MATH 250: Mathematical Data Visualization

Visualizing high dimensional data

Peter A. Gao

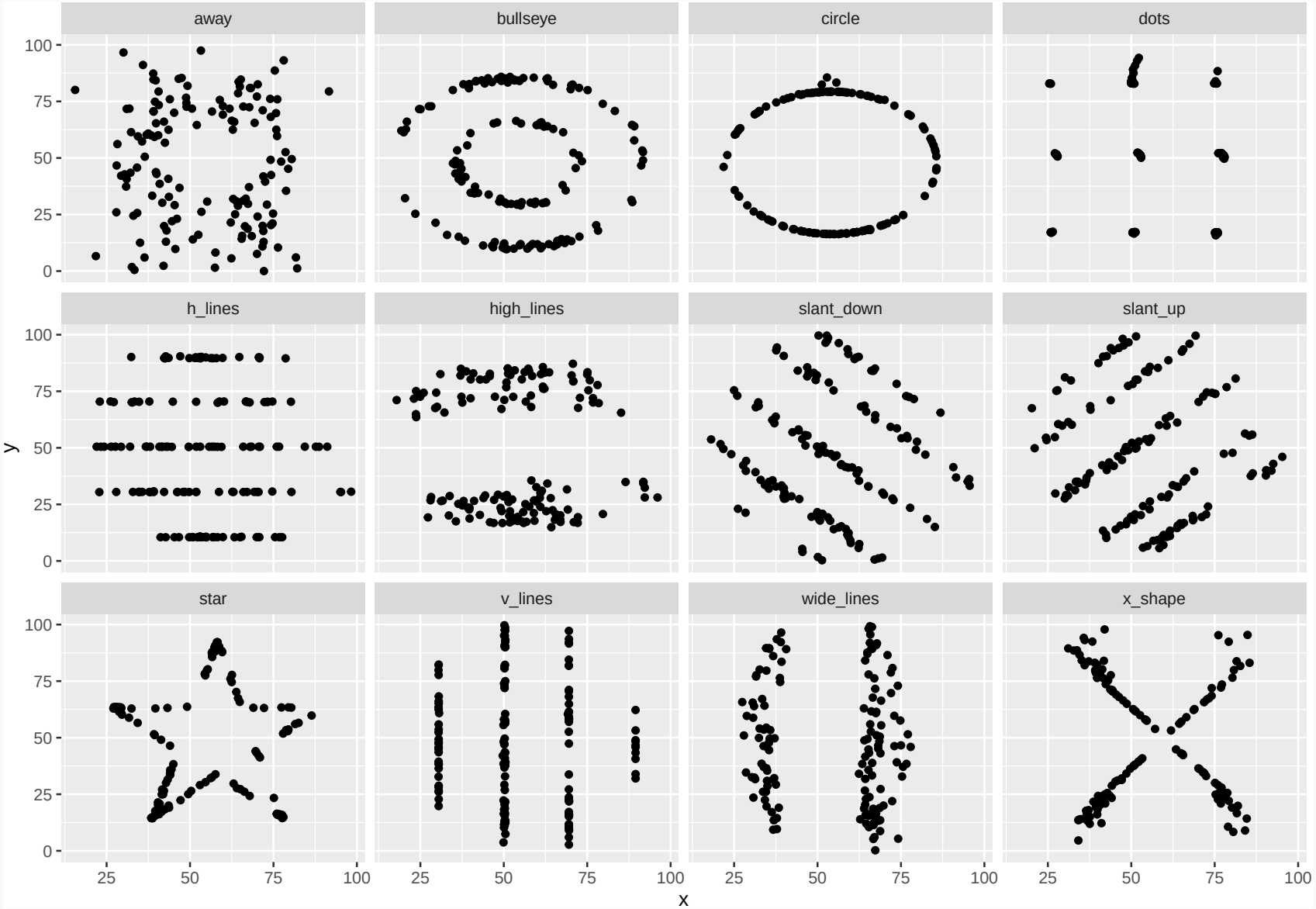
2025-03-26

San José State University

Outline

- Principles of data visualization
- Multidimensional scaling (adapted from lecture by Guangliang Chen)
- Linear and nonlinear embeddings

Why look at data?

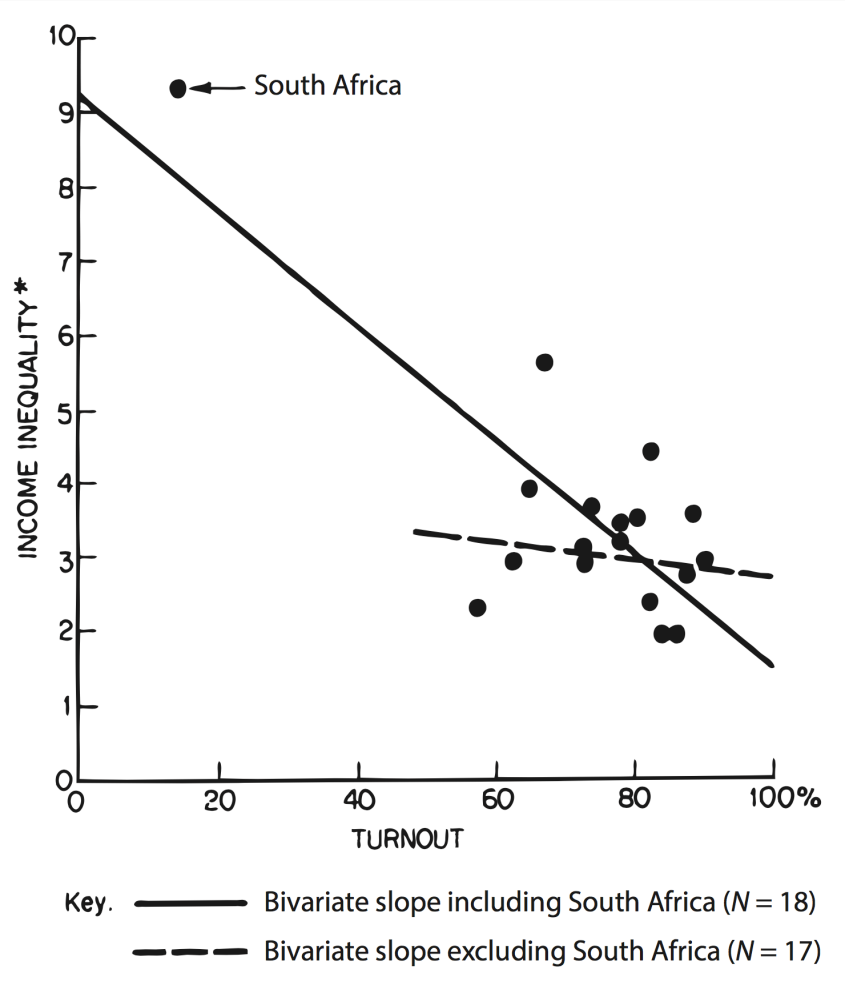


Why look at data?

All of these datasets have the same means for `x` and `y`, and (approximately) the same correlation:

```
# A tibble: 13 x 4
   dataset    mean_x mean_y correlation
   <chr>       <dbl>  <dbl>       <dbl>
 1 away         54.3   47.8        -0.1
 2 bullseye     54.3   47.8        -0.1
 3 circle       54.3   47.8        -0.1
 4 dino         54.3   47.8        -0.1
 5 dots         54.3   47.8        -0.1
 6 h_lines      54.3   47.8        -0.1
 7 high_lines   54.3   47.8        -0.1
 8 slant_down   54.3   47.8        -0.1
 9 slant_up     54.3   47.8        -0.1
10 star         54.3   47.8        -0.1
11 v_lines      54.3   47.8        -0.1
12 wide_lines   54.3   47.8        -0.1
13 x_shape      54.3   47.8        -0.1
```

Why look at data?



from Jackson (1980) as reproduced in Kieran Healy's *Data Visualization*.

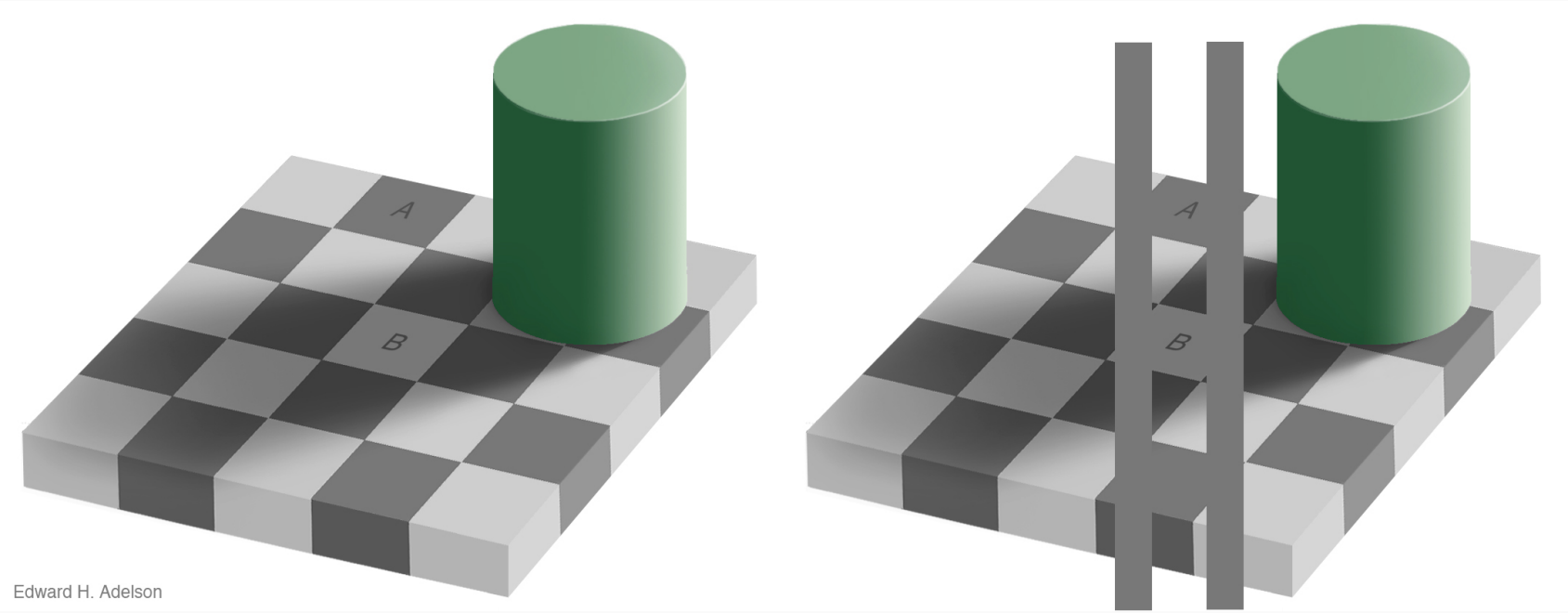
What makes a good data visualization?

1. Legibility: Is it easy for the viewer to understand and interpret your visual?
2. Integrity: Does the visual accurately communicate some insight about the data?
3. Beauty: Is the visual appealing and uncluttered?

Other Readings:

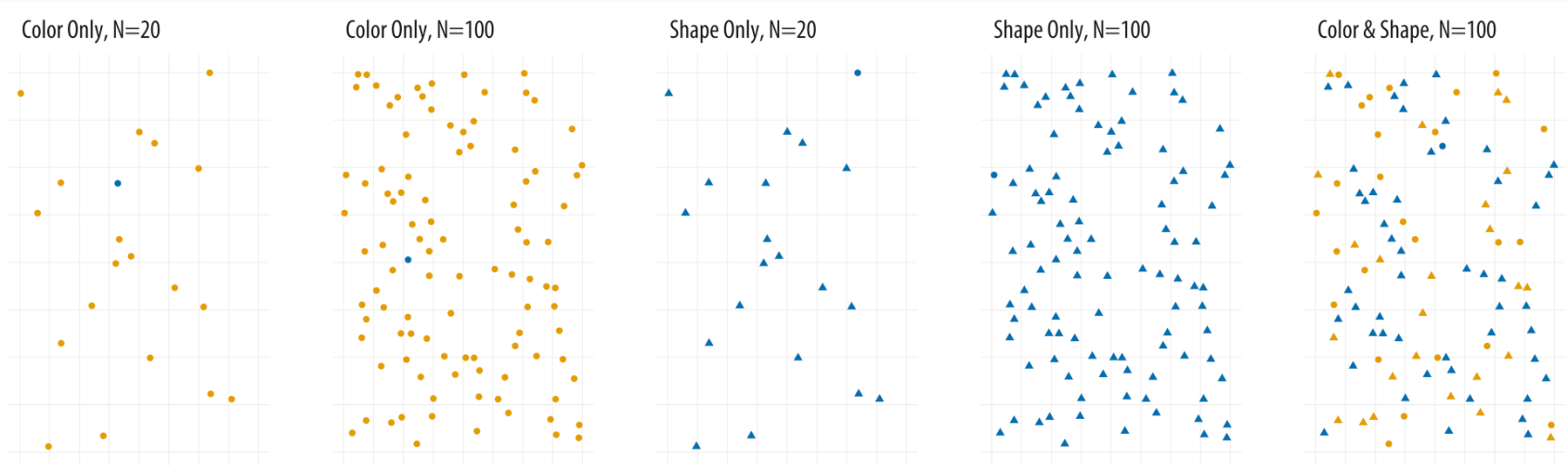
- Information is Beautiful

Legibility

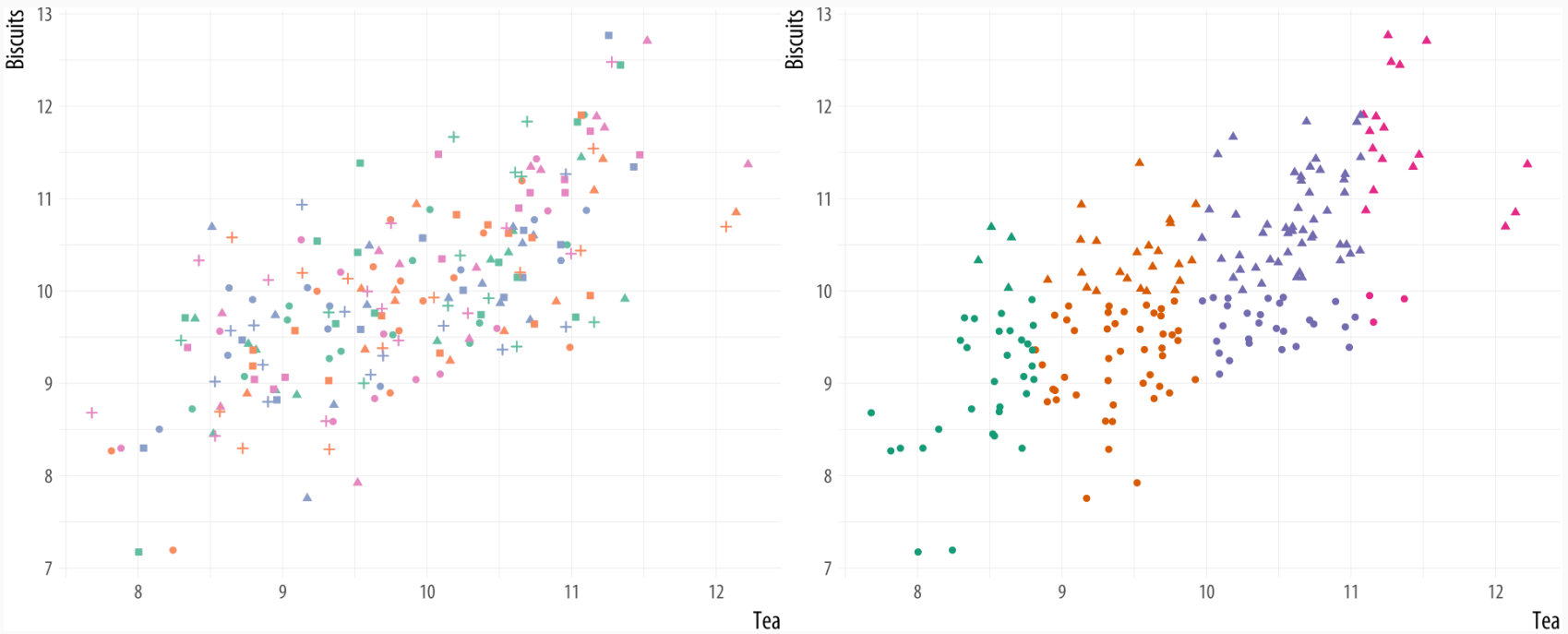


from Edward H. Adelson as reproduced in Kieran Healy's *Data Visualization*.

