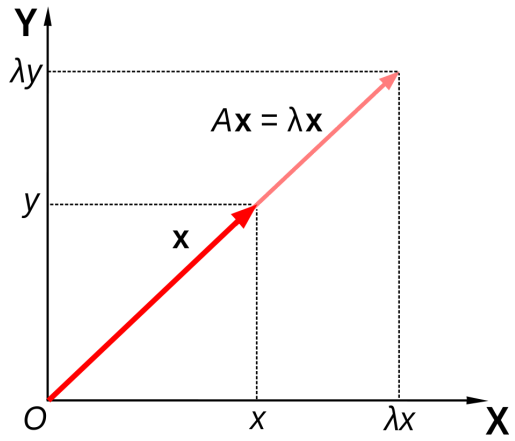
# Geometry

For a simple example, suppose $n = 2$ and $\mathbf{x}$ is an eigenvalue of $\mathbf{A}$. Then:

$$\mathbf{A}\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \end{bmatrix} = \begin{bmatrix} \lambda x_1 \\ \lambda x_2 \end{bmatrix} = \lambda \mathbf{x}$$

So the eigenvalue problem equation says that the *linear transformation of* $\mathbf{x}$ *by* $\mathbf{A}$ is simply a rescaling of $\mathbf{x}$ by a factor of $\lambda$.

# Geometry



# The decomposition

If $\lambda, \mathbf{x}$ are an eigenvalue and eigenvector of $\mathbf{A}$, then so are $c\lambda, c\mathbf{x}$ for any constant $c$; so usually attention is constrained to solutions such that $\|\mathbf{x}\| = 1$; these are unique to within the sign of $\mathbf{x}$.

The **eigendecomposition** of a square matrix consists in ***finding its unique nonzero eigenvalues and (unit-length) eigenvectors***.

This task is considered a 'decomposition' because the eigenvectors $\mathbf{V}$ and eigenvalues $\Lambda$ satisfy

$$\underbrace{\mathbf{A}}_{\text{original matrix}} = \underbrace{\mathbf{V}\Lambda\mathbf{V}'}_{\text{eigendecomposition}}$$

evectors as columns

evals on diagonal

So the original matrix can be *reconstructed* from the eigenvalues and eigenvectors.

# Special case

We will be applying eigendecomposition to correlation matrices.

- $\mathbf{V}'\mathbf{V} = \mathbf{I}$, in other words, the eigenvectors are an orthonormal basis;
- $\lambda_i \geq 0$, in other words, all eigenvalues are nonnegative.

# Computations

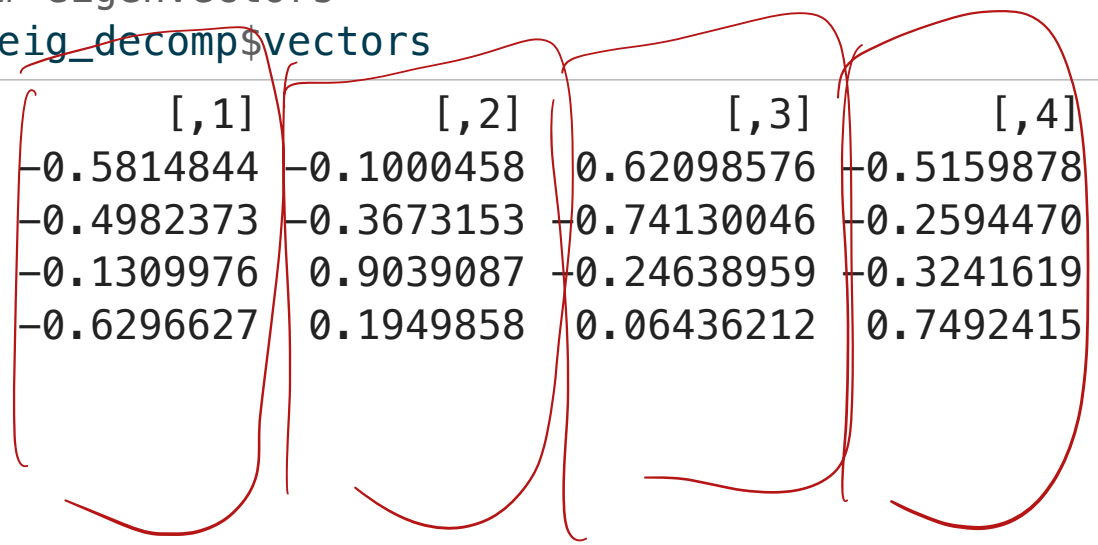
The eigendecomposition is computed numerically using iterative methods. Luckily, these are very easy to implement:

```r
# compute decomposition
eig_decomp <- eigen(corr_mx)
eig_decomp$values
```

```
[1]  2.408209e+00  1.139747e+00  4.520440e-01 -2.276671e-18
```

```r
# eigenvectors
eig_decomp$vectors
```

```
           [,1]       [,2]       [,3]       [,4]
[1,] -0.5814844 -0.1000458  0.62098576 -0.5159878
[2,] -0.4982373 -0.3673153 -0.74130046 -0.2594470
[3,] -0.1309976  0.9039087 -0.24638959 -0.3241619
[4,] -0.6296627  0.1949858  0.06436212  0.7492415
```



4 vars →

4 evals,

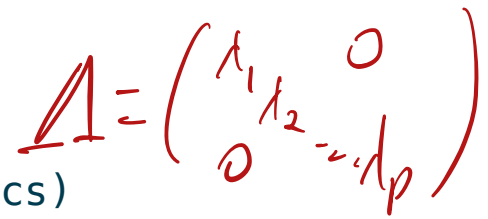
4 evecs

# Does the decomposition really work?

Let's check that in fact $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}'$:

```
# store eigenvalues and eigenvectors as dataframes
eigenvecs = eig_decomp$vectors
eigenvals = diag(eig_decomp$values)

eigenvecs %*% eigenvals %*% t(eigenvecs)
```



```
           [,1]       [,2]        [,3]      [,4]
[1,] 1.00000000  0.5314906  0.01120601 0.8775728
[2,] 0.53149064  1.0000000 -0.13867434 0.6523087
[3,] 0.01120601 -0.1386743  1.00000000 0.3923506
[4,] 0.87757283  0.6523087  0.39235065 1.0000000
```

```
corr_mx # correlation matrix (our 'A')
```

| | Econ_Domain | Social_Domain | Env_Domain | Sustain_Index |
|---|---|---|---|---|
| Econ_Domain | 1.00000000 | 0.5314906 | 0.01120601 | 0.8775728 |
| Social_Domain | 0.53149064 | 1.0000000 | -0.13867434 | 0.6523087 |
| Env_Domain | 0.01120601 | -0.1386743 | 1.00000000 | 0.3923506 |
| Sustain_Index | 0.87757283 | 0.6523087 | 0.39235065 | 1.0000000 |

# Orthogonality

Let's check also that in fact $\mathbf{V}'\mathbf{V} = \mathbf{I}$:

```
1 t(eigenvecs) %*% (eigenvecs) # V'V
```

```
              [,1]          [,2]          [,3]          [,4]
[1,]  1.000000e+00 -3.035792e-17 -1.455419e-16 7.571631e-17
[2,] -3.035792e-17  1.000000e+00  1.139010e-16 5.172949e-17
[3,] -1.455419e-16  1.139010e-16  1.000000e+00 1.046674e-16
[4,]  7.571631e-17  5.172949e-17  1.046674e-16 1.000000e+00
```

The significance of this property is that $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ form an *orthonormal basis*.

$$V = \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_4 \\ | & | & & | \end{bmatrix}$$

$v_i$ have length 1,

$v_i' v_j = 0 \quad i \neq j$

# What's a basis?

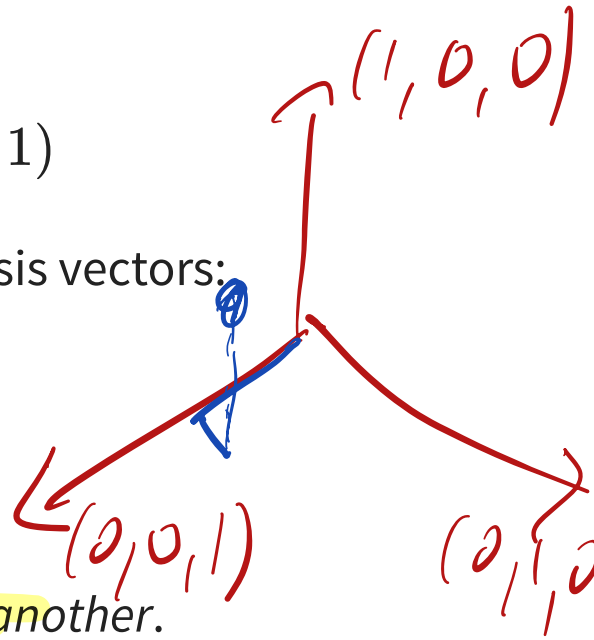
A **basis** in linear algebra is a ***set of vectors that span a linear space***. Think of a basis as a set of axes.

The usual basis for $\mathbb{R}^3$ are the unit vectors:

$$\mathbf{e}_1 = (1, 0, 0) \quad \mathbf{e}_2 = (0, 1, 0) \quad \mathbf{e}_3 = (0, 0, 1)$$

Coordinates in $\mathbb{R}^3$ are usually represented as multiples of these basis vectors:

$$(1, 2, 1) = 1\mathbf{e}_1 + 2\mathbf{e}_2 + 1\mathbf{e}_3$$



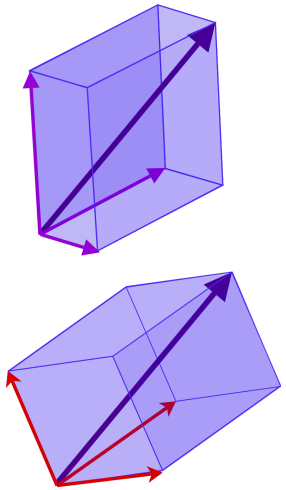

**Terminology**:

- A basis is *orthogonal* if *all basis vectors are at right angles to one another*.
- A basis is *orthonormal* if *it is orthogonal and basis vectors are of unit length*.

