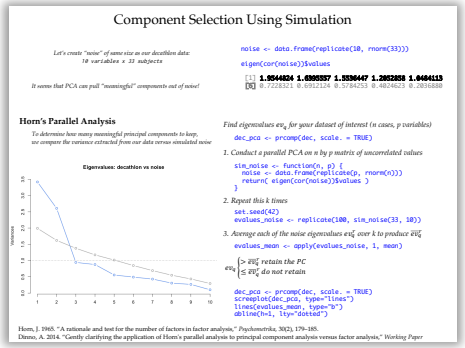
Business Analytics Using Computational Statistics

Week 13
Components and Composites

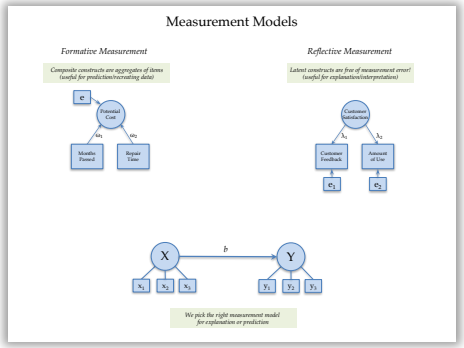
Week 14
PCA and Factors

Week 15
PLS Path Modeling

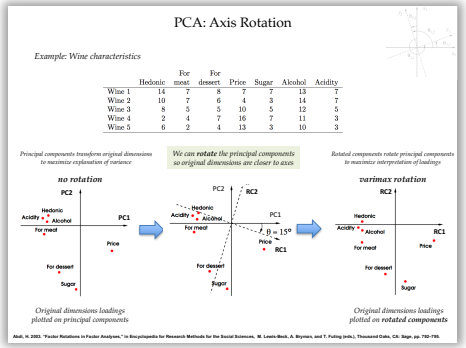
Parallel Analysis



Measurement Models



Rotation



Anonymous functions

temporary function created without a name
(similar to a "lambda" in other languages)

Long way of creating an anonymous function:

```
apply(cars_log, 2, function(x) { max(x) - min(x) } )
```

Short way of creating an anonymous function:

```
apply(cars_log, 2, \(x) { max(x) - min(x) } )
```

R v4.1+

Do NOT use this new syntax to define named functions

```
range_length <- \(x) {
  max(x) - min(x)
}
```

```
range_length <- function(x) {
  max(x) - min(x)
}
```

Forward Moving Scripts

Traditional data processing script

```
cars_log <- read.csv("cars_log.csv")
cars_log_cor <- round(cor(cars_log), 2)
```

We typically assign every line to a variable and use the variable in the next line…

- ✅ *variable name helps describe what line does*
- 😟 *hard to see actual steps; flow can jump right to left*

Native Piping in R!

R v4.1+

```
cars_log_cor <-
  read.csv("cars_log.csv") |>
  cor() |>
  round(2)
```

*notice that the result of last step must implicitly become **first** parameter*

*R now has a **native piping** operator |> that makes is even easier to use*

- ✅ *easy to read*
- 😟 *must pass last step's result to first parameter*

sometimes easier to read if code moves forward

pipe operator:

```
read.csv("cars_log.csv") |> cor( ) |> round( , 2)
```

Result can be sent to later parameters of next function:

R v4.2+

```
grepl("at", c("dogs", "cats", "rats"))
# [1] FALSE  TRUE  TRUE
```

Result of previous step can be sent to a later parameter using the underscore (_) symbol,
but only if you name the parameter

```
c("dogs", "cats", "rats") |> grepl("at", x = _)
```

Magrittr package for piping

```
library(magrittr)

cars_log_cor <- read.csv("cars_log.csv") %>%
  cor() %>%
  round(2)

c("dogs", "cats", "rats") %>% grepl("at", .)
```

*Magrittr package has a **pipe** operator %>% that also makes it easy to describe steps in your process*

✅ *no need to explicitly name output of each line*

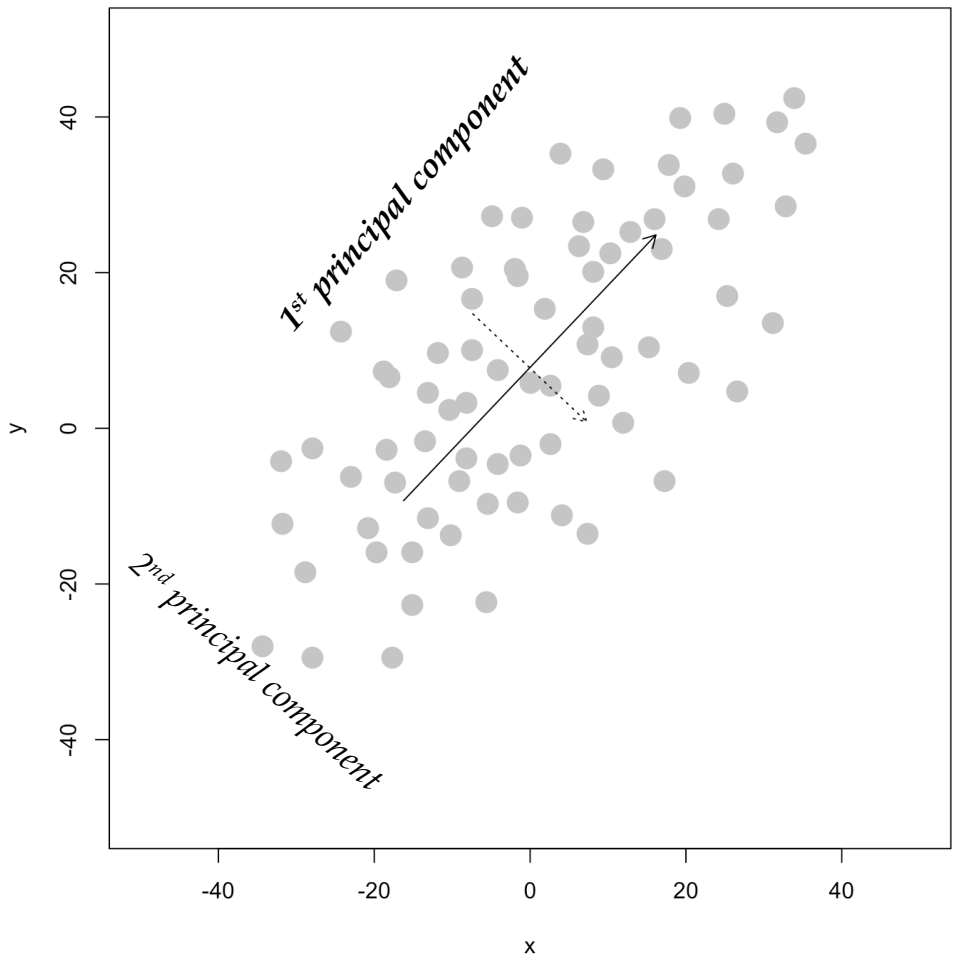
😟 *tiny bit slower than native piping*

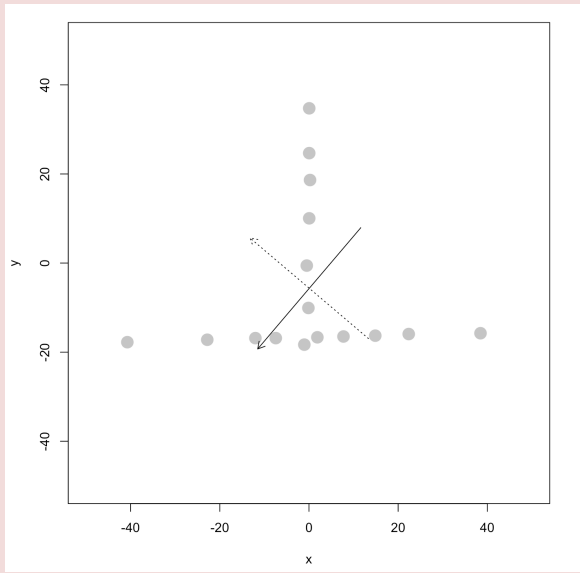
Magrittr was favored by most data scientists in the past.
I recommend using native pipes now, but know what else magrittr can do

https://magrittr.tidyverse.org/

PCA Visualized

PCA finds underlying dimensions in your data





But it may not always match your intuition…

```
install.packages("devtools")
devtools::install_github("soumyaray/compstatslib")
interactive_pca()
```

Dimension Reduction: Cars Example

One use of PCA is to reduce the dimensionality of data

Correlates:

```
correlates <- with(cars_log, data.frame(log.cylinders., log.displacement., log.horsepower., log.weight.))
cor(correlates) |> round(2)
```

	log.cylinders.	log.displacement.	log.horsepower.	log.weight.
log.cylinders.	1.00	0.95	0.83	0.88
log.displacement.	0.95	1.00	0.87	0.94
log.horsepower.	0.83	0.87	1.00	0.87
log.weight.	0.88	0.94	0.87	1.00

*These correlates capture **Heftiness***

***Hefty:** "large, heavy, and powerful"*
*(opposite of **light**)*

Eigenvalues:

```
hefty_eigen <- eigen(cor(correlates))

hefty_eigen$values
  [1] 3.674 0.188 0.104 0.034

sum(hefty_eigen$values)
  [1] 4

hefty_eigen$values / sum(hefty_eigen$values)
  [1] 0.919 0.047 0.026 0.009
```

PC1 captures 91.9% of original data's variance!

Eigenvalues

`eigen(cor(correlates))`

***Variance** of each component*

*Sums to **number of original dimensions***

Proportion of each eigenvalue/dimensions is variance captured!