Legibility

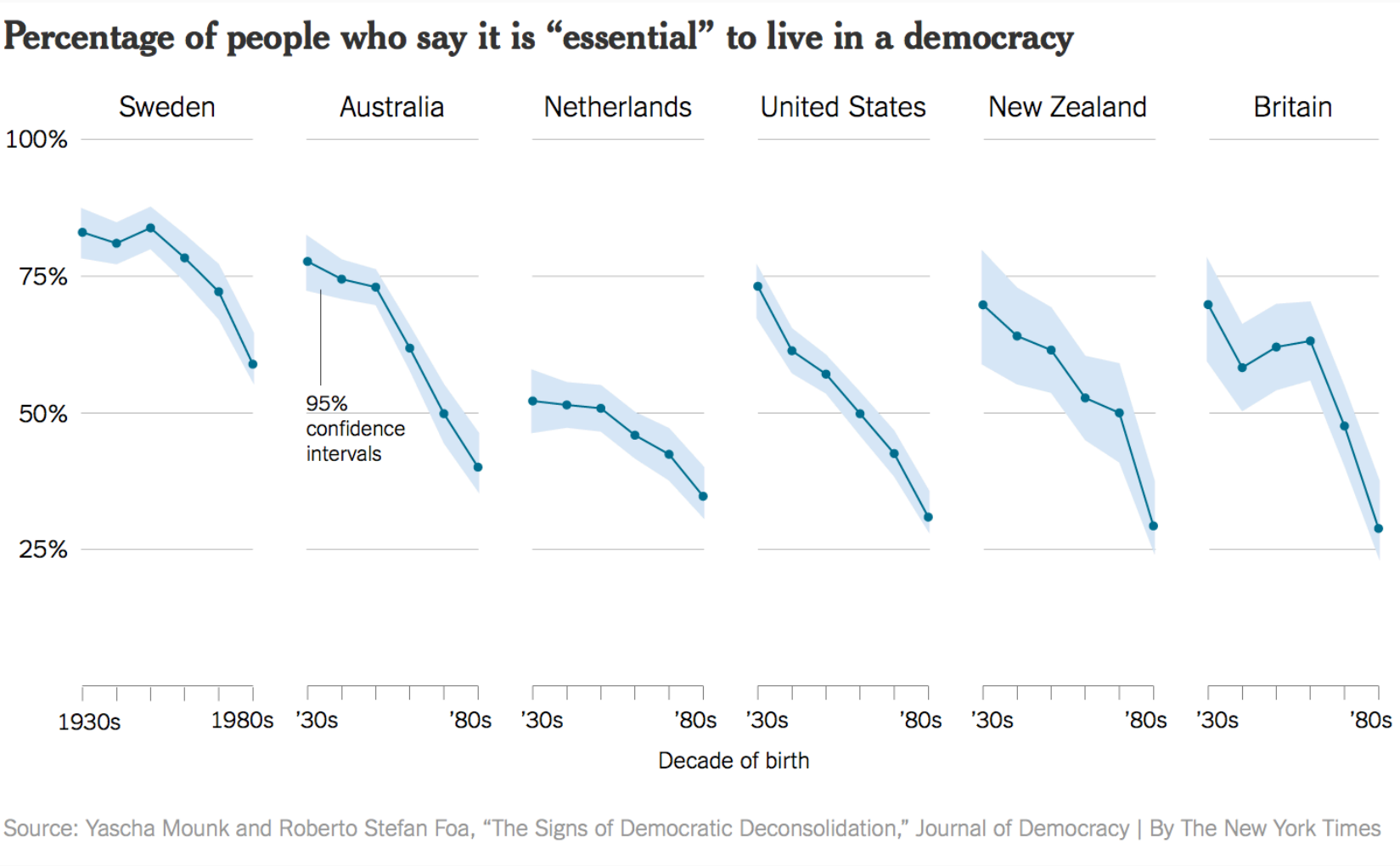

Color Only, N=20
Color Only, N=100
Shape Only, N=20
Shape Only, N=100
Color & Shape, N=100

Legibility

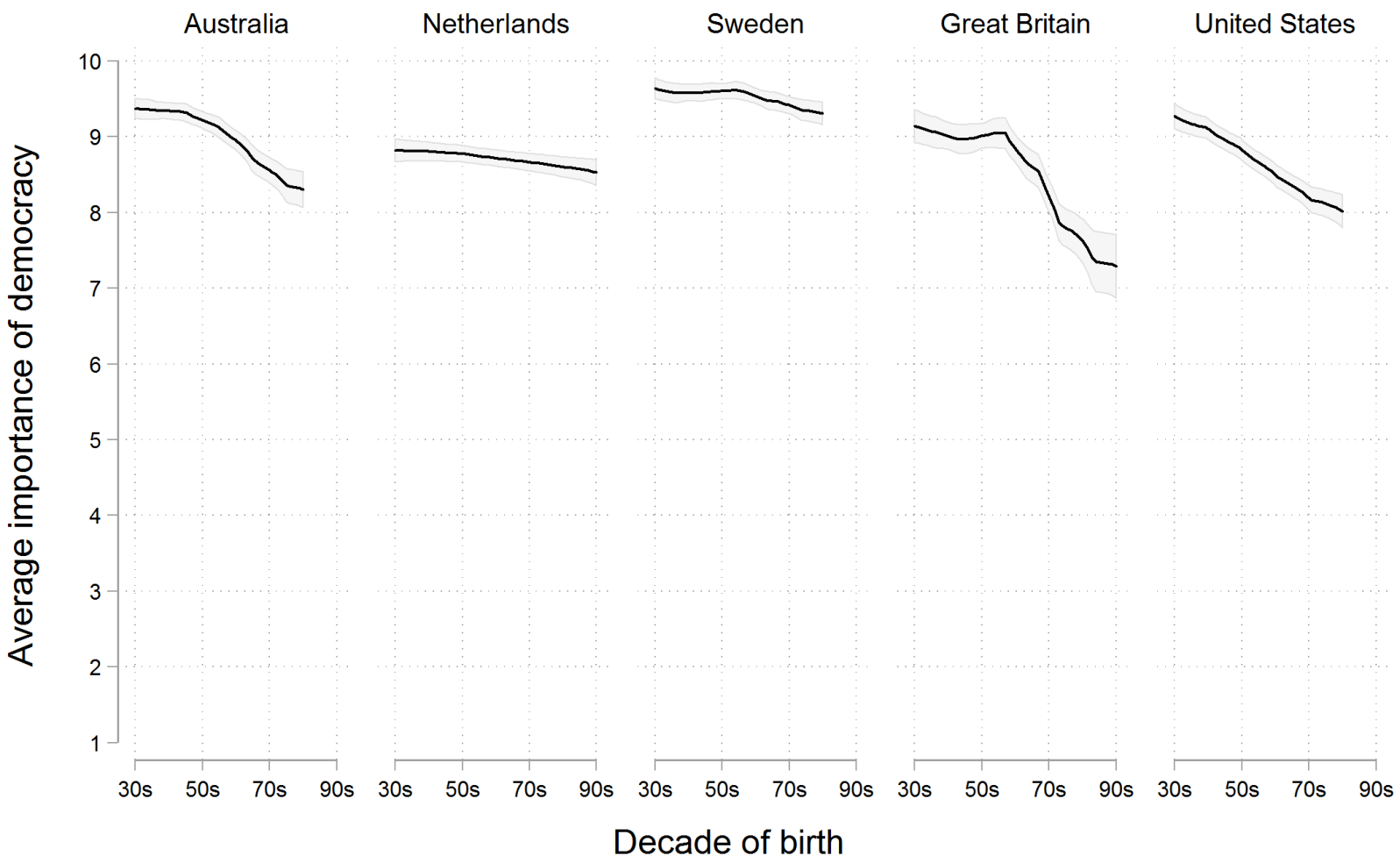


from Kieran Healy's *Data Visualization*.

Integrity


Percentage of people who say it is “essential” to live in a democracy
Sweden
Australia
Netherlands
United States
New Zealand
Britain
100%
75%
50%
25%
95% confidence intervals
1930s
1980s
'30s
'80s
'30s
'80s
'30s
'80s
'30s
'80s
'30s
'80s
Decade of birth
Source: Yascha Mounk and Roberto Stefan Foa, “The Signs of Democratic Deconsolidation,” Journal of Democracy | By The New York Times

Integrity



Graph by Erik Voeten, based on WVS 5

Example: Movies data

```
movies <- readr::read_csv(
  paste0("https://raw.githubusercontent.com/",
         "rfordatascience/tidytuesday/master/",
         "data/2018/2018-10-23/movie_profit.csv")
)
```

```
# A tibble: 6 x 9
   ...1 release_date movie  production_budget domestic_gross
  <dbl> <chr>        <chr>              <dbl>          <dbl>
1     1 6/22/2007    Evan ~         175000000      100289690
2     2 7/28/1995    Water~         175000000       88246220
3     3 5/12/2017    King ~         175000000       39175066
4     4 12/25/2013   47 Ro~         175000000       38362475
5     5 6/22/2018    Juras~         170000000      416769345
6     6 8/1/2014     Guard~         170000000      333172112
# i 4 more variables: worldwide_gross <dbl>,
#   distributor <chr>, mpaa_rating <chr>, genre <chr>
```

Univariate visualizations

Univariate visualizations describe **one** variable at a time.

For categorical variables, usually bar plots or pie charts are used.

For numerical variables, usually histograms, box plots, or density plots are used.

Summarizing categorical data with a table

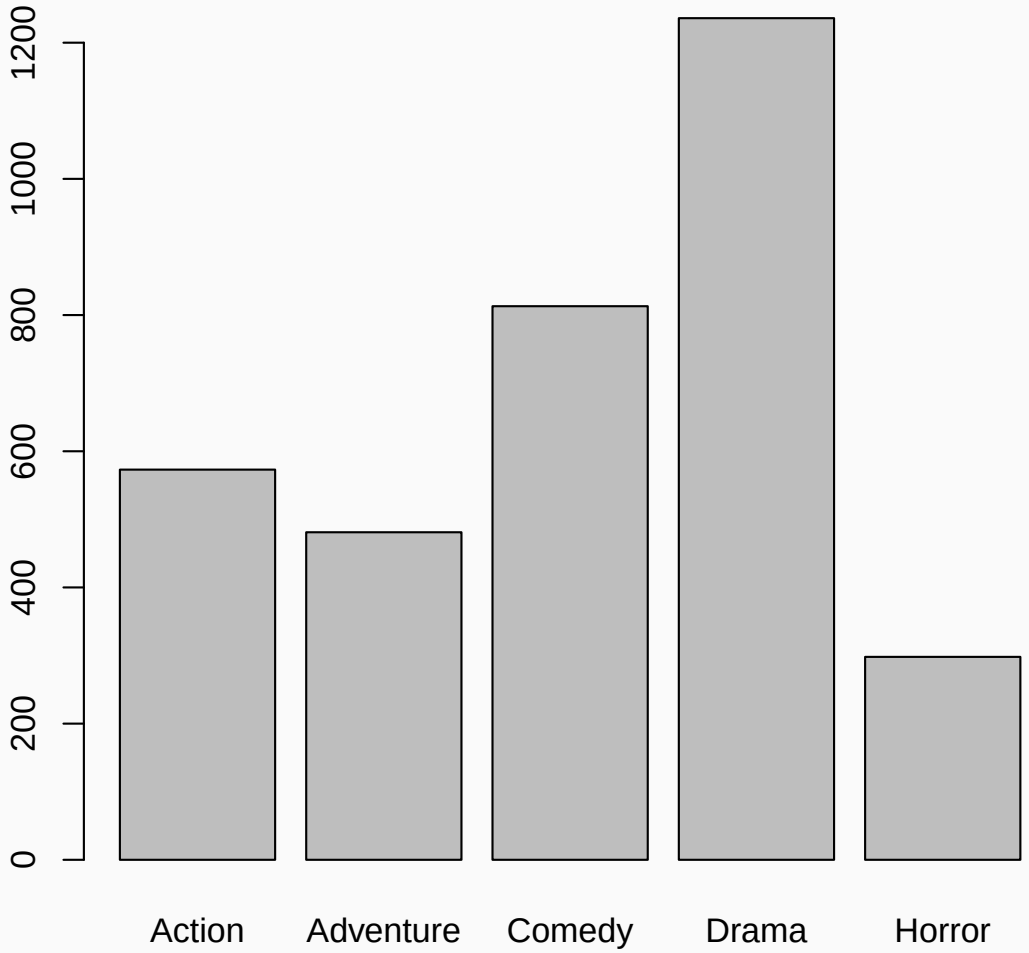
```
table(movies$genre)
```

```
   Action Adventure     Comedy      Drama     Horror
      573       481        813       1236        298
```

Visualizing a categorical variable

The `barplot` function takes a **named numeric vector** as its first argument:

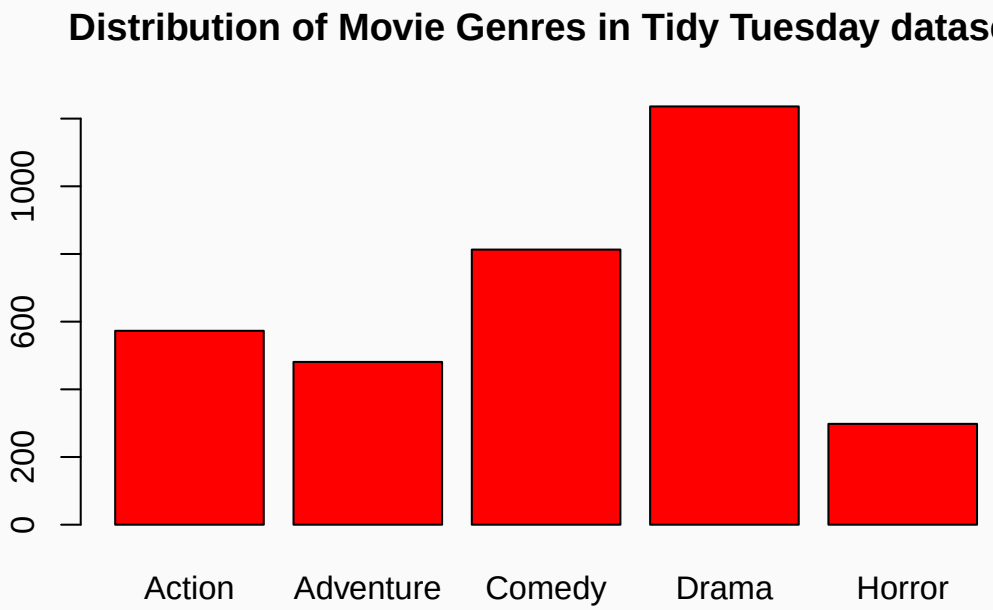
```
barplot(height = table(movies$genre))
```



Customizing base R plots

Base R plotting functions have many optional arguments allowing customization:

```
barplot(
  height = table(movies$genre),
  main = "Distribution of Movie Genres in Tidy Tuesday dataset",
  col = "red"
)
```



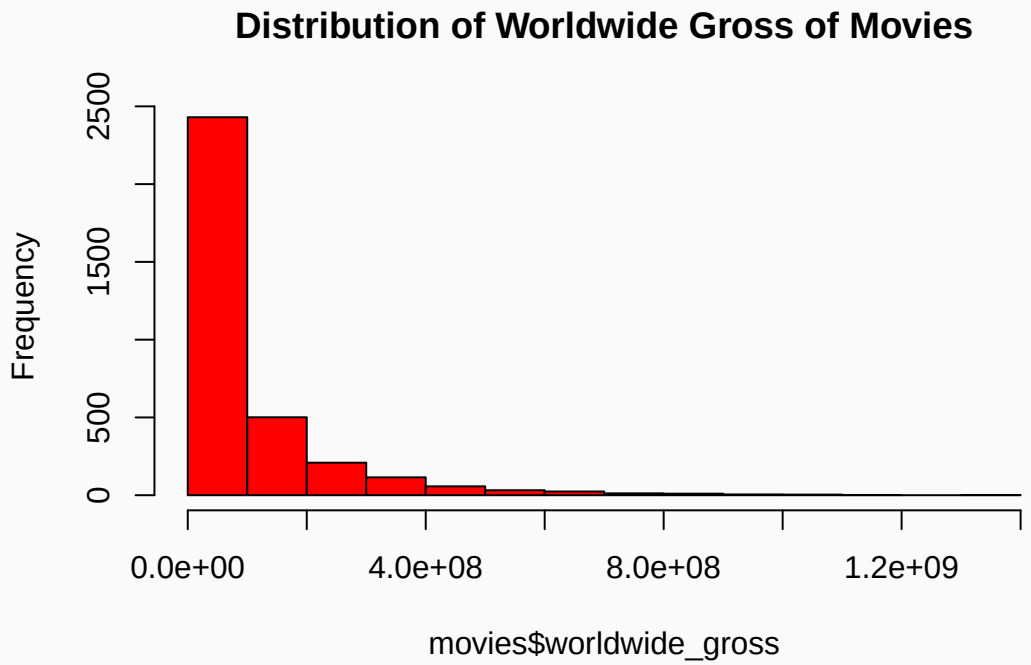
Summarizing a numerical variable

```
summary(movies$worldwide_gross)
```

```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.000e+00 1.062e+07 4.016e+07 9.412e+07 1.176e+08 1.305e+09
```