# Nonstandard bases

To an extent the usual choice of $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ is arbitrary.

It's possible to choose *any* system based on three directions to represent the same point relative to the origin.



# Interpreting a change of basis

***Different bases are simply different coordinate systems.***

It's akin to the possibility of locating the distance to the corner store by different units and relative to different directions. For example:

- half a block to the right from the front door; or
- sixty steps diagonally in a straight line; or
- fifteen steps north, fifty steps east.

# Eigendecomposition of the correlation matrix

So, decomposing the correlation matrix yields a basis.

It turns out that when data are represented on that basis, patterns of covariation vanish.

> Let $\mathbf{Y} = \mathbf{XV}$; this is the representation of $\mathbf{X}$ on the basis given by $\mathbf{V}$. Then
>
> $$\mathbf{Y}'\mathbf{Y} = \mathbf{V}'\mathbf{X}'\mathbf{XV} = \mathbf{V}'\mathbf{V}\mathbf{\Lambda}\mathbf{V}'\mathbf{V} = \mathbf{\Lambda}$$
>
> which implies that $\text{cov}(\mathbf{y}_j, \mathbf{y}_k) = 0$ since $\mathbf{\Lambda}$ is diagonal.

So the change of basis to $\mathbf{V}$ in a sense 'captures' the covariation in the data.

Notice that the change of basis is a linear transformation $\mathbf{XV}$.

This is the essence of PCA: using eigendecomposition to obtain a linear transformation that captures covariation.

# Principal components analysis

- What is PCA exactly?z
- PCA in the low-dimensional setting
- Variation capture and loss
- Interpreting principal components
- Example application

# Principal components analysis

**Principal components analysis** (PCA) consists in ***finding a linear transformation of one's data that captures covariation***.

Applications are varied, and so are descriptions of just what the method is supposed to do, but the core technique is simple:

1. Compute the eigendecomposition of the correlation matrix.
2. Choose a subset of eigenvectors.
3. Then do other stuff (the 'analysis' part).

**PCA terminology**:

- Eigenvectors are called ‘principal component directions’
- The values of eigenvectors are called ‘loadings’
- The projected data consist of the ‘principal components’

evals are the variances of the projected data.

"weights in the weighted avg"

# PCA in the low-dimensional setting

Let's consider what happens if we follow the steps of PCA outlined above with just two variables, say the social index and the economic index, which were the most correlated pair in the example data.

Denote the observations on these two variables by

$$\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2]$$

Almost always center & scale.

$$\text{where} \qquad \mathbf{x}_1 = \text{social index and } \mathbf{x}_2 = \text{economic index}$$

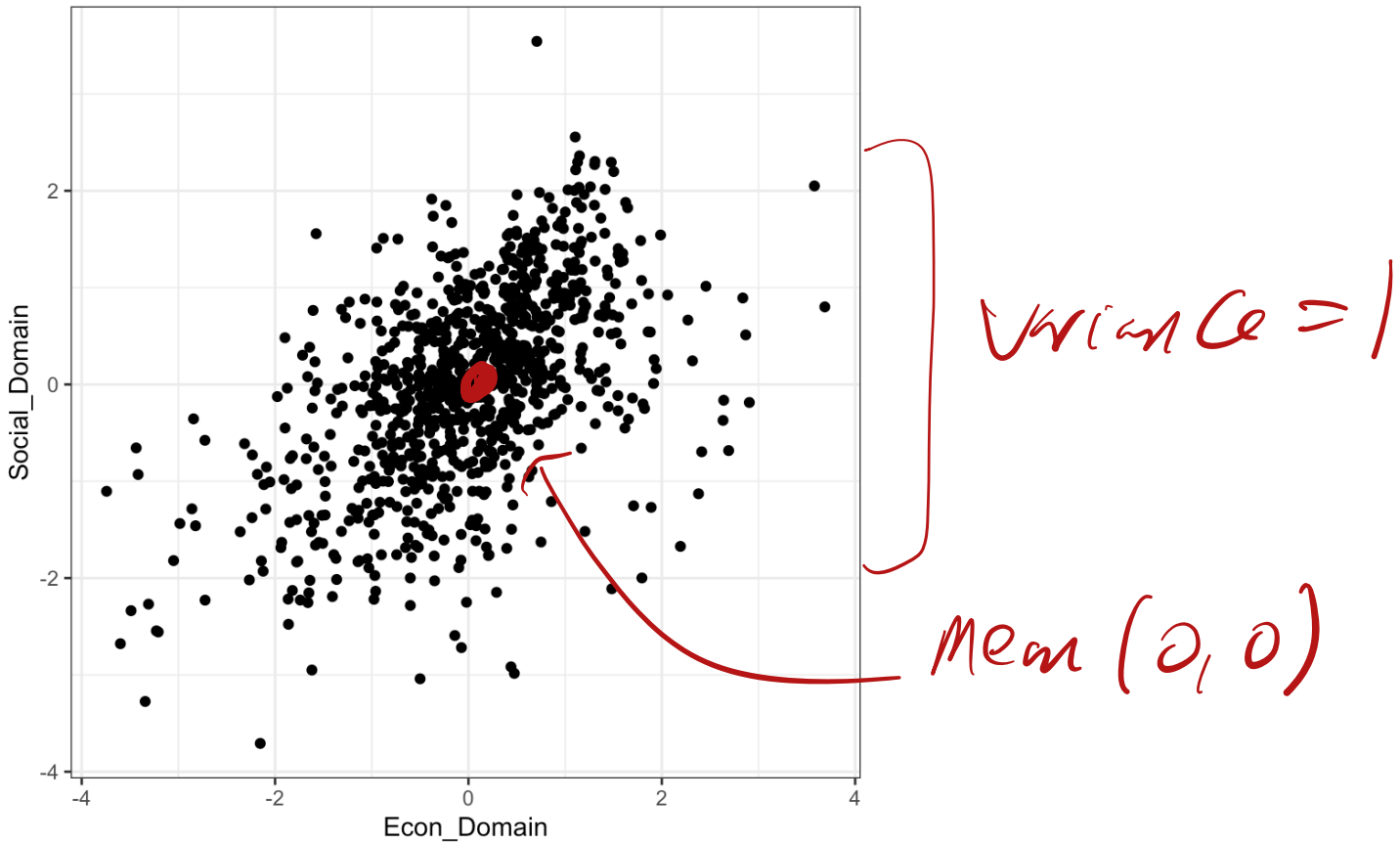
To get the correlation matrix, first compute $\mathbf{Z} = \left\{ \frac{x_i - \bar{x}}{s_x} \right\}$.

```
1 # scatterplot of scaled data
2 xscaled <- x |> scale()
```

```r
scaled_plot <- xscaled |> ggplot() +
  geom_point(aes(x=Econ_Domain, y=Social_Domain)) +
  theme_bw() + coord_fixed()

scaled_plot
```

# PCA in the low-dimensional setting



# PCA in the low-dimensional setting

Now we'll compute the eigendecomposition. This is pretty much all there is to it.

```r
z <- xscaled[, c("Econ_Domain", "Social_Domain")]

cor_mat <- t(z) %*% z / (nrow(z) - 1)
eigen_decomp <- eigen(cor_mat)

evecs <- eigen_decomp$vec
evals <- eigen_decomp$values
evecs
```

```
          [,1]       [,2]
[1,] 0.7071068 -0.7071068
[2,] 0.7071068  0.7071068
```

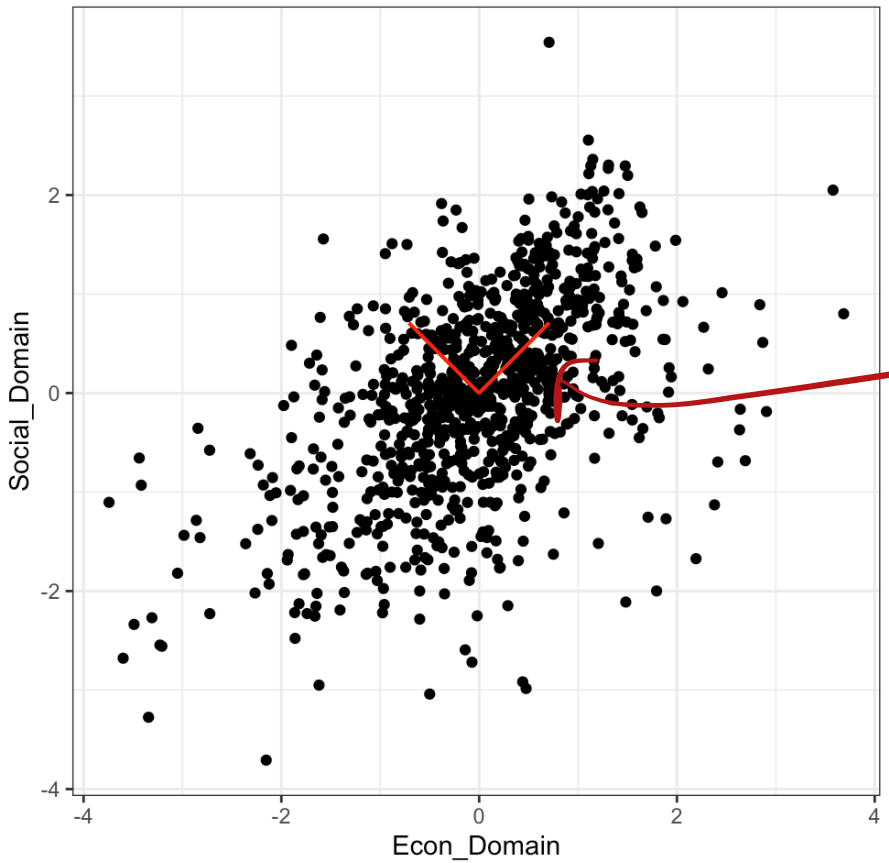
```r
evals
```

```
[1] 1.5314906 0.4685094
```

