Principal Components

Principal Components

- Have measurements on (possibly large) number of variables on some individuals.
- Question: can we describe data using fewer variables (because original variables correlated in some way)?
- Look for direction (linear combination of original variables) in which values *most spread out*. This is *first principal component*.
- Second principal component then direction uncorrelated with this in which values then most spread out. And so on.

Principal components

- See whether small number of principal components captures most of variation in data.
- Might try to interpret principal components.
- If 2 components good, can make plot of data.
- (Like discriminant analysis, but no groups.)
- "What are important ways that these data vary?"

Packages

You might not have installed the first of these. See over for instructions.

```
library(ggbiplot) # see over
library(tidyverse)
library(ggrepel)
```

Installing ggbiplot

- `ggbiplot` not on CRAN, so usual `install.packages` will not work. This is same procedure you used for `smmr` in C32:
- Install package `devtools` first (once):

```
install.packages("devtools")
```

- Then install `ggbiplot` (once):

```
library(devtools)
install_github("vqv/ggbiplot")
```

Small example: 2 test scores for 8 people

```
my_url <- "http://ritsokiguess.site/datafiles/test12.txt"
test12 <- read_table2(my_url)
test12
```

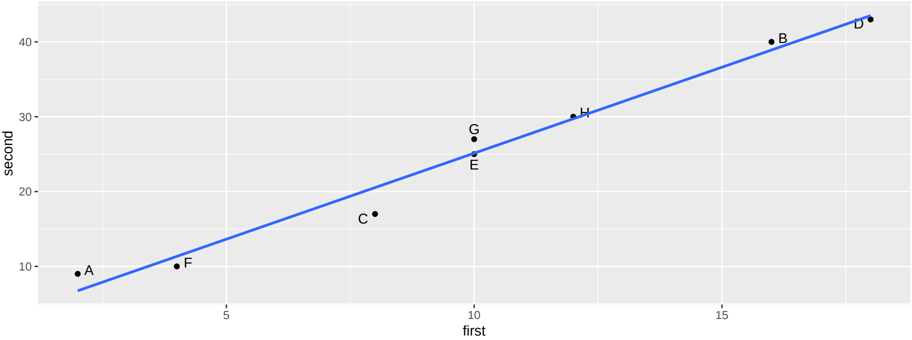
first	second	id
2	9	A
16	40	B
8	17	C
18	43	D
10	25	E
4	10	F
10	27	G
12	30	H

```
g <- ggplot(test12, aes(x = first, y = second, label = id)) +
  geom_point() + geom_text_repel()
```

The plot

```
g + geom_smooth(method = "lm", se = F)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Principal component analysis

- Grab just the numeric columns:

```
test12 %>% select(where(is.numeric)) -> test12_numbers
```

- Strongly correlated, so data nearly 1-dimensional:

```
cor(test12_numbers)
```

```
##           first   second
## first  1.000000 0.989078
## second 0.989078 1.000000
```

Finding principal components

- Make a score summarizing this one dimension. Like this:

```
test12.pc <- princomp(test12_numbers, cor = T)
summary(test12.pc)
```

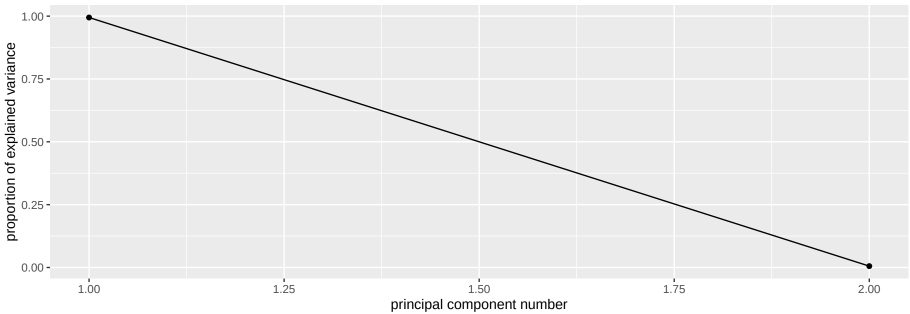
```
## Importance of components:
##                           Comp.1      Comp.2
## Standard deviation     1.410347 0.104508582
## Proportion of Variance 0.994539 0.005461022
## Cumulative Proportion  0.994539 1.000000000
```

Comments

- "Standard deviation" shows relative importance of components (as for LDs in discriminant analysis)
- Here, first one explains almost all (99.4%) of variability.
- That is, look only at first component and ignore second.
- `cor=T` standardizes all variables first. Usually wanted, because variables measured on different scales. (Only omit if variables measured on same scale and expect similar variability.)

Scree plot