Eigenvectors

All have magnitude 1:
doesn't tell us the importance of dimensions

```
hefty_eigen$vectors |> round(3)

       [,1]   [,2]   [,3]   [,4]
[1,] -0.498  0.536 -0.526 -0.434
[2,] -0.512  0.257  0.074  0.816
[3,] -0.486 -0.804 -0.342 -0.021
[4,] -0.504 -0.015  0.775 -0.381
```

magnitude $|\vec{v}| = \sqrt{\sum v_1{}^2 + v_2{}^2 + v_3{}^2 + v_4{}^2}$

```
apply(hefty_eigen$vectors, 2,
      FUN = \(x) {sqrt(sum(x^2)) } )
[1] 1 1 1 1
```

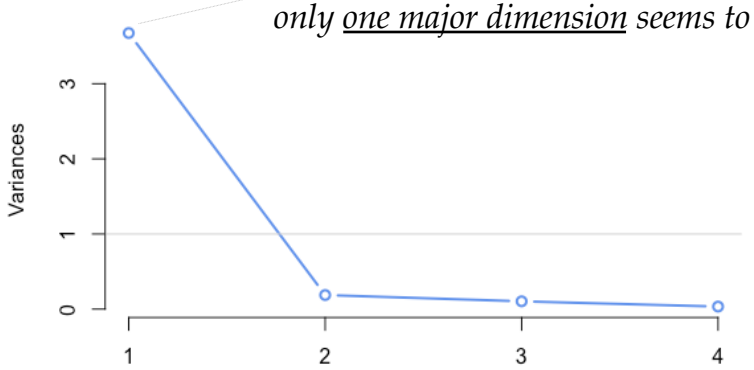
Principal Component Analysis of "Heftiness":

```
hefty_pca <- prcomp(correlates, scale. = TRUE)
```

standardize data for PCA

```
screeplot(hefty_pca, type="lines")
```

*Using both the '**eigenvalues > 1**' and **screeplot** criteria, only <u>one major dimension</u> seems to exist in these correlates*



Eigenvectors

```
hefty_pca$rotation
                        PC1         PC2         PC3        PC4
log.cylinders.    -0.4979145 -0.53580374  0.52633608  0.4335503
log.displacement. -0.5122968 -0.25665246 -0.07354139 -0.8162556
log.horsepower.   -0.4856159  0.80424467  0.34193949  0.0210980
log.weight.       -0.5037960  0.01530917 -0.77500928  0.3812031
```

PC1 is negatively associated with all heftiness items
*Let's call PC1: "**lightness**"*

*The **sign of eigenvectors** is useful in interpretation but it is arbitrary: different ways of estimating eigenvectors can produce positive/negative values*

```
summary(hefty_pca)
  Importance of components:
                          PC1     PC2     PC3     PC4
  Standard deviation     1.9168 0.43316 0.32238 0.18489
  Proportion of Variance 0.9186 0.04691 0.02598 0.00855
  Cumulative Proportion  0.9186 0.96547 0.99145 1.00000
```

***Eigenvalues** are variances*
*(**square of standard devations**)*

```
pca_report <- summary(hefty_pca)
pca_report$sdev^2 |> round(3)
  [1] 3.674 0.188 0.104 0.034
```

***Proportion of variance** in data captured by principal components*
(Same as we calculated on previous page!)

Raw Data vs. PC Scores

```
heft_scores <- heft_pca$x |> round(3)
          PC1    PC2    PC3    PC4
  [1,] -2.037 -0.379  0.380 -0.014
  [2,] -2.594  0.120  0.455 -0.129
  [3,] -2.238 -0.061  0.572 -0.086
         .      .      .      .
[392,]  1.122  0.214 -0.457  0.072
```

Dimension Reduction

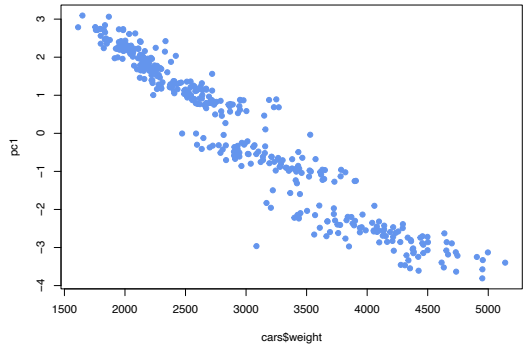
```
heft_scores <- heft_pca$x
pc1 <- heft_scores[, 1]
```

Using only the first PC
(four dimensions → 1 dimension)

Let's exanine the first principal component's relationship to weight…

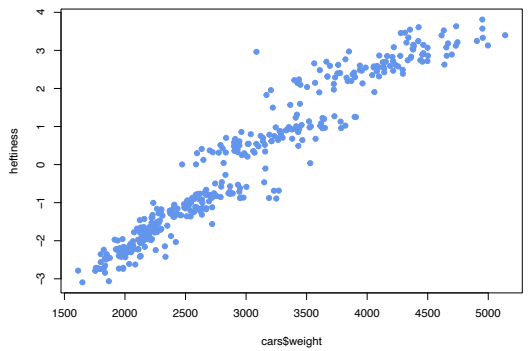
```
plot(cars$weight, pc1, pch=19, col="cornflowerblue")
```

The first principal component seems to be the opposite of "heftiness"!



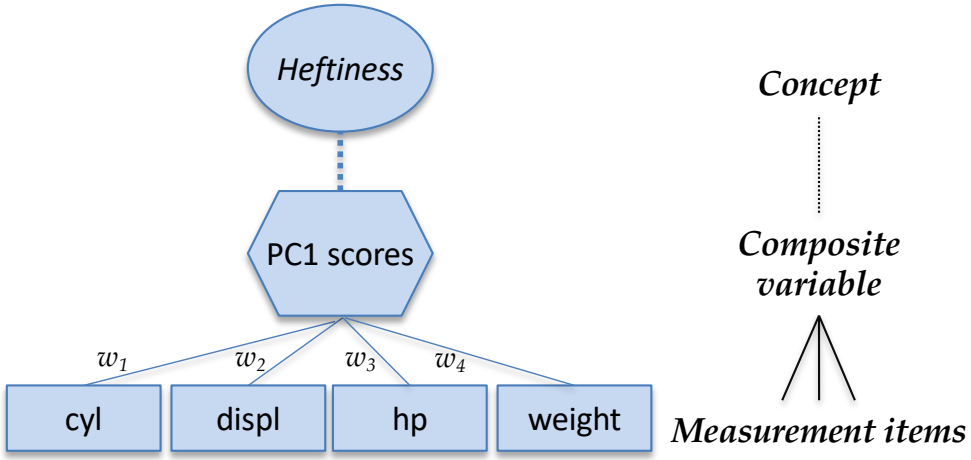

```
heftiness <- -1 * pc1
plot(cars$weight, heftiness, pch=19, col="cornflowerblue")
```

If we take the negative of PC1, it seems to match our idea of "heftiness"



Composite Measurement

(combining measurements)



Resolving Multicollinearity using PCA

Recall our full regression with all correlates

```
full_regr <- lm(
  log.mpg. ~ log.cylinders. + log.displacement. +
             log.horsepower. + log.weight. +
             log.acceleration. + model_year +
             factor(cars$origin),
  data = as.data.frame(scale(cars_log)))
```

```
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)             -0.05386    0.02626  -2.051  0.04094 *
log.cylinders.          -0.07281    0.05432  -1.340  0.18094
log.displacement.        0.03194    0.09144   0.349  0.72707
log.horsepower.         -0.28755    0.05851  -4.914 1.32e-06 ***
log.weight.             -0.49035    0.07043  -6.962 1.46e-11 ***
log.acceleration.       -0.09029    0.03174  -2.845  0.00469 **
model_year               0.32759    0.01918  17.078  < 2e-16 ***
factor(cars$origin)2     0.14915    0.06152   2.424  0.01580 *
factor(cars$origin)3     0.13885    0.06064   2.290  0.02259 *
```

```
vif(full_regr)
                        GVIF Df GVIF^(1/(2*Df))
log.cylinders.     10.456738  1        3.233688
log.displacement.  29.625732  1        5.442952
log.horsepower.    12.132057  1        3.483110
log.weight.        17.575117  1        4.192269
log.acceleration.   3.570357  1        1.889539
model_year          1.303738  1        1.141814
factor(origin)      2.656795  2        1.276702
```

We cannot simultaneously model these four correlates because they have high correlations with each other

Regression after dimension reduction (4 correlates → PC1)

```
hefty_regr <- lm(
  log.mpg. ~ scale(heftiness) +
             log.acceleration. +
             model_year + factor(cars$origin),
  data=as.data.frame(scale(cars_log)))
```

```
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)             -0.01588    0.02563  -0.620    0.536
scale(heftiness)        -0.82113    0.02851  28.805   <2e-16 ***
log.acceleration.       -0.10192    0.02220  -4.590    6e-06 ***
model_year               0.31612    0.01961  16.122   <2e-16 ***
factor(cars$origin)2     0.02431    0.05775   0.421    0.674
factor(cars$origin)3     0.05788    0.05704   1.015    0.311
```

```
vif(hefty_regr)
                        GVIF Df GVIF^(1/(2*Df))
scale(heftiness)    2.555047  1        1.598451
log.acceleration.   1.549984  1        1.244984
model_year          1.208794  1        1.099452
factor(cars$origin) 1.845987  2        1.165621
```

*We have **composited** our correlates into a **single construct***

Note that heftiness-vs-mpg is not linear… how would you fix it?

Dimensionality Analysis: Security Example

Another use of PCA is to <u>understand</u> the dimensions in our data

```
sec <- read.csv("security_questions.csv")

sec_eigen <- eigen(cor(sec))
sec_eigen$values
  [1] 9.31 1.60 1.15 0.76 0.68 0.61 0.50 0.47 0.45 0.39 0.35 0.30 0.29 0.26 0.23 0.23 0.21 0.20

sec_pca <- prcomp(sec, scale. = TRUE)
sec_pca$rotation[, 1:3] |> round(2)
```

	PC1	PC2	PC3
Q1	-0.27	0.11	0.00
Q2	-0.22	0.01	0.08
Q3	-0.25	0.03	0.08
Q4	-0.20	**-0.51**	0.10
Q5	-0.23	0.02	**-0.51**
Q6	-0.22	0.08	0.19
Q7	-0.22	0.25	0.30
Q8	-0.26	-0.03	-0.32
Q9	-0.24	0.18	0.19
Q10	-0.22	0.08	**-0.50**
Q11	-0.25	0.21	0.16
Q12	-0.21	**-0.50**	0.11
Q13	-0.23	0.05	0.08
Q14	-0.27	0.08	0.15
Q15	-0.23	-0.01	-0.31
Q16	-0.25	0.16	0.17
Q17	-0.20	**-0.53**	0.10
Q18	-0.26	0.09	-0.06

Authenticate correct website?