# The Geometry of PCA

```
scaled_plot +
  geom_segment(x=0, y=0,
               xend=evecs[1, 1],
               yend=evecs[2, 1],
               col='red') +
  geom_segment(x=0, y=0,
               xend=evecs[1, 2],
               yend=evecs[2, 2],
               col='red')
```

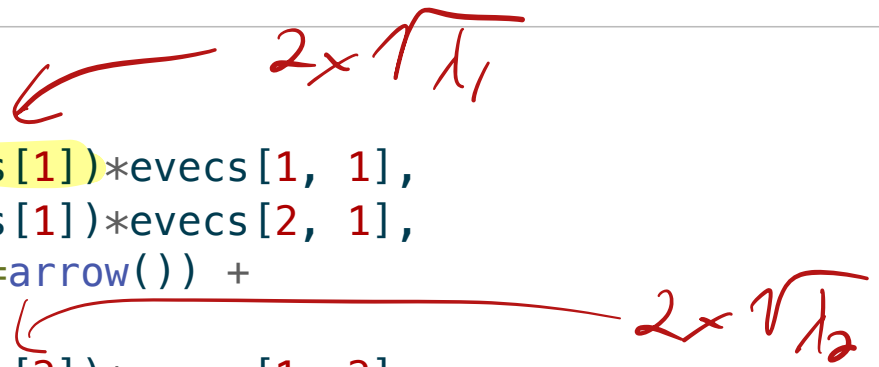
# The Geometry of PCA




evectors.

# The Geometry of PCA

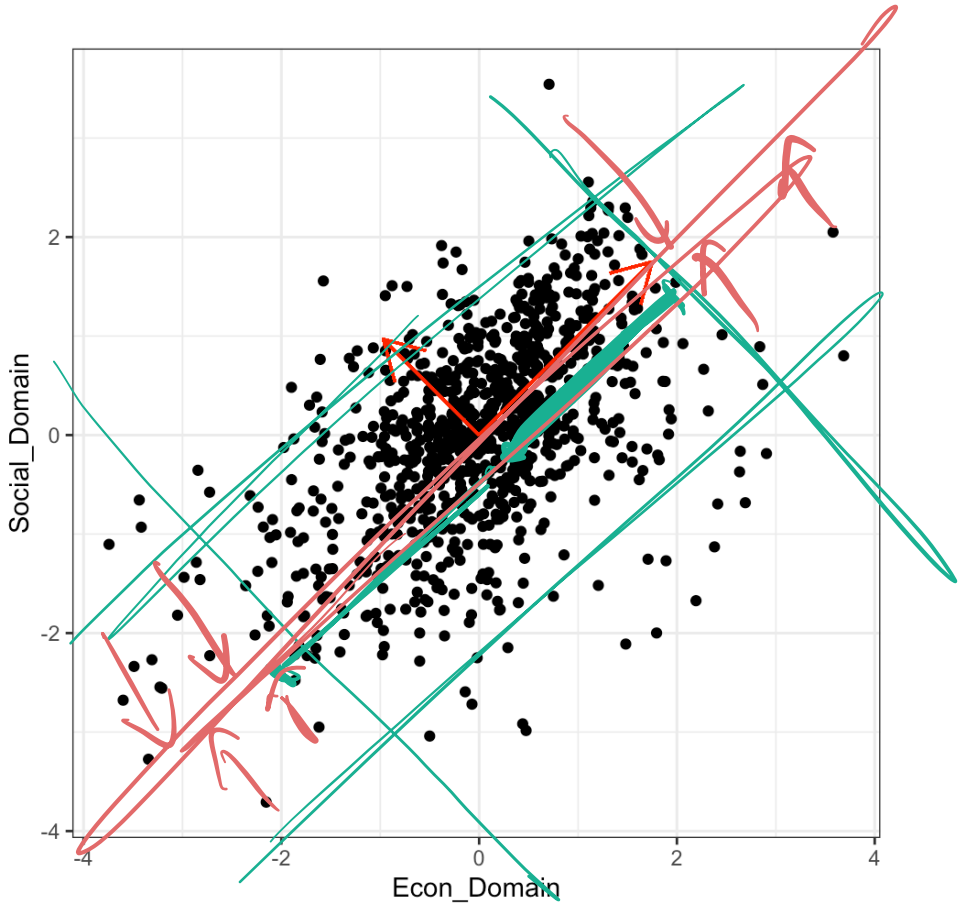
```
scaled_plot +
  geom_segment(x=0, y=0,
               xend=2*sqrt(evals[1])*evecs[1, 1],
               yend=2*sqrt(evals[1])*evecs[2, 1],
               col='red', arrow=arrow()) +
  geom_segment(x=0, y=0,
               xend=2*sqrt(evals[2])*evecs[1, 2],
               yend=2*sqrt(evals[2])*evecs[2, 2],
               col='red', arrow=arrow())
```




The difference here comes from scaling the directions by the corresponding eigenvalues (plotting $\lambda_1 \mathbf{v}_1$ and $\lambda_2 \mathbf{v}_2$).

*The principal component directions are the axes along which the data vary most, and the eigenvalues give the magnitude of that variation.*

# The Geometry of PCA



# Geometry of PCA: projection

So if we wanted to look at just *one* quantity that retains variation and covariation in the original data, we could:

- project the data onto the new basis
- and look at the values on the axis of the first PC direction (the one with the most variation, *i.e.*, largest $\lambda_i$).

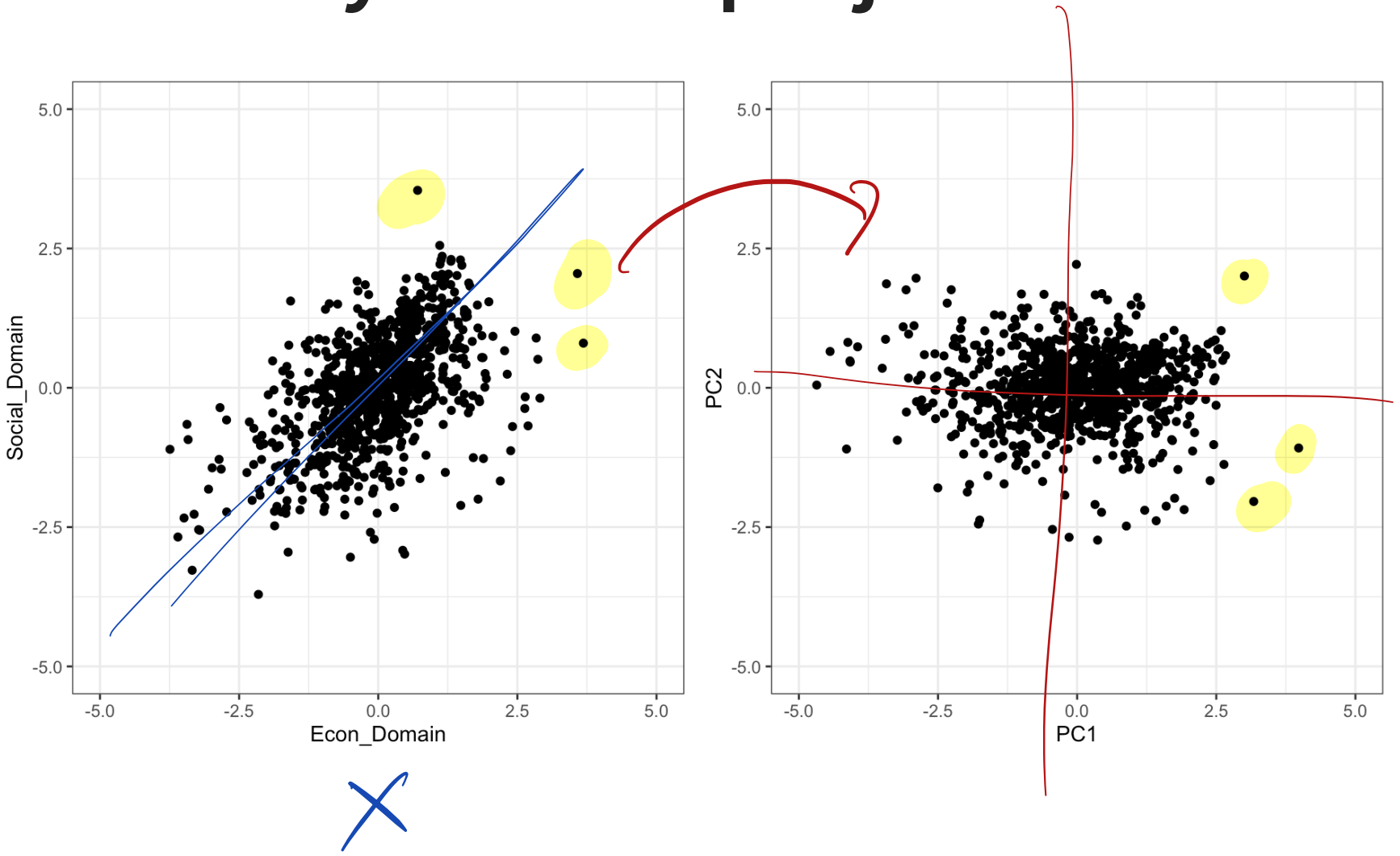
```
# project data onto pc directions
projected_data <- z %*% evecs

projected_plot <- tibble(PC1 = projected_data[, 1],
                         PC2 = projected_data[, 2]) |>
  ggplot() + geom_point(aes(x=PC1, y=PC2)) + theme_bw()

(scaled_plot + xlim(c(-5, 5)) + ylim(c(-5, 5))) + (projected_plot + xlim(c(
```

If we select just the first principal component to ‘do other stuff’ with, we lose the variation in PC2. Is this a problem?