Visualizing a numerical variable

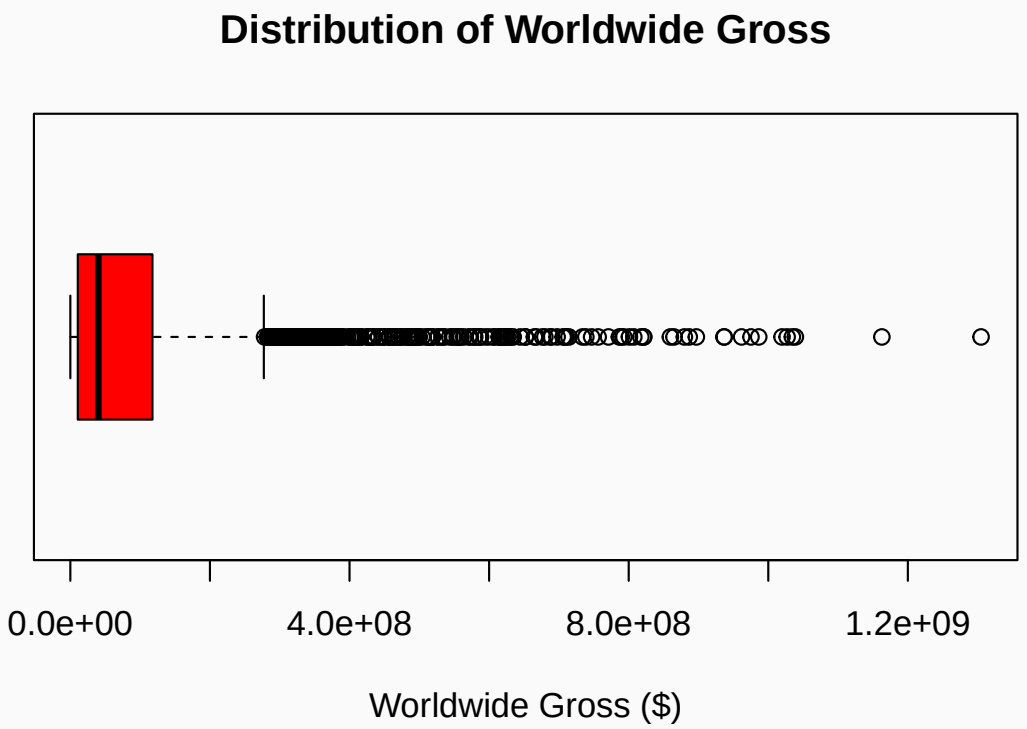
```
hist(
  x = movies$worldwide_gross,
  main = "Distribution of Worldwide Gross of Movies",
  col = "red"
)
```



What shape does this distribution have?

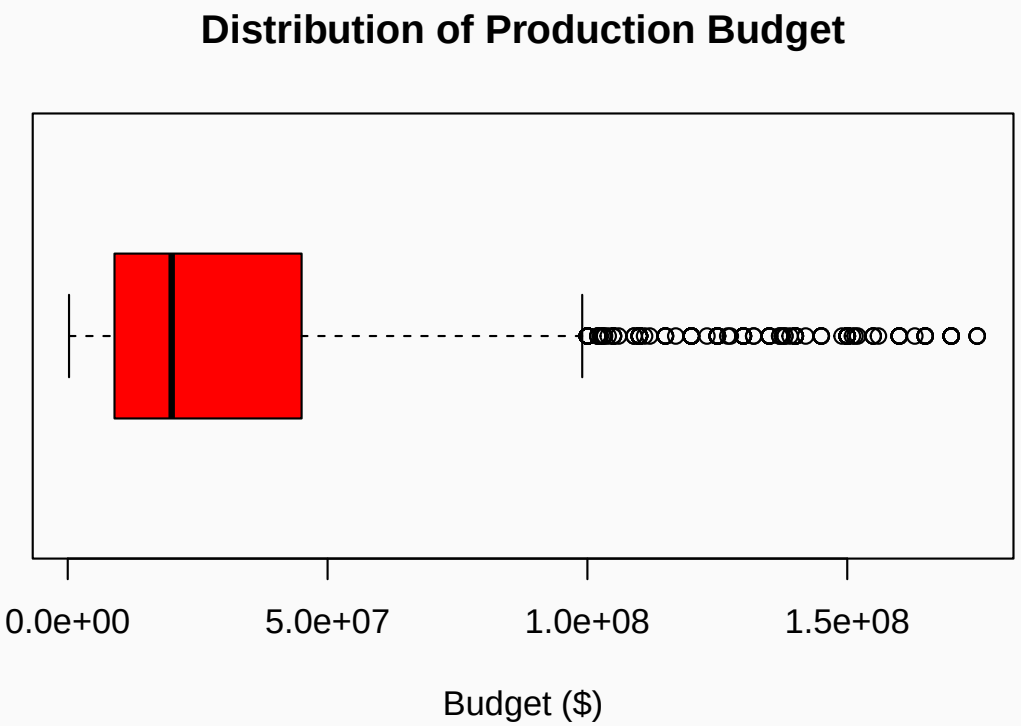
Visualizing a numerical variable

```
boxplot(
  x = movies$worldwide_gross,
  main = "Distribution of Worldwide Gross",
  xlab = "Worldwide Gross ($)",
  col = "red",
  horizontal = T
)
```



Visualizing a numerical variable

```
boxplot(
  x = movies$production_budget,
  main = "Distribution of Production Budget",
  xlab = "Budget ($)",
  col = "red",
  horizontal = T
)
```



Multivariate visualizations

How might we plot the relationship between a numerical variable and a categorical variable?

How about between two categorical variables? Between two numerical variables?

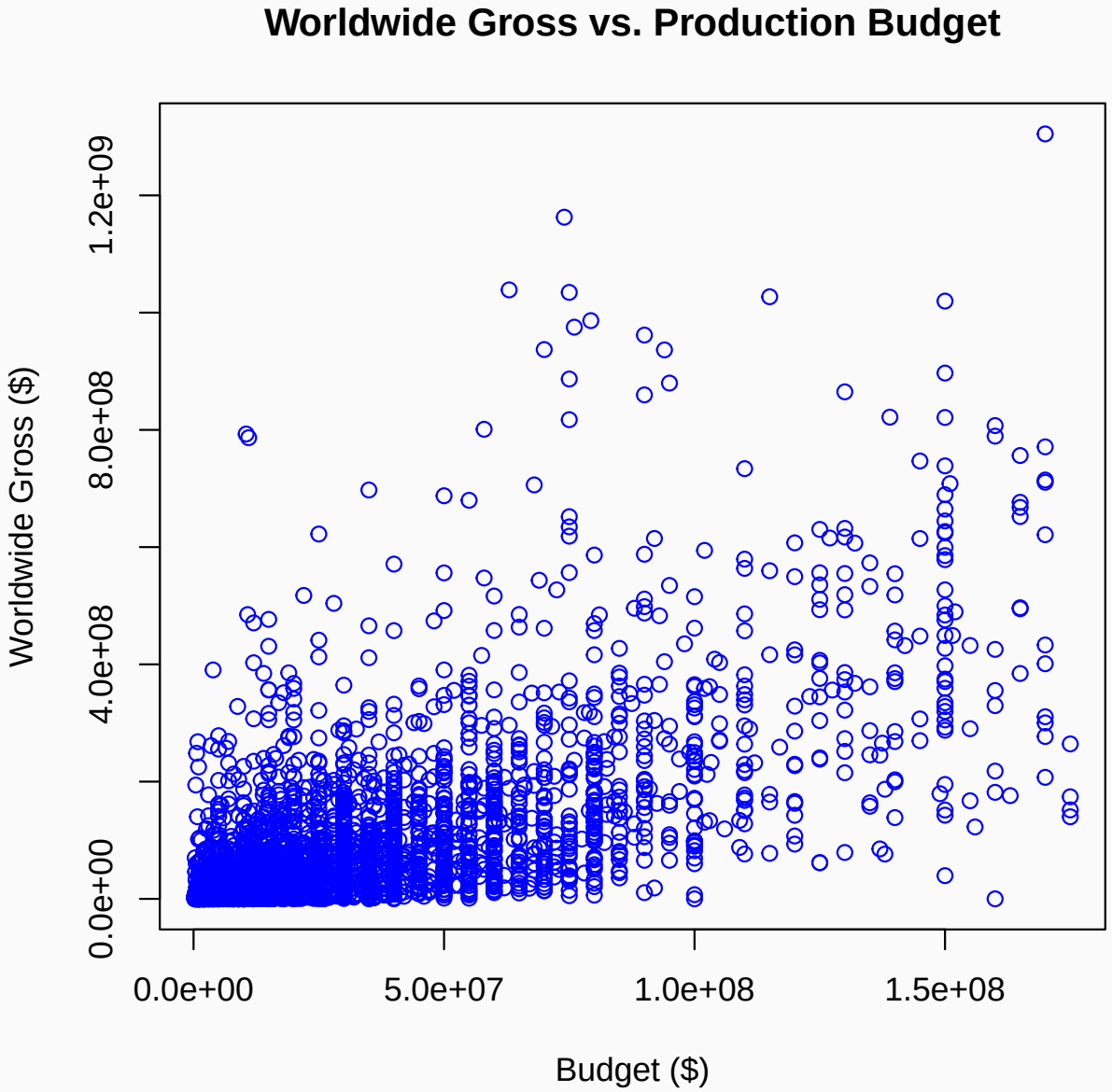
What if we want to visualize three variables at once?

Visualizing two numerical variables

The `plot` function can be used to obtain scatter plots for two numerical variables:

```
plot(
  x = movies$production_budget,
  y = movies$worldwide_gross,
  main = "Worldwide Gross vs. Production Budget",
  xlab = "Budget ($)",
  ylab = "Worldwide Gross ($)",
  col = "blue",
)
```

Visualizing two numerical variables



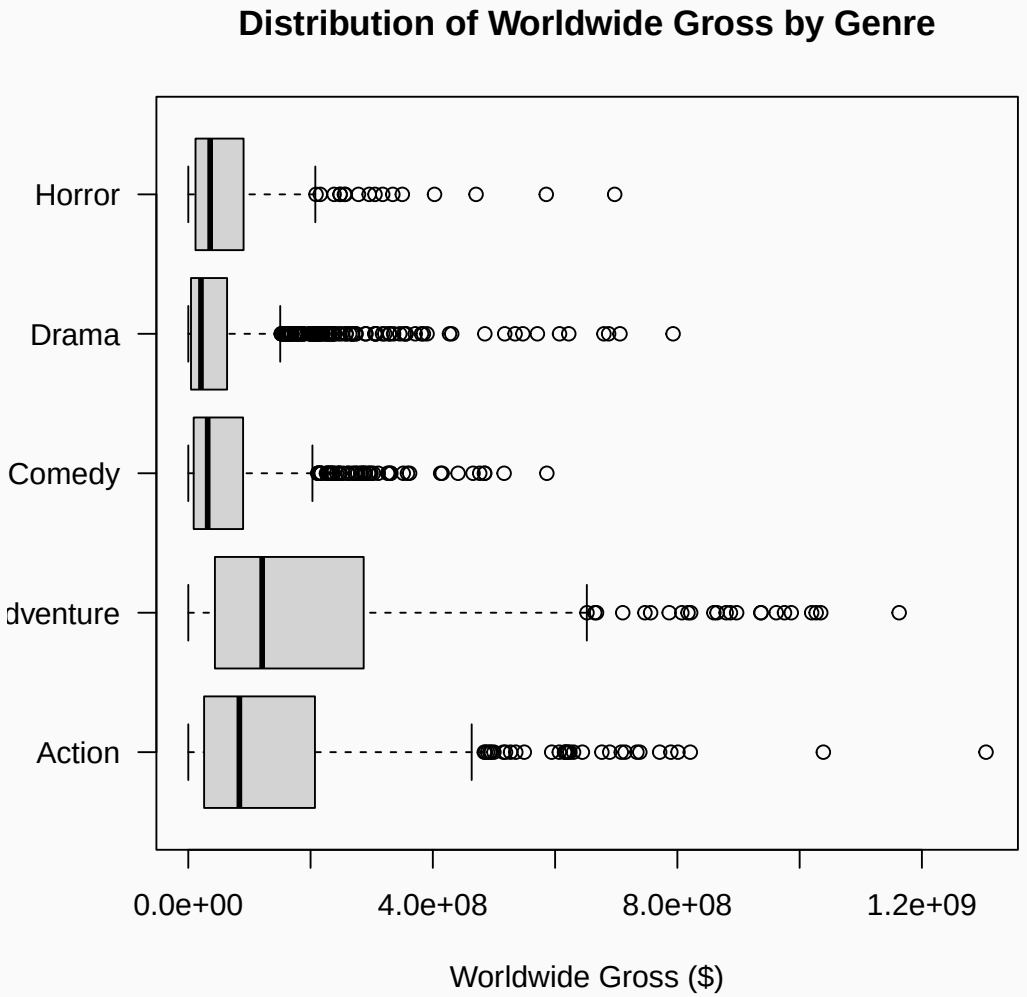
Visualizing a numerical variable and a categorical variable

The `boxplot` function can be used to obtain stacked boxplots:

```
boxplot(worldwide_gross ~ genre,
        data = movies,
        main = "Distribution of Worldwide Gross by Genre",
        xlab = "Worldwide Gross ($)",
        ylab = "",
        horizontal = T,
        las = 1)
```

Visualizing a numerical variable and a categorical variable

The `boxplot` function can be used to obtain stacked boxplots:



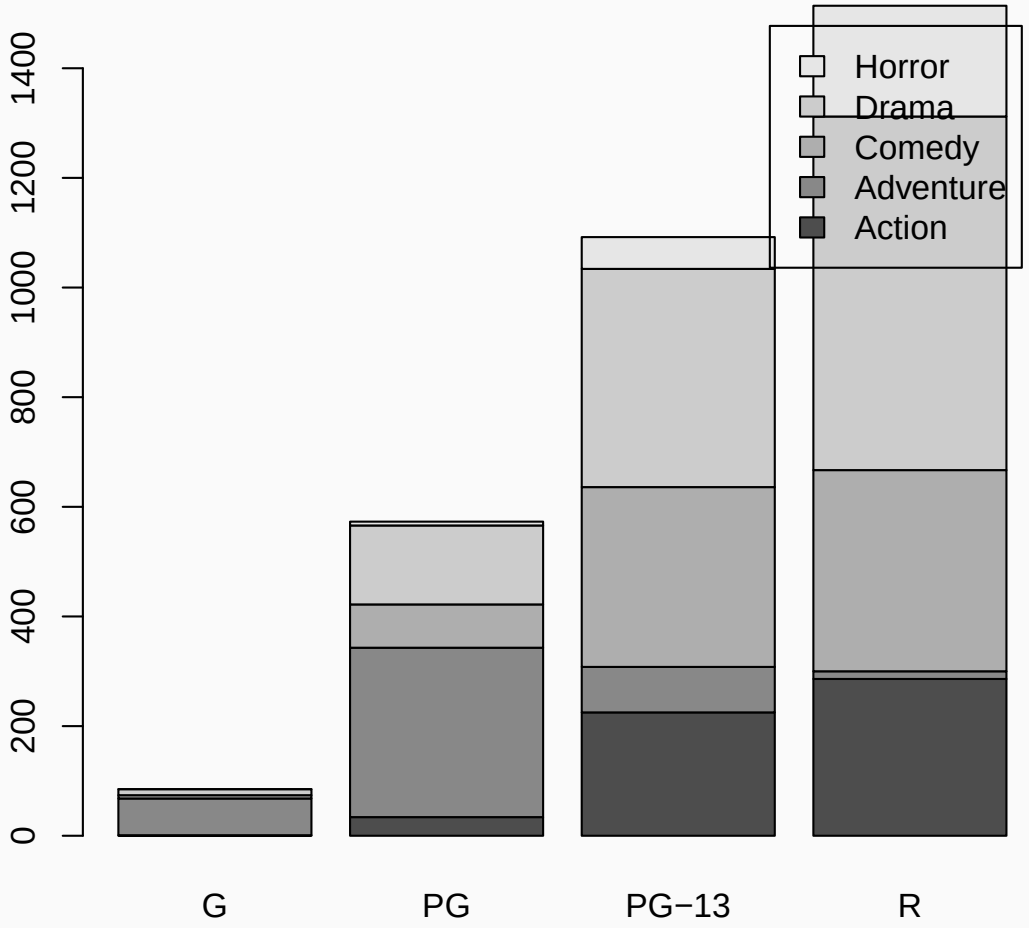
Visualizing two categorical variables

```
table(movies$genre, movies$mpaa_rating)
```

```
             G  PG PG-13   R
  Action     1  34   225 286
  Adventure 67 309    83  14
  Comedy     6  79   328 367
  Drama     11 144   398 645
  Horror     0   7    58 202
```