```r
ggscreeplot(test12.pc)
```



Imagine scree plot continues at zero, so 2 components is a *big* elbow (take one component).

Component loadings

explain how each principal component depends on (standardized) original variables (test scores):

```
test12.pc$loadings
```

```
##
## Loadings:
##        Comp.1 Comp.2
## first   0.707  0.707
## second  0.707 -0.707
##
##                Comp.1 Comp.2
## SS loadings       1.0    1.0
## Proportion Var    0.5    0.5
## Cumulative Var    0.5    1.0
```

First component basically sum of (standardized) test scores. That is, person tends to score similarly on two tests, and a composite score would summarize performance.

Component scores

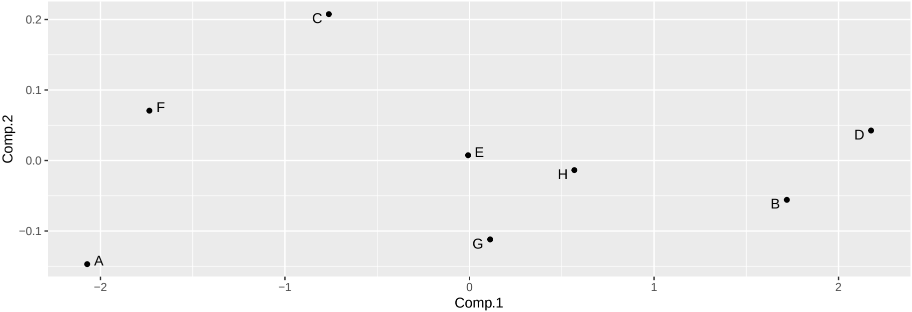
```r
d <- data.frame(test12, test12.pc$scores)
d
```

first	second	id	Comp.1	Comp.2
2	9	A	-2.0718190	-0.1469818
16	40	B	1.7198628	-0.0557622
8	17	C	-0.7622897	0.2075895
18	43	D	2.1762675	0.0425333
10	25	E	-0.0074606	0.0074606
4	10	F	-1.7347840	0.0706834
10	27	G	0.1119091	-0.1119091
12	30	H	0.5683139	-0.0136137

- Person A is a low scorer, very negative `comp.1` score.
- Person D is high scorer, high positive `comp.1` score.
- Person E average scorer, near-zero `comp.1` score.

Plot of scores

```r
ggplot(d, aes(x = Comp.1, y = Comp.2, label = id)) +
  geom_point() + geom_text_repel()
```



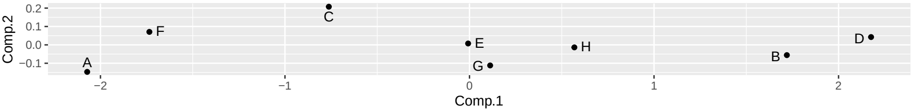
Comments

- Vertical scale exaggerates importance of `comp.2`.
- Fix up to get axes on same scale:

```
g <- ggplot(d, aes(x = Comp.1, y = Comp.2, label = id)) +
  geom_point() + geom_text_repel() +
  coord_fixed()
```

- Shows how exam scores really spread out along one dimension:

```
g
```

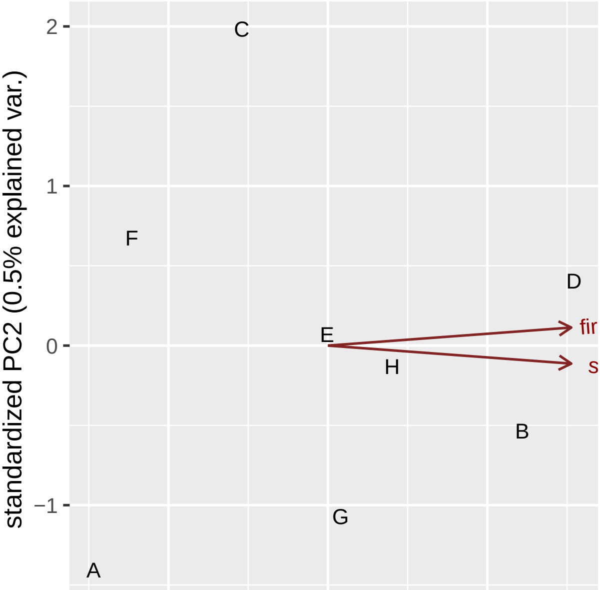


The biplot

- Plotting variables and individuals on one plot.
- Shows how components and original variables related.
- Shows how individuals score on each component, and therefore suggests how they score on each variable.
- Add `labels` option to identify individuals:

```
g <- ggbiplot(test12.pc, labels = test12$id)
```

The biplot



Comments

- Variables point almost same direction (left). Thus very negative value on `comp.1` goes with high scores on both tests, and test scores highly correlated.
- Position of individuals on plot according to scores on principal components, implies values on original variables. Eg.:
- D very negative on `comp.1`, high scorer on both tests.
- A and F very positive on `comp.1`, poor scorers on both tests.
- C positive on `comp.2`, high score on first test relative to second.
- A negative on `comp.2`, high score on second test relative to first.

Places rated

Every year, a new edition of the Places Rated Almanac is produced. This rates a large number (in our data 329) of American cities on a number of different criteria, to help people find the ideal place for them to live (based on what are important criteria for them).

The data for one year are in http://ritsokiguess.site/datafiles/places.txt.

The criteria

There are nine of them:

- `climate`: a higher value means that the weather is better
- `housing`: a higher value means that there is more good housing or a greater choice of different types of housing
- `health`: higher means better healthcare facilities
- `crime`: higher means more crime (bad)
- `trans`: higher means better transportation (this being the US, probably more roads)
- `educate`: higher means better educational facilities, schools, colleges etc.
- `arts`: higher means better access to the arts (theatre, music etc)
- `recreate`: higher means better access to recreational facilities
- `econ`: higher means a better economy (more jobs, spending power etc)

Each city also has a numbered `id`.

Read in the data

Places rated data:

```
my_url <- "http://ritsokiguess.site/datafiles/places.txt"
places0 <- read_table2(my_url)
```

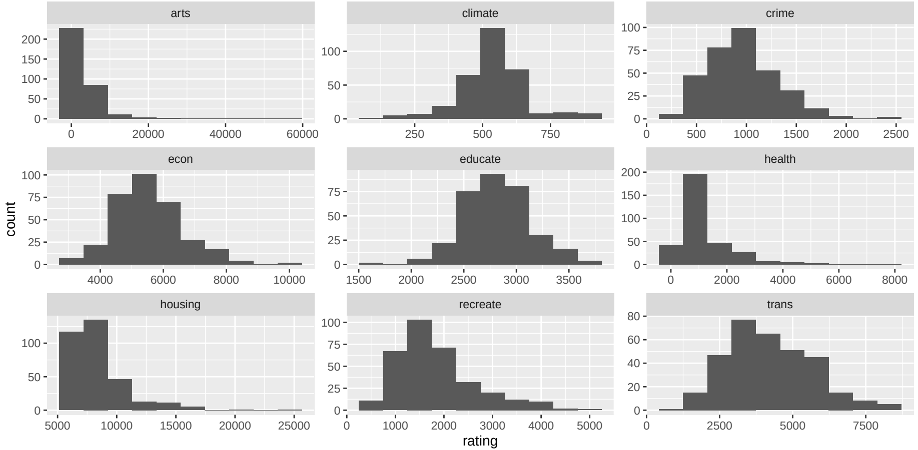
```
##
## -- Column specification ------------------------------
## cols(
##   climate = col_double(),
##   housing = col_double(),
##   health = col_double(),
##   crime = col_double(),
##   trans = col_double(),
##   educate = col_double(),
##   arts = col_double(),
##   recreate = col_double(),
##   econ = col_double(),
```

Look at distributions of everything