# Geometry of PCA: projection



# Projection & Dimension Reduction

```r
city_sust_with_pc <- city_sust
city_sust_with_pc$PC1 <- projected_data[, 1]
city_sust_with_pc$PC2 <- projected_data[, 2]

city_sust_with_pc <- city_sust_with_pc |>
  arrange(PC1) |>
  select(Name, PC1, PC2, Econ_Domain, Social_Domain)

head(city_sust_with_pc)
```

```
# A tibble: 6 × 5
  Name                          PC1     PC2 Econ_Domain Social_Domain
  <chr>                       <dbl>   <dbl>       <dbl>         <dbl>
1 Greenville, MS Micro Area -4.68  0.0489       0.178         0.420
2 Clarksdale, MS Micro Area -4.44  0.652        0.156         0.446
3 Las Vegas, NM Micro Area  -4.14 -1.10         0.281         0.401
4 Indianola, MS Micro Area  -4.12  0.815        0.166         0.460
5 Cleveland, MS Micro Area  -4.08  0.483        0.188         0.451
6 Cordele, GA Micro Area    -4.08  0.461        0.190         0.451
```

# Projection and Dimension Reduction

```
1 tail(city_sust_with_pc)
```

```
# A tibble: 6 × 5
  Name                                      PC1    PC2 Econ_Domain
Social_Domain
  <chr>                                   <dbl>  <dbl>       <dbl>
<dbl>
1 Rochester, MN Metro Area                 2.62  0.492       0.596
0.657
2 Washington-Arlington-Alexandria, DC-VA…  2.64 -1.38        0.711
0.600
3 Torrington, CT Micro Area                2.67  0.578       0.594
0.661
4 The Villages, FL Metro Area              3.01  2.01        0.527
0.715
5 San Jose-Sunnyvale-Santa Clara, CA Met…  3.17 -2.04        0.784
```

# Projection & Dimension Reduction

```
1 head(city_sust_with_pc |> arrange(PC2))
```

```
# A tibble: 6 × 5
  Name                                  PC1   PC2 Econ_Domain Social_Domain
  <chr>                               <dbl> <dbl>       <dbl>         <dbl>
1 Breckenridge, CO Micro Area         0.368 -2.73       0.655         0.489
2 Fairbanks, AK Metro Area           -0.145 -2.68       0.621         0.475
3 Ketchikan, AK Micro Area           -0.444 -2.54       0.594         0.470
4 Williston, ND Micro Area            0.884 -2.48       0.671         0.513
5 Fort Leonard Wood, MO Micro Area -1.77    -2.45       0.507         0.432
6 Juneau, AK Micro Area               1.42  -2.39       0.698         0.532
```

```
1 tail(city_sust_with_pc |> arrange(PC2))
```

```
# A tibble: 6 × 5
  Name                                   PC1   PC2 Econ_Domain Social_Domain
  <chr>                                <dbl> <dbl>       <dbl>         <dbl>
1 Raymondville, TX Micro Area         -3.07   1.76       0.172         0.521
2 Deming, NM Micro Area               -2.26   1.76       0.221         0.546
3 Rio Grande City, TX Micro Area      -3.43   1.87       0.144         0.514
4 Gallup, NM Micro Area               -2.89   1.97       0.170         0.533
5 The Villages, FL Metro Area          3.01   2.01       0.527         0.715
6 Magnolia, AR Micro Area             -0.0132 2.21       0.331         0.629
```

# Quantifying variation capture/loss

To figure out how much variation/covariation is captured and lost, we need to know how much is available in the first place.

The **total variation in $\mathbf{X}$** is given in terms of the correlation matrix by:

$$\underbrace{\text{total variation}}_{\text{in our data}} = \det(\mathbf{R}) = \sum_{i=1}^{p} \lambda_i$$

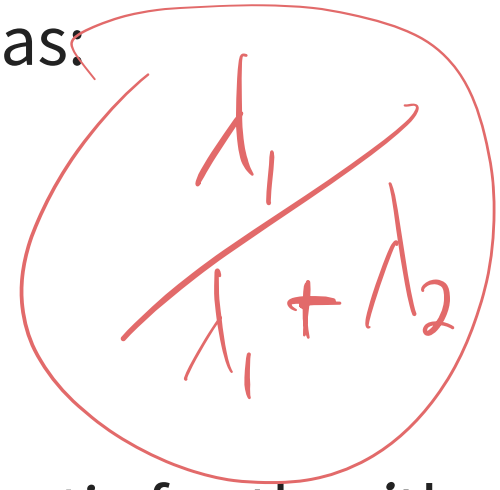
Now let $\mathbf{y}_j = \mathbf{Z}\mathbf{v}_j$ be the $j$th principal component. Its variance is:

# Quantifying variation capture/loss

So the total variation is the sum of the eigenvalues, and the variance of each PC is the corresponding eigenvalue.

We can therefore define the **proportion of total variation explained** by the $j$th principal component as:

$$\frac{\lambda_j}{\sum_{j=1}^{p} \lambda_j}$$




This is sometimes also called the *variance ratio* for the $j$th principal component.

So in our example:

```r
evals[1] / sum(evals)
```

```
[1] 0.7657453
```

The first principal component captures 77% of total variation.

# Interpreting principal components

So we have obtained a single derived variable that captures 3/4 of all variation and covariation in both variables. But what is this?

The values of the first principal componentare given by:

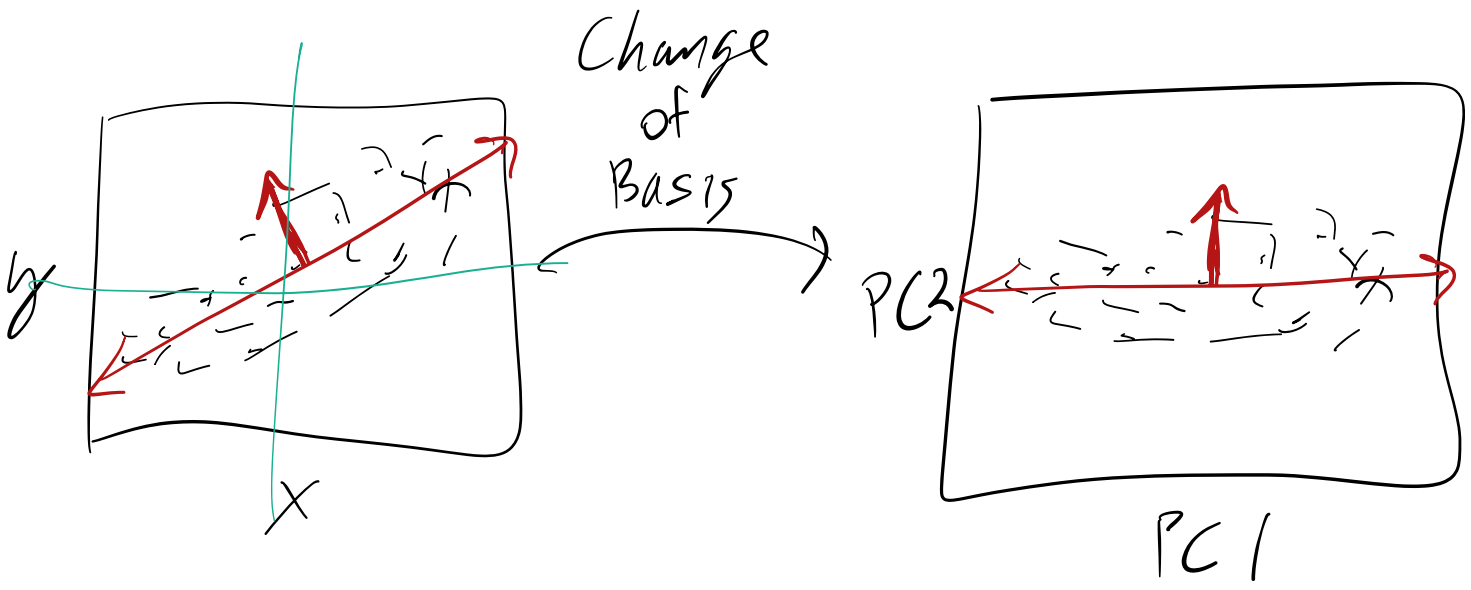
$$\mathrm{PC1}_i = \mathbf{x}_i' \mathbf{v}_1 = 0.7071 \times \mathrm{economic}_i + 0.7071 \times \mathrm{social}_i$$

So the principal component is a linear combination of the underlying variables.

Those coefficients are known as **loadings**: they determine the ***relative weights*** by which the variables contribute to the principal component.

In this case, the PC1 loadings are equal; so this principal component is simply the average of the social and economic indices.

# PCA




$$PC1 = w_{11} X + w_{12} y$$

PC "loadings"

$$\sum w_i^2 = 1$$

(loadings have "unit norm" or "length 1")

$$Cov((X, Y)) = \Sigma_{2\times 2} = U \Lambda U^T$$

eigenvectors

e-values

$$U = \left[ \begin{pmatrix} w_{11} \\ w_{12} \end{pmatrix}, \begin{pmatrix} w_{21} \\ w_{22} \end{pmatrix} \right]$$ columns of $U$, e-vecs, are the loadings for each PC.