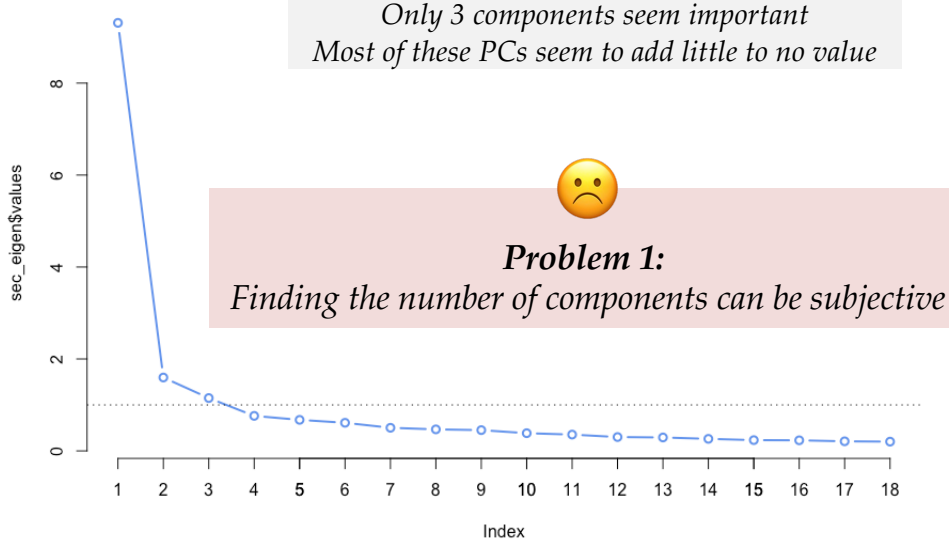
Q4+Q17: Prevent deniability of transactions

Cannot tell: all coefficients are almost the same…

How can we interpret
the meaning of our components

***PC1**: almost everything?*
***PC2**: Q4, Q12, Q17*
***PC3**: Q5, Q10*

Only 3 components seem important
Most of these PCs seem to add little to no value



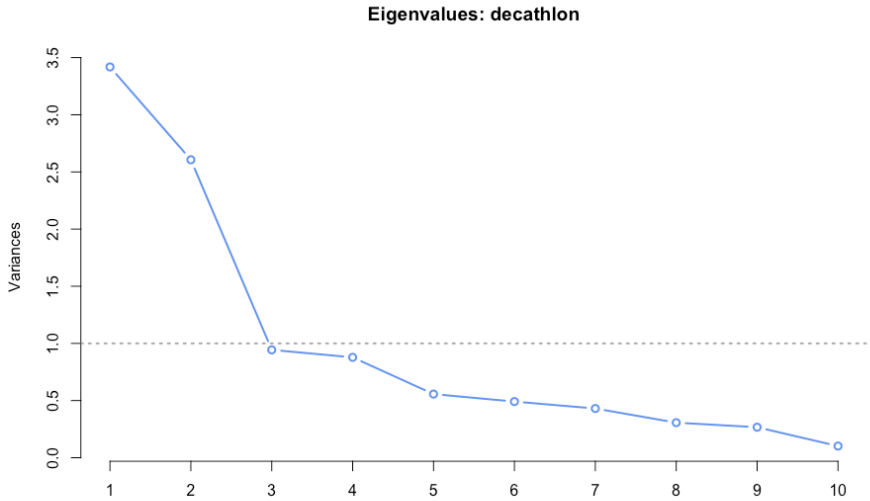
Problem 1:
Finding the number of components can be subjective

Problem 2:
It is difficult to interpret the
meaning of principle components

Comparing Data vs. Noise with PCA

Let's remember the screeplot of our decathlon data

```
dec <- read.table("decathlon_data.txt", header=TRUE)
screeplot(dec_pca, …)
```



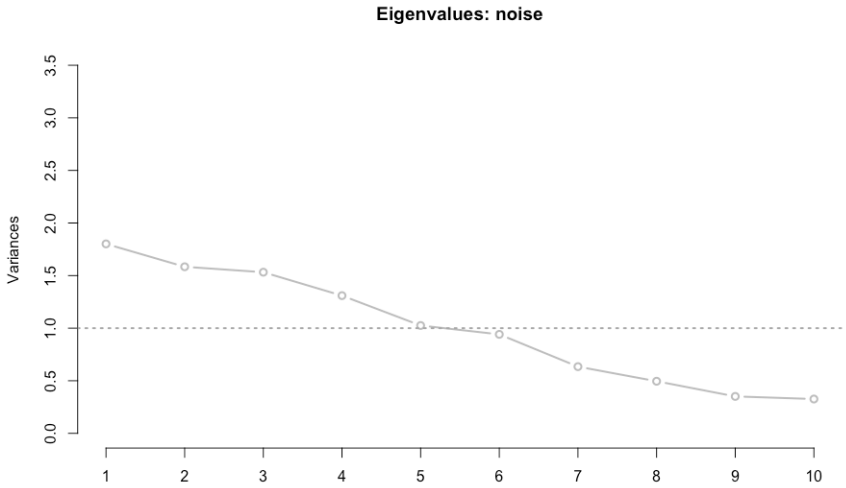
*Let's create "**noise**" of same size as our **decathlon** data:*
10 variables x 33 subjects

It seems that PCA can pull "meaningful" components (eigenvalue ≥ 1) out of noise!

```
noise <- data.frame(replicate(ncol(dec), rnorm(nrow(dec))))

eigen(cor(noise))$values |> round(2)
 [1] 1.80 1.58 1.53 1.31 1.03 0.94 0.63 0.50 0.35 0.33
```

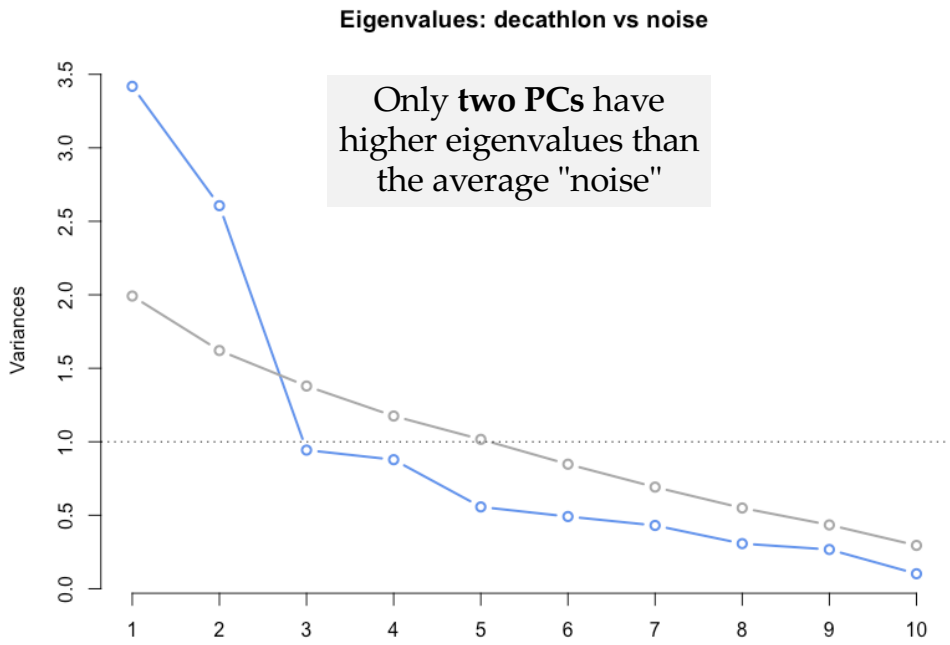
```
noise_pca <- prcomp(noise, scale. = TRUE)

screeplot(noise_pca, ylim=c(0,3.5), …)
abline(h=1, lty="dotted", col="darkgray", lwd=2)
```

Eigenvalue ≥ 1 can be found even in noise!



Horn's Parallel Analysis

To determine many meaningful principal components, we compare variance extracted from our data versus noise



Only **two PCs** have higher eigenvalues than the average "noise"

Find eigenvalues ev_q for your dataset of interest (n cases, p variables)

```
dec_pca <- prcomp(dec, scale. = TRUE)
```

1. Function to run a PCA on n × p dataframe of random values

```
sim_noise_ev <- function(n, p) {
  noise <- data.frame(replicate(p, rnorm(n)))
  eigen(cor(noise))$values
}
```

2. Repeat this k times

```
evalues_noise <- replicate(100, sim_noise_ev(33, 10))
```

3. Average each of the noise eigenvalues ev_q^r over k to produce $\overline{ev}_q^r$

```
evalues_mean <- apply(evalues_noise, 1, mean)
```

$$ev_q \begin{cases} > \overline{ev}_q^r \ \textit{retain the PC} \\ \leq \overline{ev}_q^r \ \textit{do not retain} \end{cases}$$

Compare eigenvalues ev_q to averaged eigenvalues of noise $\overline{ev}_q^r$:

```
screeplot(dec_pca, type="lines")
lines(evalues_mean, type="b")
abline(h=1, lty="dotted")
```

***Parallel analysis gives us an objective sense** of whether our principal components have value*

Use parallel analysis in conjunction with other criteria (ev > 1; screeplot)

Horn, J. 1965. "A rationale and test for the number of factors in factor analysis," *Psychometrika*, 30(2), 179–185.
Dinno, A. 2014. "Gently clarifying the application of Horn's parallel analysis to principal component analysis versus factor analysis," *available at: http://doyenne.com/Software/files/PA_for_PCA_vs_FA.pdf*