```
places0 %>%
  pivot_longer(-id, names_to = "criterion", values_to = "ratin
  ggplot(aes(x = rating)) + geom_histogram(bins = 10) +
  facet_wrap(~criterion, scales = "free") -> g
```

The histograms

```
g
```



Transformations

- Several of these variables have long right tails
- Take logs of everything but id:

```
places0 %>%
  mutate(across(-id, ~log(.))) -> places
places
```

climate	housing	health	crime	trans	educate	arts	recreate	econ
6.255750	8.732305	5.468060	6.827629	8.301770	7.921898	6.903747	7.247793	8.940236
6.354370	9.004300	7.412160	6.786717	8.493515	7.798933	8.624073	7.875499	8.377931
6.148468	8.900958	6.426488	6.877296	7.836370	7.847762	5.468060	6.755769	8.565983
6.165418	8.975630	7.266129	6.413459	8.836810	8.131236	8.445697	7.388328	8.676587
6.490723	9.035153	7.524561	7.301822	8.788441	8.014997	8.410943	7.867871	8.652947
6.253829	8.668884	6.461468	6.588927	7.801391	7.996990	5.811141	6.925595	8.566745
6.326149	9.022564	6.431331	6.242223	7.965893	8.053251	7.754910	7.018402	8.536407

Just the numerical columns

- get rid of the id column

```
places %>% select(-id) -> places_numeric
places_numeric
```

climate	housing	health	crime	trans	educate	arts	recreate	econ
6.255750	8.732305	5.468060	6.827629	8.301770	7.921898	6.903747	7.247793	8.9402
6.354370	9.004300	7.412160	6.786717	8.493515	7.798933	8.624073	7.875499	8.3779
6.148468	8.900958	6.426488	6.877296	7.836370	7.847762	5.468060	6.755769	8.5659
6.165418	8.975630	7.266129	6.413459	8.836810	8.131236	8.445697	7.388328	8.6765
6.490723	9.035153	7.524561	7.301822	8.788441	8.014997	8.410943	7.867871	8.6529
6.253829	8.668884	6.461468	6.588927	7.801391	7.996990	5.811141	6.925595	8.5667
6.326149	9.022564	6.431331	6.242223	7.965893	8.053251	7.754910	7.018402	8.5364
6.285998	8.777556	6.872128	6.559615	8.512181	7.987864	7.304516	7.154615	8.6647
6.329721	8.730852	6.068426	5.988961	8.353733	7.929487	5.545177	7.098376	8.3499
6.411818	8.786610	6.505784	6.978214	8.497390	7.955776	7.118826	7.011214	8.7388

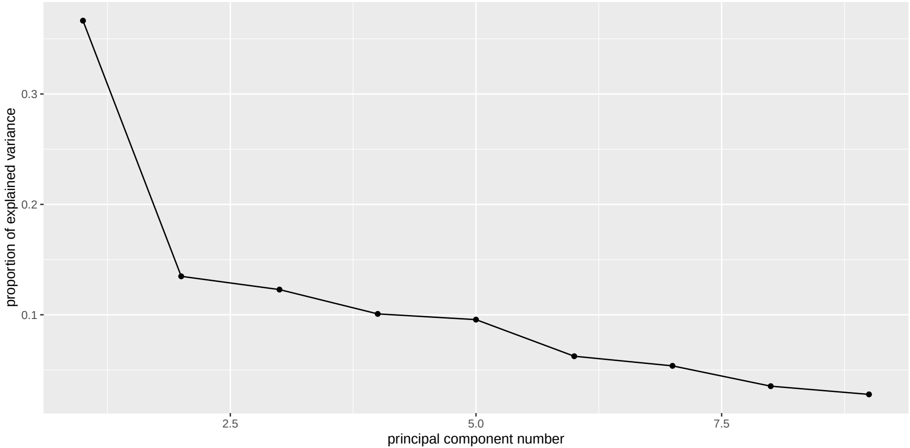
Principal components