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \quad \lambda_1 > \lambda_2.$$

$$\lambda_1 = Var(PC1)$$

$$\lambda_2 = Var(PC2)$$

In R: "prcomp"

prcomp(dat, scale = false)

Returns a list, with:

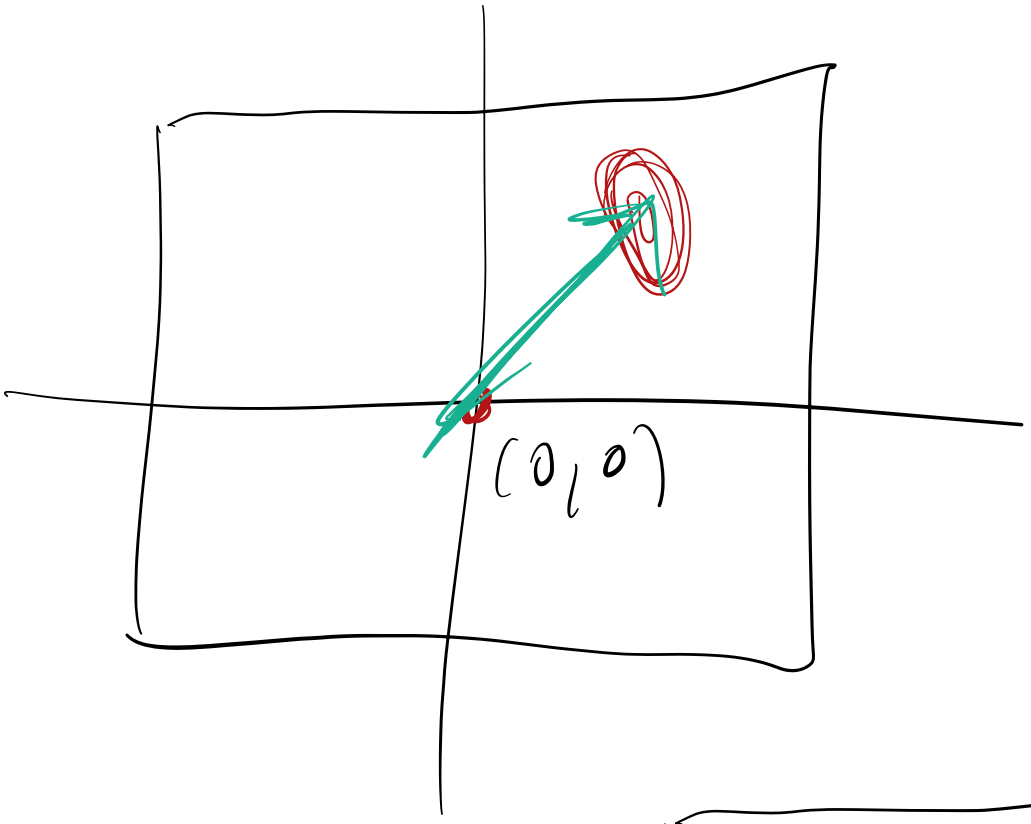
"rotation", this is V, loadings.

"sdev", this is $(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots)$
vector of root-evals,
or sdev of the PCs.

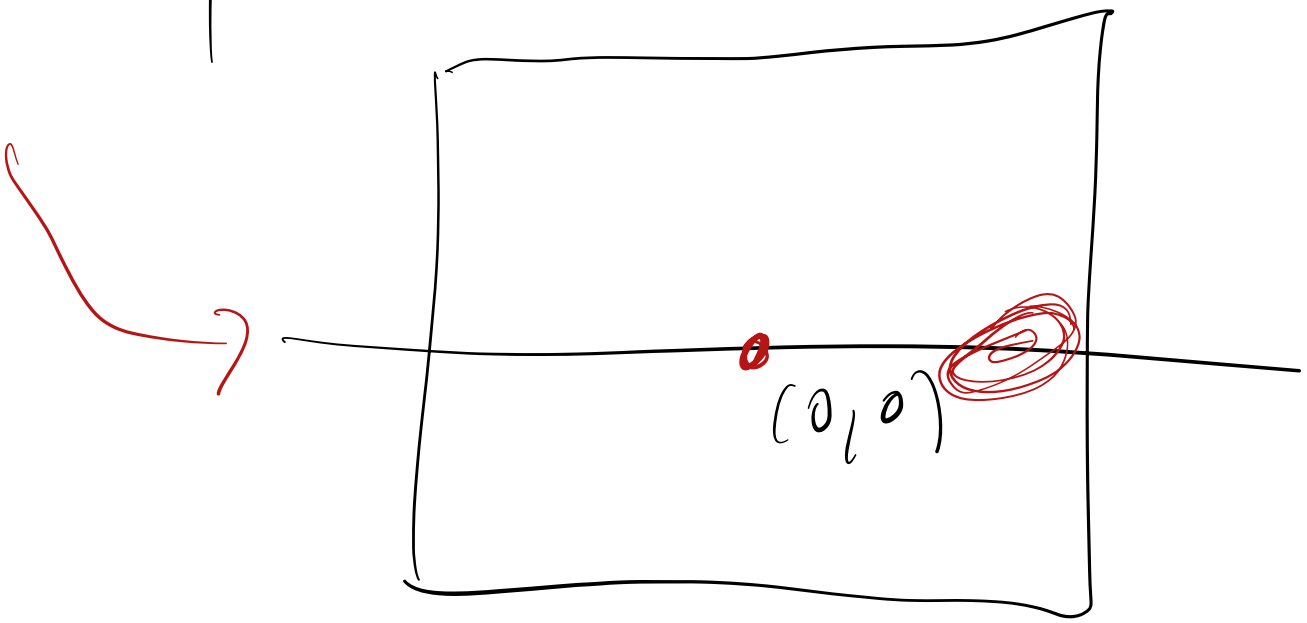
"x", the projected data,
data after rotating
"PC scores"

# Centering & Scaling

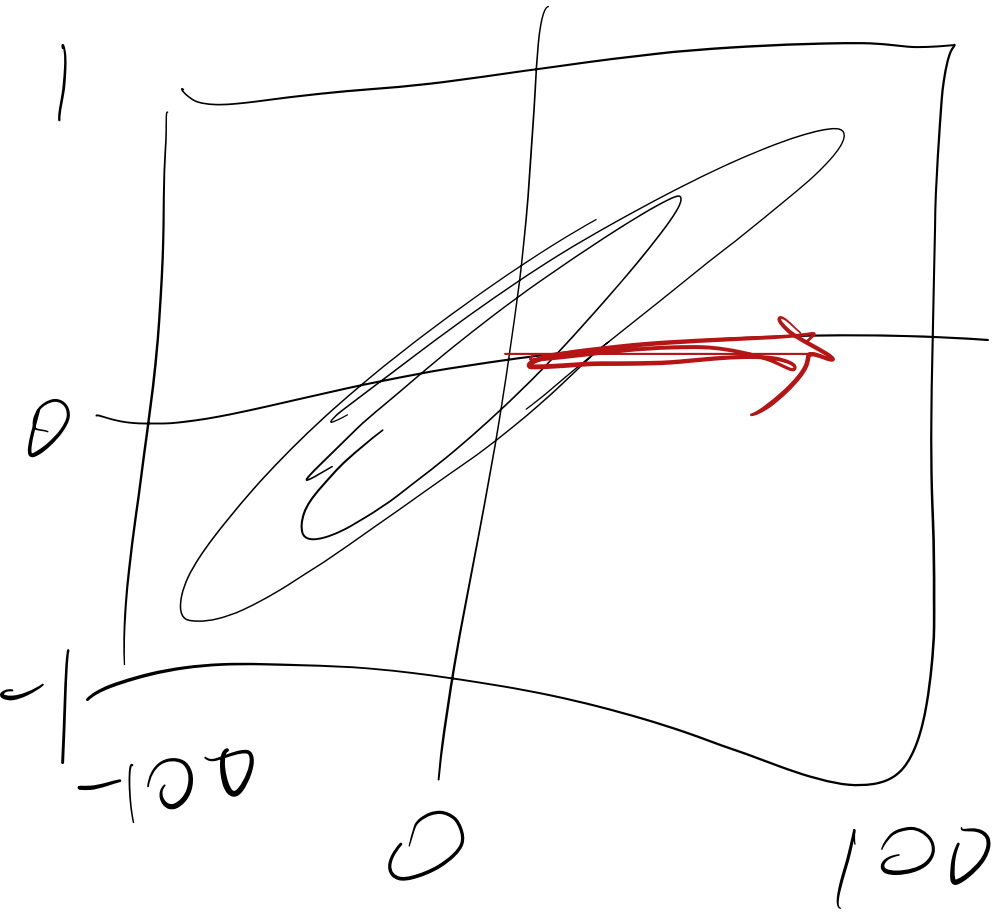
```
prcomp(dat, center = true,
       scale = true)
```




Importance of centering.



# Scaling.



# Example application

Now that we've reviewed the basic technique, we can consider cases with a larger number of variables.

This is really where PCA is most useful.

- It can help answer the question: which variables are driving variation in the data?
- It can help reduce data dimensions by finding combinations of variables that preserve variation.
- It can provide a means of visualizing high-dimensional data.