Interpreting Principal Components: Decathlon Example

Examining the results of PCA

```
dec_pca <- prcomp(dec, scale. = TRUE)

dec_pca$rotation
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
X100m	**-0.42**	0.15	-0.27	0.09	-0.44	0.03	0.25	-0.66	0.11	-0.11
lj	**0.39**	-0.15	-0.17	0.24	0.37	-0.09	0.75	-0.14	-0.05	-0.06
sp	**0.27**	0.48	0.10	0.11	-0.01	0.23	-0.11	-0.07	-0.42	-0.65
hj	**0.21**	0.03	-0.85	-0.39	0.00	0.07	-0.14	0.16	0.10	-0.12
X400m	**-0.36**	0.35	-0.19	-0.08	0.15	-0.33	0.14	0.15	-0.65	0.34
X110h	**-0.43**	0.07	-0.13	0.38	-0.09	0.21	0.27	0.64	0.21	-0.26
dis	**0.18**	0.50	0.05	-0.03	0.02	0.61	0.14	-0.01	0.17	0.53
pv	**0.38**	0.15	0.14	-0.14	-0.72	-0.35	0.27	0.28	0.02	0.07
jav	**0.18**	0.37	-0.19	0.60	0.10	-0.44	-0.34	-0.06	0.31	0.13
X1500m	**-0.17**	0.42	0.22	-0.49	0.34	-0.30	0.19	-0.01	0.46	-0.24

eigenvectors are the *"weights"* of our composite PC scores!
regression coefficients between PC score and items
(but they are still hard to interpret)

Component Scores

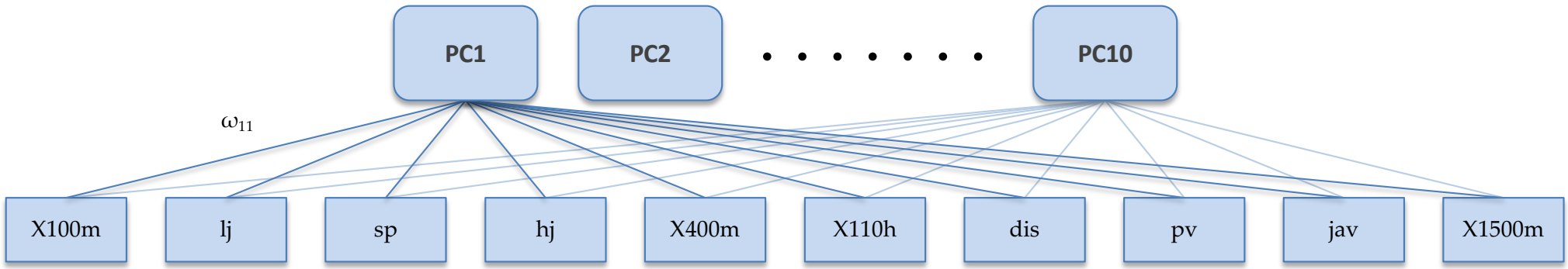
```
scores <- dec_pca$x
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
[1,]	1.73	1.23	-2.79	-0.29	0.16	1.74	0.27	0.09	0.25	-0.18
[2,]	2.79	-0.10	0.84	0.13	0.06	0.10	0.04	-0.02	0.33	-0.10
[3,]	1.88	-0.14	0.04	0.97	-0.91	-0.22	0.44	-0.23	0.28	0.29
.	.	.	.	.	.	.	.	.	.	.
[33,]	-4.12	-2.40	-0.90	1.00	0.51	-0.79	1.12	0.23	0.01	0.31

Confirming orthogonality of components

```
cor(scores) |> round(2)
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
PC1	1	0	0	0	0	0	0	0	0	0
PC2	0	1	0	0	0	0	0	0	0	0
PC3	0	0	1	0	0	0	0	0	0	0
PC4	0	0	0	1	0	0	0	0	0	0
PC5	0	0	0	0	1	0	0	0	0	0
PC6	0	0	0	0	0	1	0	0	0	0
PC7	0	0	0	0	0	0	1	0	0	0
PC8	0	0	0	0	0	0	0	1	0	0
PC9	0	0	0	0	0	0	0	0	1	0
PC10	0	0	0	0	0	0	0	0	0	1



$$PC_i = w_{i1} \cdot X100m + w_{i2} \cdot lj + w_{i3} \cdot sp + w_{i4} \cdot hj + w_{i5} \cdot X400m + w_{i6} \cdot X110h + w_{i7} \cdot dis + w_{i8} \cdot pv + w_{i9} \cdot jav + w_{i10} \cdot X1500m$$

*The **scores** of each principal component is a **weighted sum** of our original dimensions*

```
dec_scaled <- scale(dec)
pc_regr <- lm(scores[,"PC1"] ~ ., data=as.data.frame(dec_scaled))
pc_regr$coefficients |> round(2) |> as.vector()      # [1]  0.00 -0.42  0.39  0.27  0.21 -0.36 -0.43  0.18  0.38  0.18 -0.17
```

Eigenvectors are the regression weights of each component!

Reproducing Data from Components

```
dec_pca <- prcomp(dec, scale. = TRUE)
```

Our data has been decomposed into scores and weights

Weights (transposed)

```
weights <- dec_pca$rotation
```

	X100m	lj	sp	hj	X400m	X110h	dis	pv	jav	X1500m
PC1	-0.42	0.39	0.27	0.21	-0.36	-0.43	0.18	0.38	0.18	-0.17
PC2	0.15	-0.15	0.48	0.03	0.35	0.07	0.50	0.15	0.37	0.42
PC3	-0.27	-0.17	0.10	-0.85	-0.19	-0.13	0.05	0.14	-0.19	0.22
.	.	.	.	.	.	.	.	.	.	.
PC10	-0.11	-0.06	-0.65	-0.12	0.34	-0.26	0.53	0.07	0.13	-0.24

Scores

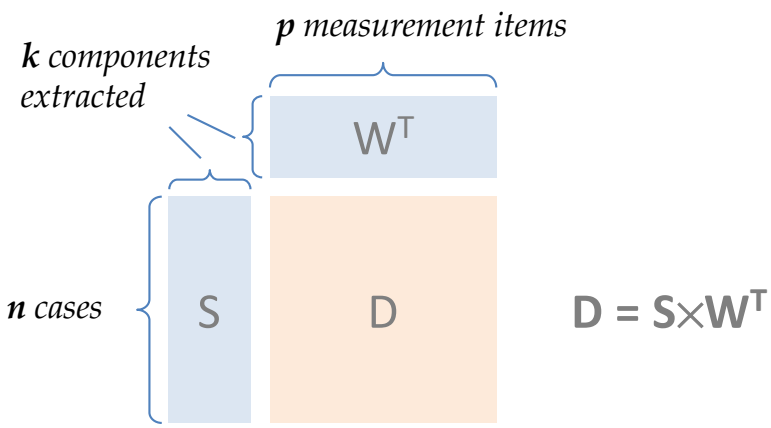
```
scores <- dec_pca$x
```

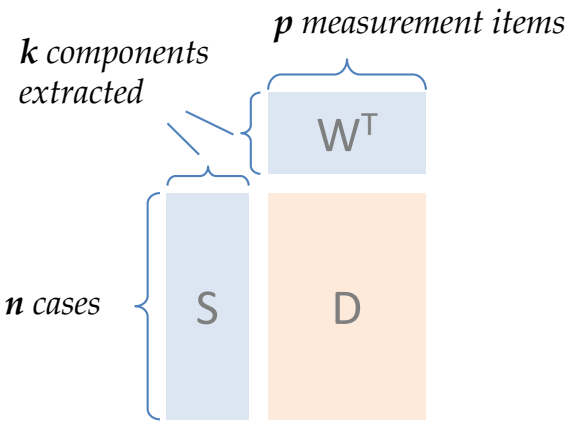
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
[1,]	1.73	1.23	-2.79	-0.29	0.16	1.74	0.27	0.09	0.25	-0.18
[2,]	2.79	-0.10	0.84	0.13	0.06	0.10	0.04	-0.02	0.33	-0.10
[3,]	1.88	-0.14	0.04	0.97	-0.91	-0.22	0.44	-0.23	0.28	0.29
.	.	.	.	.	.	.	.	.	.	.
[33,]	-4.12	-2.40	-0.90	1.00	0.51	-0.79	1.12	0.23	0.01	0.31

Original Data (scaled)

```
dec_scaled <- scale(dec)
```

	X100m	lj	sp	hj	X400m	X110h	dis	pv	jav	X1500m
[1,]	0.22	0.97	1.13	3.06	-0.35	0.16	1.86	-0.12	0.34	-0.52
[2,]	-1.34	1.04	0.75	-0.14	-1.46	-1.16	0.54	1.08	0.42	-0.22
[3,]	-0.07	1.01	0.17	-0.14	-0.92	-0.47	0.35	1.38	0.86	-0.94
.	.	.	.	.	.	.	.	.	.	.
[33,]	1.54	0.19	-2.78	-0.77	1.34	2.27	-2.15	-1.91	-0.82	-0.44



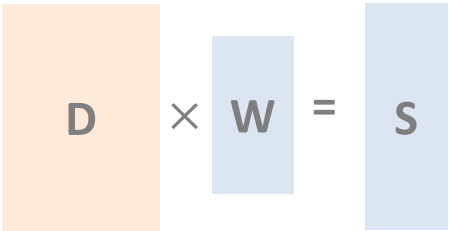


$$D = S \times W^T$$

Computing Scores From Weights

$$PC_i = w_{i1} \cdot X100m + w_{i2} \cdot lj + w_{i3} \cdot sp + w_{i4} \cdot hj + w_{i5} \cdot X400m + w_{i6} \cdot X110h + w_{i7} \cdot dis + w_{i8} \cdot pv + w_{i9} \cdot jav + w_{i10} \cdot X1500m$$