Visualizing two categorical variables

```
barplot(table(movies$genre, movies$mpaa_rating),
        legend =  c("Action", "Adventure", "Comedy", "Drama", "Horror"))
```



General principles for data visualization

Avoid:

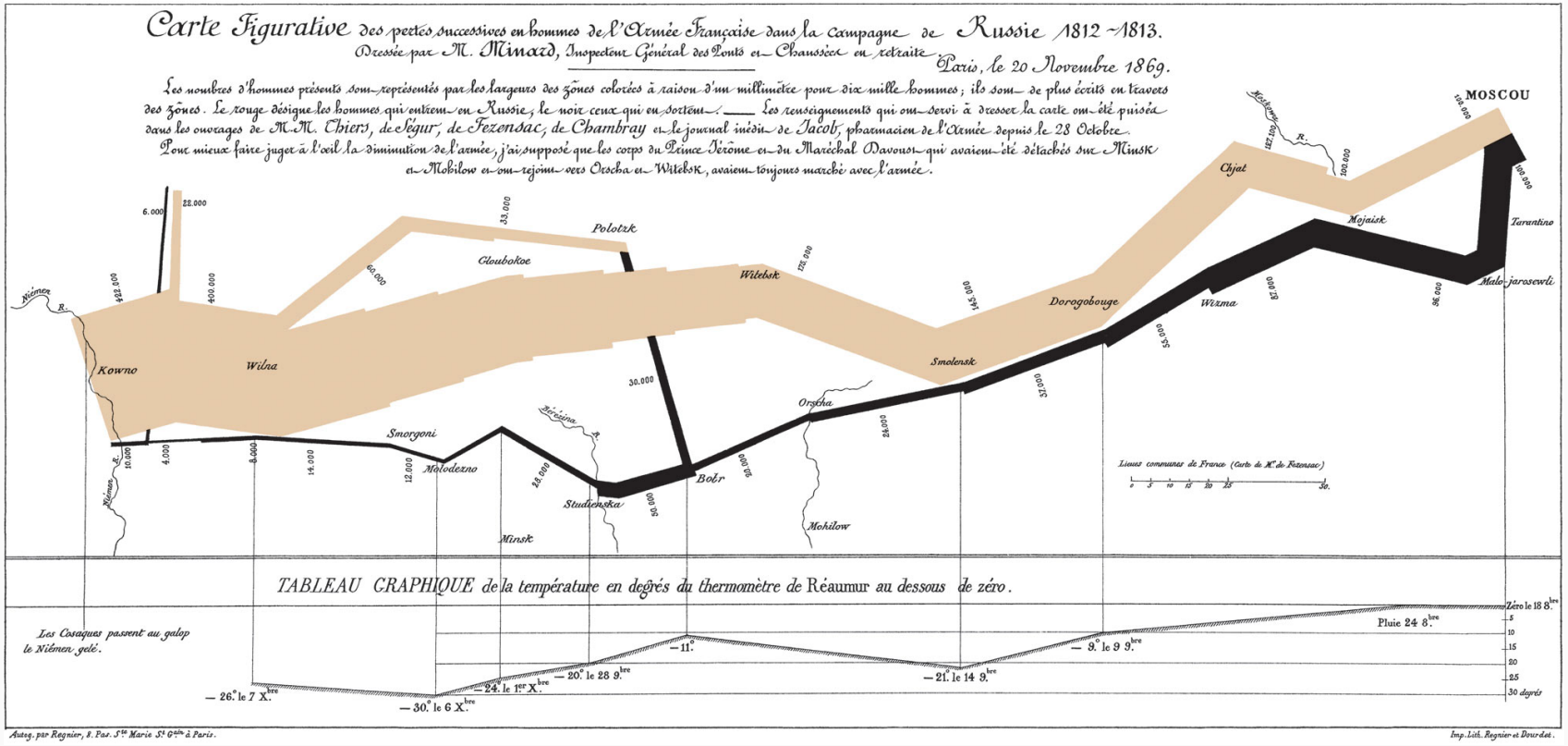
- 3D plots
- Deceptive axes
- Excessive labels
- Excessive/bad coloring
- Fancy shading/effects
- Bad variable/axis names
- Unreadable labels
- Overloaded with information

General principles for data visualization

Strive for:

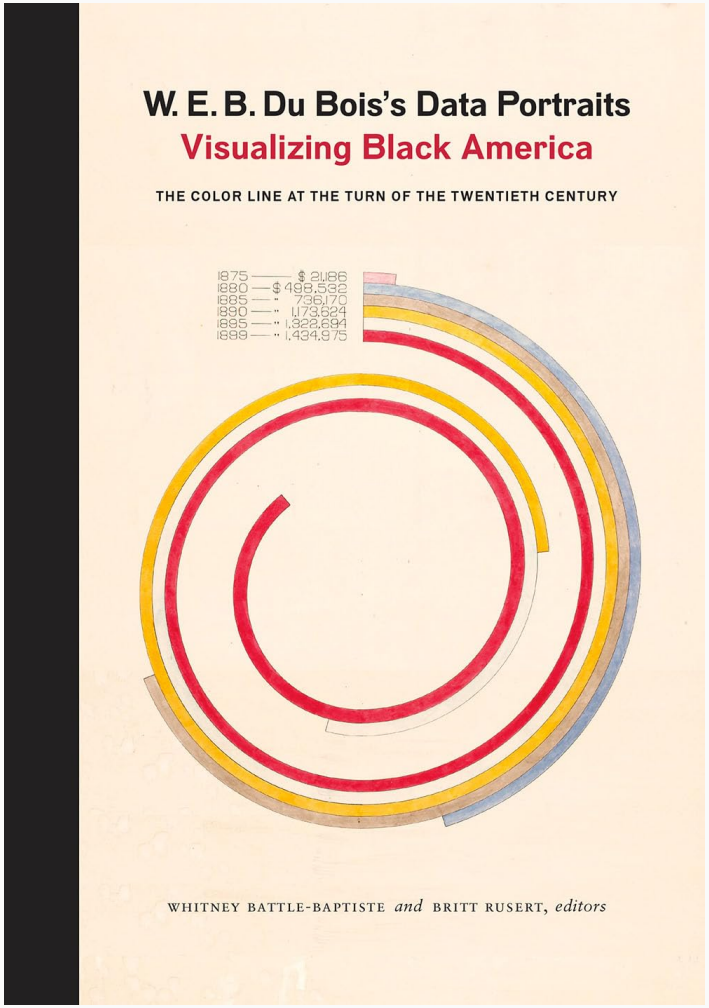
- Simple, clean graphics
- Neat and human readable text
- Appropriate data range (bar charts should *always* start from 0!)
- Consistent intervals
- Roughly ~6 colors or less
- Size figures appropriately

Sometimes breaking the rules is okay...



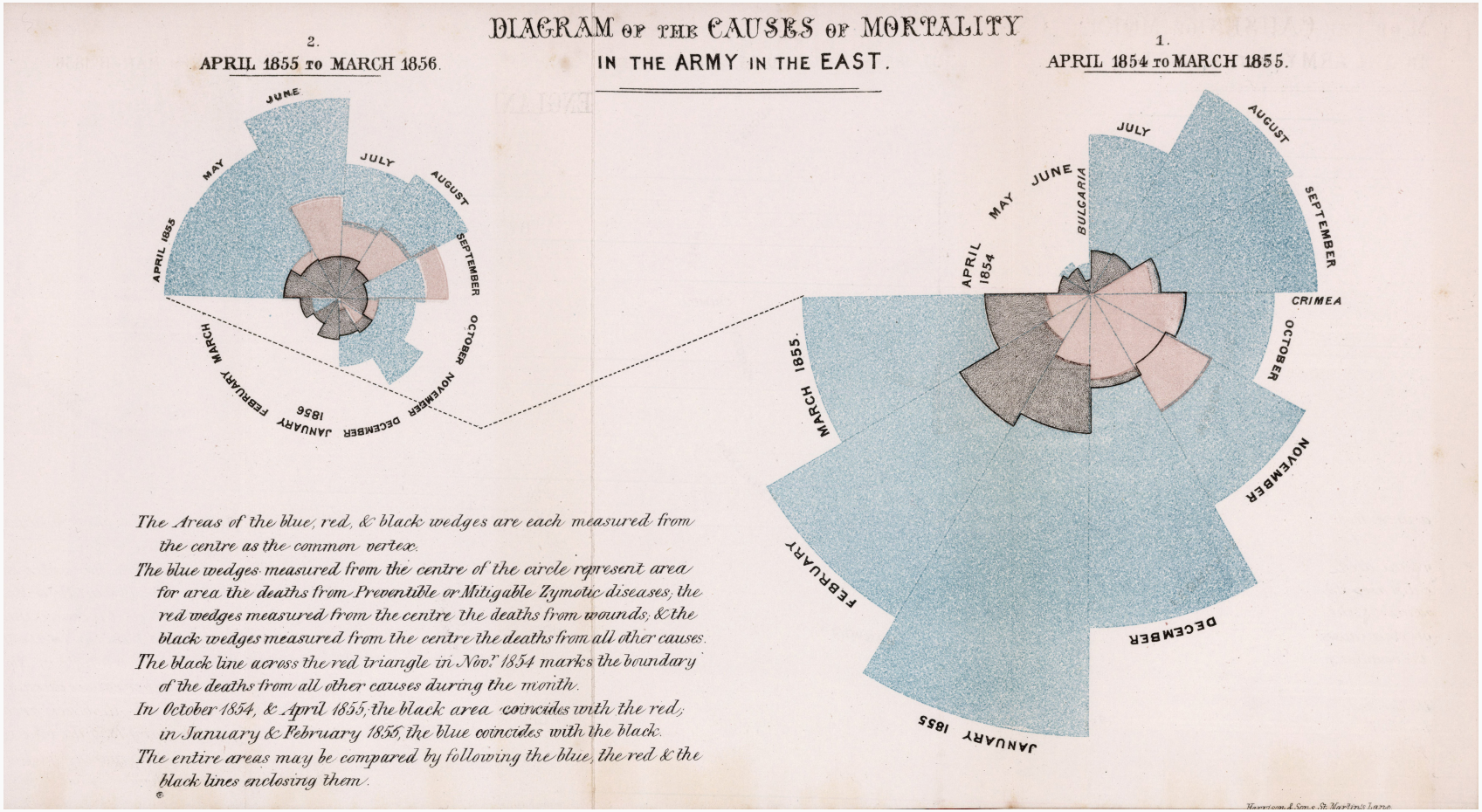
Charles Minard

Sometimes breaking the rules is okay...



W.E.B. DuBois

Sometimes breaking the rules is okay...

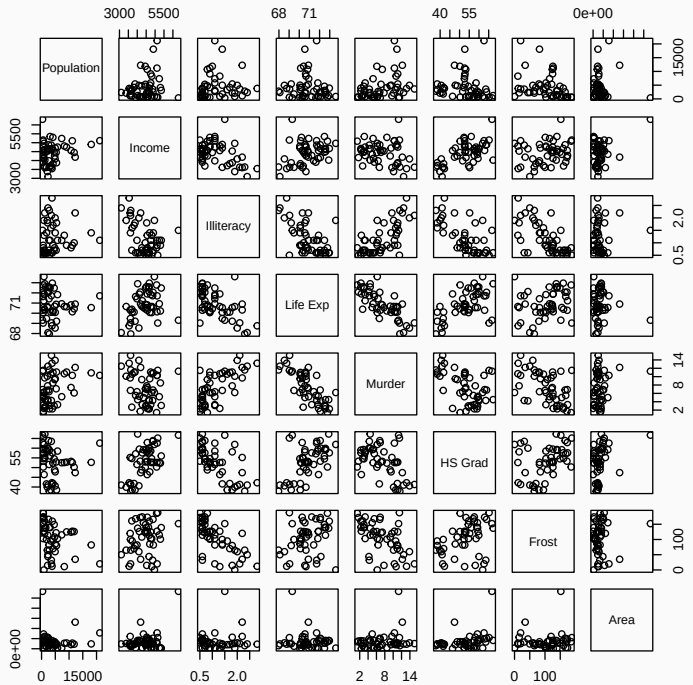


Florence Nightingale

Looking at high dimensional data

Our focus will be on visualizing high dimensional data. When p is relatively small, strategies like looking at paired scatter plots still work:

```
pairs(state.x77)
```



Looking at high dimensional data

For the rest of the semester, we will be exploring techniques to represent high dimensional data in lower dimensional spaces for visualization.

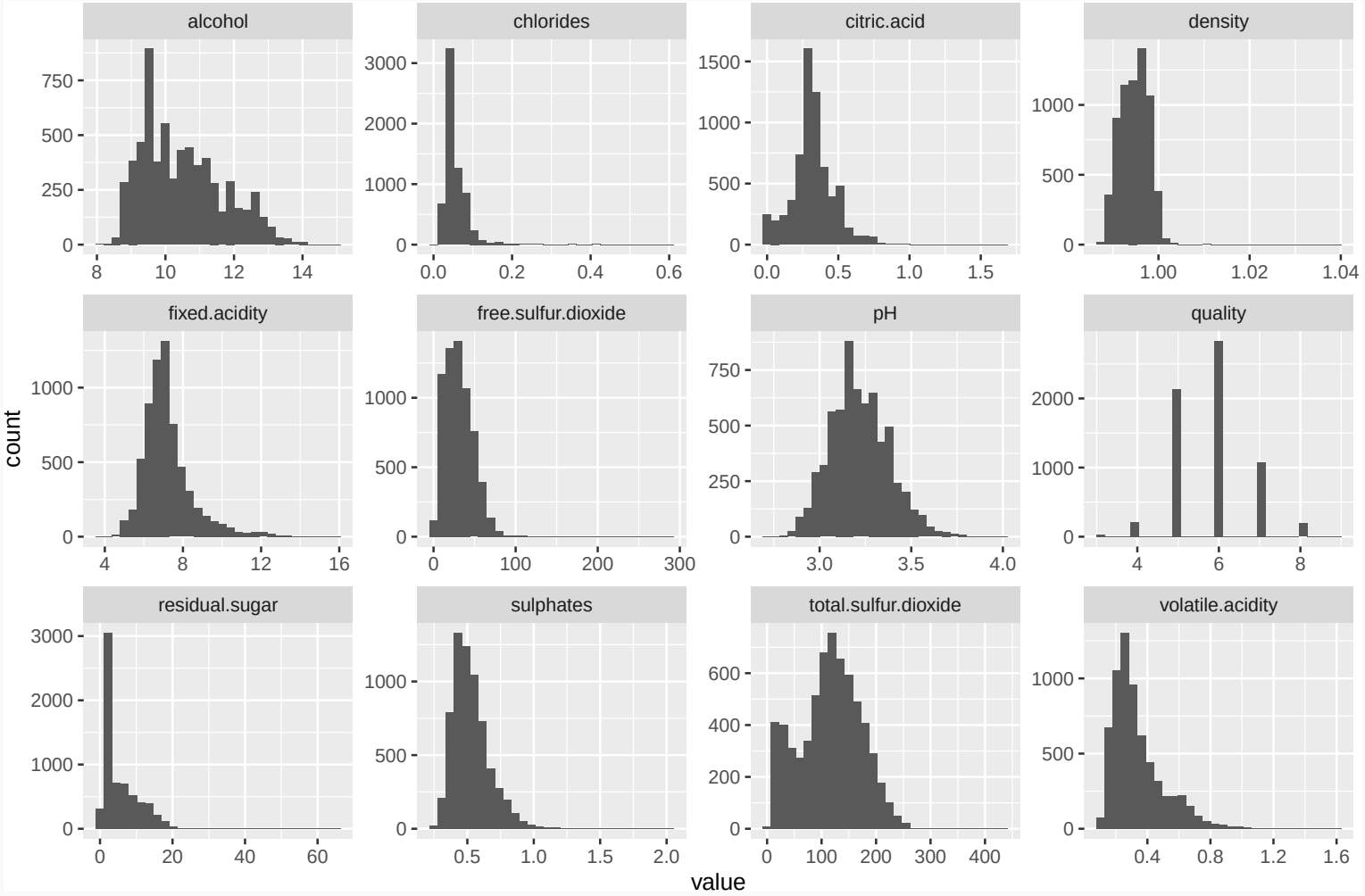
However, standard methods for multivariate visualization should always be your **first step**: typically they are easier to read and do not require the audience to understand your dimension reduction technique.