# Example application

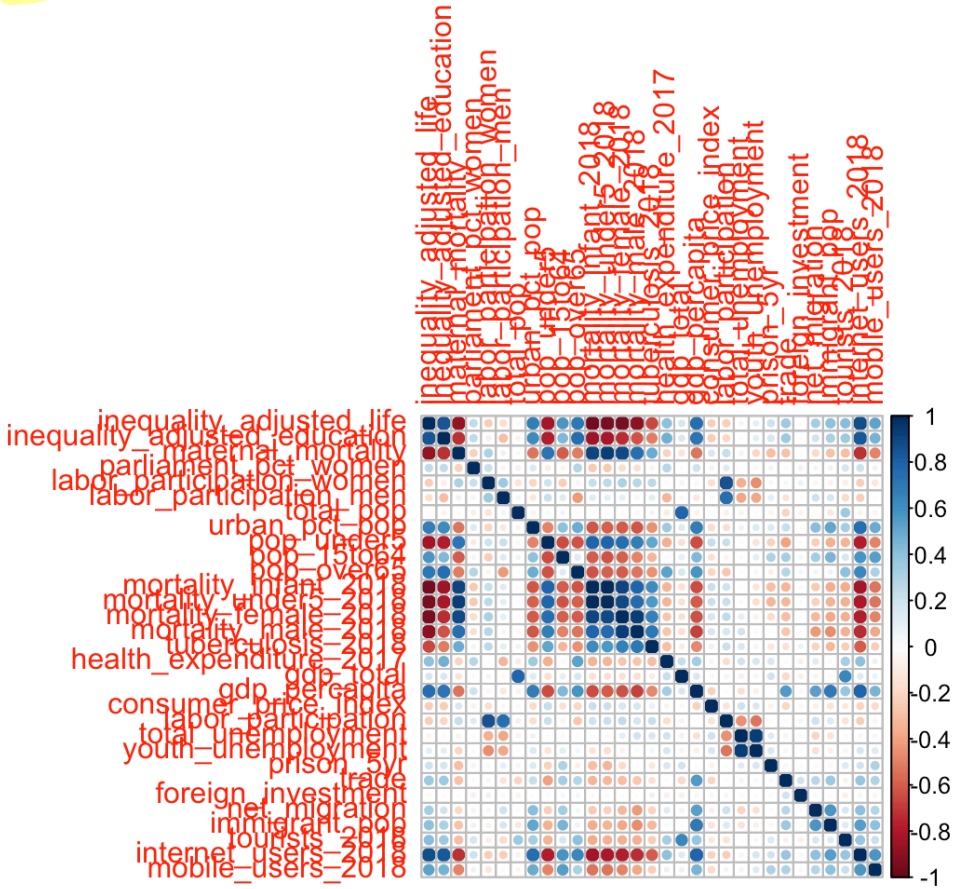
So, let's look at a different example: world development indicators.

```
1 wdi <- read_csv('data/wdi-data.csv') |> select(-1)
2 wdi
```

```
# A tibble: 139 × 33
   hdi_rank country     inequality_adjusted_life inequality_adjusted_education
      <dbl> <chr>                          <dbl>                          <dbl>
 1        1 Norway                         0.931                          0.908
 2        2 Ireland                        0.926                          0.892
 3        2 Switzerland                    0.947                          0.883
 4        4 Iceland                        0.946                          0.9
 5        6 Germany                        0.908                          0.922
 6        7 Sweden                         0.938                          0.884
 7        8 Australia                      0.94                           0.899
 8        8 Netherlands                    0.928                          0.865
 9       10 Denmark                        0.903                          0.894
10       11 Finland                        0.924                          0.907
# ℹ 129 more rows
```

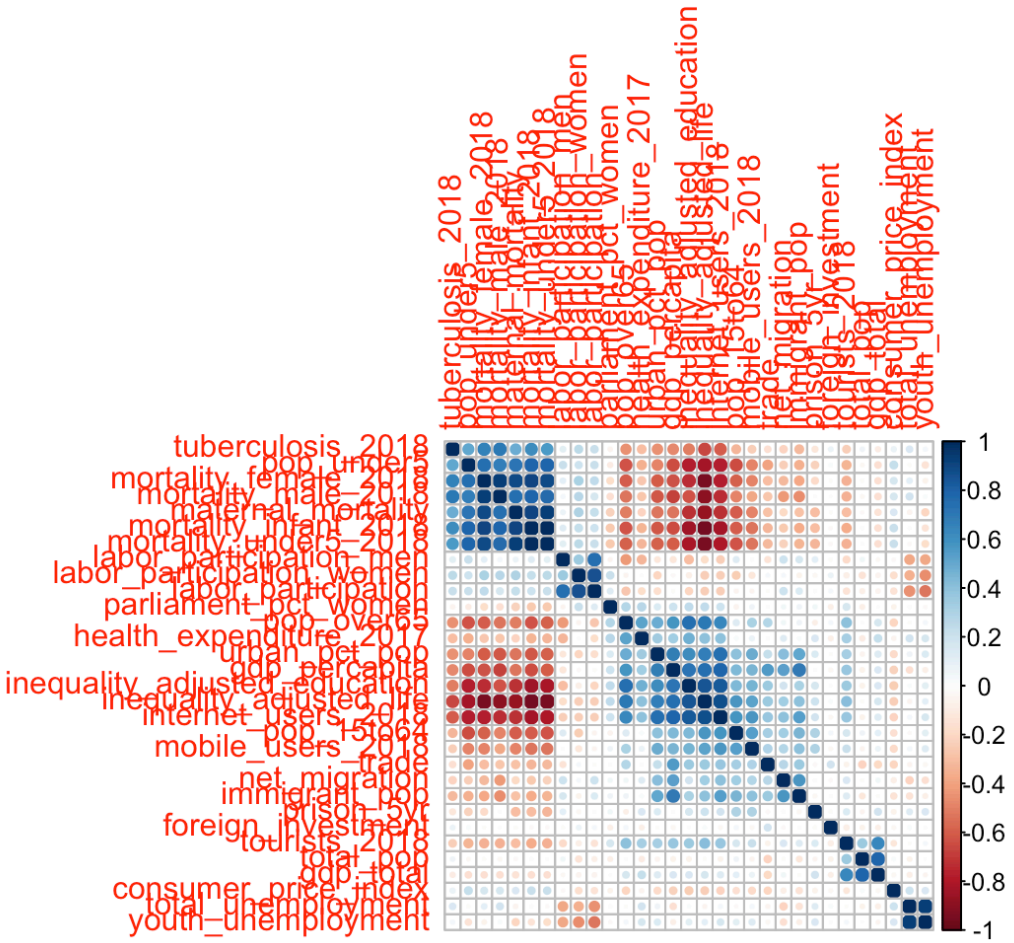
# Correlation Plot

```
1 country_corr <- cor(wdi |> select(inequality_adjusted_life:last_col()))
2 corrplot(country_corr)
```



# Clustered Corrplot

```
1 corrplot(country_corr, order='hclust')
```



# Computation via `prcomp`

Luckily, `prcomp` is an easy-to-use implementation that saves the trouble of going through all the calculations we did before 'by hand'.

```
1 pca_result <-
2   wdi |> select(inequality_adjusted_life:last_col()) |>
3   prcomp(x=_, scale=TRUE)
```

# Variance ratios

Selecting the number of principal components to use is somewhat subjective, but always based on the variance ratios.

- how many PCs capture substantial variation individually before the variance ratio ‘tails off’?
- how many PCs are needed to capture a certain proportion of the total variation?

$$\lambda_i \Big/ \sum \lambda_i$$

# Variance Ratios

This can be assessed by plotting the variance ratios and their cumulative sum:

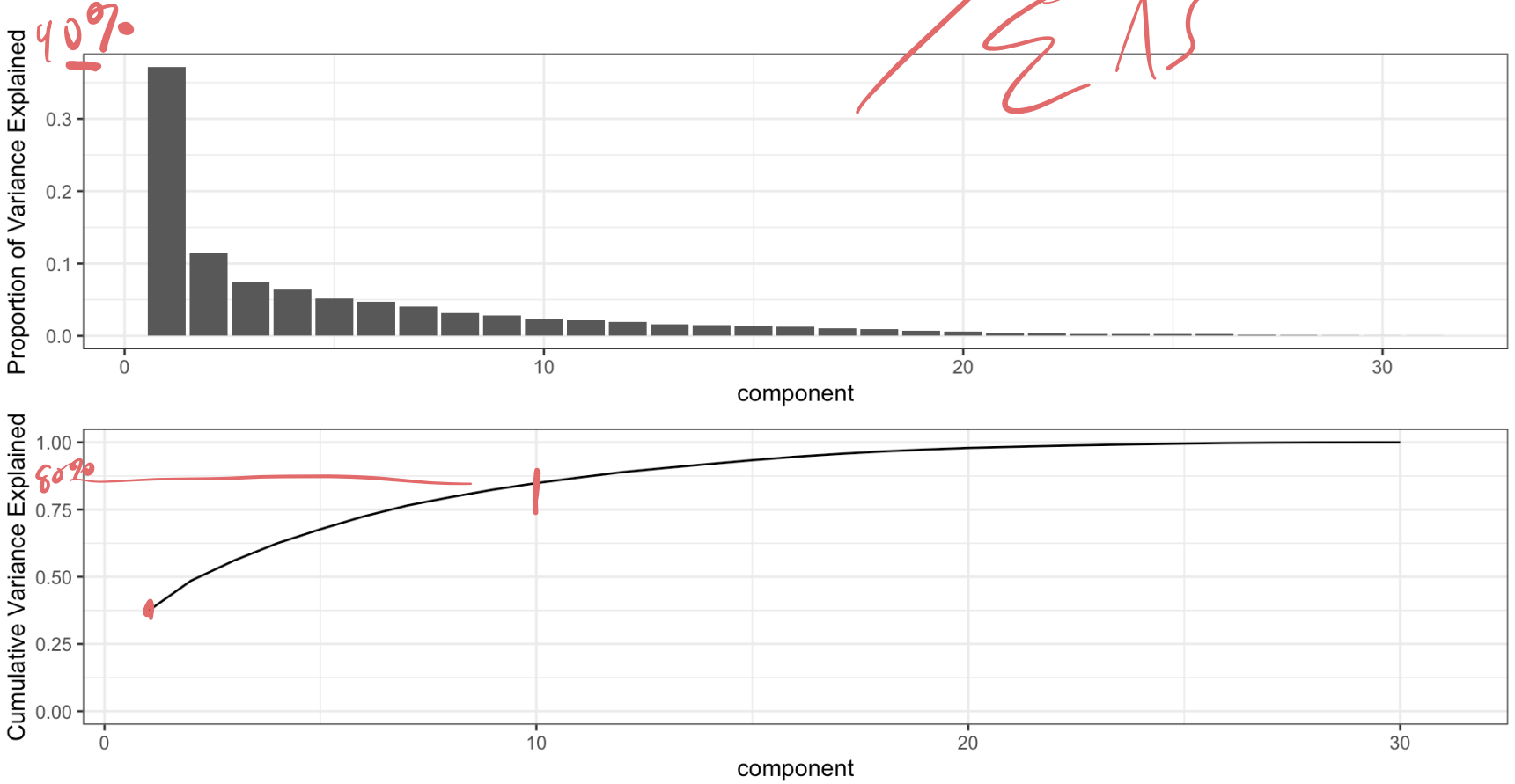
```
# store proportion of variance explained as a dataframe
pcvars <- tibble(component=1:31,
                 variance_explained = pca_result$sdev^2/sum(pca_result$sdev
                 cumulative_var_explained = cumsum(variance_explained))


var_exp_plot <- pcvars |> ggplot() +
  geom_col(aes(x=component, y=variance_explained)) +
  theme_bw() + ylab("Proportion of Variance Explained")

cum_var_exp_plot <- pcvars |> ggplot() +
  geom_line(aes(x=component, y=cumulative_var_explained)) +
  theme_bw() + ylab("Cumulative Variance Explained") + ylim(c(0, 1))

var_exp_plot / cum_var_exp_plot
```