```
as.matrix(dec_scaled) %*% weights
```

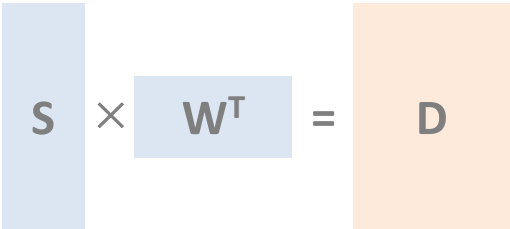


Scores

```
        PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10
 [1,]  1.73  1.23 -2.79 -0.29  0.16  1.74  0.27  0.09  0.25 -0.18
 [2,]  2.79 -0.10  0.84  0.13  0.06  0.10  0.04 -0.02  0.33 -0.10
 [3,]  1.88 -0.14  0.04  0.97 -0.91 -0.22  0.44 -0.23  0.28  0.29
   .     .     .     .     .     .     .     .     .     .     .
[33,] -4.12 -2.40 -0.90  1.00  0.51 -0.79  1.12  0.23  0.01  0.31
```

Reproducing Data from PCs

```
scores %*% t(weights)
```



Reproduced Data (scaled)

```
      X100m    lj    sp    hj X400m X110h   dis    pv   jav X1500m
 [1,]  0.22  0.97  1.13  3.06 -0.35  0.16  1.86 -0.12  0.34  -0.52
 [2,] -1.34  1.04  0.75 -0.14 -1.46 -1.16  0.54  1.08  0.42  -0.22
 [3,] -0.07  1.01  0.17 -0.14 -0.92 -0.47  0.35  1.38  0.86  -0.94
   .     .     .     .     .     .     .     .     .     .     .
[33,]  1.54  0.19 -2.78 -0.77  1.34  2.27 -2.15 -1.91 -0.82  -0.44
```

The full 10 PCs hold the same information as our 10 original items of data

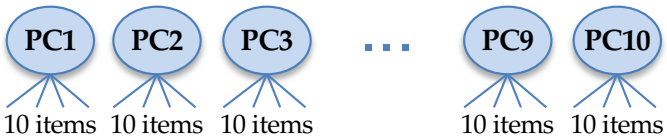
Let's make a function to reproduce our original data given the number of PCs to retain

```
reproduce_data <- function(original_data, k_pc) {
  pca_results <- prcomp(original_data, scale=TRUE)
  scores = pca_results$x[,1:k_pc]
  weights = pca_results$rotation[,1:k_pc]
  scores %*% t(weights)
}
```

$$S \times W^T = D$$

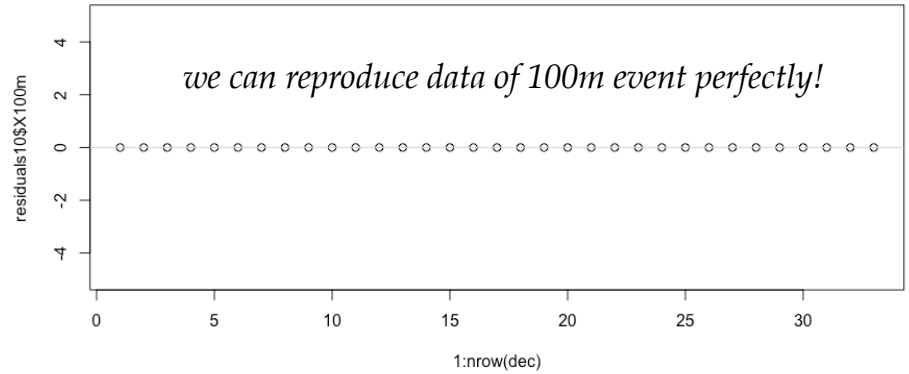
Reproducing Data From All PC dimensions

We can fully recreate our data using all 10 principal components



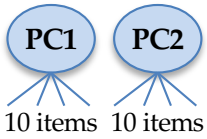

```
reproduce10 <- reproduce_data(dec_scaled, 10)
residuals10 = as.data.frame(dec_scaled - reproduce10)

plot(1:nrow(dec), residuals10$X100m, ylim = c(-5, 5))
abline(h = 0, col = "lightgray")
```

No residuals in reproducing original data



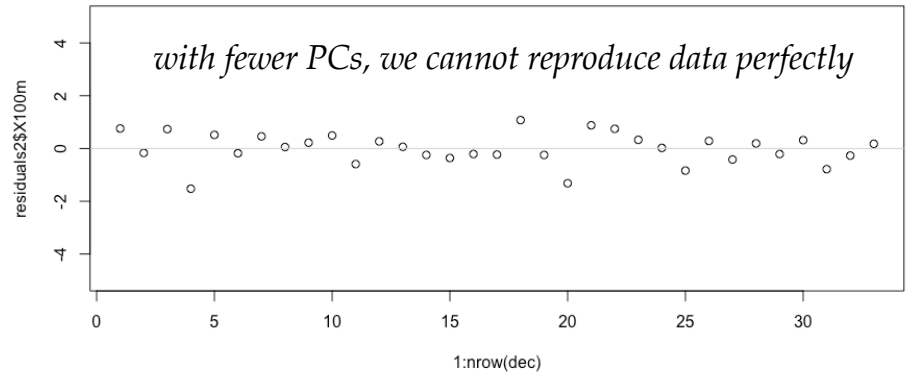
Reproducing Data After Dimension Reduction

By reducing dimensions (choosing fewer PCs), we must give up some precision of each measurement item




```
reproduce2 <- reproduce_data(dec, 2)
residuals2 = as.data.frame(scale(dec) - reproduce2)
plot(1:nrow(dec), residuals2$X100m, ylim=c(-5, 5))

plot(1:nrow(dec), residuals10$X100m, ylim = c(-5, 5))
abline(h = 0, col = "lightgray")
```

Residuals emerge in reproducing original data

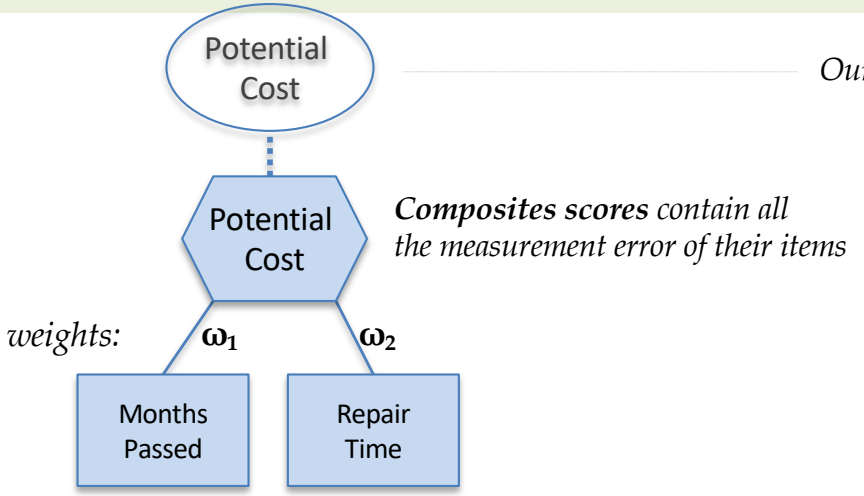


Measurement Models

How we conceptualize our construct will influence how we statistically represent it

Composite Measurement

Composite constructs are materially composed of items
(useful for prediction/recreating data)

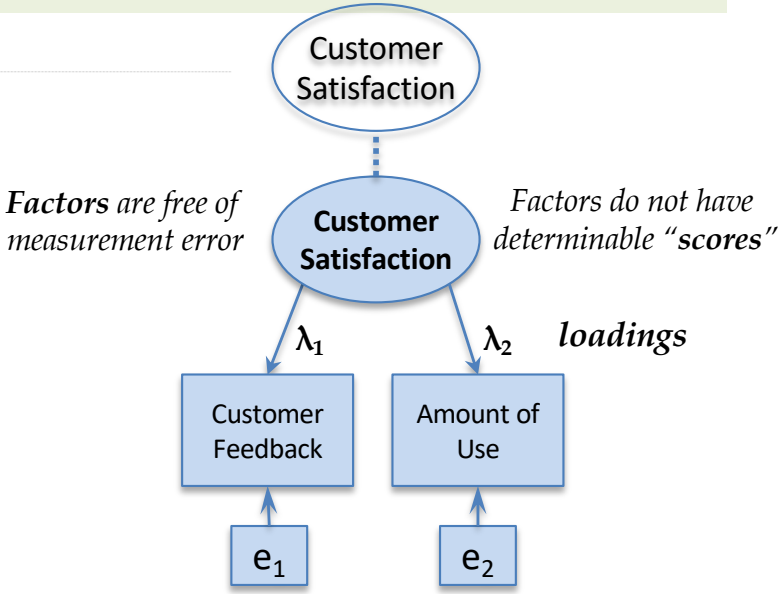


Composites scores *contain all the measurement error of their items*

*Our **concepts** of interest*

Factor Measurement

Latent constructs are the common cause of items
(useful for explanation/interpretation)

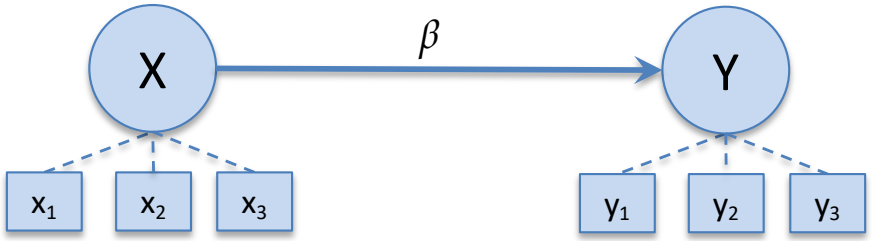


Factors *are free of measurement error*

*Factors do not have determinable "**scores**"*

*The **errors** capture items' unique sources of variance*

We pick the right measurement model based on the nature of our items



Using Components to Estimate a Factor Model