Wine quality dataset

The wine quality dataset, available via the UC Irvine Machine Learning Repository, provides chemical properties of red and white versions of Portuguese “Vinho Verde” wine Cortez et al. (2009).

```
# A tibble: 6 x 13
  fixed.acidity volatile.acidity citric.acid residual.sugar
          <dbl>            <dbl>       <dbl>          <dbl>
1           7.4             0.7         0              1.9
2           7.8             0.88        0              2.6
3           7.8             0.76        0.04           2.3
4          11.2             0.28        0.56           1.9
5           7.4             0.7         0              1.9
6           7.4             0.66        0              1.8
# i 9 more variables: chlorides <dbl>,
#   free.sulfur.dioxide <dbl>, total.sulfur.dioxide <dbl>,
#   density <dbl>, pH <dbl>, sulphates <dbl>,
#   alcohol <dbl>, quality <int>, type <chr>
```

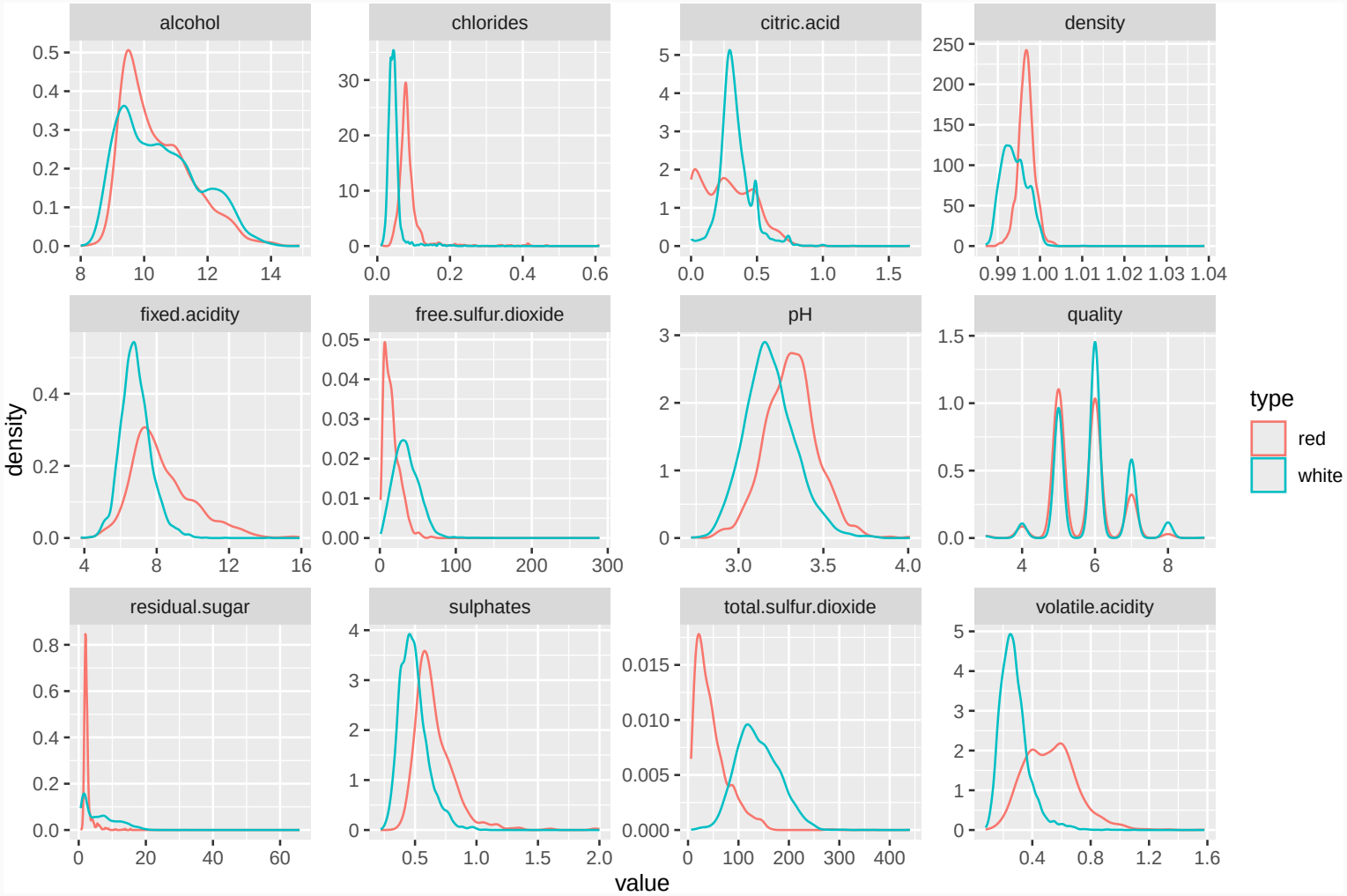
Small multiples: histograms



Small multiples: histograms

```
# transform data
wine |>
  pivot_longer(cols = !type,
               names_to = "variable",
               values_to = "value") |>
  ggplot(aes(x = value)) +
  geom_histogram() +
  facet_wrap(~variable, scale = "free") # let x-axis vary
```

Small multiples: density plots

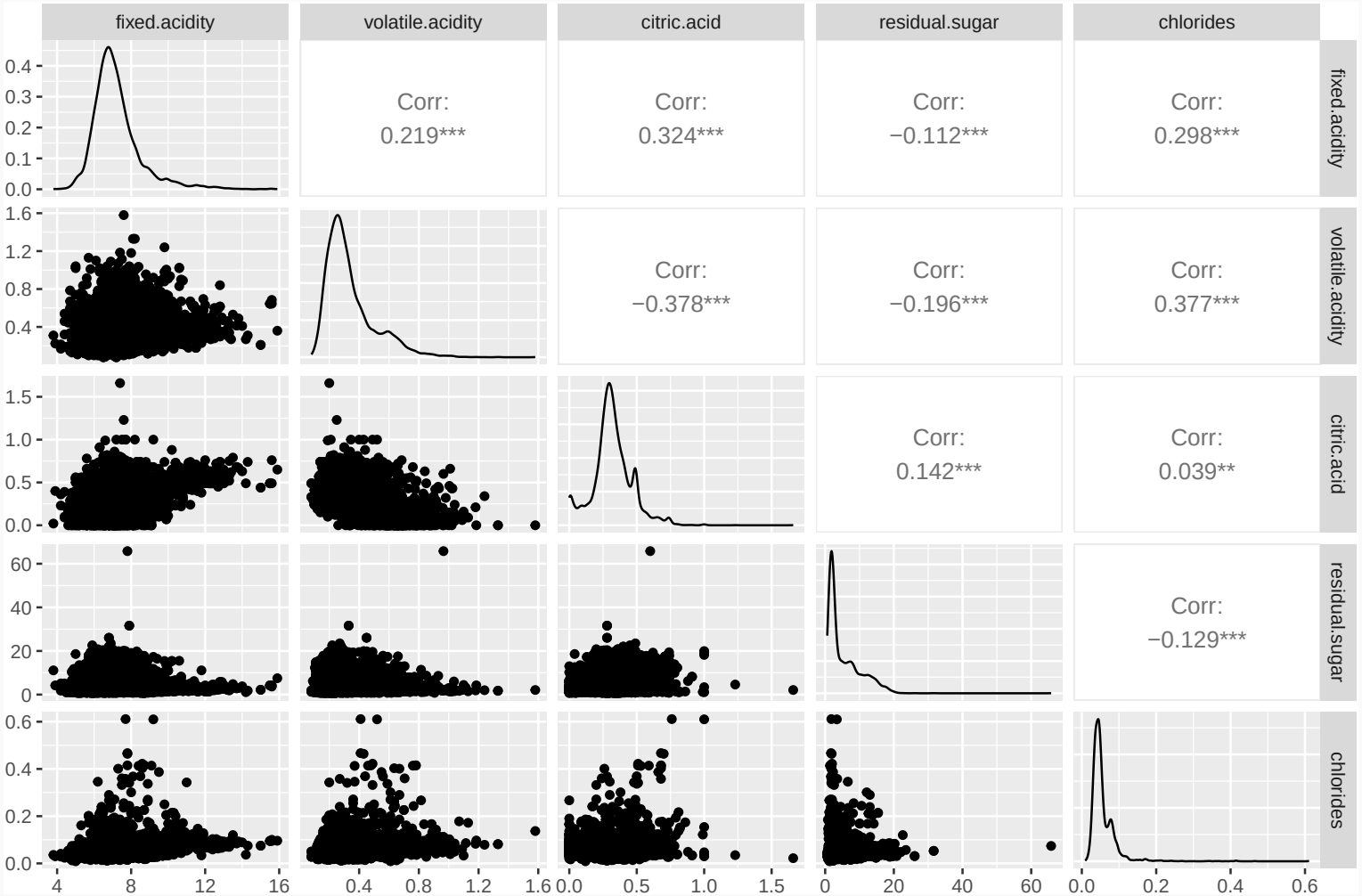


Small multiples: density plots

```
# transform data
wine |>
  pivot_longer(cols = !type,
               names_to = "variable",
               values_to = "value") |>
  ggplot(aes(x = value, color = type)) +
  geom_density() +
  facet_wrap(~variable, scale = "free") # let x-axis vary
```

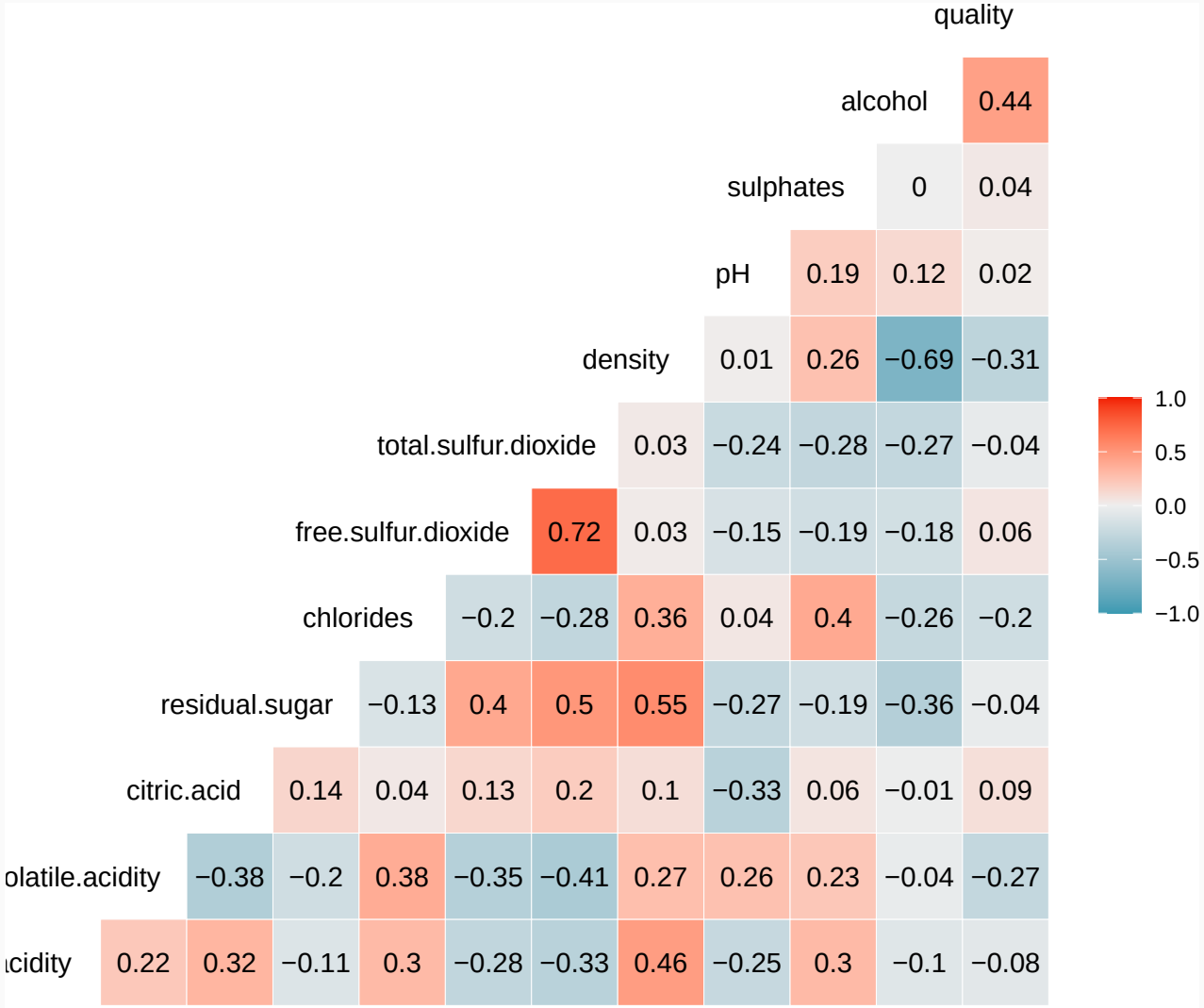
Paired scatter plots

```
library(GGally)
ggpairs(wine[, 1:5])
```



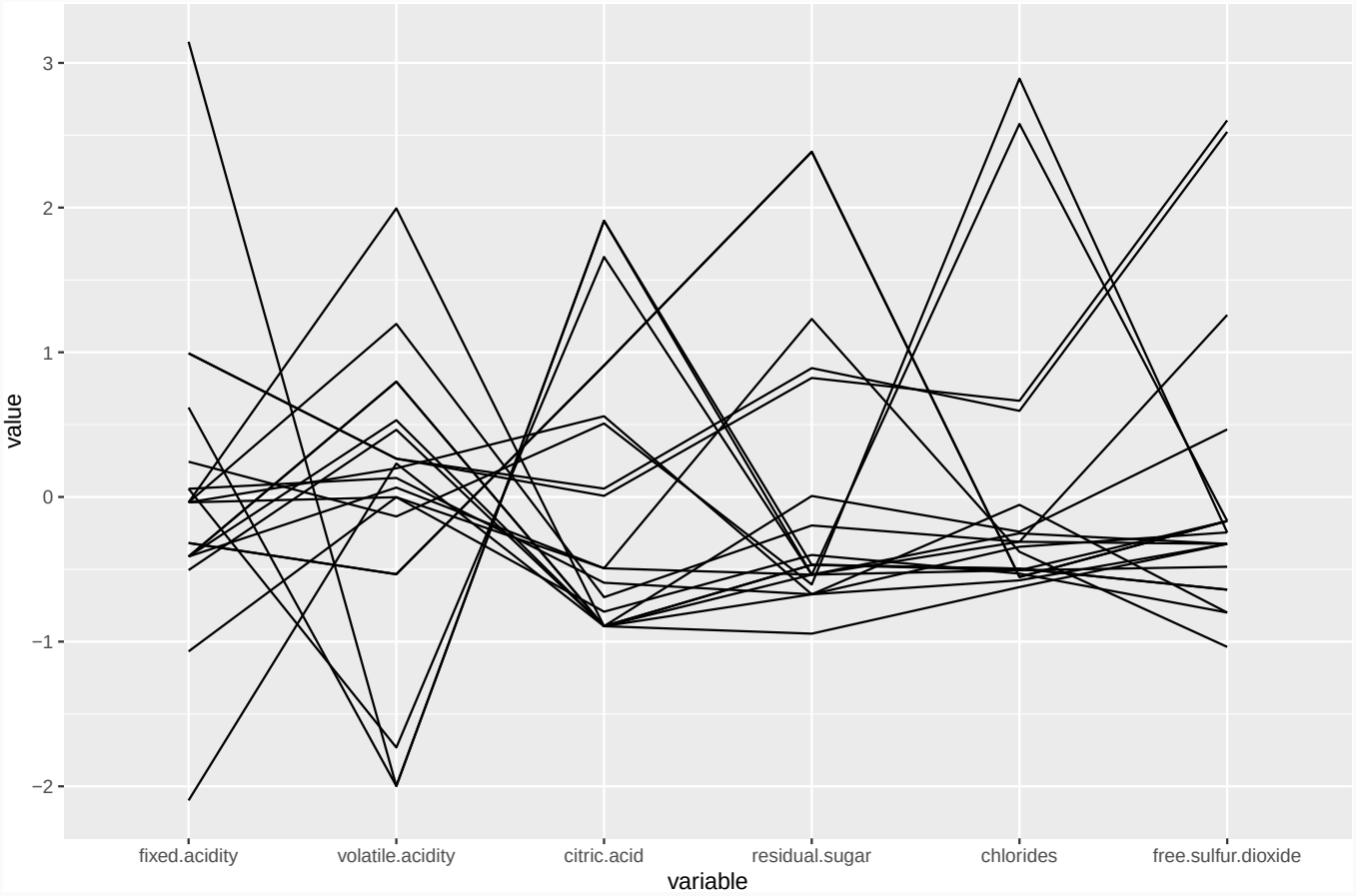
Correlation heatmaps

```
ggcorr(wine, label = TRUE, label_round = 2, hjust = .9)
```