# Variance Ratios




$\lambda_i / \sum \lambda_s$

40%

80%

$PC_i = w_{i1} y_1 + w_{i2} y_2 + \dots w_{i35} y_{35}$

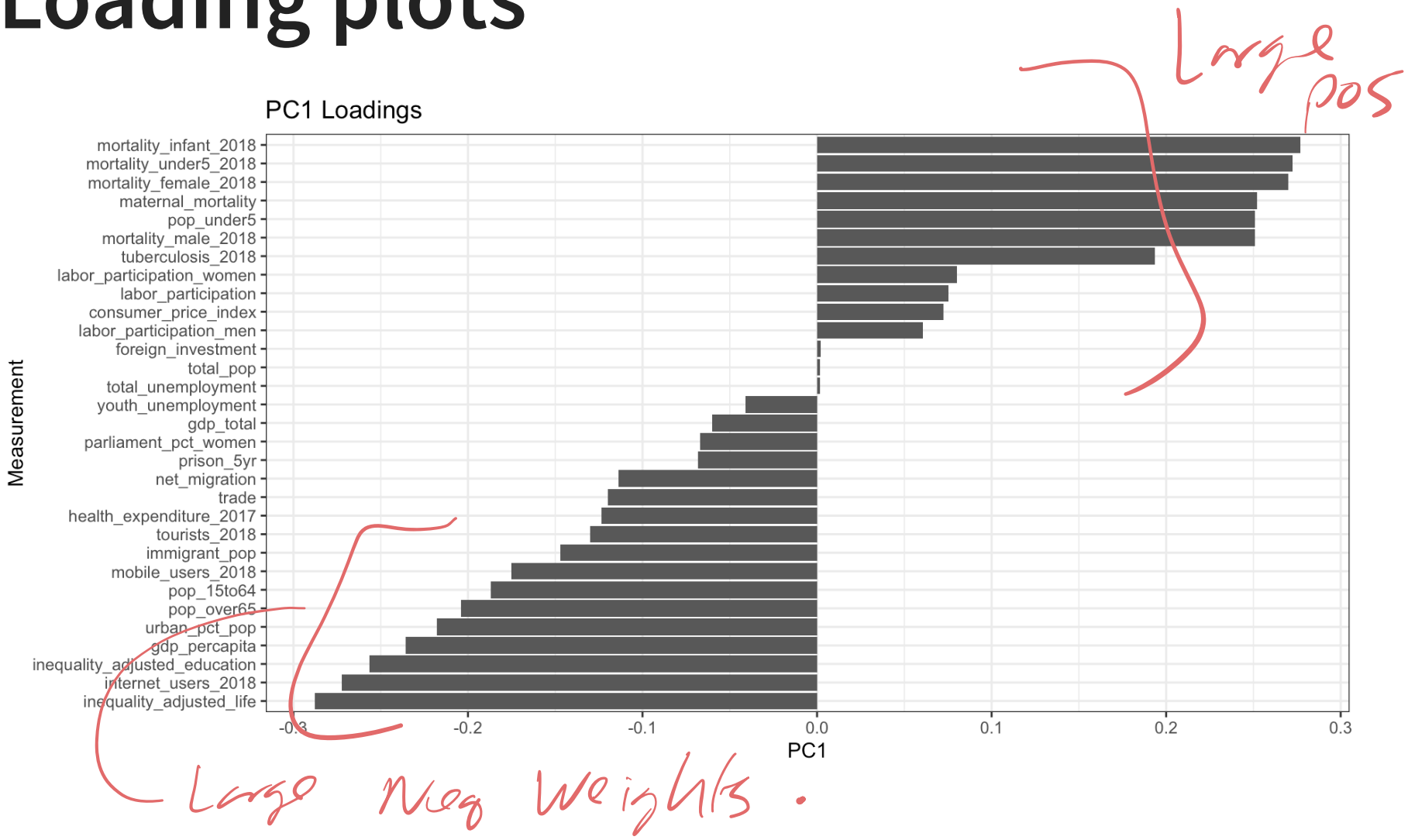
# Loading plots

Examining the loadings can help address the question: *which variables drive variation in the data?*

```r
# store the loadings as a data frame with appropriate names
loading_df <- pca_result$rotation |> as.data.frame() |>
  rownames_to_column(var = "Measurement")

loading_df |> mutate(Measurement = fct_reorder(Measurement, PC1)) |>
  ggplot() + geom_col(aes(x=Measurement, y=PC1)) + coord_flip() +
  ggtitle("PC1 Loadings") +
  theme_bw()
```

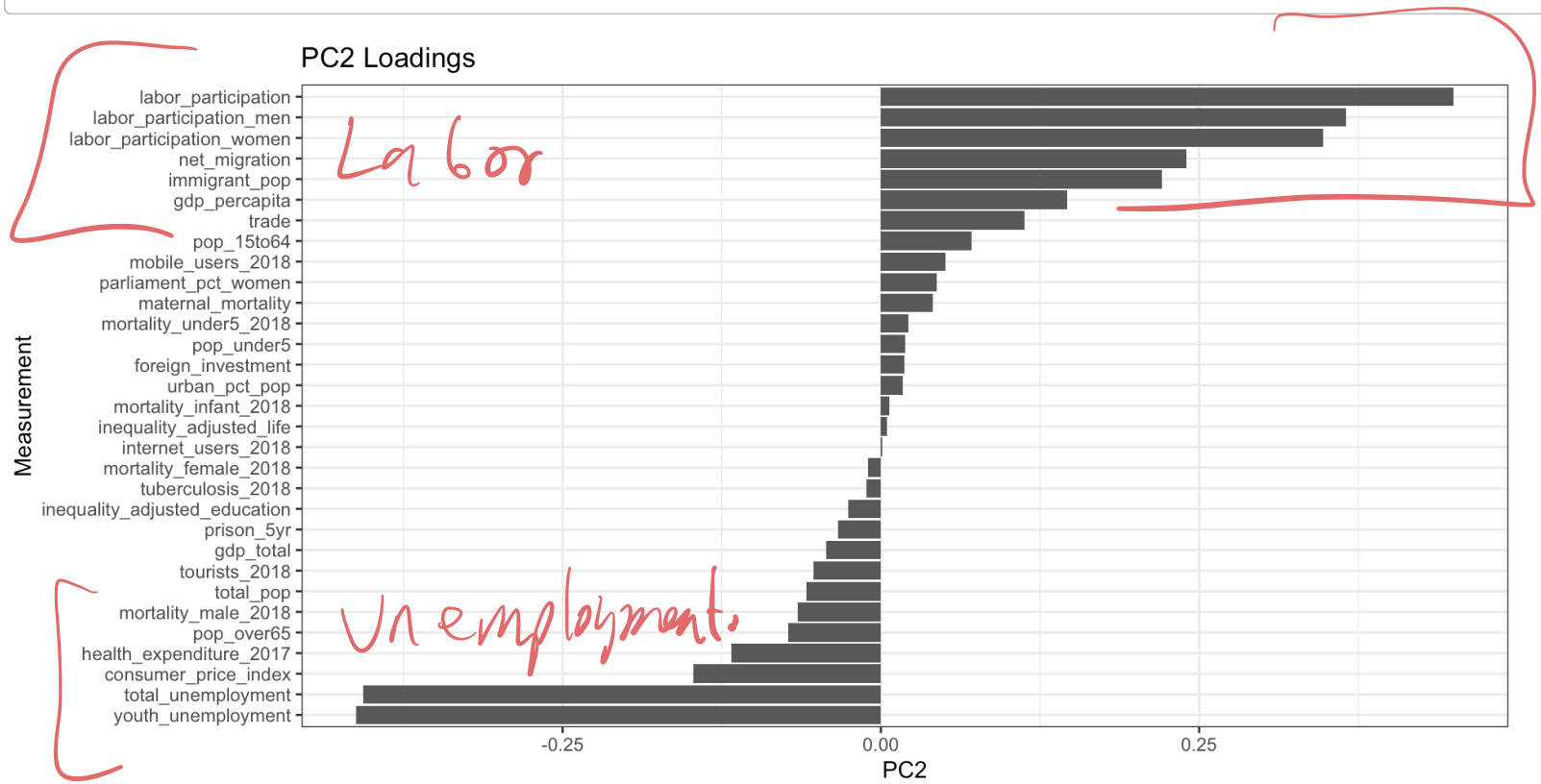
# Loading plots



# Loading Plots

PC 2.

```
1 loading_df |> mutate(Measurement = fct_reorder(Measurement, PC2)) |>
2   ggplot() + geom_col(aes(x=Measurement, y=PC2)) + coord_flip() +
3   ggtitle("PC2 Loadings") +
4   theme_bw()
```