```
library(psych)
dec_principal <- principal(dec, nfactor=10, rotate="none", scores=TRUE)
 Standardized loadings (pattern matrix) based upon correlation matrix
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
X100m	-0.77	0.24	0.26	0.08	0.33	-0.02	0.17	-0.37	0.06	0.03
lj	0.73	-0.25	0.16	0.23	-0.28	0.07	0.49	-0.08	-0.02	0.02
sp	0.50	0.78	-0.10	0.10	0.01	-0.16	-0.07	-0.04	-0.22	0.21
hj	0.39	0.05	0.83	-0.36	0.00	-0.05	-0.09	0.09	0.05	0.04
X400m	-0.66	0.57	0.18	-0.08	-0.11	0.23	0.09	0.08	-0.34	-0.11
X110h	-0.80	0.11	0.12	0.36	0.07	-0.15	0.18	0.35	0.11	0.08
dis	0.33	0.81	-0.04	-0.02	-0.01	-0.43	0.09	-0.01	0.09	-0.17
pv	0.71	0.24	-0.13	-0.13	0.53	0.24	0.18	0.15	0.01	-0.02
jav	0.33	0.60	0.19	0.56	-0.07	0.31	-0.22	-0.03	0.16	-0.04
X1500m	-0.31	0.68	-0.22	-0.46	-0.25	0.21	0.12	0.00	0.24	0.08

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
SS loadings	3.42	2.61	0.94	0.88	0.56	0.49	0.43	0.31	0.27	0.10
Proportion Var	0.34	0.26	0.09	0.09	0.06	0.05	0.04	0.03	0.03	0.01
Cumulative Var	0.34	0.60	0.70	0.78	0.84	0.89	0.93	0.96	0.99	1.00

`principal()` – *Performs PCA, reports factor loadings*

`nfactor` – *number of components to extract*

`loadings` – *Scaled eigenvectors: these have meaning!*

Loadings = $\textbf{eigenvector} \times \sqrt{\textbf{eigenvalue}}$

$$L = \vec{v} \cdot s$$

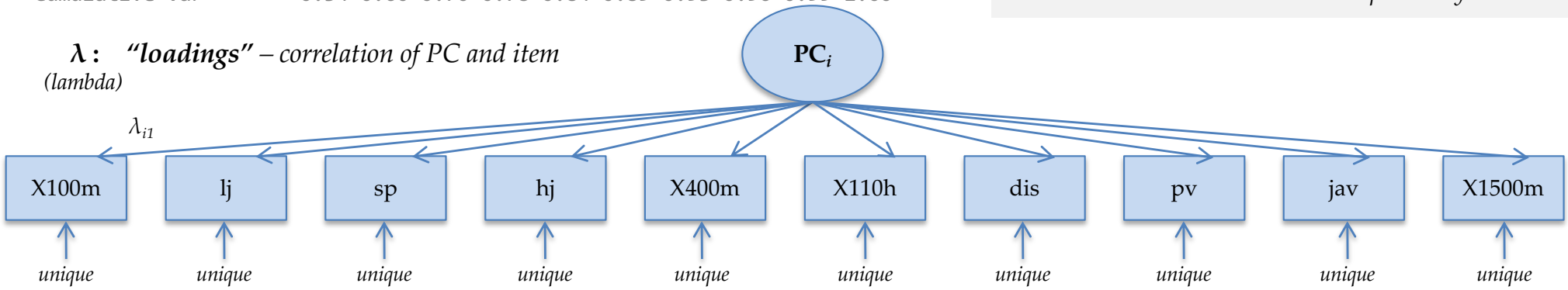
$\vec{v}$ = *eigenvector*
s = *standard deviation (square root of eigenvalue)*

`SS loadings` – *sum-of-square loadings is* ***eigenvalue***

```
sum(dec_principal$loadings[,"PC1"]^2)
  [1] 3.42
```

`Proportion Var` – *variance explained by each PC*
`Cumulative Var` – *total variance explained by all PCs*

λ **:** ***"loadings"*** – *correlation of PC and item*
(lambda)



Interpreting PC-item Loading

Simple regression coefficients

Correlations
(correlation of PC1 and X100m is -0.77)

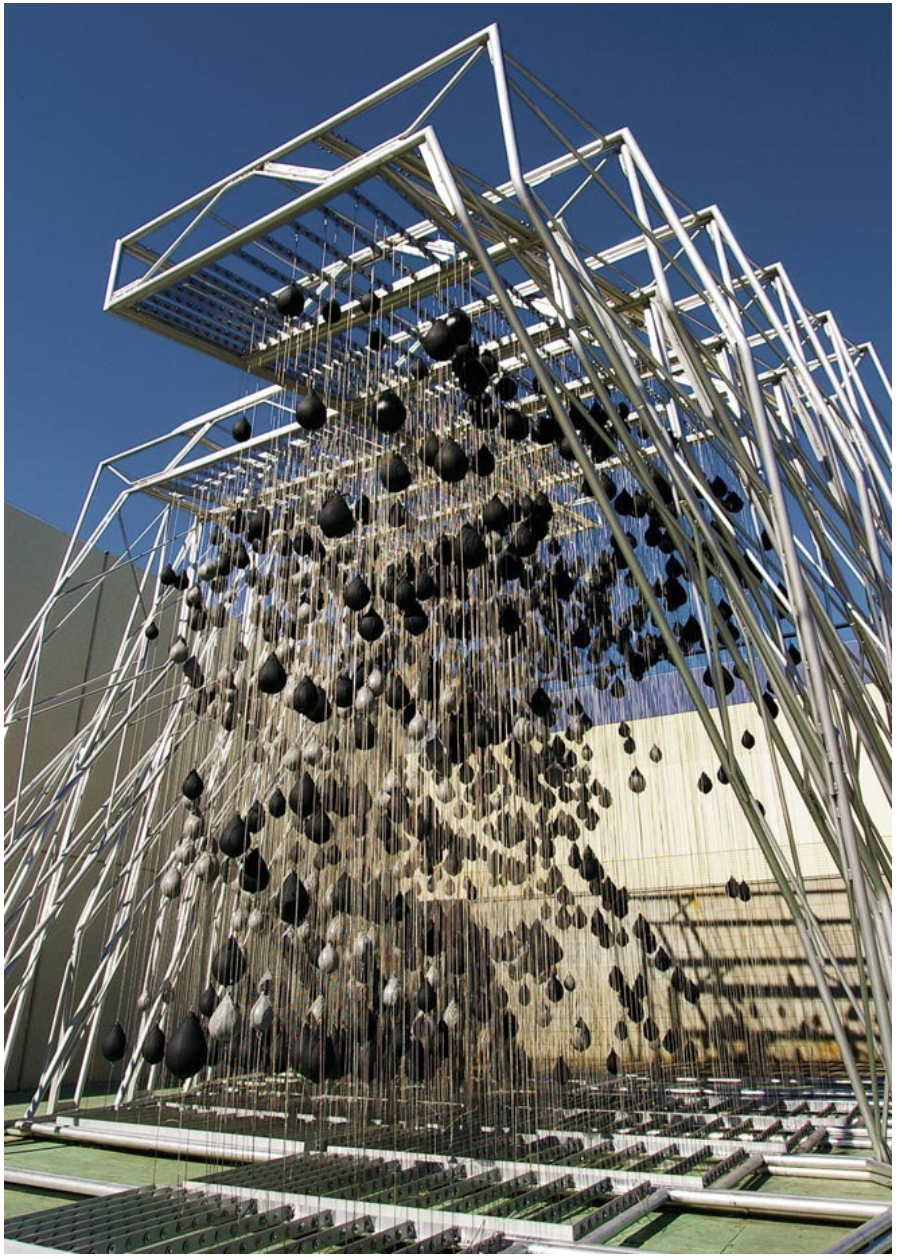
$$X100m = \lambda_1 \cdot PC + \varepsilon$$
$$lj = \lambda_2 \cdot PC + \varepsilon$$
$$sp = \lambda_3 \cdot PC + \varepsilon$$
$$\ldots$$
$$X1500m = \lambda_{10} \cdot PC + \varepsilon$$

Loadings*, which include magnitude and direction are easier to interpret than* ***eigenvectors***

λ > 0.70 is considered a good loading, more than half of item variance explained by PC

λ^2 : *"variance explained"*
proportion of item variance explained by factor

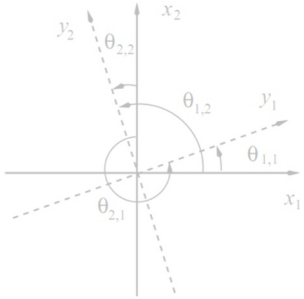
Rotation: Change in Perspective



Video: https://www.youtube.com/watch?v=1VcWXwV0vW8

Axis Rotation: Wine Example

We can rotate PCs relative to original dimensions (not data) to increase interpretabilty



Wine characteristics

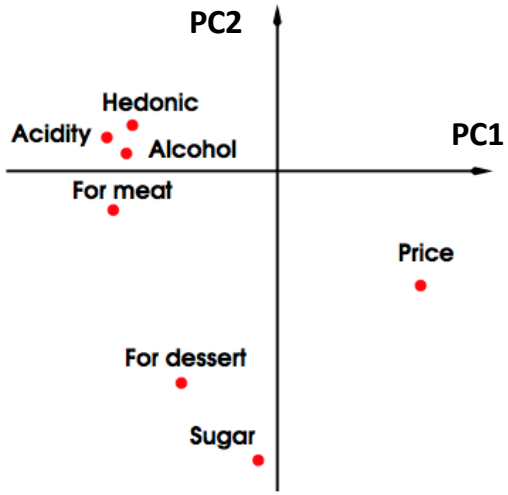
	Hedonic	For meat	For dessert	Price	Sugar	Alcohol	Acidity
Wine 1	14	7	8	7	7	13	7
Wine 2	10	7	6	4	3	14	7
Wine 3	8	5	5	10	5	12	5
Wine 4	2	4	7	16	7	11	3
Wine 5	6	2	4	13	3	10	3