```
places.1 <- princomp(places_numeric, cor = TRUE)
summary(places.1)
```

```
## Importance of components:
##                           Comp.1    Comp.2
## Standard deviation     1.8159827 1.1016178
## Proportion of Variance 0.3664214 0.1348402
## Cumulative Proportion  0.3664214 0.5012617
##                           Comp.3    Comp.4
## Standard deviation     1.0514418 0.9525124
## Proportion of Variance 0.1228367 0.1008089
## Cumulative Proportion  0.6240983 0.7249072
##                            Comp.5     Comp.6
## Standard deviation     0.92770076 0.74979050
## Proportion of Variance 0.09562541 0.06246509
## Cumulative Proportion  0.82053259 0.88299767
```

scree plot

```
ggscreeplot(places.1)
```



- big elbow at 2 (1 component); smaller elbow at 6 (5) and maybe 4 (3).

What is in each component?

```
places.1$loadings
```

```
##
## Loadings:
##          Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## climate   0.158         0.800  0.377         0.217
## housing   0.384  0.139         0.197 -0.580
## health    0.410 -0.372         0.113        -0.535
## crime     0.259  0.474  0.128         0.692 -0.140
## trans     0.375 -0.141 -0.141 -0.430  0.191  0.324
## educate   0.274 -0.452 -0.241  0.457  0.225  0.527
## arts      0.474 -0.104        -0.147        -0.321
## recreate  0.353  0.292        -0.404 -0.306  0.394
## econ      0.164  0.540 -0.507  0.476
##          Comp.7 Comp.8 Comp.9
## climate   0.151  0.341
```

Assessing the components

Look at component loadings and make a call about “large” (in absolute value) vs “small”. Large loadings are a part of the component and small ones are not. Thus, if we use 0.4 as cutoff:

- component #1 depends on health and arts
- #2 depends on economy and crime, and negatively on education.
- #3 depends on climate, and negatively on economy.
- #4 depends on education and the economy, negatively on transportation and recreation opportunities.
- #5 depends on crime and negatively on housing.

Comments

- The use of 0.4 is arbitrary; you can use whatever you like. It can be difficult to decide whether a variable is "in" or "out".
- The large (far from zero) loadings indicate what distinguishes the cities as places to live, for example:
    - places that are rated high for health also tend to be rated hight for arts
    - places that have a good economy tend to have a bad climate (and vice versa)
    - places that have a lot of crime tend to have bad housing.

Making a plot 1/3

How can we make a visual showing the cities? We need a "score" for each city on each component, and we need to identify the cities (we have a numerical `id` in the original dataset):

```r
cbind(city_id = places$id, places.1$scores) %>%
  as_tibble() -> places_score
places_score
```

city_id	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6	Comp.7	Comp.8	C
1	- 1.2006820	1.4766156	- 0.9457658	0.5570233	0.6763275	0.9534050	1.5379046	0.9569135	0
2	0.9397421	- 0.2488284	1.1075997	- 1.6593654	- 0.4120840	- 1.7082160	- 1.1965465	0.4896324	0
3	- 2.3510699	0.3402470	1.0254193	0.6504600	0.3985502	- 0.7400013	0.2592204	- 0.7342527	0
4	1.3813627	- 1.6161080	- 0.2566714	0.1791818	0.0709810	1.8077640	0.5482947	0.6379999	0

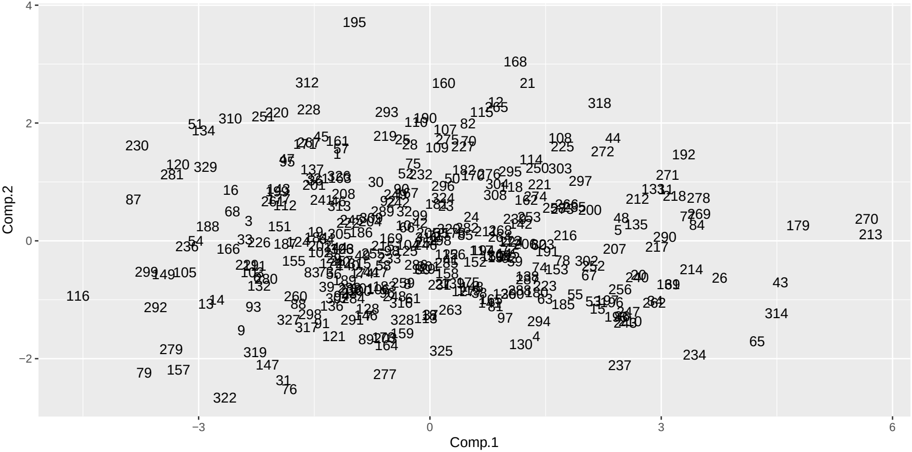
Making a plot 2/3

- Plot the first two scores against each other, labelling each point by the `id` of the city it belongs to:

```
ggplot(places_score, aes(x = Comp.1, y = Comp.2,
                         label = city_id)) +
  geom_text() -> g
```

Making a plot 3/3

```
g
```



Comments

- Cities 213 and 270 are high on component 1, and city 116 is low. City 195 is high on component 2, and city 322 is low.
- This suggests that cities 213 and 270 are high on health and arts, and city 116 is low. City 195 should be high on economy and crime and low on education, and city 322 should be the other way around.

Checking this 1/2

- The obvious way of checking this is in two steps: first, work out what high or low means for each variable:

```
summary(places)
```

```
##     climate          housing           health
##  Min.   :4.654   Min.   : 8.548   Min.   :3.761
##  1st Qu.:6.174   1st Qu.: 8.819   1st Qu.:6.368
##  Median :6.295   Median : 8.972   Median :6.725
##  Mean   :6.260   Mean   : 8.997   Mean   :6.805
##  3rd Qu.:6.384   3rd Qu.: 9.107   3rd Qu.:7.276
##  Max.   :6.813   Max.   :10.071   Max.   :8.968
##      crime           trans          educate
##  Min.   :5.730   Min.   :7.043   Min.   :7.439
##  1st Qu.:6.561   1st Qu.:8.052   1st Qu.:7.871
##  Median :6.853   Median :8.314   Median :7.935
##  Mean   :6.796   Mean   :8.283   Mean   :7.936
```

Checking this 2/2

and then find the values on the variables of interest for our cities of interest, and see where they sit on here.

Cities 270, 213, and 116 were extreme on component 1, which depended mainly on health and arts:

```
places %>% select(id, health, arts) %>%
  filter(id %in% c(270, 213, 166))
```

id	health	arts
166	6.142037	5.010635
213	8.968269	10.946323
270	8.223091	9.562827

City 166 is near or below Q1 on both variables. City 213 is the highest of all on both `health` and `arts`, while city 270 is well above Q3 on both.

Checking component 2

- Component 2 depended positively on economy and crime and negatively on education. City 195 was high and 322 was low:

```
places %>% select(id, econ, crime, educate) %>%
  filter(id %in% c(195, 322))
```

id	econ	crime	educate
195	9.208338	7.061334	7.789869
322	8.098035	6.139885	7.974533

- City 195 is the highest on economy, just above Q3 on crime, and below Q1 on education. City 322 should be the other way around: nearly the lowest on economy, below Q1 on crime, and between the median and Q3 on education. This is as we'd expect.

A better way: percentile ranks

- It is a lot of work to find the value of each city on each variable in the data summary.
- A better way is to work out the percentile ranks of each city on each variable and then look at those:

```
places %>%
  mutate(across(-id, ~percent_rank(.))) -> places_pr
places_pr
```

climate	housing	health	crime	trans	educate	arts	recreate	econ
0.3871951	0.0975610	0.0274390	0.4725610	0.4939024	0.4542683	0.2957317	0.2957317	0.957317
0.6707317	0.5762195	0.7804878	0.4268293	0.6829268	0.0945122	0.8506098	0.8475610	0.112804
0.2195122	0.3902439	0.3140244	0.5304878	0.1189024	0.2073171	0.0823171	0.0762195	0.429878
0.2378049	0.5182927	0.7439024	0.1615854	0.9573171	0.9512195	0.8079268	0.4695122	0.66768
0.9115854	0.6615854	0.8201220	0.9085366	0.9390244	0.7621951	0.7957317	0.8445122	0.618902

Look up cities and variables again

```
places_pr %>% select(id, health, arts) %>%
  filter(id %in% c(270, 213, 166))
```

id	health	arts
166	0.1524390	0.0487805
213	1.0000000	1.0000000
270	0.9695122	0.9817073

This shows that city 270 was also really high on these two variables: in the 97th percentile for `health` and the 98th for `arts`.

Component 2

- What about the extreme cities on component 2?

```
places_pr %>% select(id, econ, crime, educate) %>%
  filter(id %in% c(195, 322))
```

id	econ	crime	educate
195	1.0000000	0.7621951	0.0884146
322	0.0060976	0.0731707	0.6310976

City 322 was really low on economy and crime, but only just above average on education. City 195 was the highest on economy and really low on education, but only somewhat high on crime (76th percentile).

This, as you see, is much easier when you set it up.

The biplot

- draw it
- explain why little value
- redraw with cities minimized

Principal components from correlation matrix

Create data file like this:

```
 1         0.9705 -0.9600
 0.9705    1      -0.9980
-0.9600   -0.9980  1
```

and read in like this:

```
my_url <- "http://ritsokiguess.site/datafiles/cov.txt"
mat <- read_table(my_url, col_names = F)
mat
```

X1	X2	X3
1.0000	0.9705	-0.960
0.9705	1.0000	-0.998
-0.9600	-0.9980	1.000

Pre-processing

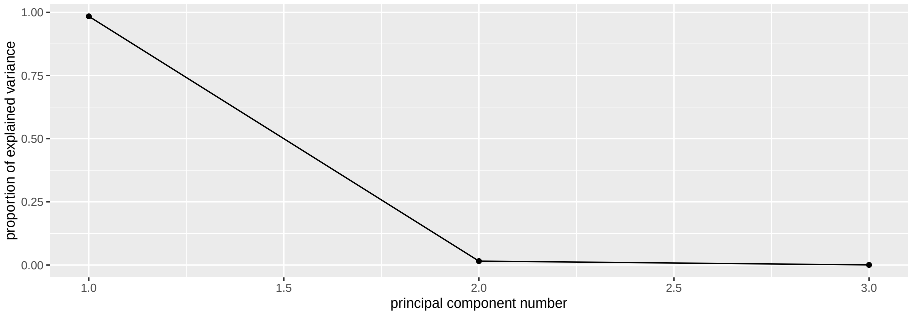
A little pre-processing required:

- Turn into matrix (from data frame)
- Feed into `princomp` as `covmat=`

```
mat.pc <- mat %>%
  as.matrix() %>%
  princomp(covmat = .)
```

Scree plot: one component fine

```r
ggscreeplot(mat.pc)
```



Component loadings

Compare correlation matrix:

```
mat
```

X1	X2	X3
1.0000	0.9705	-0.960
0.9705	1.0000	-0.998
-0.9600	-0.9980	1.000

with component loadings

```
mat.pc$loadings
```

```
##
## Loadings:
##    Comp.1 Comp.2 Comp.3
## X1  0.573  0.812  0.112
## X2  0.581 -0.306 -0.755
## X3 -0.578  0.498 -0.646
##
##                Comp.1 Comp.2 Comp.3
## SS loadings     1.000  1.000  1.000
```

Comments

- When X1 large, X2 also large, X3 small.
  - Then `comp.1` *positive*.
- When X1 small, X2 small, X3 large.
  - Then `comp.1` *negative*.

No scores

- With correlation matrix rather than data, no component scores
    - So no principal component plot
    - and no biplot.