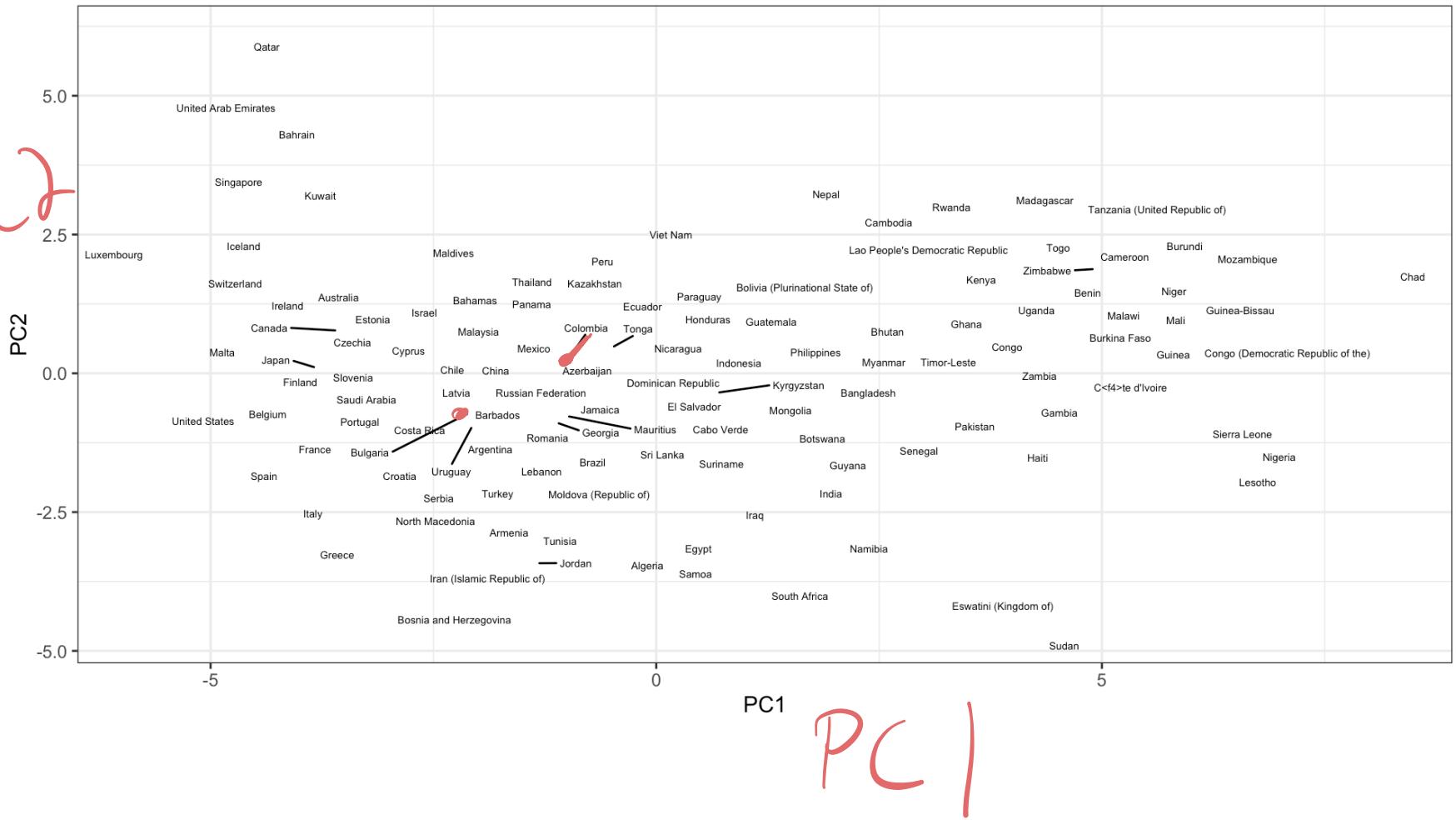
# Visualization

Finally, we might want to use the first two principal components to visualize the data. Often it's helpful to merge the principal components with the original data and apply visualization techniques you already know to search for interesting patterns.

```
# project pcdata onto first two components; store as data frame
projected_data <- pca_result$x |> as.data.frame() |>
  mutate(Country = wdi$country)

projected_data |> ggplot() +
  ggrepel::geom_text_repel(aes(x=PC1, y=PC2, label=Country), size=1.8) + th
```

# Visualization

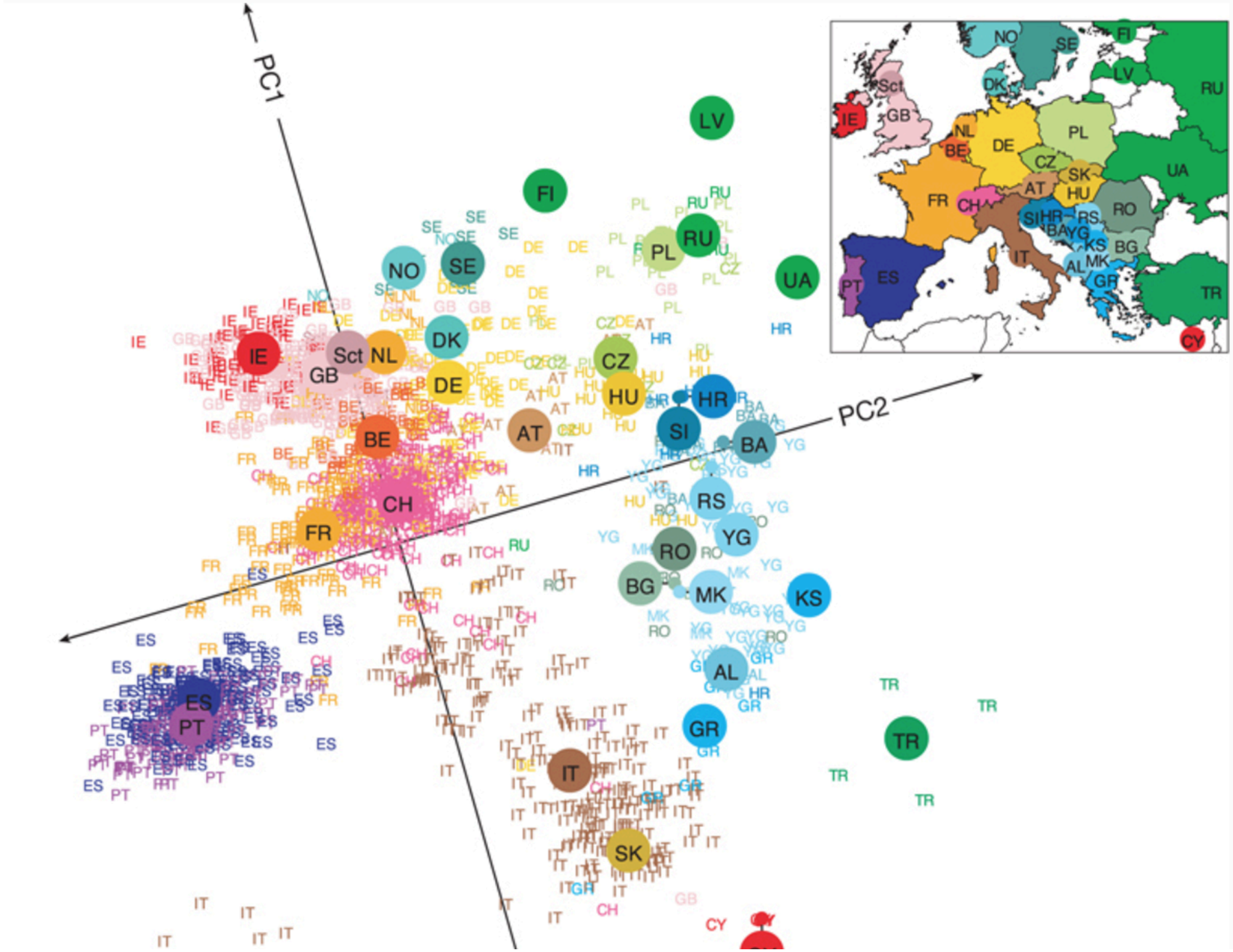


# WDI Data

```
1 wdi |> filter(country %in% c("Qatar", "United States", "Italy", "United Ara
```

```
# A tibble: 5 × 6
  country      labor_participation_…¹ labor_participation_…²
labor_participation
  <chr>                         <dbl>                  <dbl>
<dbl>
1 United Stat…                   56.1                   68.2                  62
2 Italy                          40.8                   59
49.6
3 United Arab…                   52.4                   93.4
82.1
4 Qatar                          56.8                   94.7
86.8
5 Madagascar                     83.4                   88.9
86.1
```

# Genetics and Geography


PC1
PC2
LV
FI
RU
PL
NO
SE
UA
DK
IE
Sct
NL
GB
DE
CZ
HU
HR
SI
BA
AT
BE
CH
FR
RS
YG
RO
BG
MK
KS
AL
ES
PT
GR
TR
IT
SK
CY
NO
SE
FI
LV
RU
Sct
DK
IE
GB
NL
PL
DE
BE
CZ
SK
HU
AT
UA
FR
CH
SI
HR
RS
BA
YG
RO
KS
BG
IT
MK
AL
GR
ES
PT
TR
CY

# Summary

Our focus this week was on introducing PCA as an exploratory technique for understanding covariation in mulitvariate data.

By way of introduction, we discussed **covariation**, the tendency of values of multiple variables to change together.

- Measures of covariation: covariance and correlation
- Correlation matrices and heatmap visualization

We then moved on to **eigendecomposition**, a method of factoring matrices.

- For correlation (and covariance) matrices, eigendecomposition produces an orthonormal basis.
- Representing the data on this basis – a linear transformation – ‘captures’ covariation.

# Summary

This provided the background needed to understand **principal components analysis** (PCA) – finding covariation-preserving linear transformations of data.

- Technically amounts to eigendecomposition of a correlation matrix.
- Useful for identifying variables that drive covariation, and for visualizing multivariate data.