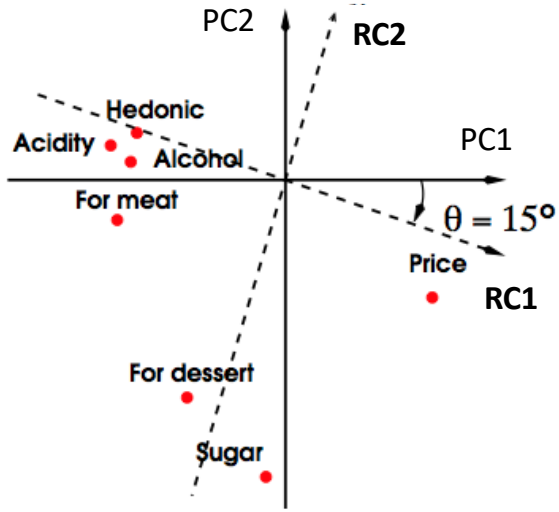
*Principal components transform original dimensions to **maximize variance explained***

*We can **rotate** the principal components so original dimensions are closer to axes*

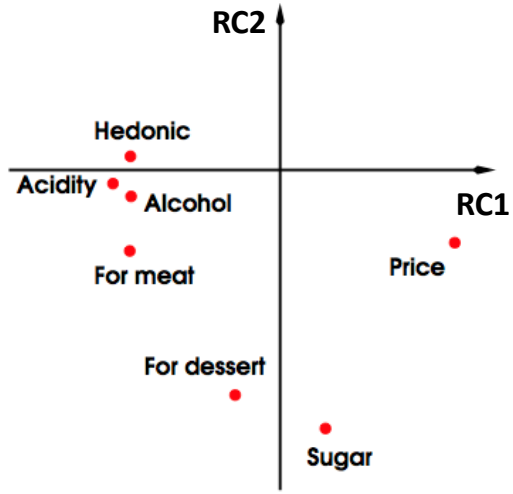
*Rotated components rotate principal components to **maximize interpretation of loadings***

no rotation





varimax rotation



Original dimensions loadings plotted on principal components

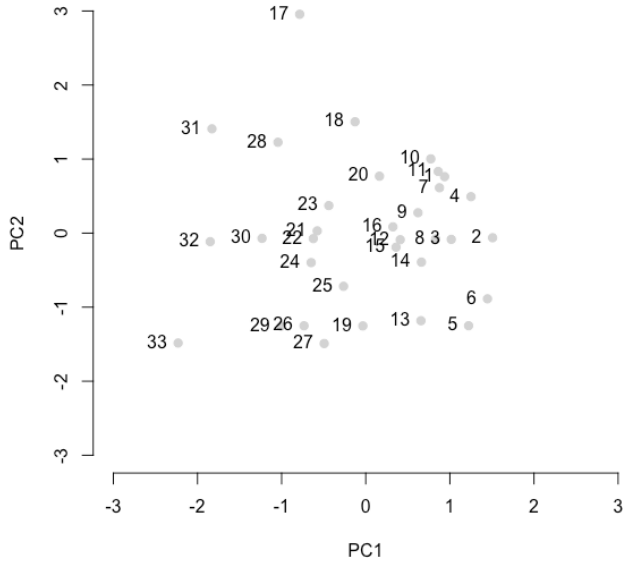
*Original dimensions loadings plotted on **rotated components***

Abdi, H. 2003. "Factor Rotations in Factor Analyses," in Encyclopedia for Research Methods for the Social Sciences, M. Lewis-Beck, A. Bryman, and T. Futing (eds.), Thousand Oaks, CA: Sage, pp. 792–795.

PCA: No Rotation

```
dec_pca2_orig <- principal(dec, nfactor=2, rotate="none", scores=TRUE)
```

Subject Scores (33 atheletes)



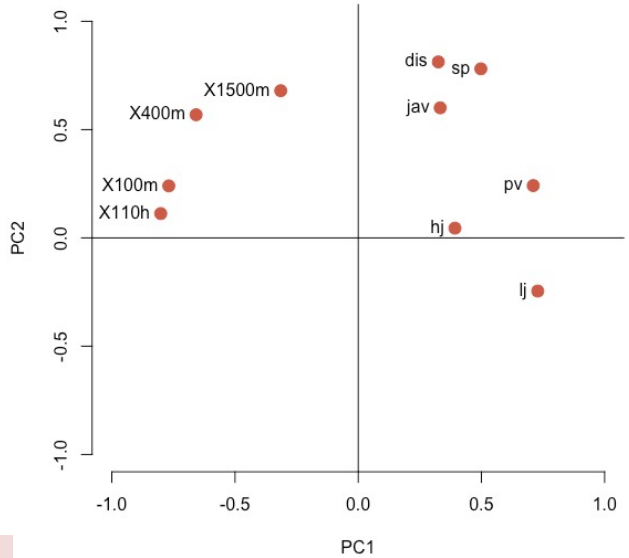

```
plot(dec_pca2_orig$scores, …)
text(dec_pca2_orig$scores,
     labels=1:nrow(dec_pca2_orig$scores))
```

Structure Matrix
(Unrotated Loadings)

	PC1	PC2
X100m	-0.77	0.24
lj	0.73	-0.25
sp	0.50	0.78
hj	0.39	0.05
X400m	-0.66	0.57
X110h	-0.80	0.11
dis	0.33	0.81
pv	0.71	0.24
jav	0.33	0.60
X1500m	-0.31	0.68

Loadings are clearer than eigenvectors, but several sports are still hard to distinguish

Dimension Comparisons




```
plot(dec_pca2_orig$loadings, …)
text(dec_pca2_orig$loadings,
     labels=rownames(dec_pca2_orig$loadings))
abline(h=0, v=0)
```

Original 10 dimensional data

100m	lj	sp	hj	400m	110h	dis	pv	jav	1500m
11.25	7.43	15.48	2.27	48.9	15.13	49.28	4.7	61.32	269
10.87	7.45	14.97	1.97	47.71	14.46	44.36	5.1	61.76	273
11.18	7.44	14.2	1.97	48.29	14.81	43.66	5.2	64.16	263.2
10.62	7.38	15.02	2.03	49.06	14.72	44.8	4.9	64.04	285.1
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
11.47	6.43	12.33	1.94	50.3	15	38.72	4	57.26	293.7
11.57	7.19	10.27	1.91	50.71	16.2	34.36	4.1	54.94	270

Reduced 2 dimensional scores

```
dec_pca2_orig$scores
```

```
             PC1         PC2
[1,]  0.93731950  0.76272582
[2,]  1.50755550 -0.06236234
[3,]  1.01642865 -0.08479291
[4,]  1.25112810  0.49216869
   .          .          .
   .          .          .
   .          .          .
[32,] -1.84912989 -0.11539163
[33,] -2.23052370 -1.48413203
```

```
cor(dec_pca2_orig$scores)
```

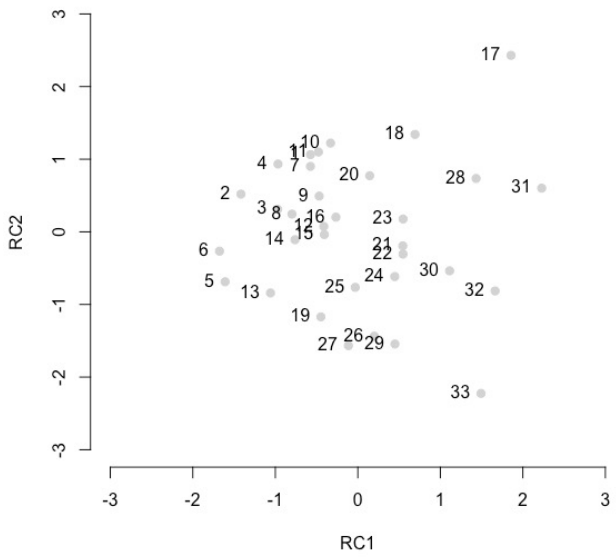
```
     PC1 PC2
PC1   1   0
PC2   0   1
```

Principal components are orthogonal

PCA: Varimax Rotation

```
dec_pca_rot <- principal(dec, nfactor=2, rotate="varimax", scores=TRUE)
```

It is now clearer that pv, hj, *and* X1500m *are unusual sports compared to the others*

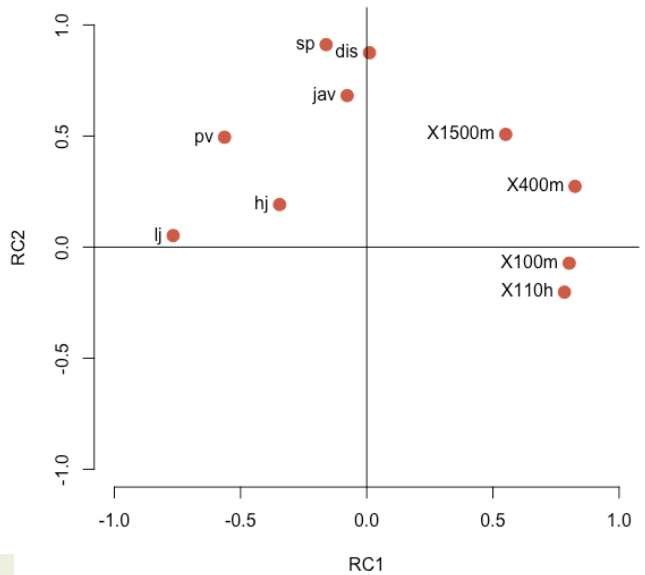



```
plot(dec_pca_orig$scores)
text(dec_pca_orig$scores, pos=2,
     labels=1:nrow(dec_pca_orig$scores))
abline(h=0, v=0)
```

Pattern Matrix
(Rotated Loadings)

	RC1	RC2
X100m	0.80	-0.07
lj	-0.77	0.05
sp	-0.16	0.91
hj	-0.35	0.19
X400m	0.83	0.27
X110h	0.78	-0.20
dis	0.01	0.88
pv	-0.56	0.49
jav	-0.08	0.68
X1500m	0.55	0.51