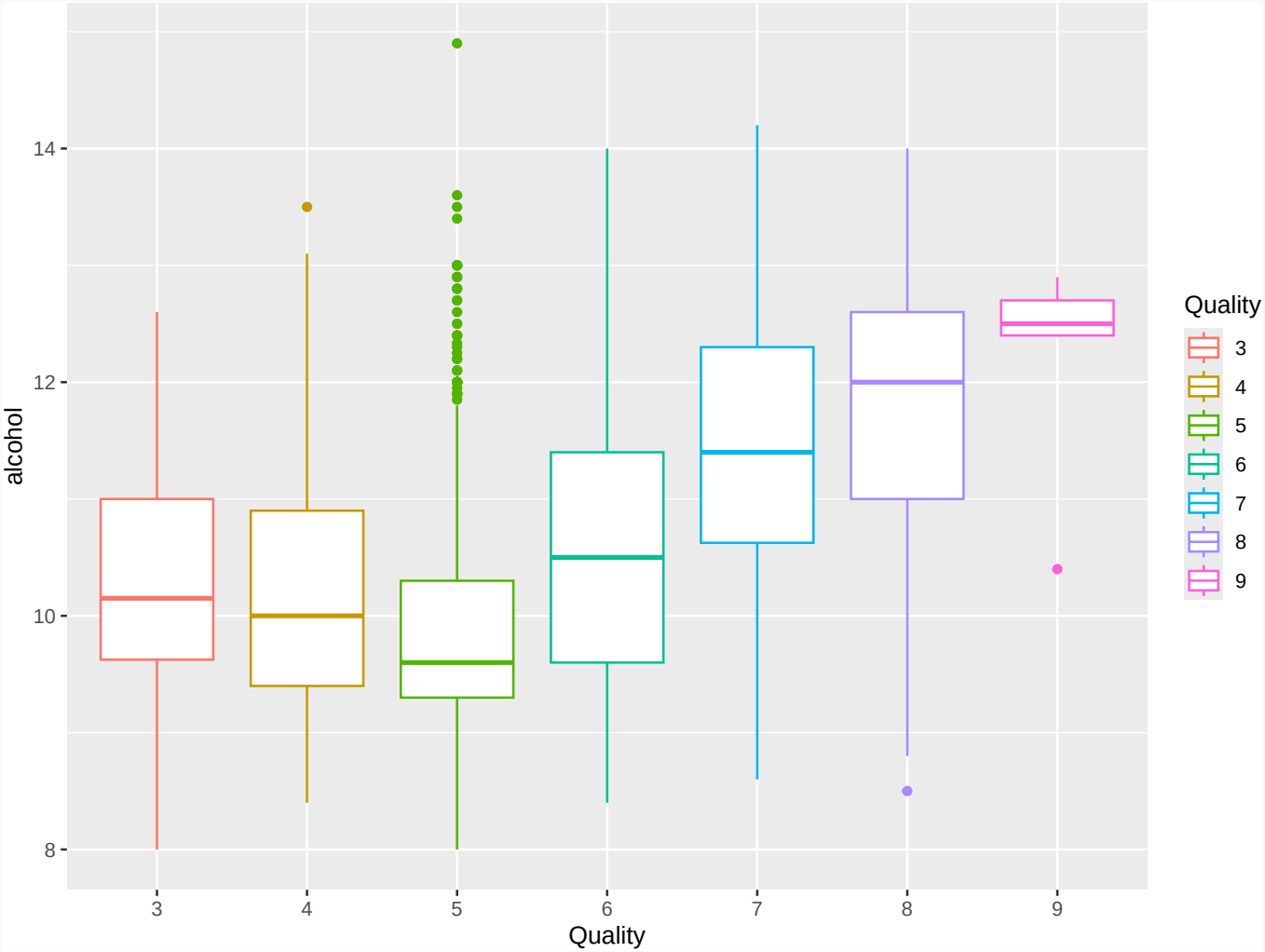


Parallel coordinate maps

```
ggparcoord(data = wine[wine$type == "red",][1:20,], columns = 1:6)
```



Parallel boxplots



Parallel boxplots

```
# transform data
wine |>
  ggplot(aes(y = alcohol,
             x =  as.factor(quality),
             group = as.factor(quality),
             color = as.factor(quality))) +
  geom_boxplot() +
  scale_color_discrete(name = "Quality") +
  xlab("Quality")
```

Multidimensional scaling

We have already seen how singular value decomposition and principal components analysis may be used to construct low-rank approximations to any data matrix and obtain 2D (or 3D) visualizations of high-dimensional data.

Multidimensional scaling is a dimension reduction technique that projects high dimensional data onto a lower dimensional subspace while preserving, as much as possible, the **distances** between observations.

We will see that in certain cases, MDS is identical to PCA, but it also can yield very different results.

Comparing MDS and PCA

Principal components analysis projects data onto the lower dimensional space that maximizes the variances of the ℓ^2-lengths of the projections.

Multidimensional scaling projects data onto the lower dimensional spaces that best preserve the distances (ℓ^2 or other) between observations.

When the Euclidean distance is used for MDS, it is equivalent to PCA.

Multidimensional scaling

Suppose we have n observations $\{o_1, \dots, o_n\}$. Further assume that for each pair of observations, we can define

$$d_{ij} = \text{dissimilarity}(o_i, o_j), \qquad D = (d_{ij}) \in \mathbb{R}^{n \times n}$$

where we call D the dissimilarity matrix. Any distance metric can be used to construct D, but we can also use dissimilarity functions that do **not** meet the definition of a distance metric.

Example dissimilarity metrics

1. Euclidean distance (ℓ_2): $||\mathbf{x} - \mathbf{y}||_2$
2. Manhattan/cityblock distance (ℓ_1): $||\mathbf{x} - \mathbf{y}||_1$
3. Chebyshev/maximum coordinate distance (ℓ_∞): $||\mathbf{x} - \mathbf{y}||_\infty$
4. Cosine dissimilarity: $\left\| \frac{\mathbf{x}}{||\mathbf{x}||} - \frac{\mathbf{y}}{||\mathbf{y}||} \right\|^2 = 2 - 2\cos\theta$
5. Correlation coefficient: $1 - \mathrm{Cor}(\mathbf{x}, \mathbf{y})$

Multidimensional scaling

Objective: Given a dissimilarity matrix D, represent the objects as vectors $\mathbf{y}_1, \dots, \mathbf{y}_n$ in some Euclidean space (ex. $\mathbb{R}^q$) such that for all i, j

$$d_{ij} = ||\mathbf{y}_i - \mathbf{y}_j||_2$$

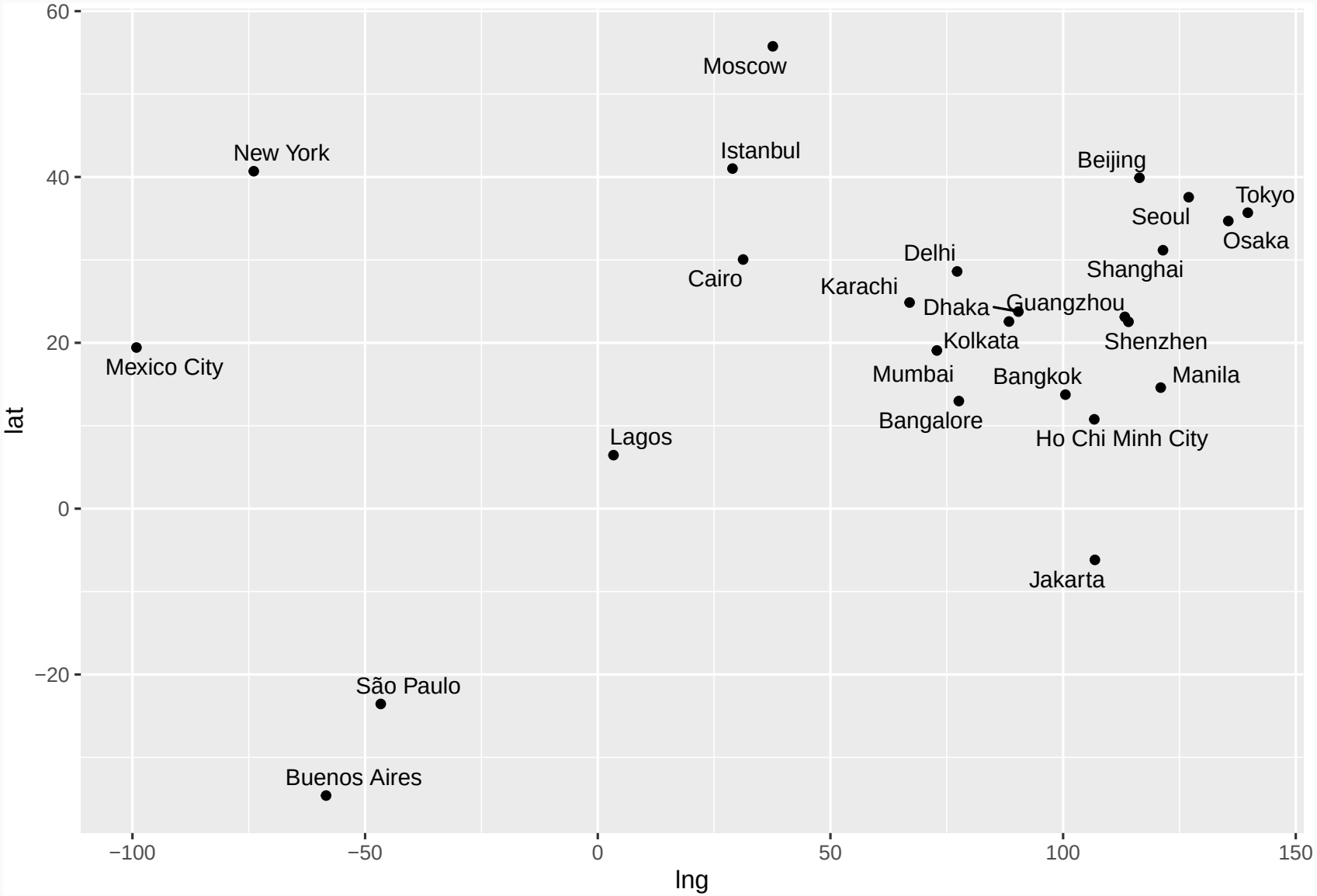
In some cases, we relax the above equality to hold only approximately.

Example: Distances between cities

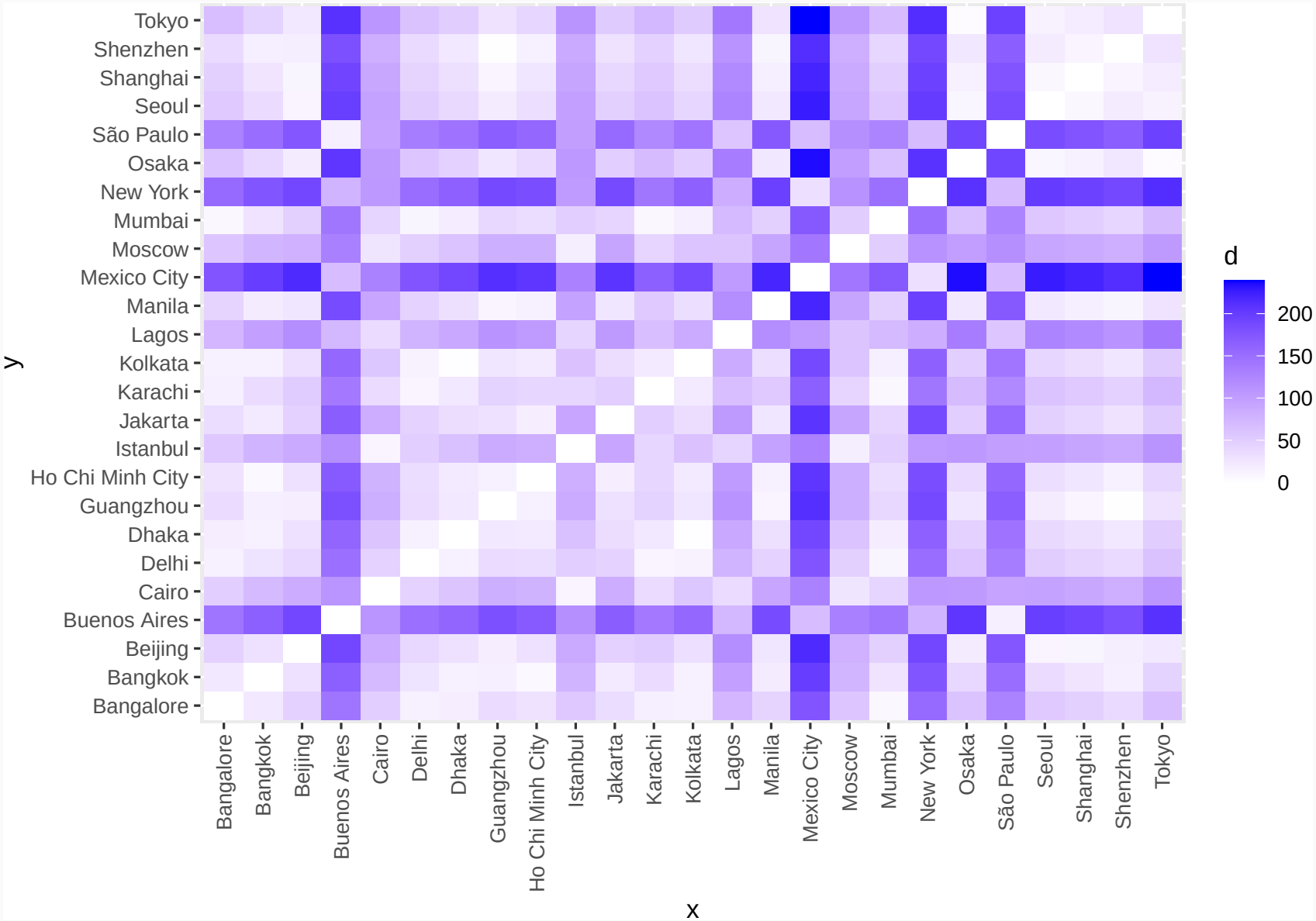
Consider a map of the twenty most populous cities, as provided by simplemaps. Example originally by Andrew Irwin.

```
# transform data
cities <-
  readr::read_csv(paste0("~/Dropbox/teaching/math250/",
                         "examples/MDS/select-cities.csv"))
library(ggplot2)
library(ggrepel)
ggplot(cities, aes(x=lng, y=lat)) +
  geom_point() +
  geom_text_repel(aes(label=city), size = 3.5)
```

Example: Distances between cities



Example: Distances between cities



Example: Distances between cities

```
# transform data
heatmap_dat <- expand_grid(x = cities$city, y = cities$city)
heatmap_dat$d <-
  as.vector(
    as.matrix(
      dist(cbind(cities$lng, cities$lat),
           upper = T, diag = T)
      )
    )
ggplot(heatmap_dat, aes(x, y, fill= d)) +
  geom_tile() +
  scale_fill_gradient(low="white", high="blue") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

Multidimensional scaling

Objective: Given $D = (d_{ij})$, compute a representation $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^2$ such that

$$d_{ij} = ||\mathbf{y}_i - \mathbf{y}_j||_2 \text{ for all } i, j$$

Multidimensional scaling

First, note that any solution to the above set of equations is **not unique** as ℓ_2 distance is invariant under translation:

$$d_{ij} = ||\mathbf{y}_i - \mathbf{y}_j||_2 = ||(\mathbf{y}_i + \mathbf{c}) - (\mathbf{y}_j + \mathbf{c})||_2 \text{ for all } i, j$$

We can place an additional constraint that the vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n$ are centered to ensure a unique solution:

$$\sum_{i=1}^{n} \mathbf{y}_i = \mathbf{0}$$

Our goal will be to identify a set of vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n$ that satisfies both sets of equations.

Estimating $\mathrm{y}_1, \ldots, \mathrm{y}_n$

Consider that

$$d_{ij}^2 = ||\mathbf{y}_i||^2 + ||\mathbf{y}_j||^2 - 2\mathbf{y}_i^\top \mathbf{y}_j$$

Let D_2 be the matrix (d_{ij}^2) and denote its column, row, and overall sums:

$$d_{\cdot j}^2 = \sum_i d_{ij}^2$$
$$d_{i\cdot}^2 = \sum_j d_{ij}^2$$
$$d_{\cdot\cdot}^2 = \sum_i \sum_j d_{ij}^2$$

Estimating $\mathrm{y}_1, \ldots, \mathrm{y}_n$

If

$$d_{ij}^2 = ||\mathbf{y}_i||^2 + ||\mathbf{y}_j||^2 - 2\mathbf{y}_i^\top \mathbf{y}_j$$

we get the following equations by summing over i and j separately:

$$d_{\cdot j}^2 = \sum_i d_{ij}^2 = \sum_i ||\mathbf{y}_i||^2 + n||\mathbf{y}_j||^2$$

$$d_{i\cdot}^2 = \sum_j d_{ij}^2 = n||\mathbf{y}_i||^2 + \sum_j ||\mathbf{y}_j||^2$$

Estimating $\mathbf{y}_1, \dots, \mathbf{y}_n$

Analogously, we can show that

$$d_{\cdot\cdot}^2 = n \sum_i ||\mathbf{y}_i||^2 + n \sum_j ||\mathbf{y}_j||^2 = 2n \sum_i ||\mathbf{y}_i||^2$$

so

$$\sum_i ||\mathbf{y}_i||^2 = \frac{1}{2n} d_{\cdot\cdot}^2$$

and therefore

$$||\mathbf{y}_j||^2 = \frac{1}{n} d_{\cdot j}^2 - \frac{1}{2n^2} d_{\cdot\cdot}^2$$

$$||\mathbf{y}_i||^2 = \frac{1}{n} d_{i\cdot}^2 - \frac{1}{2n^2} d_{\cdot\cdot}^2$$

Estimating $\mathbf{y}_1, \dots, \mathbf{y}_n$

From the equation

$$d_{ij}^2 = ||\mathbf{y}_i||^2 + ||\mathbf{y}_j||^2 - 2\mathbf{y}_i^\top \mathbf{y}_j$$

we get

$$d_{ij}^2 = \frac{1}{n} d_{i\cdot}^2 - \frac{1}{2n^2} d_{\cdot\cdot}^2 + \frac{1}{n} d_{\cdot j}^2 - \frac{1}{2n^2} d_{\cdot\cdot}^2 - 2\mathbf{y}_i^\top \mathbf{y}_j$$

and thus for all i, j:

$$\mathbf{y}_i^\top \mathbf{y}_j = \frac{1}{2}\left(\frac{1}{n} d_{i\cdot}^2 + \frac{1}{n} d_{\cdot j}^2 - \frac{1}{n^2} d_{\cdot\cdot}^2 - d_{ij}^2\right)$$

This equation will be key to identifying $\mathbf{y}_1, \dots, \mathbf{y}_n$.

Estimating $\mathbf{y}_1, \dots, \mathbf{y}_n$

Define a new matrix $G = (g_{ij}) \in \mathbb{R}^{n \times n}$ with entries

$$g_{ij} := \mathbf{y}_i^\top \mathbf{y}_j = \frac{1}{2}\left(\frac{1}{n} d_{i\cdot}^2 + \frac{1}{n} d_{\cdot j}^2 - \frac{1}{n^2} d_{\cdot\cdot}^2 - d_{ij}^2\right)$$

This matrix is a **Gram matrix** containing the pairwise dot products for $\mathbf{y}_1, \dots, \mathbf{y}_n$.

If we define $Y = [\mathbf{y}_1, \dots, \mathbf{y}_n]^\top \in \mathbb{R}^{n \times q}$, then this last equation can be rewritten as

$$YY^\top = G$$

Estimating $\mathrm{y}_1, \ldots, \mathrm{y}_n$

We can decompose the Gram matrix as follows:

$$G = -\frac{1}{2} C D_2 C \quad \text{where} \quad C = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$$

where $\mathbf{1}$ is a length-n vector of ones. Then:

$$\begin{aligned} CD_2C &= \left(I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top\right) D_2 \left(I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top\right) && (1) \\ &= D_2 - \frac{1}{n}\mathbf{1}\mathbf{1}^\top D_2 - \frac{1}{n} D_2 \mathbf{1}\mathbf{1}^\top + \frac{1}{n^2}\mathbf{1}\mathbf{1}^\top D_2 \mathbf{1}\mathbf{1}^\top && (2) \\ &= -2G && (3) \end{aligned}$$

Estimating $\mathbf{y}_1, \dots, \mathbf{y}_n$

Here, C is a centering matrix. What are $\mathbf{1}\mathbf{1}^\top D_2$ and $D_2\mathbf{1}\mathbf{1}^\top$? Thus, CD_2C centers the columns and rows of D_2 so that both column and row means are zero.

As a result, the column and row sums of G are also zero.

In order for a solution to

$$YY^\top = G$$

to exist, G must be positive semidefinite. Since G is symmetric and real-valued, the spectral decomposition is

$$G = U\Lambda U^\top = (U\Lambda^{1/2})(U\Lambda^{1/2})^\top$$

where $\Lambda^{1/2}$ is defined since G is positive semidefinite.

Estimating $\mathrm{y}_1, \ldots, \mathrm{y}_n$

Thus, if $G = -\frac{1}{2} C D_2 C \;\; \text{where} \;\; C$ is positive semidefinite and $q \geq r = \operatorname{rank}(G)$, then the solution to the multidimensional scaling problem is

$$Y = U_q \Lambda_q^{1/2} = \left[\sqrt{\lambda_1} \mathbf{u}_1, \ldots, \sqrt{\lambda_r} \mathbf{u}_r, \sqrt{\lambda_{r+1}} \mathbf{u}_{r+1}, \ldots, \sqrt{\lambda_q} \mathbf{u}_q \right] \in \mathbf{R}^{n \times q}$$

where λ_i is the ith eigenvalue and $\mathbf{u}_i$ is the ith eigenvector of G. Note that for $i = r+1, \ldots, q$, $\lambda_i = 0$.

Non-unique solutions

Note that the solution to

$$YY^\top = G$$

is not unique since any rotation of Y by an orthogonal matrix Q yields another solution:

$$(YQ)(YQ)^\top = YQQ^\top Y^\top = YY^\top = G$$

Comparison wth PCA

Suppose the original objects are Euclidean vectors $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^p$ and the dissimilarities are Euclidean distances $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||_2$. Then consider that $G = X_c X_c^\top$ where $X_c = CX$ is the centered data matrix.

Then the embedding matrix $Y \in \mathbb{R}^{n \times q}$ satisfies

$$G = YY^\top = X_c X_c^\top$$

Comparison wth PCA

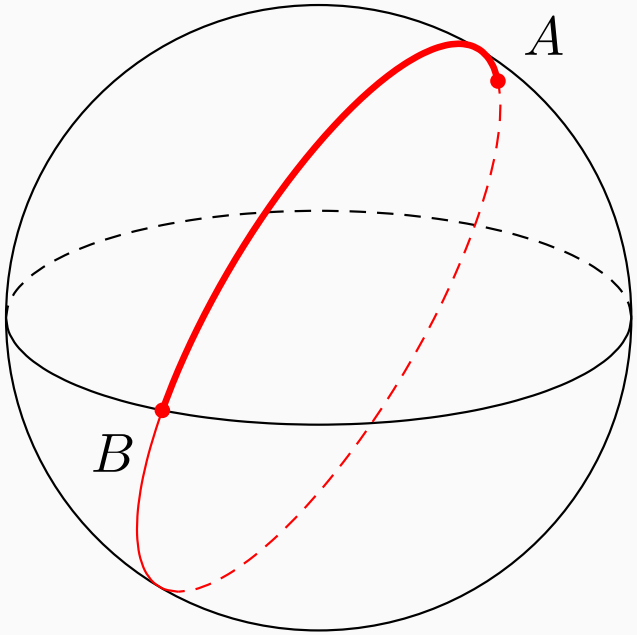
The solution is given by solving $YY^\top = U_r \Sigma_r V_r^\top V_r \Sigma_r U_r^\top = U_r \Sigma_r^2 U_r^\top$ where $U_r \Sigma_r V_r^\top$ is the compact SVD of X_c.

Thus, $Y_r = U_r \Sigma_r$ if $q = r$, and if $q < r$, then there is no exact solution, but the best approximation is $Y_q = U_q \Sigma_q$: the matrix of the first q principal components.

When does a solution exist?

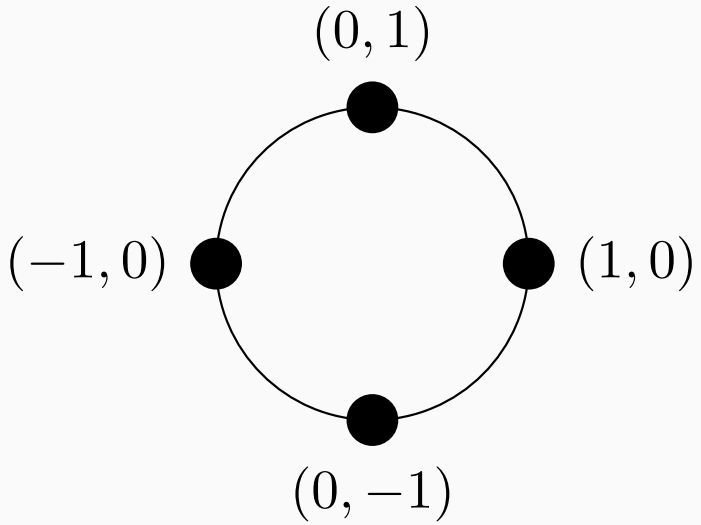
Remember that G must be symmetric positive semidefinite to yield a solution.

Example: Great circle distance



Non-Euclidean geometry

Consider the following four points on the unit circle:



Non-Euclidean geometry

If the dissimilarity of two points is defined as the length of the shortest arc between them along the circle, then the dissimilarity matrix is

$$D = \begin{bmatrix} 0 & \pi/2 & \pi & \pi/2 \\ \pi/2 & 0 & \pi/2 & \pi \\ \pi & \pi/2 & 0 & \pi/2 \\ \pi/2 & \pi & \pi/2 & 0 \end{bmatrix}$$

Then

$$G = -\frac{1}{2} C D_2 C = \frac{\pi^2}{16} \begin{bmatrix} 3 & 1 & -5 & 1 \\ 1 & 3 & 1 & -5 \\ -5 & 1 & 3 & 1 \\ 1 & -5 & 1 & 3 \end{bmatrix}$$

This matrix is symmetric but not positive semidefinite because G has a negative eigenvalue:

$$\lambda_1 = \lambda_2 = \frac{\pi^2}{2}, \lambda_3 = 0, \lambda_4 = -\frac{\pi^2}{4}$$

Non-Euclidean geometry

Instead of constructing an exact solution to $YY^\top = G$, we construct Y using only the eigenvectors associated with positive eigenvalues. If we let $\lambda_1, \dots, \lambda_{r^+}$ the positive eigenvalues of G, then we can use the following approximate solution as our embedding.

$$Y_{r^+} = U_q \Lambda_q^{1/2} = \left[\sqrt{\lambda_1}\mathbf{u}_1 \ \cdots \ \sqrt{\lambda_{r^+}}\mathbf{u}_{r^+}\right]$$

where we assume $\mathbf{u}_1, \dots, \mathbf{u}_{r^+}$ are unit-norm eigenvectors.

This solution satisfies

$$Y_{r^+}(Y_{r^+})^\top = \lambda_1 \mathbf{u}_1 \mathbf{u}_1^\top + \cdots \lambda_{r^+} \mathbf{u}_{r^+} \mathbf{u}_{r^+}^\top$$

which is the closest positive semidefinite matrix to G under the Frobenius norm.

Non-Euclidean geometry

In the unit circle example, the first two eigenvalues are positive $\lambda_1 = \lambda_2 = \frac{\pi^2}{2}$ with eigenvectors

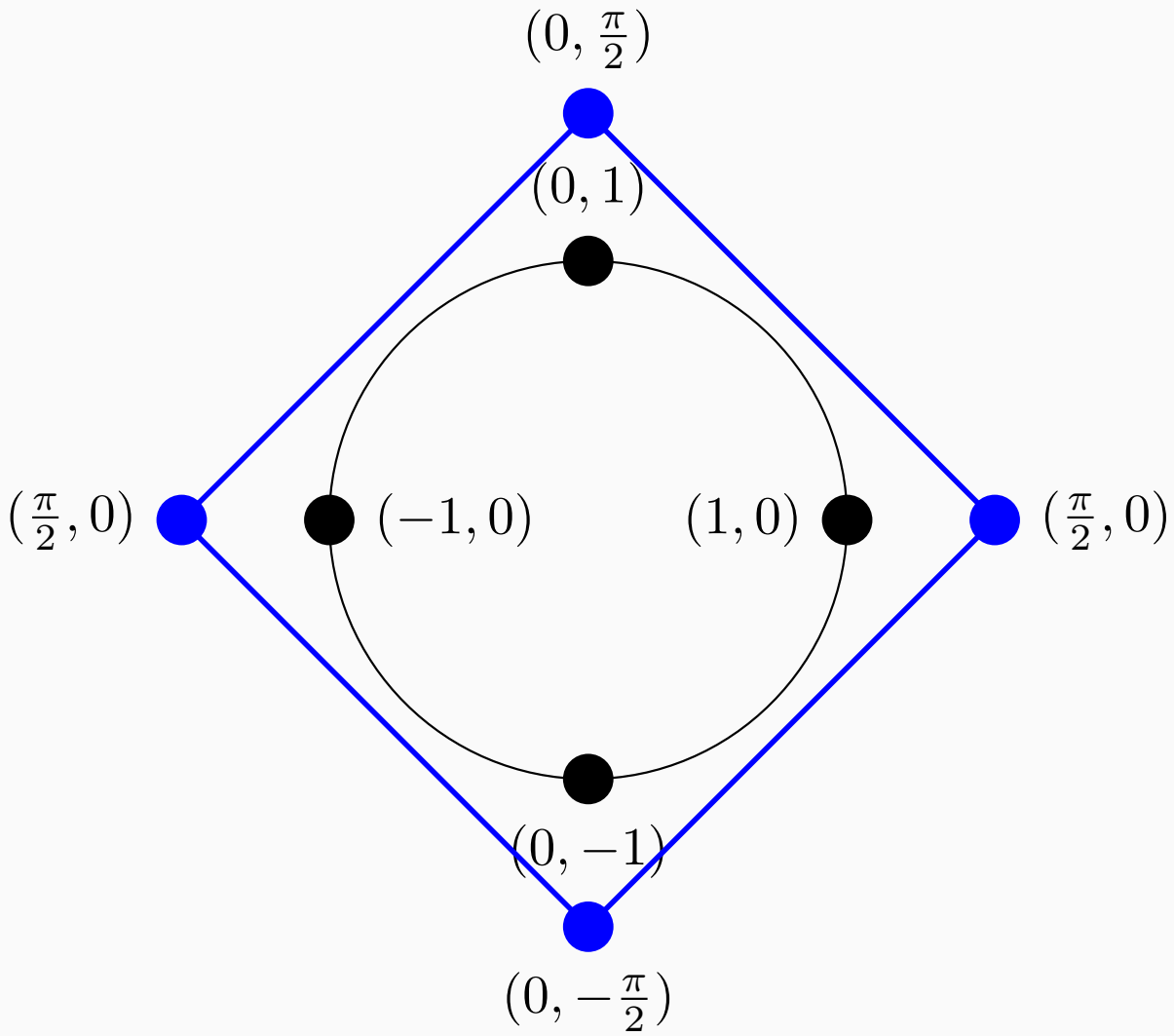
$$\mathbf{u}_1 = \left(1/\sqrt{2}, 0, -1/\sqrt{2}, 0\right)^\top, \quad \mathbf{u}_2 = \left(0, 1/\sqrt{2}, 0, -1/\sqrt{2}\right)^\top$$

which yields the following embedding

$$Y_2 = [\sqrt{\lambda_1}\mathbf{u}_1 \quad \sqrt{\lambda_2}\mathbf{u}_2] = \begin{bmatrix} \pi/2 & 0 \\ 0 & \pi/2 \\ -\pi/2 & 0 \\ 0 & -\pi/2 \end{bmatrix}$$

Non-Euclidean geometry

This yields the following embedding



Non-Euclidean geometry

The embedding has the following distances

$$\widehat{D} = \begin{bmatrix} 0 & \pi/\sqrt{2} & \pi & \pi/\sqrt{2} \\ \pi/\sqrt{2} & 0 & \pi/\sqrt{2} & \pi \\ \pi & \pi/\sqrt{2} & 0 & \pi/\sqrt{2} \\ \pi/\sqrt{2} & \pi & \pi/\sqrt{2} & 0 \end{bmatrix}$$

Kruskal stress

To assess the fit of the approximate solution Y_2, we can use the **Kruskal stress**:

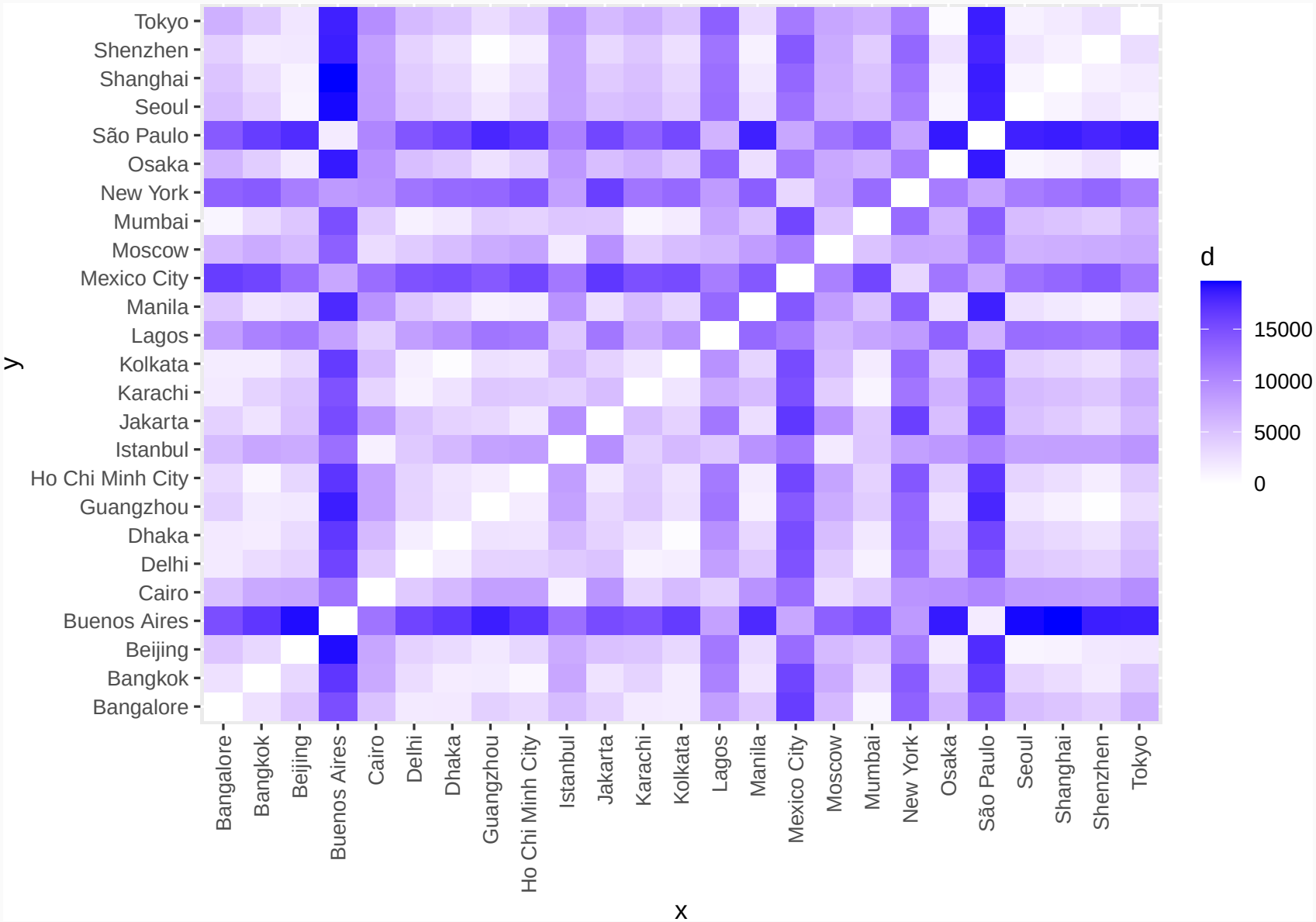
$$\text{Stress} = \frac{||D - \widehat{D}||_F}{||D||_F}$$

which is the relative approximation error of D by $\widehat{D}$. A rough rule of thumb is that the fit is acceptable if stress < 0.1 and unacceptable if stress > 0.15. For the unit lcircle example,

$$\text{Stress} = \frac{\sqrt{2} - 1}{\sqrt{3}} = .2391$$

suggesting that the embedding does not capture the nonlinearity well.

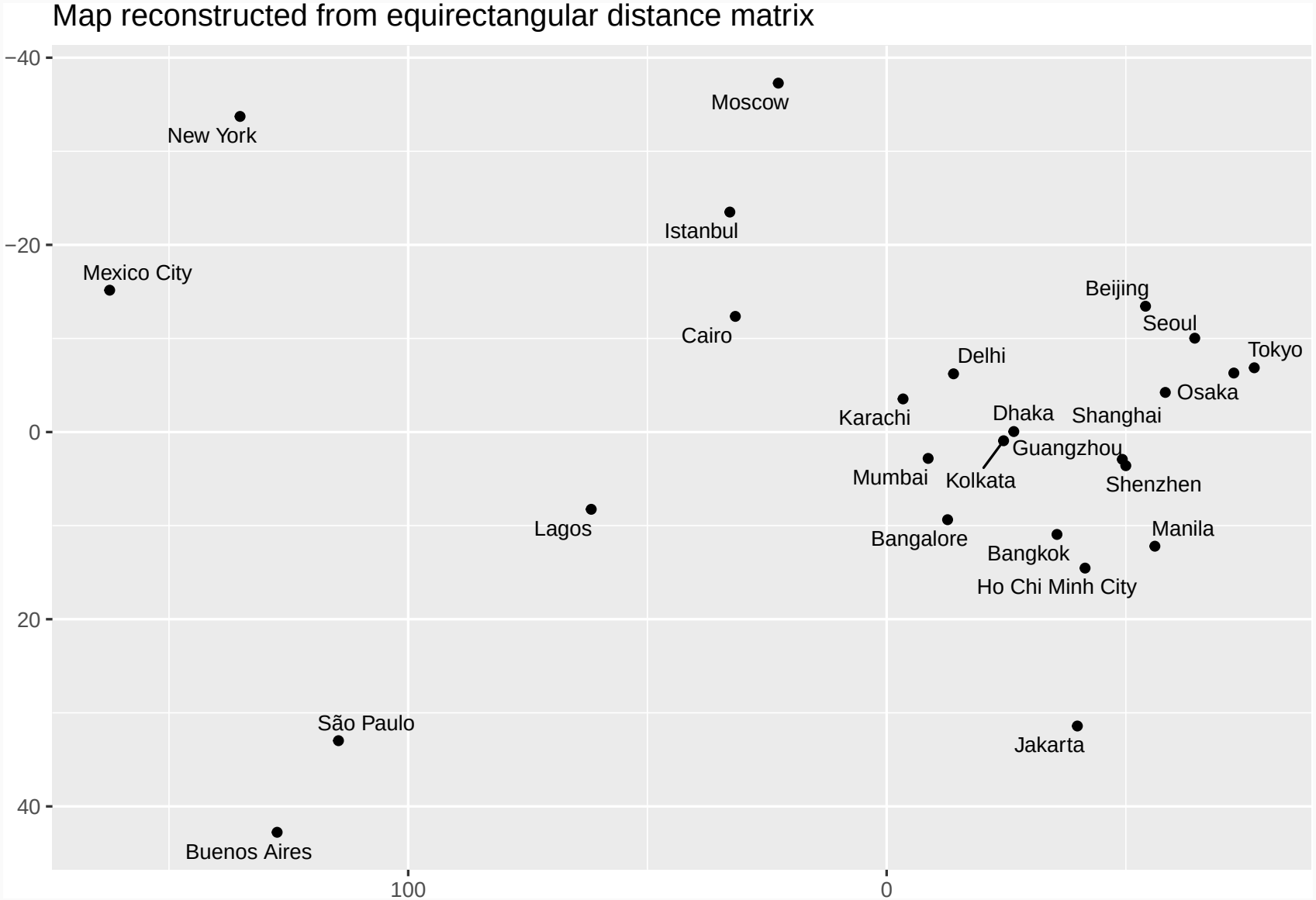
Example: Distances between cities



Example: Distances between cities