After rotation, most sports have distinct loadings on rotated components




```
plot(dec_pca_orig$loadings)
text(dec_pca_orig$loadings, pos=2,
    labels=rownames(dec_pca_orig$loadings))
abline(h=0, v=0)
```

Original 10 dimensional data

100m	lj	sp	hj	400m	110h	dis	pv	jav	1500m
11.25	7.43	15.48	2.27	48.9	15.13	49.28	4.7	61.32	269
10.87	7.45	14.97	1.97	47.71	14.46	44.36	5.1	61.76	273
11.18	7.44	14.2	1.97	48.29	14.81	43.66	5.2	64.16	263.2
10.62	7.38	15.02	2.03	49.06	14.72	44.8	4.9	64.04	285.1
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
11.47	6.43	12.33	1.94	50.3	15	38.72	4	57.26	293.7
11.57	7.19	10.27	1.91	50.71	16.2	34.36	4.1	54.94	270

Reduced 2 dimensional scores

```
dec_pca2_rot$scores
```

```
           RC1        RC2
[1,] -0.5741143  1.0633491
[2,] -1.4166777  0.5192659
[3,] -0.9715142  0.3106077
[4,] -0.9675701  0.9334664
        :          :         :
[32,] 1.6642351 -0.8141978
[33,] 1.4928380 -2.2247064
```

```
cor(dec_pca2_rot$scores)
```

```
     RC1 RC2
RC1    1   0
RC2    0   1
```

Rotated components are also orthogonal!

```r
principal(dec, nfactor=2, rotate="none", scores=TRUE)
```

```
Principal Components Analysis
Call: principal(r = dec, nfactors = 2, rotate = "none", scores = TRUE)
Standardized loadings (pattern matrix) based upon correlation matrix
        PC1   PC2   h2   u2 com
X100m  -0.77  0.24 0.65 0.35 1.2
lj      0.73 -0.25 0.59 0.41 1.2
sp      0.50  0.78 0.86 0.14 1.7
hj      0.39  0.05 0.16 0.84 1.0
X400m  -0.66  0.57 0.76 0.24 2.0
X110h  -0.80  0.11 0.65 0.35 1.0
dis     0.33  0.81 0.77 0.23 1.3
pv      0.71  0.24 0.56 0.44 1.2
jav     0.33  0.60 0.47 0.53 1.6
X1500m -0.31  0.68 0.56 0.44 1.4

                       PC1  PC2
SS loadings           3.42 2.61
Proportion Var        0.34 0.26
Cumulative Var        0.34 0.60
Proportion Explained  0.57 0.43
Cumulative Proportion 0.57 1.00
```

"Communality"

variance of X100m explained by both principal components

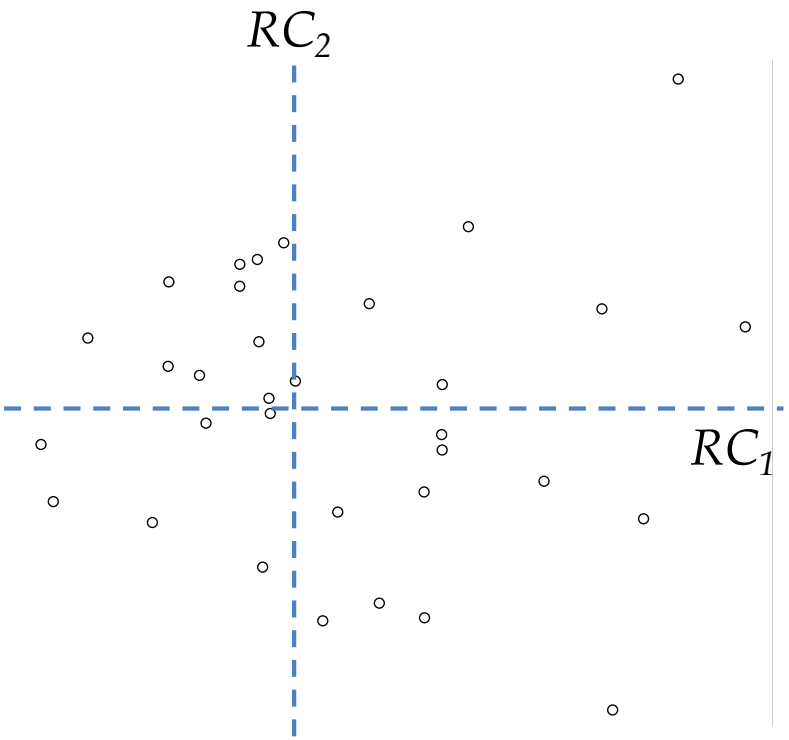
$h^2 = RC1^2 + RC2^2$

"Uniqueness"

Unexplained variance of X100m

$u^2 = 1 - \text{Communality}$

Sum-squared loadings: Eigenvalues




```r
principal(dec, nfactor=2, rotate="varimax", scores=TRUE)
```

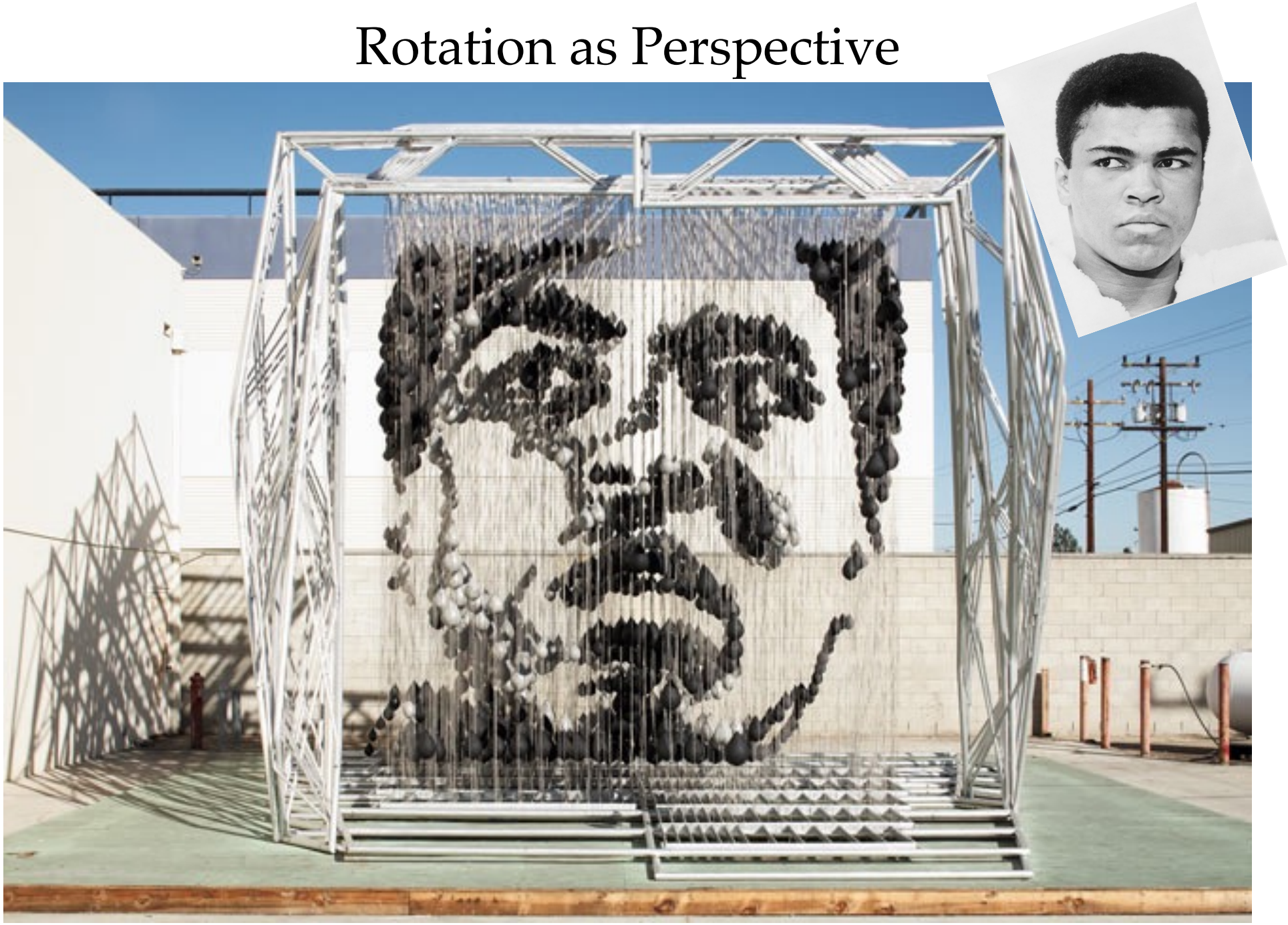
```
Principal Components Analysis
Call: principal(r = dec, nfactors = 2, rotate = "varimax", scores = TRUE)
Standardized loadings (pattern matrix) based upon correlation matrix
         RC1   RC2   h2   u2 com
X100m   0.80 -0.07 0.65 0.35 1.0
lj     -0.77  0.05 0.59 0.41 1.0
sp     -0.16  0.91 0.86 0.14 1.1
hj     -0.35  0.19 0.16 0.84 1.6
X400m   0.83  0.27 0.76 0.24 1.2
X110h   0.78 -0.20 0.65 0.35 1.1
dis     0.01  0.88 0.77 0.23 1.0
pv     -0.56  0.49 0.56 0.44 2.0
jav    -0.08  0.68 0.47 0.53 1.0
X1500m  0.55  0.51 0.56 0.44 2.0

                       RC1  RC2
SS loadings           3.30 2.73
Proportion Var        0.33 0.27
Cumulative Var        0.33 0.60
Proportion Explained  0.55 0.45
Cumulative Proportion 0.55 1.00
```

Rotated components ARE NOT principal Components!

Loadings are different
Variances explained are different
Scores are different

But communality of measurement items are the same
Components remain orthogonal

Rotation as Perspective



Video: https://www.youtube.com/watch?v=1VcWXwV0vW8