```
# transform data
library(fields)
great_earth_dist <-
  rdist.earth(cities |>
                dplyr::select(lng, lat) |>
                as.matrix(),
              miles = FALSE)
heatmap_dat <- expand_grid(x = cities$city, y = cities$city)
heatmap_dat$d <- as.vector(as.matrix(
  great_earth_dist
))
ggplot(heatmap_dat, aes(x, y, fill= d)) +
  geom_tile() +
  scale_fill_gradient(low="white", high="blue") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

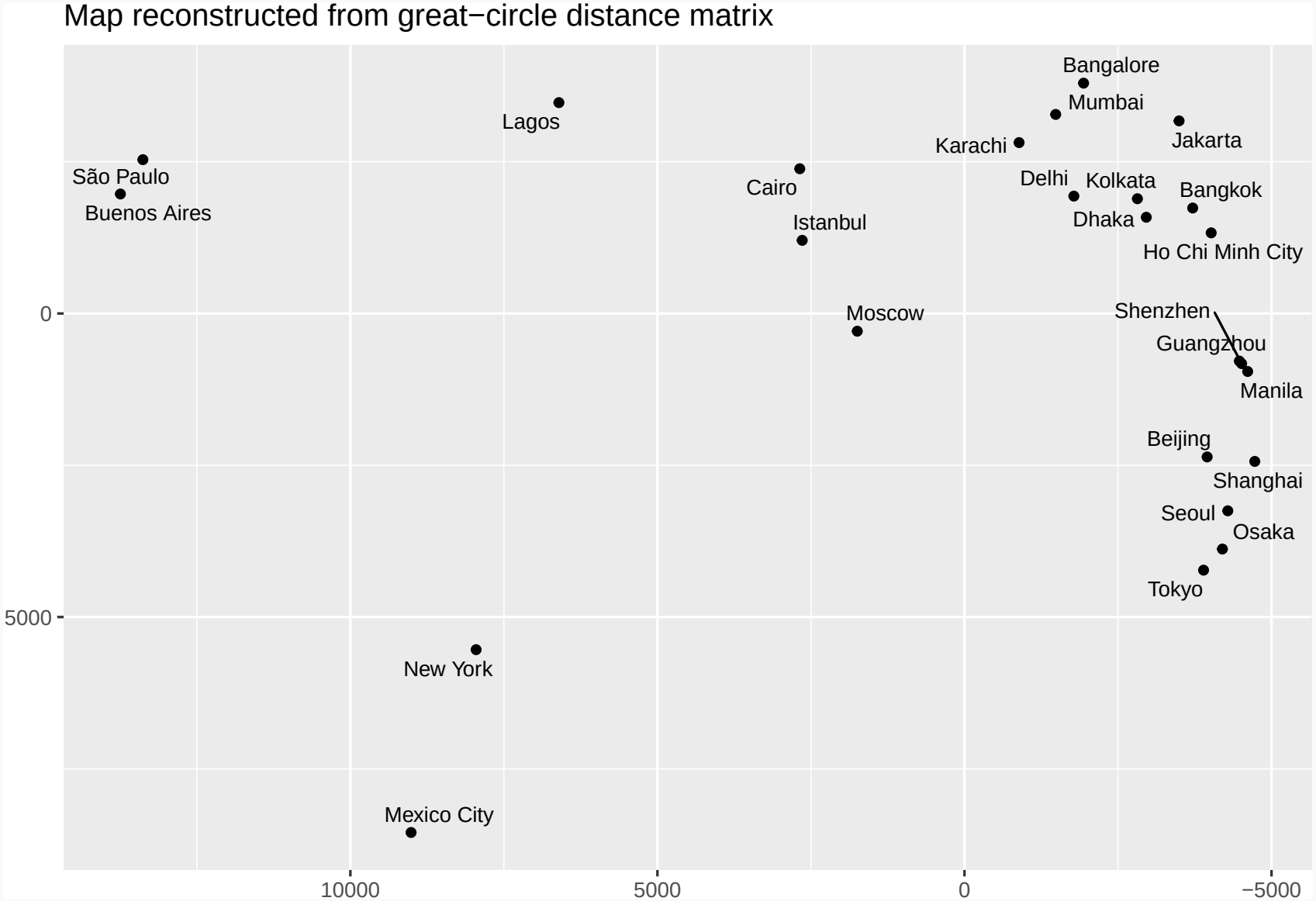
Example: Distances between cities



Example: Distances between cities

```
equirec_dist <-  dist(cbind(cities$lng, cities$lat),
                      upper = T, diag = T) |>
  as.matrix()
mds_equirect <- cmdscale(equirec_dist)
colnames(mds_equirect) <- c("V1", "V2")
bind_cols(cities, as_tibble(mds_equirect)) %>%
  ggplot(aes(x = V1, y = V2)) +
  geom_point() +
  geom_text_repel(aes(label = city), size = 3) +
  scale_x_reverse() + scale_y_reverse() +
  labs(title = "Map reconstructed from equirectangular distance matrix",
       x = "", y = "")
```

Example: Distances between cities



Example: Distances between cities

```
mds_great_earth <- cmdscale(great_earth_dist)
colnames(mds_great_earth) <- c("V1", "V2")
bind_cols(cities, as_tibble(mds_great_earth)) %>%
  ggplot(aes(x = V1, y = V2)) +
  geom_point() +
  geom_text_repel(aes(label = city), size = 3) +
  scale_x_reverse() + scale_y_reverse() +
  labs(title = "Map reconstructed from great-circle distance matrix",
       x = "", y = "")
```

Example: Distances between cities

We can run `cmdscale()` with `eig = T` to get the eigenvalues. Only the positive eigenvalues are used for the embedding.

```
options(width = 60)
mds_great_earth <- cmdscale(great_earth_dist, eig = T)
mds_great_earth$eig
```

```
 [1]  7.927842e+08  2.482942e+08  9.914175e+07  1.480705e+07
 [5]  6.942207e+06  1.978123e+06  1.046758e+05  3.330188e+04
 [9]  2.029490e+03  6.438807e+02  4.426408e+01  8.169769e+00
[13]  1.680218e-02  2.786034e-08 -2.734136e-01 -2.135059e+00
[17] -3.455217e+01 -9.448057e+02 -2.716067e+03 -9.012171e+04
[21] -1.766472e+05 -7.626291e+06 -1.383487e+07 -3.374239e+07
[25] -1.242048e+08
```

Example: Distances between cities

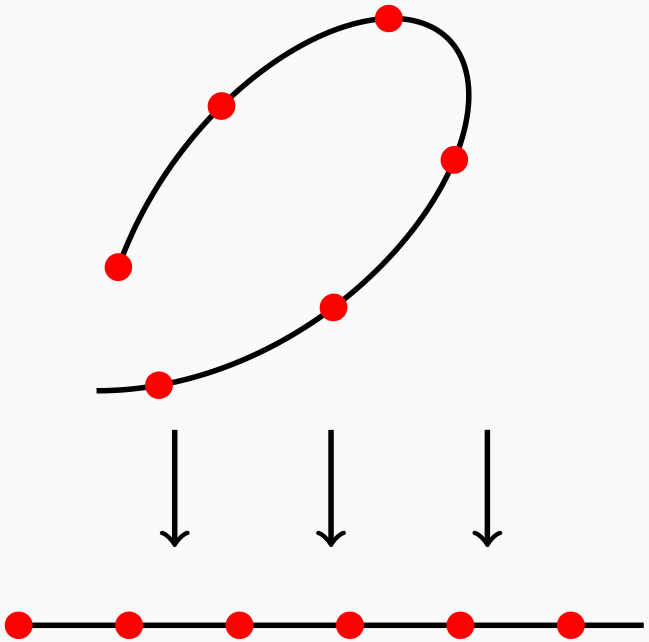
Finally, we can calculuate the stress using the `norm()` function in R:

```
stress <- norm(great_earth_dist - dist(mds_great_earth$points),
               type = "F")
stress <- stress / norm(great_earth_dist, type = "F")
stress
```

```
[1] 0.8205751
```

Non-linear embeddings

Linear embeddings (MDS with Euclidean distances) can fail to represent non-linear geometries in data. Non-linear embedding methods yield strategies for better representing such data:



References

Cortez, Paulo, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. 2009. “Modeling Wine Preferences by Data Mining from Physicochemical Properties.” *Decision Support Systems*, Smart Business Networks: Concepts and Empirical Evidence, 47 (4): 547–53. https://doi.org/10.1016/j.dss.2009.05.016.